# ENHANCED COMPUTATION TIME

# FOR FAST BLOCK MATCHING ALGORITHM

**By**

**ZAYNAB ANWER AHMED**

***BSc, MSc***

A thesis submitted in partial fulfilment of the requirements of Liverpool John Moores University for the degree of Doctor of Philosophy

October 2013

# ACKNOWLEDGMENTS

My thanks go firstly, as it should always be, to **Allah** who gave me the energy, health, and courage to spend a lot of determination and time until I completed this work.

My most sincere gratitude and appreciation to my supervisor **Dr Abir Hussain** for all her patience, valuable advice, discussions, convincing arguments and more during the life of this thesis, I wish her all the best for the future. I also owe a great debt to **Dr. Dhiya Al-Jumeily** for his support, advice and encouragement from the beginning until now and I would like to thank him for his constructive comments on my thesis.

My gratitude also extends to the **Director and staff of the School of Computing & Mathematical Sciences, Liverpool John Moores University** for their advice and help.

I would like to express my deepest sense of gratitude to my sponsor, the **Ministry of Higher Education and Scientific Research in Iraq** and the **Iraqi Culture Attaché in London** for financial support.

My heartiest and warm thanks to my family, who deserve great thanks for their support, patience and understanding throughout my PhD time: starting with my husband, **Hussein Al-Bayati, my mother, my father, my brother and my sisters**, ending with my lovely children **Khadija, Abdullah** and **Sohaib** for their true understanding and sacrifice in uncountable ways.

Finally, I would like to thank all my friends for their support during my PhD study, and who supported me during the times of weakness. I would like also to thank and appreciate Dr. Waleed Al-Nuaimy and his family for helping me in a lot of things from the first day and during my PhD.

# ABSTRACT

Video compression is the process of reducing the amount of data required to represent digital video while preserving an acceptable video quality. Recent studies on video compression have focused on multimedia transmission, videophones, teleconferencing, high definition television (HDTV), CD-ROM storage, etc. The idea of compression techniques is to remove the redundant information that exists in the video sequences.

Motion compensated predictive coding is the main coding tool for removing temporal redundancy of video sequences and it typically accounts for 50-80% of the video encoding complexity. This technique has been adopted by all of the existing international video coding standards. It assumes that the current frame can be locally modelled as a translation of the reference frames. The practical and widely method used to carry out motion compensated prediction is block matching algorithm. In this method, video frames are divided into a set of non-overlapped macroblocks; each target macroblock of the current frame is compared with the search area in the reference frame in order to find the best matching macroblock. This will carry out displacement vectors that stipulate the movement of the macroblocks from one location to another in the reference frame. Checking all these locations is called full Search, which provides the best result. However, this algorithm suffers from long computational time, which necessitates improvement. Several methods of Fast Block Matching algorithm were developed to reduce the computation complexity.

This thesis focuses on two classifications: the first is called the *lossless block matching algorithm* process, in which the computational time required to determine the matching macroblock of the full search is decreased while the resolution of the predicted frames is the same as for the full search. The second is called the *lossy block matching algorithm* process, which reduces the computational complexity effectively but the search result's quality is not the same as for the full search.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# ACRONYMS

| | |
|---|---|
| 1BT | One–Bit Transformation |
| 2BT | Two-Bit Transformation |
| 2D-LOG | Two-Dimensional Logarithmic Search |
| 4SS | Four Step Search |
| AC | Alternating current |
| ARPS | Adaptive Rood Pattern Search |
| ARP-ZMP | Adaptive Rood Pattern-Zero Motion Prejudgment |
| BDM | Block Distortion Measure |
| BMA | Block Matching Algorithm |
| BMME | Block Matching Motion Estimation |
| BPM | Bit-Plane Matching |
| CABAC | Context-Adaptive Binary Arithmetic Coding |
| CAVLC | Context-Adaptive VLC |
| CDS | Cross-Diamond Search algorithm |
| CIF | Common Intermediate Format |
| dB | decibels |
| DC | Direct current |
| DCT | Discrete Cosine Transform |
| DFD | Displaced Frame Difference |
| DS | Diamond Search |
| DWT | Discrete Wavelet Transform |
| EMPBM | Enhanced Mean Predictive Block Matching |
| ES | Exhaustive Search |
| ETSS | Efficient Three Step Search |
| FBSME | Fixed Block-Size Motion Estimation |
| FCsFS | Fast Computations of Full Search |
| FPS | Frames Per Second |
| FS | Full Search |
| GOP | Group Of Pictures |
| HDTV | High Definition Television |
| HEVC | High Efficiency Video Coding |

| | |
|---|---|
| HVS | Human Visual System |
| ISO/IEC | International Organization for Standardization/ International Electrotechnical Commission |
| JABMS | Joint Adaptive Block Matching Search |
| JCT-VC | Joint Collaborative Team on Video Coding |
| JVT | Joint Video Team |
| KLT | Karhunen–Loeve Transform |
| LDSP | large diamond search pattern |
| LSP | Large Search Pattern |
| MAD | Mean Absolute Difference |
| MBl | MacroBlock |
| MBls | MacroBlocks |
| MC | Motion Compensation |
| MCP | Motion Compensated Prediction |
| ME | Motion Estimation |
| MODS | Modified DS |
| MPBM | Mean Predictive Block Matching |
| MPEG | Moving Picture Experts Group |
| MRFME | Multiple Reference Frames' ME |
| MSE | Mean Square Error |
| MV | Motion Vector |
| NHS | Novel Hexagon-based Search |
| NNMP | Number of Non-Matching Points |
| NTSC | National Television System Committee |
| OSA | Orthogonal Direction Search |
| PAL | Phase Alternation Line |
| PDE | Partial Distortion Elimination |

| | |
|---|---|
| PDEDS | PDE Diamond Search |
| PDENTSS | PDE New Three Step Search |
| PSNR | Peak Signal-to-Noise Ratio |
| Q | Quantisation |
| QCIF | Quarter Common Intermediate Format |
| QSIF | Quarter Source Input Format |
| RLC | Run Length Coding |
| RPE | Residual Prediction Error |
| SAD | Sum of Absolute Differences |
| SDSP | small diamond search pattern |
| SEA | Successive Elimination Algorithm |
| SECAM | SEquential Couleur Avec Memoire |
| SESTSS | Simple and Efficient Search |
| SIF | Source Input Format |
| SQCIF | Sub-Quarter Common Intermediate Format |
| SSP | Small Search Pattern |
| SVD | Singular Value Decomposition |
| TC | Transform Coding |
| TCon | Multi-pattern-based search |
| TSS | Three Step Search |
| UMHexagonS | Unsymmetrical Multi-Hexagon search |
| UVLC | Universal VLC |
| VBSME | Variable Block-Size Motion Estimation |
| VCEG | Visual Coding Experts Group |
| VLC | Variable Length Coding |
| VQ | Vector Quantisation |
| XOR | exclusive-or |

# LIST OF PUBLICATIONS

1. AHMED, Z., HUSSAIN, A. J. & AL-JUMEILY, D. (2011). "Mean Predictive Block Matching (MPBM) for fast block-matching motion estimation". *In Proc.* IEEE 3rd European Workshop on Visual Information Processing (EUVIP), Paris, France : pp. 67-72.

2. AHMED, Z., HUSSAIN, A. J. & AL-JUMEILY, D. (2011). "Fast Computations of Full Search Block Matching Motion Estimation (FCsFS)". The 12th Annual Conference on the Convergence of Telecommunications, Networking & Broadcasting (PGNet), 27- 28 June. Liverpool, UK.

3. AHMED, Z., HUSSAIN, A. J. & AL-JUMEILY, D. (2011). "Enhanced Computation Time for Fast Block Matching Algorithm". *In Proc.:* IEEE Developments in E-systems Engineering (DeSE), Abu Dhabi: pp. 289-293.

4. AHMED, Z., HUSSAIN, A. J. & AL-JUMEILY, D. (2012). "Edge detection for fast block-matching motion estimation to enhance Mean Predictive Block Matching algorithm". *In Proc.:* IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA), Trabzon, Turkey: pp. 1-5.

5. AHMED, Z., HUSSAIN, A. J. & AL-JUMEILY, D. (Expected 2014); "Mean Predictive Block Matching for Motion Estimation", submitted to PLOS ONE journal (accepted with revision).

# CHAPTER 1: INTRODUCTION

Digital video is a series of orthogonal bitmap digital images called frames displayed in rapid succession at a constant rate to give the illusion of a motion picture. Digital video applications have been extended to a wide range of industrial applications, especially in the area of entertainment, communications, and broadcasting. As a result of technological advances, several commercial products are becoming an integral part of modern life, such as High Definition Television (HDTV), digital cinema, smart phones, and other mobile devices. Huge revenue from these products and services is being gained since the number of end users increases continuously. Currently, more than one billion unique users visit YouTube each month [YouTube, 2013], and video chat reaches tens of thousands of users online at any time during a day [Tian et al., 2013]. In addition, the digital video industry invests a lot of money in the research and development of video technology (around £1.5 billion in 2013 and is expected to be more than £2.73 billion in 2017 in the UK alone [eMarketer, 2013]) to ensure continuous growth in the long term. The major challenge for efficient digital video storage and transmission lies in the huge amount of data needed to display digital video, and hence a large memory space is required to store video images, and equally large bandwidth is required for their transmission. To reduce this amount of data while preserving an acceptable video quality, different *video compression* techniques have been actively proposed and developed by researchers and companies since the 1980s [Al-Mualla et al., 2002]. The idea of these techniques is to provide efficient solutions to represent video data in a more compact and robust way so that the information can be stored or transmitted faster in videoconferencing and videophone, digital broadcasting, interactive games (internet), etc. Well-known international *video coding standards* include the former MPEG series and H.26x series [ISO/IEC, 1993; ISO/IEC, 1996; ITU-T and ISO/IEC, 2003; Sullivan et al., 2004; Sullivan and Wiegand, 2005; Ohm and Sullivan, 2013].

The main idea of compression techniques is to remove the redundant information that exists in video sequences. Digital video carries four types of redundancy: colour space redundancy, spatial redundancy, temporal redundancy and statistical redundancy [Richardson, 2010]. These redundancies are processed separately because of the differences in their characteristics. Video compression contains two systems: *video*

*encoders* and *video decoders*. A *video encoder* compresses the original video for storage and transmission, after which the encoded video is decompressed by a *video decoder* back to the displayable video before playback and editing.

A video encoder consists of three main functional units: colour subsampling (to remove colour redundancy), inter-frame encoder (to remove temporal redundancy) or intra-frame encoder (to remove spatial redundancy), and an entropy encoder (to remove statistical redundancy), as shown in Figure 1.1.

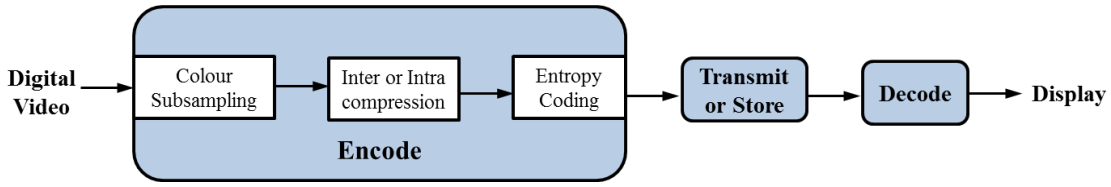


**Figure 1.1: Encoder/decoder**

Video compression efficiency is achieved by an *inter-frame* encoder, which reduces or eliminates temporal redundancy [Bhattacharyya and Deprettere, 2010]. An inter-frame encoder exploits the high correlation that exists between successive frames in video sequences especially if the frame rate is high. This correlation leads to temporal redundancy. The goal of inter-frame encoding is to reduce this redundancy. Video coding standards share a number of common features for inter-frame encoding. Each standard assumes that after colour subsampling there are four stages of inter-frame encoding to produce the compressed bitstream, which are: temporal prediction, transform, quantisation and entropy coding.

Temporal prediction is the main tool that reduces temporal redundancy by predicting some frames from others to reduce the transmission rate of the sequence of the video images and obtain high compression. This means that the current frame ($F_c$) can be locally modelled as a translation of the reference frame ($F_r$). Reference frames have to be encoded first, while a residual (difference) between current and reference frames which contain less energy will be encoded later instead of encoding the current frame [Richardson, 2003]. To decrease this residual, the prediction can be improved by estimating the motion of the moving objects between the current and the reference

frames, which is called Motion Estimation (ME) technique [Sayood, 2006]; that is, the motion estimation used to calculate the Motion Vectors (MVs) by comparing the current frame and the reference frame. The technique that uses MVs to predict a new frame from a reference frame is called Motion Compensation (MC). The predicted frame is known as the Motion Compensated Prediction (MCP) [Richardson, 2010]. The first output of this process will be the difference between the current frame and the MCP, which is called the Residual Prediction Error (RPE) (or Displaced Frame Difference (DFD)); the second output will be the motion vectors. The MVs are encoded using entropy coding and RPE between the current frame and the MCP is encoded using transform coding, quantisation and entropy coding, as shown in Figure 1.2 [Sullivan et al., 2004; Leontaris et al., 2009; Richardson, 2010; Sayood, 2006; Marpe et al., 2006; Al-Mualla et al., 2002].

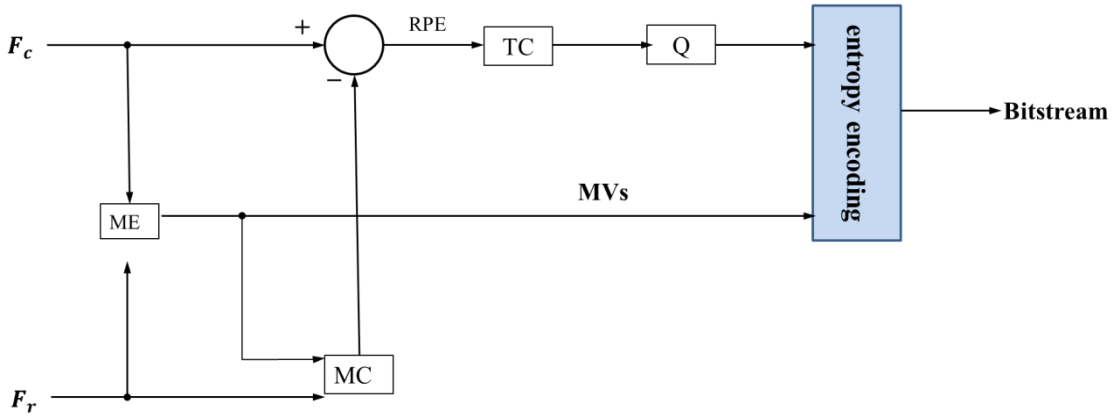


**Figure 1.2: Inter-frame encoder (adapted from [Sayood, 2006; Bovik, 2010])**

At the decoder, the received MVs will be utilised to form an MCP from the reconstructed reference frame, and then the current frame will be reconstructed by adding the reconstructed RPE to the MCP [Bhattacharyya and Deprettere, 2010].

ME technique has the highest complexity of all other stages; it typically accounts for 50-80% of the total video encoder complexity. This technique has been adopted by all existing international video coding standards such as the MPEG series and the H.26x series including its latest H.265 code [ISO/IEC, 1993; ISO/IEC, 1996; ITU-T and ISO/IEC, 2003; Sullivan et al., 2004; Sullivan and Wiegand, 2005; Ohm and Sullivan, 2013]. Therefore, ME is the main challenge for implementing real-time video encoding.

It is possible to estimate the displacement for every one or two pixel positions between successive video frames. However, this is not a practical method since the calculation of these motion vectors is very computationally intensive. Moreover, the number of motion vectors is equal to or half the number of pixels. These vectors will be sent to the decoder in order to form an MCP. As a result, a large amount of data should be transmitted. Therefore, the most practical and widely used method is to use a group of pixels, called a MacroBlock (MBl), to estimate the motion of the current frame. This method is called Block Matching Algorithm (BMA) or Block Matching Motion Estimation (BMME) [Srinivasan and Rao, 1985; Huang et al., 2006; Horn and Schunck, 1981; Richardson, 2010].

BMA is the most popular technique used for motion estimation in which video frames are divided into a set of non-overlapped MBls of size $N \times M$. Each target MBl in the current frame is compared with a number of candidate macroblocks within the search area in the reference frame in order to find the best matching macroblock. The spatial difference between the two matching macroblocks will determine a set of displacement vectors that stipulate the movement of the macroblocks from one location to another in the reference frame [Barjatya, DIP 6620 Spring 2004; Ezhilarasan and Thambidurai, 2008]. There are a number of Block Distortion Measures (BDMs) that can be used to calculate the difference between two macroblocks, namely Mean Absolute Difference (MAD), Sum of Absolute Differences (SAD) and Mean Square Error (MSE) [Sayood, 2006]. If a maximum displacement of $p$ pixels/frame is allowed, then $(2p+1)^2$ locations have to be searched in order to find the best match of the current macroblock. Checking all search area locations is referred to as the Full Search (FS) algorithm. It produces the best possible match and the highest resolution MCP. However, this algorithm suffers from long computational time, which necessitates improvement. Various methods of fast block matching algorithms have been developed to decrease and improve the computational complexity [Nie and Ma, 2002; Huang et al., 2006; Cai et al., 2009].

In this thesis two classifications of fast block matching algorithm were investigated: the first is called the *lossless block matching algorithm* process, in which the computational time required to determine the matching macroblock of the full search is decreased while the resolution of the predicted frames is the same as the full search. The second is

called the *lossy block matching algorithm* process, which reduces the computational complexity effectively but the search result's quality is not the same as that of the full search.

## 1.1 Motivation and Problem Statement

Motion estimation is the main challenge for implementing real-time video encoding since it has the highest complexity of all other stages. It typically accounts for 50-80% of the total video encoder complexity and has been adopted by all existing international video coding standards. It is also the critical part that affects the video quality and compression efficiency. For this reason, many algorithms and models have been proposed to optimise this process [ISO/IEC, 1993; ISO/IEC, 1996; ITU-T and ISO/IEC, 2003; Sullivan et al., 2004; Sullivan and Wiegand, 2005; Ohm and Sullivan, 2013]. With the advancement of video compression standards, the requirements of motion estimation have been increased and thus optimisations must be implemented to cope with the increased complexity. Variable block size and multiple reference frames have been involved in the latest video coding standards, which has led to high computational requirements and as a result motion estimation has become a problem in many video applications, especially for any video coding that requires real-time transmission such as mobile video. This indicates that this is an extremely active field of research.

A number of fast block matching motion estimation algorithms have been developed as a solution to the problem associated with the FS approach, which is the simplest algorithm used for motion estimation to find motion vectors. FS exhaustively searches for the best matching block within the search area, where the correlation window is moved to each possible candidate position within the search area. As a result, a large amount of computational complexity is involved, which means a long time is required for processing. Various algorithms have been proposed and developed to reduce the huge computational complexity. These algorithms can be classified into lossy and lossless categories. Lossy block matching motion estimation can achieve more compression ratio and faster processes than FS by sacrificing the quality of the compressed video. Lossless BMAs have the specific requirement to preserve the quality of the video [Nie and Ma, 2002; Huang et al., 2006; Cai et al., 2009]. Lossy BMAs can be classified into: Fixed Set of Search Patterns, Predictive Search, Hierarchical or

Multiresolution Search, Subsampled Pixels on Matching Error Computation, and Bit-width Reduction; while lossless BMAs include Partial Distortion Elimination (PDE) algorithm and Successive Elimination Algorithm (SEA) [Nie and Ma, 2002; Huang et al., 2006; Cai et al., 2009]. The performance of each fast block matching algorithm is evaluated and compared against the FS algorithm. Their performance is measured by the reduction in the RPE and the computational requirement.

## 1.2 Research Objective and Contributions

The objective of this thesis is to design, implement and optimise fast block matching motion estimation. The major focus of this research study is to investigate the possibility of developing novel techniques for both the lossless and lossy block matching algorithms' process for the purpose of managing both the time needed to process the block matching algorithm and the resolution of predicted frame. The contributions of the thesis can be summarised by:

1. In lossy block matching algorithms, Mean Predictive Block Matching (MPBM) [Ahmed et al., 4-6 July 2011] and Enhanced Mean Predictive Block Matching Algorithm (EMPBM) [Ahmed et al., 2012] have been proposed to decrease the time needed for processing and improve the resolution of the predicted frame in comparison to the well-known standard fast block matching algorithm.

2. In lossless block matching algorithms, Fast Computations of Full Search (FCsFS) is proposed to reduce the search time of the macroblock matching, while keeping the resolution of the predicted frames close to the one predicted by full search.

3. All the proposed algorithms use the fact that the general motion in any video frame is usually coherent; therefore the motion of previous above and left MBls could be a good a prediction for the search process of the current macroblock's motion.

4. Moreover, the PDE algorithm has been used to stop the partial sum of matching distortion between current macroblock and candidate macroblock. Good prediction leads to detecting matching MBl in the early steps; therefore applying the PDE algorithm will speed up the search process.

5. A simple edge detection technique was proposed to classify the current MBl of size 4×4 into *shade* and *edge.* The shade macroblock has a probability to move in the same direction as its neighbouring macroblocks, hence decrease the number of search points required to find the matching MB1.

6. All the proposed techniques were benchmarked with well-known standard algorithms for the purpose of evaluation. The experimental results of all proposed techniques were conducted on a luminance component for 50 frames of six popular video sequences with various motion activities of low, medium and large.

7. The simulation's results indicated that motion activity of video sequences affected the proposed algorithms FCsFS and MPBM in that where video sequences have low motion activity these algorithms are more effective; that is, the improvement has shown clearly in comparison to the benchmarked algorithm.

8. Finally, the simulations of the Enhanced Mean Predictive Block Matching algorithm indicated that using edge detection could improve the computational complexity when compared with MPBM; while keeping or enhancing the resolution of compensated frames built by EMPBM is close to the one built by MPBM. Unlike other proposed algorithms, motion activity of video sequences does not affect the computational complexity of EMPBM and the resolution of the compensated frames built by it due to its similarity with MPBM.

## 1.3 Thesis Structure

The remaining part of this thesis is structured into the following chapters:

Chapter 2 introduces some basic concepts of digital video compression such as redundant information, lossy and lossless compression, and digital video frame types.

Chapter 3 considers the fundamentals of the inter-compression system in which the main focus is on ME, motion compensation, and block matching motion estimation.

Chapter 4 surveys fast block matching motion estimation algorithms and architectures. It describes various techniques of lossy and lossless block matching algorithms.

Chapter 5 provides an overview of the designed fast block matching architectures. It introduces a novel method in lossless block matching algorithms which is called Fast Computations of Full Search Block Matching Motion Estimation (FCsFS) and two novel techniques of lossy block matching algorithms called Mean Predictive Block Matching algorithm (MPBM) and Enhanced Mean Predictive Block Matching algorithm (EMPBM).

Chapter 6 presents the analysis and the simulation results for the novel algorithms as well as the benchmarked techniques.

Chapter 7 provides the conclusion for the work outlined in this thesis as well as suggestions for future works.

# CHAPTER 2: INTRODUCTION TO VIDEO COMPRESSION

In this chapter, a brief description of some concepts of video compression is introduced, with some of the methods and techniques used for such process. This chapter is divided into two sections. Section one introduces some basic definitions of analogue and digital videos with colour representation and the different types of standard digital videos. Section two presents the fundamentals of video coding. It starts by providing the chronological development of video coding standards, and then gives the outline of video compression, introducing concepts such as that of redundant information, Lossless and Lossy compression, Quality Measure in video coding and, finally, digital video frame types. The chapter is summarised in section three.

## 2.1 Analogue and Digital Videos

Video, in common terms, is a time sequence of still images (frames) that is a spatial distribution of intensity, as shown in Figure 2.1. Also, video may be defined as a three-dimensional (3D) function, $f(x, y, t)$, where the pair $(x, y)$ denotes the spatial (plan) coordinate and *t* denotes time. The amplitude of $f$ at $(x, y, t)$ is called the intensity of the image in time $(t)$ at the location $(x, y)$. When the video $f(x, y, t)$ is continuous in both $(x, y)$ and $(t)$, the video is called analogue video. Analogue video signal refers to a one-dimensional (1D) electrical signal obtained by sampling $f(x, y, t)$ along the vertical $(y)$ coordinate and along the time $(t)$ direction and converting intensity to electrical representation [Bovik, 2009]. This sampling process is known as scanning and the result is a series of time samples, which are complete frames or pictures. The most commonly used scanning methods are progressive and interlaced, as shown in Figure 2.2. In progressive scanning, a frame is formed by a single scanning pass. In interlaced scanning, a frame is formed by two successive scanning passes. In the first pass, the odd lines are scanned to form the first field, and then the even lines are scanned to form the second field. The lines of the two fields form a single frame [Bovik, 2010; Gonzalez et al., 2009].

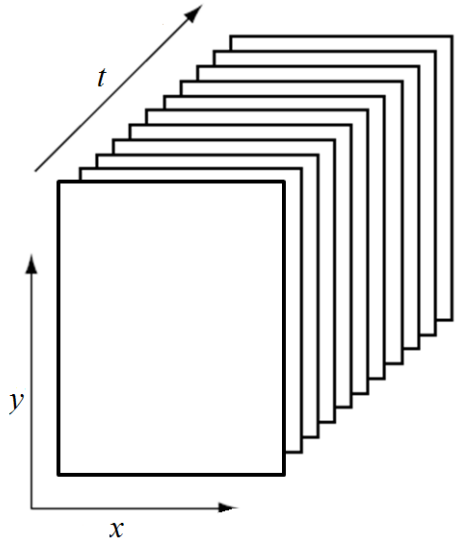


**Figure 2.1: Video sequence**

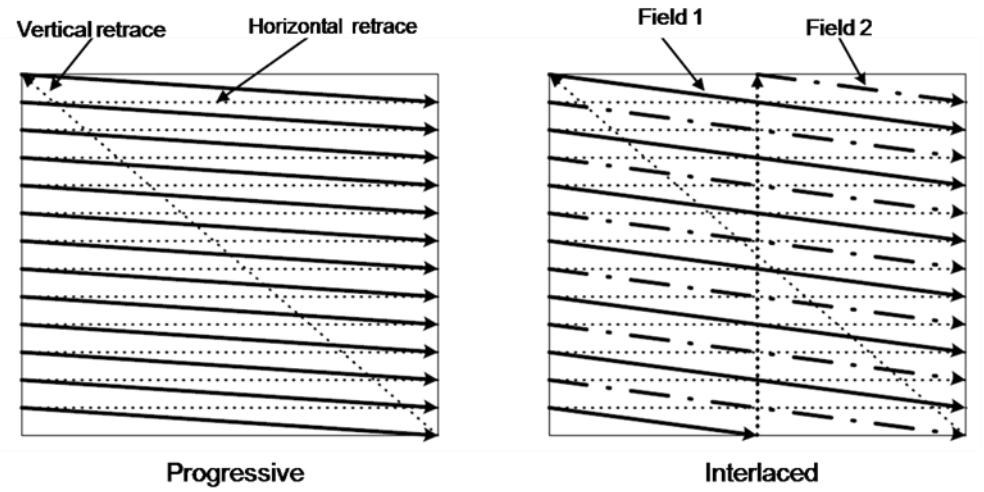


**Figure 2.2: Video scanning [Al-Mualla et al., 2002]**

The important parameters of the video signal are: the *aspect ratio*, *vertical resolution*, *frame rate* and *refresh rate*. The *aspect ratio* is the ratio of the *width* to the *height* of a frame. The *vertical resolution* is related to the number of scan lines per frame. The *frame rate* is the number of frames scanned per second measured by Frames Per Second (FPS). Smooth motion can be achieved using a frame rate of about *25–30 FPS*, but the human eye picks up the flicker produced by refreshing the display between frames. To prevent that, the display *refresh rate* must be above *50 FPS* [Al-Mualla et al., 2002; Bovik, 2010]. However, in many systems, like television, such fast refresh rates are not possible because of bandwidth limitations, unless spatial resolution is severely

compromised. Interlaced scanning is a solution for this problem; for example, to reduce bandwidth requirements, the television industry uses interlaced scanning. In this case, the field rate is set to *50* or *60* fields per second (fields/s) to avoid refresh flicker, while the frame rate is set to *25 or 30 FPS* to maintain smooth motion.

There are three main analogue video systems. In most of Western Europe, the Phase Alternation Line (PAL) system is used, which is *625/50* (625 scan lines and 50 fields/s). In Russia, France, the Middle East and Eastern Europe, a 625/50 SEquential Couleur Avec Memoire (SECAM) system is used. In North America and Japan, a *525/60* National Television System Committee (NTSC) system is used. All three systems are interlaced with a *4:3* aspect ratio [Al-Mualla et al., 2002; Sayood, 2006; Bovik, 2010].

Digital video is obtained by digitising the analogue video signal or the 3D space–time intensity distribution. Digitising involves two distinct subprocesses: digitising the coordinate values, which is called sampling, and digitising the amplitude value, which is called quantisation. Sampling a video signal at a specific time generates a sampled frame or image. The most common format for a sampled image is a rectangle with the sampling points positioned on a square or rectangular grid. Figure 2.3 shows the sampling of progressive analogue video. If interlaced analogue video is sampled, then the digital video is also interlaced as shown in Figure 2.4 [Richardson, 2003].

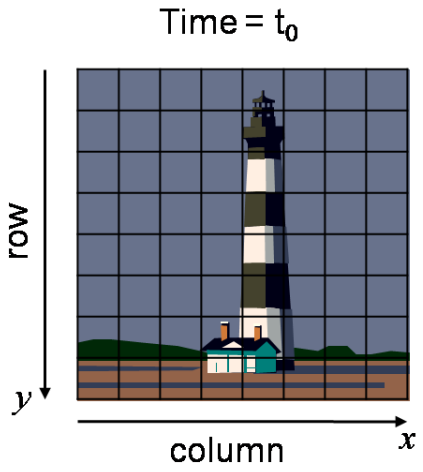


**Figure 2.3: A single frame from a sampled progressive video sequence**

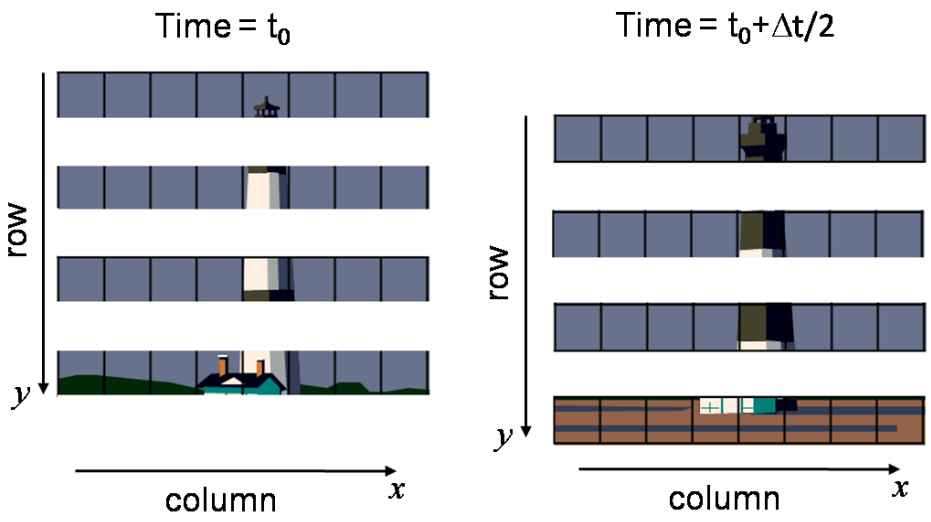


**Figure 2.4: A single frame of two fields from a sampled interlaced video sequence**

The discrete samples have continuous amplitudes. Quantisation is used to assign a finite set of discrete amplitudes to the amplitude values at each sampling instant. This finite set can be represented by a finite number of bits. A discrete location with the discrete amplitude is called an image element or pixel. This means that the pixels are arranged in a two-dimensional (2D) array to form a digital image. To eliminate errors caused by quantisation, the number of pixels should be increased. Moreover, the visual quality of the image is influenced by the number of pixels. The resolution of the frame (number of image pixels) can be calculated by multiplying the number of horizontal pixels and vertical pixels. In a monochrome image the intensity of each pixel is called the grey level and requires just one number to indicate the brightness or luminance. Colour images require at least three numbers per pixel position to represent colour accurately [Richardson, 2003]. The method chosen to represent brightness and colour is described as a colour space, as shown in the next section.

### 2.1.1 Colour Space

There are three basic colours: red, green and blue (RGB) to describe colour digital video. Colour space of digital video determines how to describe these basic colours mathematically. The suitability of the colour space is dependent on its usage. For example, *RGB* colour space is suitable for video capture and display, while *YCbCr (YUV)* colour space is more suitable for storage and transmission [Waggoner, 2002; Richardson, 2003; Kim, 2010; Sayood, 2006; Al-Mualla et al., 2002].

In *RGB* space, colour image sampling is represented by the three additive primary colours: red (R), green (G) and blue (B). Any colour can be created by combining red, green and blue in varying proportions. Each of these three colours is highly correlated with the other two, which means that the *luminance* (brightness) cannot be separated from *chrominance* (which is related to the perception of colour information). RGB space can only be poorly compressed and is not suitable for storage. This colour space is always used in computer graphics, and all digital video starts and ends as RGB, even if it is never stored as that.

*YCbCr* colour space of digital systems or *YUV (YIQ)* of analogue systems separates the brightness Y (luminance component) from the colours Cb (U) and Cr (V) (chrominance components). There are three main methods to calculate the luminance and chrominance components from the *RGB* components created by the three main analogue video systems, *PAL*, *SECAM* and *NTSC*. For example, the *PAL* system calculates the luminance and chrominance components as follows: luminance component *Y* is calculated as a weighted average of *R*, *G* and *B*:

$$Y = 0.299R + 0.587G + 0.114B \tag{2.1}$$

While the chrominance components *U* and *V* can be obtained from:

$$U = 0.493(B - Y)$$

$$V = 0.877\,(R - Y) \tag{2.1}$$

The *NTSC* and *SECAM* systems calculate luminance in the same way but use different coefficients for obtaining the chrominance components.

It should be noted that *U* and *V* may be negative in the *YUV* colour space and cannot be directly used in a digital system. In order to make chrominance components nonnegative, the *Y*, *U* and *V* are scaled and shifted to produce the YCbCr model. The *YCbCr* colour space is widely used in digital systems and converts from *RGB* space as follows:

$$Y = 219\,(+0.299R + 0.587G + 0.114B) + 16$$

$$Cb = 224\,(-0.169R - 0.331G + 0.500B) + 128 \quad (2.2)$$

$$Cr = 224\,(+0.500R - 0.419G - 0.081B) + 128$$

The human visual system (*HVS*) is more sensitive to luminance than to chrominance. Thus, the resolution or bits required for representing chrominance Cb and Cr can be reduced by colour subsampling to achieve compression while keeping acceptable quality, which is described in the next section. It should be noted that, before displaying the image, it is usually necessary to convert it back to *RGB*.

### 2.1.2 Colour Subsampling

There are three common subsampling patterns for *Y, Cb* and Cr, as shown in Figure 2.5 [Richardson, 2003; Pu, 2005]:

1. *4:4:4 YCbCr*: this is a format with no subsampling of *Y, Cb* and *Cr* components, in which the three components (*Y, Cb* and *Cr*) have the same resolution and hence a sample of each component exists at every pixel position. This means that for every four luminance samples there are four *Cb* and four *Cr* samples. 4:4:4 sampling preserves the full fidelity of the chrominance components.

2. *4:2:2 YCbCr*: this format uses 2:1 horizontal down-sampling. This means that for every four luminance samples in the horizontal direction there are two Cb and two Cr samples. Therefore, the total storage required for Cb and Cr is reduced by 50%.

3. *4:2:0 YCbCr*: due to its compression ratio this subsampling format is widely used in vide or image compression application. This format uses 2:1 horizontal down-sampling and 2:1 vertical down-sampling. This means that for every four luminance samples there is one Cb and one Cr sample. Therefore, the total storage space required for *Cb* and *Cr* is only 25% compared with the 4:4:4 *YCbCr* format. This yields a 2:1 reduction in data before further compression.

This research work has only used the luminance information approach, i.e. pixels are assumed to contain the *Y* component of the *YCbCr* colour space.

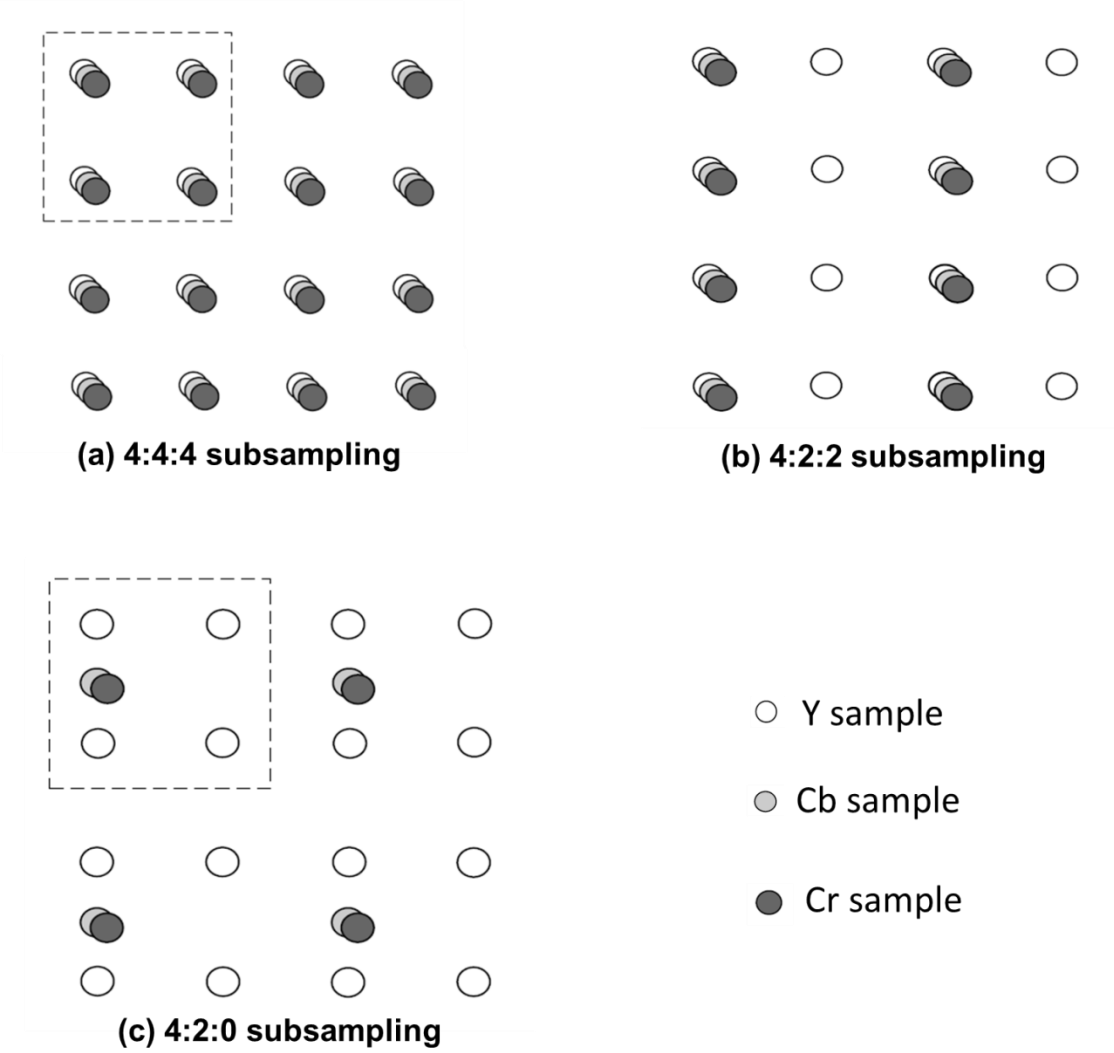


**Figure 2.5: Colour subsampling [Richardson, 2003]**

### 2.1.3 Video Format

Exchange of digital video between different industries, applications and networks requires standard digital video formats. The most common digital frame format used in standard video compression is Common Intermediate Format (CIF), which is the basis for a family of formats. In this family the colour subsampling is 4:2:0, and each pixel is usually represented by 8 bits, and a rate of 30 FPS. The luminance component of the

CIF format is represented by *352 pixels×288 lines* and the two chrominance components have half the luminance resolution in both the horizontal and vertical planes. Quarter-CIF (QCIF) has a luminance component of *176×144* pixels, whereas Sub-QCIF (SQCIF) has a luminance component of *128×96* pixels and *4CIF* has *704×576* pixels. The choice of frame resolution depends on the application and available storage or transmission capacity. For example, 4CIF is appropriate for standard definition television and DVD-video; CIF is popular for videoconferencing applications; QCIF or SQCIF are appropriate for mobile multimedia applications, where the display resolution and the bitrate are limited.

The other common formats are Source Input Format (SIF) and Quarter-SIF (QSIF), which are used for storage applications. These formats define different vertical resolution values for NTSC and PAL, while CIF and its family support the NTSC and PAL video formats using the same parameters. SIF resolution is *352×288* pixels with a frame rate of 25 frames/s for PAL, but *352×240* pixels with a frame rate of 30 FPS for NTSC. For both cases the resolution of the chrominance components is half of the luminance resolution in both the horizontal and vertical planes. QSIF has half the dimensions of SIF in both directions [Richardson, 2003; Bovik, 2010; Sayood, 2006].

## 2.2 Fundamentals of Video Compression

Video compression, or what may be known as video coding, has become an essential part of multimedia systems. A huge amount of information is needed in order to display a digital video, therefore a large memory space will be required to store digital video images and it will need an equally large bandwidth for transmission. Video compression is the process of reducing the amount of data required to represent digital video images while preserving an acceptable video quality. This technique provides efficient solutions to representing video data in a more compact and robust way so that the information can be stored or transmitted faster in videoconferencing and videophone, digital broadcasting, interactive games (internet), etc. The balance between video quality (dependent upon frame size, frame rate and bit depth) and file size should be considered.

This section gives a short overview of the fundamentals of video compression. The chronological development of video coding standards will also be introduced.

### 2.2.1 Video Coding International Standard

The existing standard of video compression techniques were developed by two public international organisations: the International Telecommunication Union–Telecommunication Standardization Sector (ITU-T), known as the Visual Coding Experts Group (VCEG), and the International Organization for Standardization/International Electrotechnical Commission (ISO/IEC), known as the Moving Picture Experts Group (MPEG). The standards approved by the ISO/IEC are called the MPEG family, whose applications range from consumer video on CD-ROM (MPEG-1 1991) to broadcast/storage standard or high definition TV (MPEG-2 1994) and object-based representation (MPEG-4 Visual or part 2 1998). On the other hand, the *H.26x* series of video standards published by the ITU-T focuses on improving the coding efficiency for bandwidth-restricted telecommunication applications as the number of video services increases. The ITU-T published its first video coding standard *H.261* in 1990, and in 1995, it evolved *H.263* video coding standards (and later enhancements of H.263 known as *H.263+* and *H.263++*) with higher compression ratios [ISO/IEC, 1993; ISO/IEC, 1996; Li Liu et al., 2010 ]. The various applications for transmitting videos over the network have created great demand for efficient video coding. VCEG and MPEG formed the Joint Video Team (JVT) in December 2001 to complete the draft of the video coding standard as *H.264/AVC (MPEG-4 Part 10)* in May 2003. The video coding standard *H.264/AVC* is reported to achieve gains in compression efficiency of up to 50% compared with its predecessor MPEG-2. However, the increasing popularity of high definition TV, video delivery on mobile devices and other multimedia applications create new demands for video coding standards. In January 2010, the Joint Collaborative Team on Video Coding (JCT-VC) was created as a group composed of VCEG and MPEG to develop a new-generation video coding international standard. In February 2012, JCT-VC introduced the committee draft video compression standard called High Efficiency Video Coding (HEVC), which is also known as H.265 and MPEG-H Part 2. The final draft international standard appeared in January 2013 [Ohm and Sullivan, 2013]. HEVC code (without reduction in visual quality) has improved the video compression ratio by at least 50%, compared with H.264, across various applications such as videoconferencing, digital storage media, television broadcasting, internet streaming and communication [Wiegand et al., 2003; Li Liu et al., 2010 ; Nightingale et al., 2012; Bross et al., 2012].

### 2.2.2 Redundant Information

For all the standard video compression techniques, video coding can be obtained by taking advantage of the redundant information in any video [Kim, 2010; ITU-T and ISO/IEC, 2003; Al-Mualla et al., 2002; Metkar and Talbar, 2010; ISO/IEC, 1993; Sayood, 2006; Chanyul, 2010].

**Colour Space Redundancy**

As mentioned in section 2.1.1, the Human Visual System (HVS) is more sensitive to luminance components than to chrominance components. Therefore, colour subsampling can reduce the resolution required to represent chrominance components. The first of several steps in compression is to transfer the information in the picture into the frequency domain. That is, the *RGB* intensity information in each pixel is transferred into luminance/brightness values as well as chrominance components in the YCbCr colour subsampling to achieve compression.

**Spatial Redundancy**

This redundancy comes from the spatial correlation in an image, where a block of an image can be predicted from its neighbouring pixels, which is called intra-frame compression, as shown in Figure 2.6. There are several spatial compression algorithms that are proposed for this purpose; the most common uses are predictive coding, transform coding such as Discrete Cosine Transform (DCT), quantisation and entropy coding.

**Temporal Redundancy**

In this case, the adjacent frames are highly correlated; that is, most of the time, the image frame looks similar to the frame before it, as shown in Figure 2.6. This redundant information can be removed using *inter-frame* compression.

There are several inter-frame compression methods of varying degrees of complexity, such as subsampling coding, difference coding, block-based difference coding and motion compensation [ISO/IEC, 1993; Metkar and Talbar, 2010]. This thesis deals with temporal redundancy and attempts to enhance the complexity computations that come from inter-frame compression, as shown in Chapter 3.


**Figure 2.6: Spatial and temporal correlation in video sequence [Richardson, 2003]**

### Statistical Redundancy

For any data, there is a minimum number of bits required to represent it without losing any information. Bit redundancy could be removed to further compress intra-frame and inter-frame compression. This can be performed by entropy coding such as Run Length Coding (RLC), Huffman Coding and Arithmetic Coding [Gonzalez et al., 2009].

## 2.2.3 Lossless and Lossy Compression

In general, video coding contains two systems: video encoders and video decoders, as shown in Figure 1.1. A video encoder consists of three main functional units: colour subsampling, a temporal model (inter-frame encoder) or a spatial model (intra-frame encoder) and an entropy encoder. The target of the encoder is to condense the huge amount of information needed to display a video frame in order to achieve a high compression ratio using the following equation:

$$Compression\ Ratio = \frac{Original\ video\ size\ (bytes)}{Compressed\ video\ size\ (bytes)} \tag{2.3}$$

The balance between decoded video quality and file size should be considered. The encoder can be classified into two approaches: lossless and lossy approaches. The

lossless technique (which is also known as bitpreserving or the reversible method) is used to compress the statistical redundancy. This method has a low compression ratio of about *3:1* or *4:1* in the best case, but the reconstructed data is identical to the original data. On the other hand, the lossy technique usually achieves a high compression ratio from *50:1* to *200:1* and even above, but the reconstructed data is not identical to the original data; that is, there is loss of information [Richardson, 2003; Vanne, 2011].

### 2.2.4 Quality Measure in Video Coding

In video compression, the lossy approach is the main method used to achieve a high compression ratio; however, this approach leads to lost information (it is called distortion) after reconstruction of the compressed video. In order to assess the quality of the reconstructed video, several methods have been developed. One of the simplest and most popular methods is to use Mean Square Error (MSE) for each frame separately and take their arithmetic mean. *MSE* is the average of the squared error measure determined according to the following equation:

$$MSE(f,\hat{f}) = \frac{1}{M \times N}\sum_{i=1}^{M}\sum_{j=1}^{N}\left(f(i,j) - \hat{f}(i,j)\right)^2 \tag{2.4}$$

Where $M$ and $N$ are the horizontal and vertical dimensions of the frame, respectively, and $f(i,j)$ and $\hat{f}(i,j)$ are the pixel values at location $(i,j)$ of the original and reconstructed frames, respectively.

A more common form of the *MSE* measure is the Peak Signal-to-Noise Ratio (PSNR), which is defined as:

$$PSNR(f,\hat{f}) = 10\ log_{10}\left(\frac{(f_{max})^2}{MSE}\right) \tag{2.5}$$

Where $f_{max}$ is the maximum possible pixel value (for example, 255 for an 8-bit resolution component). The unit measure of PSNR is decibels (dB). Equation (2.5) shows that the PSNR measures the strength of the signal relative to the strength of the error. In application, PSNR between the original and reconstructed video sequences is measured by computing the PSNR for each frame separately and taking their arithmetic

mean. A high PSNR usually indicates high quality and low PSNR usually indicates low quality. However, PSNR is an *objective* measure, which means that a particular value of PSNR does not necessarily equate to a *subjective* video quality perceived by the HVS. The easy and quick calculation of PSNR makes it a very popular quality measure and it is widely used to compare the quality of the decompressed and the original videos [Al-Mualla et al., 2002; Sayood, 2006]. Thus, to facilitate comparisons with algorithms reported by others, this research work adopts the PSNR measure.

### 2.2.5 Types of Frames

Video frames are compressed using different algorithms depending on the frame type. Figure 2.7 shows the three major frame types used in different video coding algorithms, which consist of I-frame, P-frame and B-frame. It should be noted that all information provided in this section is taken from the following reference [Bhaskaran and Konstantinides, 1997; Moeritz and Diepold, 2004; Richardson, 2010].

**I-frame 'Intra-coded frame':** this type of frame is coded independently from all other frames. This frame is compressed as a still image using a still image compression technique such as transform coding, vector quantisation or entropy coding. This type of frame is the largest size in encoding but is faster to decompress than the other frames.

**P-frame 'Predicted frame':** an inter-coded frame, which is forward predicted from the last I-frame or P-frame, i.e. it is impossible to reconstruct it without the data of the previous frame (I or P). P-frames are typically a smaller size in encoding than I-frames.

**B-frame 'Bi-predictive frame':** an inter-coded frame, which is a bi-directionally predicted frame, coded based on both the previous and next I- or P- frames, but a B-frame cannot be the reference for other B-frames, i.e. there are two other frames necessary to reconstruct them. So B-frames are an effective video coding tool to improve coding efficiency. However, using B-frames for coding requires more memory in the encoder and decoder, as an extra frame (next reference) needs to be stored during the decoding process. Furthermore, B-frames introduce extra delay (next reference send first), which is unacceptable in two-way video coding such as for a videoconferencing application; in this case, no B-frames are used [Sayood, 2006].

### 2.2.6 Group of Pictures

Frames between two successive I-frames, including the leading I-frame, are collectively called a Group Of Pictures (GOP), which is the smallest random access unit in the video sequence, as shown in Figure 2.7. A GOP pattern is defined by the ratio of P- to B-frames within a GOP. Common frame patterns used for DVD are IBP and IBBP. All three frame types do not have to be used in a pattern. For example, an IP pattern can be used in two ways for video coding, as mentioned previously. Longer GOP lengths (the term long GOP refers to the fact that there are several P- and B-frames used between I-frame intervals) encode video very efficiently by giving a good compression ratio. Smaller GOP patterns with shorter GOP lengths work better with video that has quick movements, but they do not compress the data as much. For television systems, an I-frame is sent typically every half second in order to enable channel surfing [Moeritz and Diepold, 2004].

An I-frame is often used to efficiently code frames corresponding to scene changes, i.e. frames that are different from previous frames and cannot be easily predicted. Since video sequences have variable scene durations, depending on the content, it is not possible to use a fixed GOP structure to efficiently code the video sequence. This is because the position of I-frames in the sequence depends on the time that scene changes happen. For example, video coding standards allow for macroblocks which are $16\times16$ pixels in P- and B-frames to be intra-coded if they cannot be predicted efficiently. This means that, even if all the frames are set to be of types *P* or *B*, there may be many macroblocks in each frame that are intra-coded [Turaga and Chen, 2001; Huang, 2005].

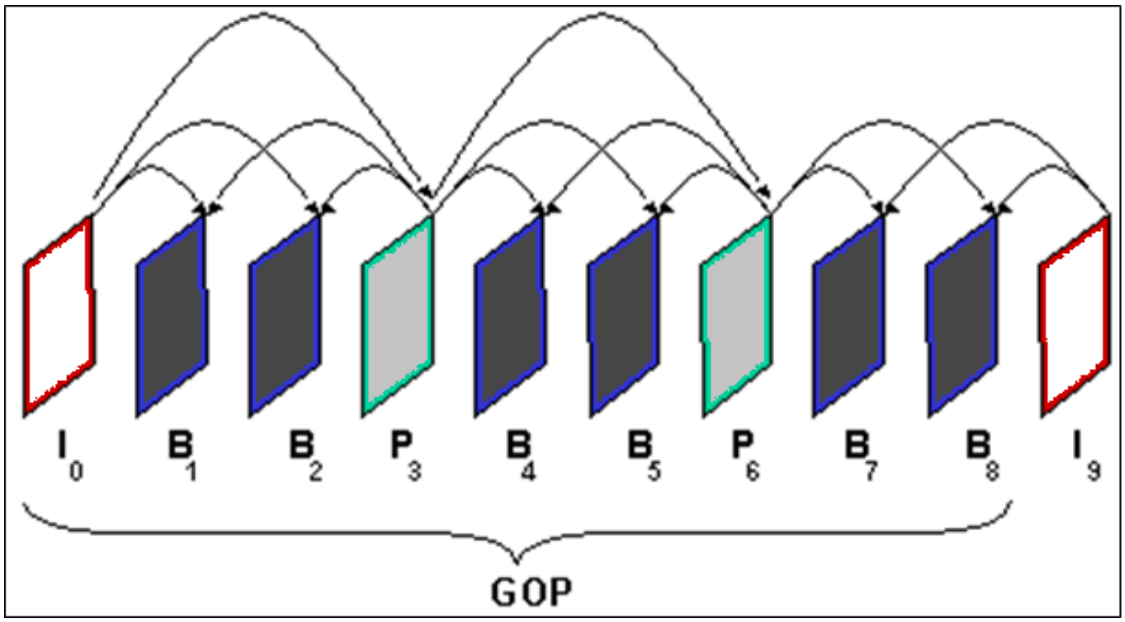


**Figure 2.7: Types of coded frames ( adapted from [Huang, 2005])**

Coding as *P*- and *B*-frames gives a higher compression rate, but it is more computationally expensive than coding an *I*-frame. This relates to the fact that coding *P*- and *B*-frames uses motion estimation and motion compensation, which will be discussed in the next chapter.

## 2.3 Chapter Summary

Digital video consists of a series of orthogonal bitmap digital images displayed in rapid succession at a constant rate. Video compression is the process of reducing the amount of data required to represent digital video images while preserving an acceptable video quality. There are four types of redundant information in any video, which are: colour space redundancy, spatial redundancy, temporal redundancy and statistical redundancy. The video compression system contains two systems: video encoders and video decoders. A video encoder consists of three main functional units to remove redundant information: colour subsampling, a temporal model (inter-frame encoder) or a spatial model (intra-frame encoder), and an entropy encoder.

Efficient video compression can be achieved by an inter-frame encoder in which the current frame can be locally modelled as a translation of the reference frames. That is, most frames will depend on the others to reduce the temporal redundancy and reduce

the transmission rate of the sequence of the video images in order to obtain high compression. Therefore inter-frame encoding is the important part in video compression. For this reason, improving video compression is an active research area, and is investigated in this research. More details about inter-frame compression will be provided in the next chapter.

# CHAPTER 3: MOTION COMPENSATION AND MOTION ESTIMATION

As seen in Chapter 2, the efficiency of the video compression process is achieved by reducing or eliminating temporal redundancy, which is called inter-frame compression. This chapter concentrates on the inter-compression system, motion compensation and block motion estimation.

## 3.1 Inter-Frame Compression

Inter-frame compression exploits the high correlation that exists between successive frames in video sequences, especially if the frame rate is high. This correlation leads to temporal redundancy. The goal of inter-frame coding is to reduce this redundancy. Video coding standards share a number of common features, as shown in Figure 1.2. Each standard assumes that after colour subsampling there will be four stages of inter-frame encoding to produce the compressed bitstream: temporal prediction between current frame and reference frame, transform coding (TC), quantisation (Q) and entropy coding.

### 3.1.1 Temporal Prediction

The goal of temporal prediction is to reduce temporal redundancy coming from high correlation between successive frames. This can be done by predicting some frames from others to reduce the transmission rate of video image sequences and obtain further compression. Reference frames of type I or P could be used to predict frames of type P or B. In *forward prediction,* past frames in the display order have been used as reference frames to the current frame; while, in *backward prediction*, the reference frames of the current frame are displayed in the display order in the future frames. The average of the forward and backward predictions may be used to predict frames of type B. In any prediction, reference frames have to be encoded first, while a residual (difference) between current and reference frames which contain less energy will be encoded later instead of the encoded current frame [Richardson, 2003].

To decrease this residual, prediction was improved by estimating the motion of the moving objects in-between the current and the reference frames, which is called Motion Estimation (ME) technique. That is, the motion estimation has been used to calculate the Motion Vectors (MVs) by comparing the current frame and the reference frame. The technique that uses the MVs to predict a new frame from a reference frame is called Motion Compensation (MC). The predicted frame is known as a Motion Compensated Prediction (MCP). The first output of this process will be the difference between the current frame and the MCP, which is called the residual prediction error (RPE) (or may be known as displaced frame difference (DFD)); the second output will be the motion vectors.

Motion vectors are encoded by lossless compression, while RPE is encoded by lossy compression to get high compression ratio [Sullivan et al., 2004; Leontaris et al., 2009; Richardson, 2010; Sayood, 2006; Marpe et al., 2006; Al-Mualla et al., 2002]. This thesis focuses on this stage and the details will be introduced in sections 3.2 and 3.3

### 3.1.2 Transform Coding (TC)

Transform coding is one of the most important tools, which is employed to reduce spatial redundancy. The RPE, which is the difference between the current frame and the MCP frame, has a high correlation between neighbouring pixels, as shown in Figure 3.1. Inter-frame compression can be coded more efficiently by exploiting these similarities and reducing the spatial redundancy. Transform coding converts the data from a spatial domain of the RPE into a transform domain to produce a set of coefficients. The energy of the transformed data (coefficients) is localised and compacted at some certain areas. The transform should be reversible and transform as much information as possible into a small number of transform coefficients. Over the years, a variety of linear transform methods have been developed. The most popular transforms can be classified into two types: block-based transform coding and image-based transform coding [Richardson, 2010; Jizheng et al., 2009].

Block-based coding is widely used in image/video coding standards systems. In block-based transforms, an image is divided into non-overlapping macroblocks and for each macroblock the 2-D transform coding is applied. Most transform coding systems employ a macroblock size of 8×8 or 16×16. Note that both sizes are powers of 2, which

reduce the computation complexity of the transform coding and requires low memory. The block-based transform coding converts the macroblock pixel information into the frequency domain where pixel correlation information is captured in a DC coefficient and pixel difference information is captured in AC coefficients. The AC coefficients normally have very small values because of the high correlation between the pixels in a macroblock. Therefore, the energy is concentrated in the DC coefficients and a small number of AC coefficients that are close to the DC coefficient. That is, the macroblock energy is usually concentrated in the low frequency region. Furthermore, block-based transform allows each macroblock to be processed in a different way according to its content in order to improve the coding performance significantly, as performed in H.264. The disadvantage of such block-based transform is that the transform can only exploit the correlations within the macroblock and hence this technique suffers from artefacts at edge macroblocks using very low bit rates, which affects the coding efficiency. Popular block-based transforms include: Discrete Cosine Transform (DCT), Karhunen–Loeve Transform (KLT), and Singular Value Decomposition (SVD) [Richardson, 2010; Bovik, 2010; Jizheng et al., 2009; Prasantha et al., 2007].

Image-based transform resolves the problem of artefacts initiated at edge macroblocks by using Discrete Wavelet Transform (DWT) on the entire image or video frame. An image-based transform would provide better energy compaction, but it tends to suffer from higher computational complexity and memory requirements in comparison to block-based transform because the whole image is processed as a unit. Therefore, the block-based transform is better compatible with the residual prediction error [Jizheng et al., 2009; Vanne, 2011; Richardson, 2010; Bovik, 2010].

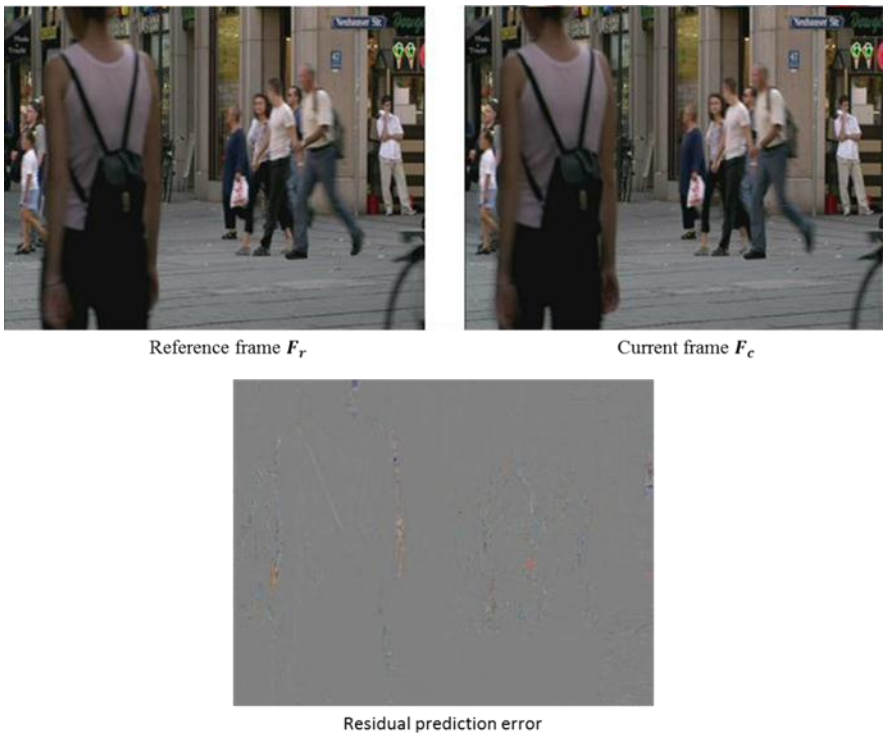


**Figure 3.1: The similarity between neighbouring pixels of the residual prediction error [Kim, 2010]**

### 3.1.3 Quantisation (Q)

Quantisation is a mapping of a large set of possible inputs into a smaller set of possible outputs. Quantisation forms the heart of lossy compression and it is an irreversible process. The goal of this scheme is to map the data from a source into as few bits as possible such that the reconstructed data from these bits is as close to the original one as possible. There are two types of quantisation, scalar and vector. Scalar quantisation maps a single value of the input signal to one quantised output value (level). A simple example of uniform scalar quantisation is the process of rounding a fractional number to the nearest integer. The reconstructed values are usually the midpoint of the two adjacent step values. The length of the interval of the output levels is called step size. A scalar quantiser of the same step size is called a uniform quantiser, while a quantiser of different step size is called a non-uniform quantiser. If the step size is large (coarse), fewer numbers of bits are required and hence high compression ratio is achieved while the quality of the reconstructed data is reduced. However, small step size gives a larger range of quantised values and hence reduces compression efficiency and improves the reconstructed data. In each video coding standard, there exists a defined set of

quantisation step size parameters that provide the best balance between decoded video quality and compression ratio for different applications.

Vector quantisation maps a group of input values (vector) (such as a block of image samples) to a group of quantised values which is the index from a "codebook". Vector quantisation can be used alone as a method of compression and is very powerful with high computational complexity.

Scalar quantisation techniques are involved in most video coding standards with the combination of transform coding. After the transformation, the energy in both the pixel and the transform domains are equal but the transform coefficients are less correlated than the original data. In the transform domain the majority of energy is concentrated on the low frequencies while little energy is concentrated on the high frequencies. Since the human eyes are more sensitive to low frequencies compared to high frequencies, therefore greater compression can be achieved by apply coarser quantisation step size at higher frequencies to remove insignificant coefficient values [Kou, 1995; Pu, 2005; Pereira and Ebrahimi, 2002; Yu and Peng Wang, 2010; Marpe et al., 2006; Sayood, 2006; Richardson, 2010].

### 3.1.4 Entropy coding (EC)

Entropy coding is the last stage in a video encoding system. It is a lossless compression scheme used to remove statistical redundancy by determining the minimum number of bits required to represent the data without losing any information. EC converts the MVs, the quantised transform coefficients and other information from the intra-compression process into a compressed bitstream suitable for transmission or storage. The widely used entropy coding are Variable Length Coding (VLC) and Arithmetic Coding. Arithmetic coding usually provides better compression efficiency, with relatively high computational complexity. These codes are improved by Context-Adaptive VLC (CAVLC) and Universal VLC (UVLC), which are based on VLC, while Context-Adaptive Binary Arithmetic Coding (CABAC) is based on arithmetic coding. CABAC provides bit-rate savings of 9-14% compared to CAVLC but this is at the cost of higher complexity. The low complexity CAVLC entropy encoding method is utilised by the H.264 standard [Wiegand et al., 2003; Richardson, 2010; Yu and Peng Wang, 2010].

### 3.1.5 Decoding of Inter-frame compression

The decoder interprets the compressed data stream of the compressed motion vectors and compressed RPE; the process is reversed to reconstruct the original frame.

In the decoder side (Figure 3.2), the reference frame was already reconstructed $\hat{F}_r$ by intra-frame decoding and is ready to compensate and predict the current frame. The MC uses the decompressed MVs from entropy decoding to predict MCP of the current frame. On the other hand, to produce decoding of residual prediction error which is denoted by $\widehat{RPE}$ in Figure 3.2, start by entropy decoding followed by inverse quantisation ($Q^{-1}$), then inverse transform coding $TC^{-1}$. Note that the irreversible quantisation process means that $\widehat{RPE}$ is not identical to RPE. Finally, $\widehat{RPE}$ is added to the predicted frame to introduce the reconstructed current frame $\hat{F}_c$.

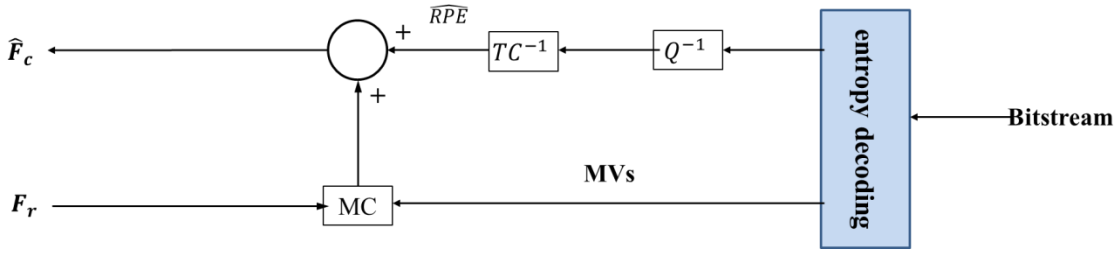


**Figure 3.2: Inter frame decoder ( adapted from [Sayood, 2006; Bovik, 2010])**

## 3.2 Motion Compensation (MC)

Motion compensation (MC) has been used as a main tool to reduce the temporal redundancy that comes from the small change in the contents from one image to another in video sequences. That is, MC is the key to achieve high compression ratio for the coding system. This technique dates back to the early 1970s and has been adopted by all of the existing international video coding standards, such as MPEG series and H.26x series including H.265 [ISO/IEC, 1993; ISO/IEC, 1996; ITU-T and ISO/IEC, 2003; Sullivan et al., 2004; Sullivan and Wiegand, 2005; Ohm and Sullivan, 2013].

Motion Compensated Prediction (MCP) assumes that the current frame can be locally modelled as a translation of the reference frames. MC uses reference frames to predict the current frame, and then encodes RPE. Normally, a P-frame is predicted from one of the previous reference frames. Similarly, a motion compensated bi-prediction or B-frame is predicted from two previous reference frames and the next frame. To achieve such a high coding efficiency, H.264/MPEG-4 AVC use Multiple Reference Frames' ME (MRFME) of up to five reference frames to predict the current frame. However, this dramatically increases the computational complexity of the encoders. Moreover, MRFME must be stored in memory until they are no longer needed for further usage, which requires a large amount of memory usage [Huang et al., 2006; Kim, 2010; Srinivasan and Rao, 1985].

The simplest method of MCP is to use the previous frame as the predictor for the current frame, and encode the difference between them. However, this prediction can be effective only if the two frames are similar and the residual values are close to zero. In any video, either the camera is moving or the object is moving with the fixed camera or scene lighting changes. In all cases, the difference between successive frames will not be close to zero and a lot of energy remains in the residual frame. This means that there is still a big amount of information to compress after this stage. To achieve further compression, a better prediction of the current frame may be formed by compensating for motion between the two frames. In order to carry out motion compensated prediction, the motion of the moving objects has to be estimated first; this is known as Motion Estimation (ME). Figure 3.3 shows the residual prediction error with/without ME [Srinivasan and Rao, 1985; Huang et al., 2006; Yu and Peng Wang, 2010].

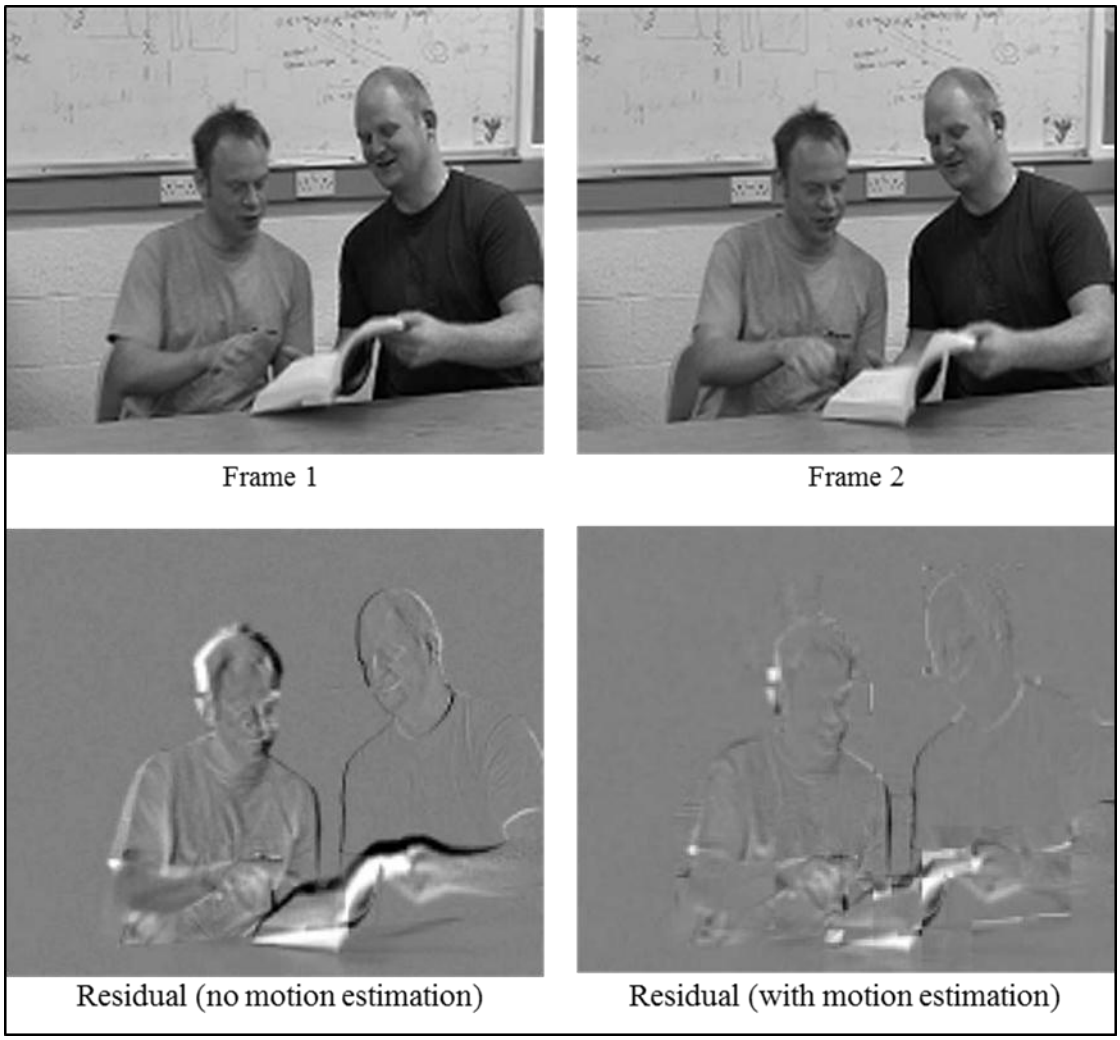


**Figure 3.3: The residual prediction error without ME and the residual prediction error with ME [Richardson, 2010]**

## 3.3 Motion Estimation (ME)

Motion Estimation is the first step of inter-frame compression and usually the most computationally intensive part (about 50% for one reference - 80% for five of the entire system) in a video encoder [Srinivasan and Rao, 1985; Huang et al., 2006; Horn and Schunck, 1981; Richardson, 2010]. It is possible to estimate the displacement for every pixel position between successive video frames, producing a field of pixel flow vectors known as the *optical flow*. The field is subsampled and hence only one vector for every two pixels is shown. However, for motion compensation, this is not a practical method since the calculation of optical flow is very computationally intensive and needs computations for each pixel. Moreover, the number of optical flow vectors is equal to or half the number of pixels. These vectors will be sent to the decoder in order to form

MCP. As a result a large amount of data should be transmitted [Srinivasan and Rao, 1985; Huang et al., 2006; Horn and Schunck, 1981; Richardson, 2010].

Nowadays ME is not only used for the application of video compression, it is used and implemented in various fields to solve their problems some of them are intelligent applications such as psychological studies of Gesture Recognition [Mr. P. Vijaykumar, 2011]. Gesture Recognition can be termed as the process in which the receiver recognizes the gestures made by the user. Gesture is a meaningful expression involving the movements of the face, hand, finger, etc. Motion estimation has been used to get the motion vector of the movement data as an important part of the hall process [Kratz and Ballagas, 2007; Mitra and Acharya, 2007]. Therefore ME attracts the attention of a lot of researchers.

The practical and widely used method to estimate the motion of a group of pixels (macroblock) of the current frame is called Block Matching Algorithm (BMA).

## 3.4 Block Matching Motion Estimation

Block matching algorithm is the most popular technique used for motion estimation, in which the current luminance frame is divided into non-overlapped MacroBlocks (MBls) of size N×M. These macroblocks are then compared with the corresponding macroblock and their adjacent neighbours in the reference frame. This will carry out displacement vectors that stipulate the movement of the macroblocks from one location to another in the reference frame [Barjatya, DIP 6620 Spring 2004]. For any macroblock in the current frame, the BMA finds the matching macroblock of the same size N×M in the search area within the reference frame. The position of the matching macroblock gives the Motion Vector (MV) of the current macroblock, as shown in Figure 3.4. This motion vector has two parts, horizontal and vertical, which can be positive or negative. A positive value means motion to the right or motion down and a negative value means motion to the left or motion up. These motion vectors will be used to form the MCP to the current frame from the reference by block motion compensation, as shown in Figure 3.5. The MVs will be encoded using entropy coding and the RPE between the current frame and the MCP will be encoded using transform coding, quantisation and entropy coding. At the decoder, the received MVs and RPE will be decoded and utilised

to form MCP from the reconstructed reference frame and use the reconstructed RPE to reconstruct the current frame.

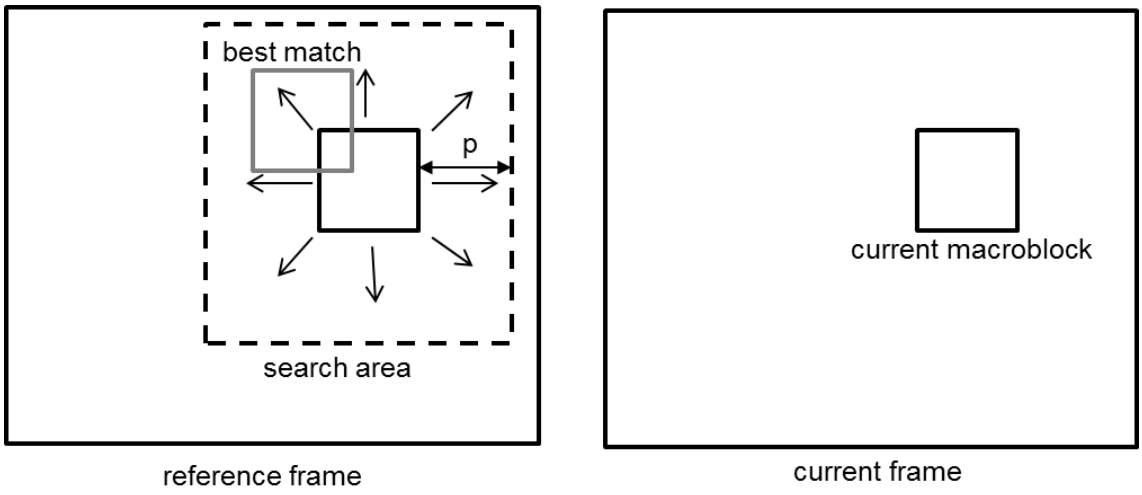


**Figure 3.4: Block matching ME (adapted from [Huang, 2006])**

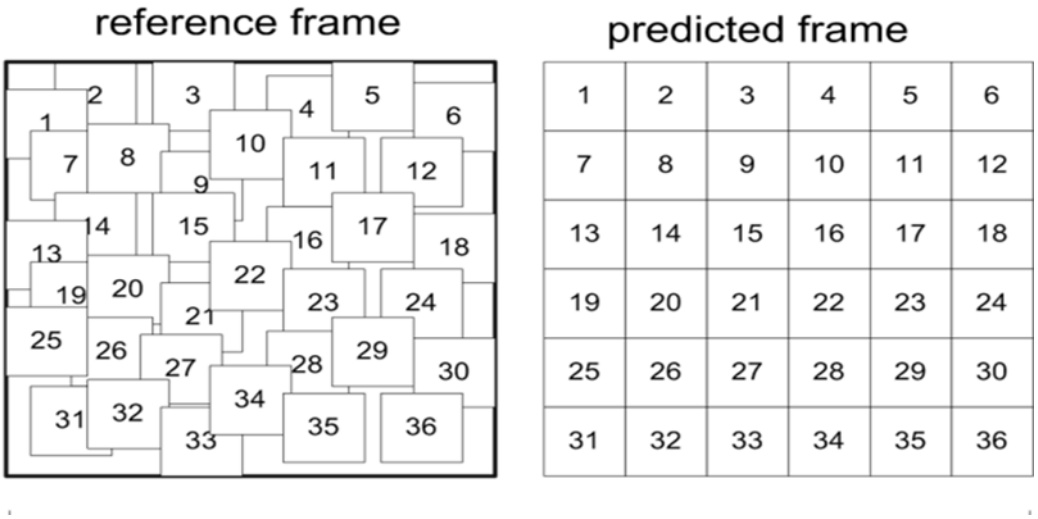


**Figure 3.5: Block motion compensation [Kim, 2010]**

The matching measure is usually determined using a Block Distortion Measure (BDM) like Mean Absolute Difference (MAD) (equation 3.1), or Sum of Absolute Differences (SAD) (equation 3.2) or Mean Square Error (MSE) (equation 3.3). The macroblock with the least BDM is considered to be the one matching the current macroblock [Metkar and Talbar, 2010].

The search area for a macroblock match is usually constrained up to $p$ pixels on all four sides around the corresponding macroblock in the reference frame, where $p$ is the search parameter. Larger motions require a larger $p$ value, which demands more computational power, as shown in Figure 3.4.

For the current macroblock *C* of dimension *N*×*N* and the candidate macroblock *R* in the reference frame with a displacement of $(v_x, v_y)$, SAD, MAD and MSE are defined as:

$$SAD\ (C, R, v_x, v_y) = \sum_{i=1}^{N}\sum_{j=1}^{N}\left|C(i,j) - R(i{+}v_x, j + v_y)\right| \tag{3.1}$$

$$MAD(C, R, v_x, v_y) = \frac{1}{N \times N}\sum_{i=1}^{N}\sum_{j=1}^{N}\left|C(i,j) - R(i{+}v_x, j + v_y)\right| \tag{3.2}$$

$$MSE(C, R, v_x, v_y) = \frac{1}{N \times N}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(C(i,j) - R(i{+}v_x, j + v_y)\right)^2 \tag{3.3}$$

where $C(i,j)$ is the pixel value of current MBl at position $(i,j)$ and R($i{+}v_x, j + v_y$) is the pixel value of the reference frame with the vector $(v_x, v_y)$ within the search range $[-p, p]$.

### 3.4.1 Block-Size Motion Estimation

Macroblock size is an important parameter of the BMA. In the BMA, increasing the size of the macroblock means that more computations are required. However, it also means that there will be fewer macroblocks per frame, so the amount of computation needed to perform motion estimation will be decreased. There is a high possibility that the big macroblock will contain different objects moving in different directions. In other words, using a larger macroblock size reduces the amount of computation; however, it provides poor prediction; while smaller macroblock size can produce better motion compensation results and hence reduces residual energy. However, smaller MBl size leads to increased complexity and increase in the number of motion vectors that need to be transmitted, which may outweigh the benefit of reduced residual energy. An effective

compromise is to adapt the macroblock size to the picture characteristics, for example choosing a large block size in the homogeneous and shade regions of a frame and choosing a small block size for areas of high details, edges, and complex motion, which is called Variable Block-Size Motion Estimation (VBSME) [Marpe et al., 2006; Richardson, 2003; Sayood, 2006; Ruiz and Michell, 2011].

The default block size for motion compensation is 16×16 samples for the luminance component. Fixed Block-Size Motion Estimation (FBSME) of size 16×16 or 8×8 has been used in the first-generation coding standards; while H.264\AVC utilises VBSME, which is more complicated. VBSME allows a macroblock of 16×16 samples of the luminance component to be partitioned into 4 ways, as shown in Figure 3.6: one 16×16 MBl, two 16×8 sub-MBls, two 8×16 sub-MBls or four 8×8 sub-MBls. In addition, each of the four 8×8 sub-MBl partitions within the MBl can be further sub-partitioned into 3 ways, as shown in Figure 3.6: two 8×4 sub-MBls, two 4×8 sub-MBls or four 4×4 sub-MBls. These partitions and sub-partitions give around 41 MBls in total for each MBl. For each type of sub-MBl, a motion vector is required. Each motion vector must be coded and transmitted with the choice of partition(s). In order to get these MVs for each MBl, the computation of comparison operations was increased. To enhance these computations, a large partition size is applied for homogeneous areas of the frame and a sub-partition size may be useful for detailed areas [V.K.Ananthashayana and Pushpa.M.K, 2009; Sayood, 2006; Sullivan and Wiegand, 2005; Wien, 2003; Ruiz and Michell, 2011].

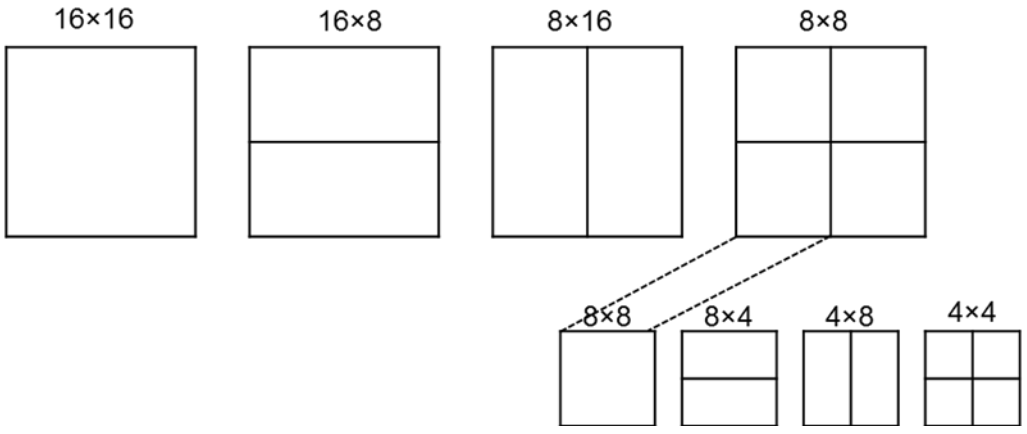


**Figure 3.6: Macroblock partitions and sub-macroblock partitions [Ruiz and Michell, 2011]**

### 3.4.2 Full Search

The simplest algorithm which can be used for motion estimation to find motion vectors is the Full Search (FS), or Exhaustive Search (ES), which exhaustively searches for the best matching block within the search area, where the correlation window is moved to each candidate position within the search area. It can be described by:

$$SAD(m,n) = \sum_{i=1}^{N}\sum_{j=1}^{N}|c(i,j) - s(i+m, j+n)| \;;\; -p \le m,n \le p \tag{3.4}$$

$$MV = \{(u,v) \mid SAD(u,v) \le SAD(m,n);\; -p \le m,n \le p\} \tag{3.5}$$

where :

SAD (m, n) is the distortion of the candidate macroblock at search position (m, n), $\{c(x,y) \mid 1 \le x \le N, 1 \le y \le N\}$ means current macroblock data, $\{s(x,y) \mid -p \le x \le p + N, -p \le y \le p + N\}$ stands for search area data; the search range is $[-p, p]$, the block size is N×N.

Input

- Convert video to frames and convert to greyscale
- Read frames
- Let frame I be the reference frame of frame I+2
- Divided the frames into macroblocks of size N×N.
- Let the search window of maximum stepsize =P

Find the motion vectors for each macroblock by using FS motion estimation

- For each macroblock MBl in frame I+2.
- compute the SAD between current MBl and each candidate macroblocks at the search windows in the reference frame I.
- let $SAD_{min}$ = MINIMUM (SUM)
- find the coordinates of the vector where $SAD_{min}$.

Output

- Motion vectors for all MBls ; Number of search points ; Time of process.

**Figure 3.7: Pseudo code of FS**

From the above, $(2p + 1)^2$ of search locations need to be examined by the FS algorithm. As a result, FS finds the best possible match and gives the highest PSNR amongst any block matching algorithm; however, a large amount of computational complexity is involved, especially with VBSME and MRFME.

Various methods of fast block matching algorithms have been developed to decrease and improve this computational complexity. If the algorithm enhances the computation and produces the same quality results as FS then it is called lossless block matching algorithm while if the algorithm could not keep the same quality results then it is called lossy block matching algorithm [Sayood, 2006; Srinivasan and Rao, 1985; Huang et al., 2006].

## 3.5 Chapter Summary

The high correlation between successive frames in video sequences leads to temporal redundancy. To reduce this redundancy and satisfy compression, inter-frame compression has been used. The first stage of inter-frame compression is temporal prediction, in which some frames can be predicted from others to reduce the transmission rate of video image sequences and obtain further compression. Motion estimation technique has been used to improve this prediction by estimating the motion of the moving objects between the reference frame and the current frame. Motion estimation is the most computationally intensive part in a video encoder. The practical and most widely used method to estimate the motion of the macroblock of the current frame is called Block Matching Algorithm (BMA). In this case, video frames are divided into a set of non-overlapped MBls. Each target macroblock in the current frame is compared with a number of candidate macroblocks within the search area in the reference frame in order to find the best matching macroblock. The spatial difference between the two matching macroblocks will determine a set of displacement vectors that stipulate the movement of the macroblocks from one location to another in the reference frame. Checking all search area locations is called the Full Search algorithm. The full search algorithm can produce the best possible matching and hence the highest resolution MCP. However, this algorithm suffers from long computational time, which necessitates improvement. Various methods of fast block matching algorithms have been developed to decrease and improve the computational complexity. These methods

can be classified into two types, lossless and lossy BMA. More details about both types will be provided in Chapter 4 and the novel algorithms to develop these methods will be introduced in Chapter 5.

# CHAPTER 4: FAST BLOCK MATCHING ALGORITHMS

As shown in Chapter 3, motion estimation shows computational complexity. Hence, the computational complexity of video coding can be reduced by efficiently coding Motion Estimation (ME). A block matching algorithm is the most common technique used for motion estimation to find the best matching macroblock for the current macroblock from the reference frame. FS is the simplest but the most computation-intensive Block Matching Algorithm (BMA), which exhaustively tests all the search locations for the best matching macroblock within the search area. As a result, Full Search (FS) finds the best possible match and gives the highest Peak Signal-to-Noise Ratio (PSNR). Moreover, variable block size and multiple reference frames have been involved in the later video coding standards. Therefore, the required computation is highly increased and motion estimation has become a problem in many video applications, for example mobile video and real-time video coding.

In the last three decades, various methods of fast BMA have been developed to reduce such high computational complexity. Some of the fast BMA algorithms have been adopted in video coding standards [ISO/IEC, 1993; ISO/IEC, 1996; ITU-T and ISO/IEC, 2003]. This indicates that this is an extremely active field of research, and most of the fast block matching algorithms are introduced first for FBSME and then extended to VBSME [Xiong et al., 2011]. The performance of each algorithm can be estimated by benchmarking with FS. The effective one minimises the RPE and saves the computational time compared with Full Search.

Fast block matching algorithms can be classified into lossy block matching algorithms and lossless block matching algorithms. Lossy BMAs reduce the computational complexity; however, the search results quality is not the same as for FS. That is, the PSNR of the decompressed video with lossy BMA is not as good as the PSNR of the one with the full search. While lossless BMA preserves the video quality as well as speeding up the FS [Nie and Ma, 2002; Huang et al., 2006; Cai et al., 2009].

This chapter discusses various lossy and lossless techniques using block matching algorithms, as shown in section 4.1 and 4.2. The chapter summary is provided in section 4.3.

## 4.1 Lossy Block Matching Algorithms

Lossy BMAs can be classified into the following categories:

### 4.1.1 Fixed Set of Search Patterns

Fixed set of search patterns or what is known as reduction in search positions is the most popular category in lossy block matching algorithms. These algorithms reduce search complexity by selecting a subset of the possible search candidate locations instead of all possible MBls within the search window. Most algorithms in this category state that the error decreases monotonically as the search location moves closer to the best-matching location. Therefore, the search starts with the locations coarsely spread over the search window according to some predefined uniform pattern. After that, the search is repeated with a smaller spread around the search location with the minimum BDM (error) obtained from the preceding step. Each search pattern has a specific shape (rectangle, diamond, hexagonal, cross, etc.) [Al-Mualla et al., 2002; Huang et al., 2006].

The first algorithm proposed in this category was the Two-Dimensional Logarithmic Search (2D-LOG), which was proposed in 1981 [Jain and Jain, 1981]. After that, some well-known similar algorithms were proposed, such as: Three Step Search (TSS) [Koga et al., 1981], Orthogonal Direction Search (OSA) [Puri et al., 1987], New Three Step Search (NTSS) [Reoxiang et al., 1994], Four Step Search (4SS) [Lai-Man and Wing-Chung, 1996], Diamond Search (DS) [Shan and Kai-Kuang, 1997], Simple and Efficient Search (SESTSS) [Jianhua and Liou, 1997], Cross-Diamond Search algorithm (CDS) [Cheung and Po, 2002], Novel Hexagon-based Search (NHS) [Ce et al., 2004], Efficient Three Step Search (ETSS) [Xuan and Lap-Pui, 2004], Modified DS (MODS) [Xiaoquan and Nam, 2005] Multi-pattern-based search (TCon) [Akram and Izquierdo, 2010] and many others.

Much of the research and coding was dependent on the Fixed Set of Search Patterns due to its high-speed search capabilities in comparison to other lossy BMA categories.

Unfortunately, these algorithms produce significant loss in visual quality when the actual motion does not match the pattern and hence these algorithms become trapped in a local minimum. As an example, a centre-biased search pattern cannot provide optimal motion estimation for videos with large motions [Hui-Yu and Shih-Hsu, 2011].

### N-Step

Three step search, new three step search and simple and efficient three step search come under the N-Step Search class. The steps of this class are summarised as follows: (1) Choose step size (which is usually slightly larger or equal to half of the search window). (2) Number of search points is selected at a distance of the step size as well as the centre point. The macroblock with the minimum BDM value becomes the centre of the next step. (3) Divide step size by two and select new search points at a distance of the new step size. (4) Repeat step 2 until the step size becomes one.

### Three Step Search (TSS)

TSS uses a maximum of three steps in a coarse to fine search patterns. For a usual search window of parameter $p = 7$ the initial step size will be $4 = round((p+1)/2)$; TSS utilises nine search points centred at the search area (eight points on the boundary of the search square and one centre point) to be compared in the first step search. As mentioned before, the point with the minimum BDM value becomes the centre of the next step. Therefore, there are eight search points to be compared in the second and third step searches, i.e. the total number of search points is $(9+8+8=25)$, as shown in Figure 4.1.

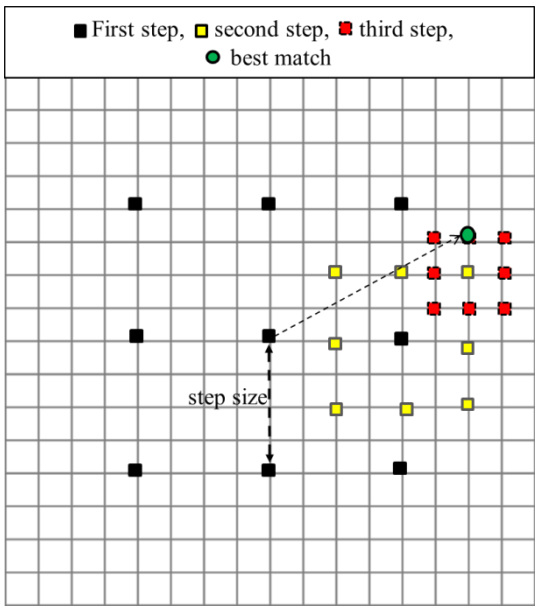


**Figure 4.1: TSS [Jong-Nam and Tae-Sun, 1998]**

Due to its simplicity and reasonable performance, the TSS is widely used for research purposes [Chao-Feng et al., 2012]. The drawback of the TSS is the reality of its not being efficient with small motion video, since the search points forming the search pattern in the first step are positioned at a relatively large distance from the search centre; while 80% of the MBls in various motion video sequences can be regarded as stationary or quasi-stationary MBls, which means that 80% of MVs are centre-biased, i.e. lie within a region of 5×5 of the central area [Cheung and Po, 2002]; therefore TSS is not efficient for most video sequences. This problem was solved in 1994 by proposing a new search called NTSS [Reoxiang et al., 1994].

**New TSS (NTSS)**

NTSS provided improvement over the quality results of TSS [Reoxiang et al., 1994]. This algorithm is considered as one of the first widely accepted fast block matching algorithms. Moreover, it has been used in earlier standards like MPEG 1 and H.261 [Mogus et al., 2010].

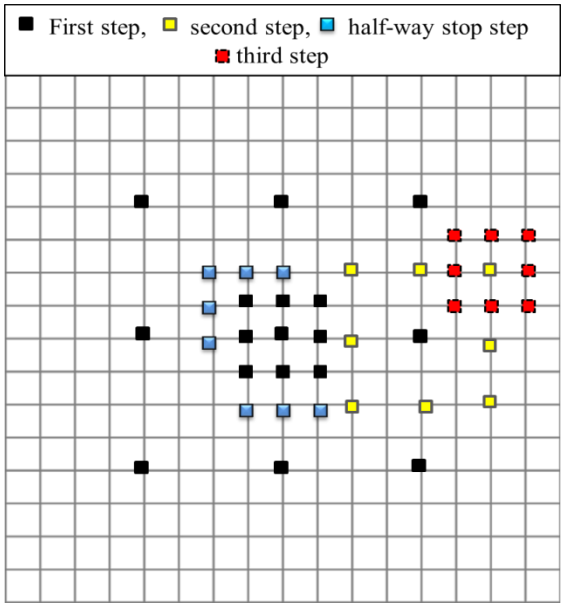


**Figure 4.2: : NTSS [Reoxiang et al., 1994]**

NTSS added a smaller search pattern of eight points at the central area to the first step of the original TSS search pattern. That is, NTSS requires more search points compared to TSS. For search windows of parameter p= 7, NTSS requires 33 search points for

large motion MBls while TSS always required 25, which means more computations may be needed. In order to compensate the disadvantage of adding a centre-biased searching pattern, NTSS used a halfway stop technique for stationary and quasi-stationary MBls as follows:

Step 1: similar to the first step of the TSS algorithm, the matching macroblock is determined first using eight search points on the boundary of the step size search square and the centre point: (±stepsize,0), (0, ±stepzise), (0,0), (±stepsize,±stepsize), and eight extra neighbours of the centre-biased search pattern will be searched: (±1,0), (0, ±1), (±1,±1) as shown in Figure 4.2.

Step 2: if the minimum BDM in the first step is already at the centre of the search window, the search will be stopped and the motion vector is set as (0, 0), which mean that the total number of search points is 9+8; this is called the first-step-stop. Otherwise, the centre will move to the minimum BDM. In stationary and quasi-stationary MBls, the new centre will move to the centre-biased search points and the search in the second step will be performed only for three or five neighbouring points to complete 8 points adjacent to this centre, as illustrated in Figure 4.2. The minimum BDM of this step search gives the matching MBl, i.e. the total number of search points will be only 9+8+3 or 9+8+5, and this is called the second-step-stop. Otherwise, for the large motion MBls, the new centre will move to the boundary search square then the same procedure of TSS is applied and hence the total number of search points will be 33 [Reoxiang et al., 1994; Barjatya, 2004; Mogus et al., 2010; Jae-Yong and Sung-Bong, 1999; Goel and Bayoumi, 2006].

Therefore, for typical video sequences, NTSS is faster than TSS, while for high motion video sequences the computational complexity for NTSS will be higher than that of TSS. In general, NTSS works better than TSS by producing smaller motion compensation errors, and in terms of computational complexity it is similar to TSS, being simple in nature. Therefore, it is utilised as one of the comparison algorithms in this thesis and the pseudo code of NTSS is illustrated in Figure 4.3.

Input

- Convert video to frames and convert to greyscale
- Read frames
- Let frame I be the reference frame of frame I+2
- Divided the frames into macroblocks of size N×N.
- Let the search window of maximum stepsize =P

Find the motion vectors for each macroblock by using NTSS motion estimation

- For each macroblock MBl in frame I+2
- Let L = round((p+1)/2) and
- Let stepsize =L
- Compute the MAD between MBl and the 17 candidate macroblocks at the positions (±stepsize,0), (0, ±stepzise), (0,0), (±stepsize,±stepsize), (±1,0), (0, ±1), (±1,±1).
- make sure the position of the candidate macroblock is not out of the frame
- find the coordinate of the vector V=(v1,v2) where the MAD is minimum
- If V =(0,0) then V is the motion vector and the search will end.
- else V become the centre of new search
- If V is one of the candidate macroblocks (±1,0), (0, ±1), (±1,±1)
- compute the MAD between MBl and the candidate macroblocks at eight points adjacent to this centre by add only for three or five neighbouring points depend on the position of the new centre.
- Make sure that don't calculate the same points again that were calculate in the initial search.
- Make sure the position of the candidate macroblock is not out of the frame.
- find the coordinates of the new vector where the MAD is minimum and stop the search.
- else let stepsize = round (L / 2);
- while (stepsize >= 1) do
- compute the MAD between MBl and the candidate macroblocks at eight search points in distance of the step size around the new centre.
- Make sure that don't calculate the same points again that were calculate in the previous search.
- Make sure the position of the candidate macroblock is not out of the frame.
- find the coordinates of the new vector where the MAD is minimum and store it.
- move the centre of the search to new vector.
- stepsize = stepsize / 2
- end do
- find the coordinates of the new vector where the MAD is minimum and stop the search.

Output

- Motion vectors for all MBls ; Number of search points ; Time of process.

**Figure 4.3: Pseudo code of NTSS**

**Simple and Efficient TSS (SESTSS)**

Another extension illustrated to speed up TSS was done by Simple and Efficient TSS [Jianhua and Liou, 1997]. SESTSS requires around half of the computation for TSS while keeping the same regularity and good performance. It exploits the fact that the uniform distribution search pattern in TSS is not effective since the error decreases monotonically as the search location moves closer to the best-match location, i.e. minimum points cannot occur in two directions opposite to each other, which means that, for the search pattern in TSS, at most half of the total eight points are actually required to be searched in each step, and, thus, the computational complexity can be further reduced. Additional computation is needed to determine which directions are to be chosen. The algorithm still has three steps like TSS but each step has two phases as follows [Jianhua and Liou, 1997]:

Step 1: first phase: compute MAD of the three locations *A*, *B* and *C* as shown in Figure 4.4. Point *A* refers to the centre location. *B* and C are located at step size =4 away from A, towards the right-hand side and bottom. In the second phase, a few more points are added depending on the following conditions:

$$If\ MAD(A) \geq MAD(B)\ and\ MAD(A) \geq MAD(C), select\ (b);$$

$$If\ MAD(A) \geq MAD(B) and\ MAD(A) < MAD(C), select\ (c);$$

$$If\ MAD(A) < MAD(B) and\ MAD(A) < MAD(C), select\ (d);$$

$$If\ MAD(A) < MAD(B) and\ MAD(A) \geq MAD(C), select\ (e);$$

Where:

(b) is the second phase of one point more add to phase one located at step size =4 away from B towards bottom side.

(c) is the second phase of two points more add to phase one located at step size =4 away from A and B towards above side.

(d) is the second phase of three points more add to phase one located at step size =4 away from A, towards left-hand side, above and up-left corner.

(e) is the second phase of two points more add to phase one located at step size =4 away from A and C towards left-hand side.

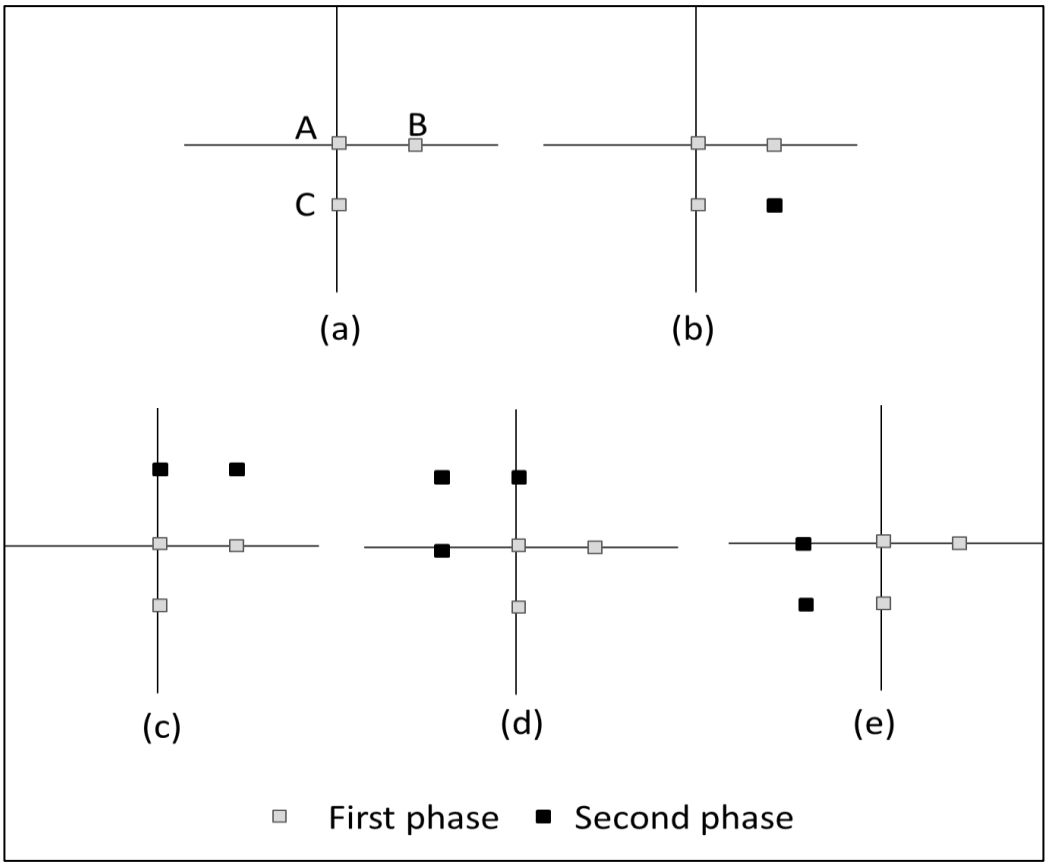


**Figure 4.4: Search patterns of SESTSS depending on MAD of A, B and C [Jianhua and Liou, 1997]**

Step 2: the point with the minimum MAD value from step 1 becomes the centre of the current step and the step size will be 2. The pattern of the first phase in this step is similar to first phase in step 1.

Step 3: repeat step 2 with step size equal to 1.

Figure 4.5 shows an example for the SESTSS, and the pseudo code of SESTSS is illustrated in Figure 4.6

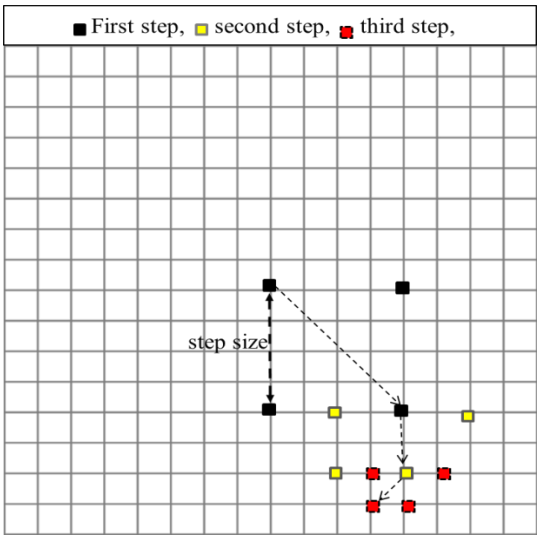


**Figure 4.5: Example of the SESTSS search procedure [Jianhua and Liou, 1997]**

Input

- Convert video to frames and convert to greyscale
- Read frames
- Let frame I be the reference frame of frame I+2
- Divided the frames into macroblocks of size N×N.
- Let the search window of maximum stepsize =P

Find the motion vectors for each macroblock by using SESTSS motion estimation

- For each macroblock MBl in frame I+2
- Let L = round((p+1)/2) and
- Let stepsize =L
- While stepsize >= 1 do
- Compute the MAD between MBl and the 3 candidate macroblocks at the positions A=(0,0), B=(stepsize,0), and C=(0,stepsize).
- make sure the position of the candidate macroblock is not out of the frame
- $If\ MAD(A) >= MAD(B)\ and\ MAD(A) >= MAD(C)$
    - Compute MAD at (stepsize,stepsize)
- $else\ If\ MAD(A) >= MAD(B) and\ MAD(A) < MAD(C)$
    - Compute MAD at (0,-stepsize) and (stepsize,-stepzise)
- $else\ If\ MAD(A) < MAD(B) and\ MAD(A) < MAD(C)$
    - Compute MAD at (0,-stepsize) , (-stepsize,stepzise) and (-stepzise,0)
- $else\ If\ MAD(A) < MAD(B) and\ MAD(A) >= MAD(C)$
    - Compute MAD at (-stepsize,-stepzise) and (-stepzise,0)
- Make sure the position of the candidate macroblock is not out of the frame.
- find the coordinate of the vector V=(v1,v2) where the MAD is minimum
- let V become the centre of new search
- let stepsize = round (L / 2);
- end do
- find the coordinates of the new vector where the MAD is minimum and stop the search.

Output

- Motion vectors for all MBls ; Number of search points ; Time of process.

**Figure 4.6: Pseudo code of SESTSS**

**Diamond Search (DS)**

DS is one of the most common and widely used algorithms. DS requires significantly less computation by reducing the average search points while achieving acceptable performance in comparison with its prior fixed set of search pattern algorithms. Therefore, it is adopted by the reference software of MPEG-4 [ISO/IEC, 1999; Huang et al., 2006].

Similar to NTSS, the DS is based on the assumption that most motion vectors of typical video sequences are centre-biased. Also, it is based on the fact that the MBl displacement of real-world video sequences could be in any direction, but mainly in horizontal and vertical directions [Shan and Kai-Kuang, 1997] .

This technique utilises two search patterns, a large diamond search pattern (LDSP) of 9 search points and a small diamond search pattern (SDSP) of five search points, as follows: in the first step the matching MBl is searched within the search points of the LDSP which are {(±2,0), (0, ±2), (0,0), (±1,±1)}, as shown in Figure 4.7. The position of the minimum BDM for the LDSP becomes the centre of the new search. If the minimum BDM is already at the centre of the LDSP, then the search pattern is switched from the LDSP to a SDSP of four points {(±1, 0), (0, ±1)}. Otherwise, the search in the next step will be performed only for three or five neighbouring points that complete the LDSP of this new centre, as illustrated in Figure 4.7. The LDSP is repeatedly used in the searching procedure until the step in which the minimum BDM point stays at the centre of the LDSP. The search pattern is then switched to a SDSP. The minimum BDM point found from the SDSP will be the best matching block [Zhu and Ma, 2000; Barjatya, 2004; Mogus et al., 2010; Shan and Kai-Kuang, 1997].

The search pattern of the DS algorithm is neither too small nor too big since the step size has two pixels in horizontal and vertical directions and one pixel in each diagonal direction. Also, the DS algorithm does not have a limited number of search steps. Therefore, for both large motion MBls, and stationary or quasi-stationary MBls, the DS algorithm is not so easily trapped into a local minimum point; this algorithm can find the global minimum accurately. In addition, the compact shape of the search patterns used in the DS algorithm increases the possibility of finding the global minimum point located inside the search pattern. The pseudo code of DS is shown in Figure 4.8.

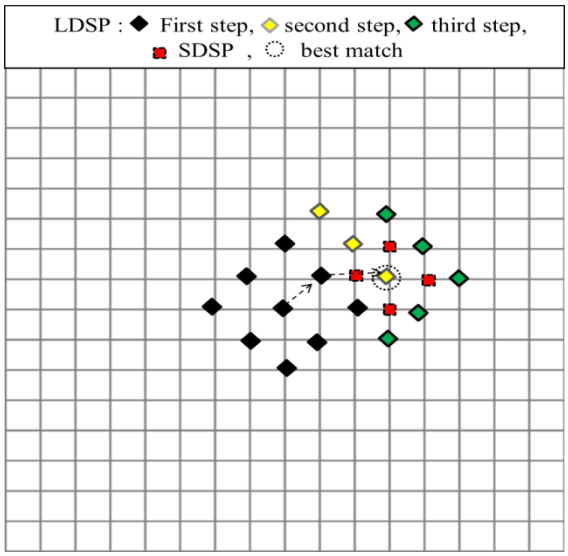


**Figure 4.7: DS [Shan and Kai-Kuang, 1997]**

Input

- Convert video to frames and convert to greyscale
- Read frames
- Let frame I be the reference frame of frame I+2
- Divided the frames into macroblocks of size N×N.
- Let the search window of maximum stepsize =P

Find the motion vectors for each macroblock by using DS motion estimation

- For each MBl in frame I+2
- Compute the MAD between MBl and 9 search points of large diamond search pattern (LDSP) which are {(±2,0), (0, ±2), (0,0), (±1,±1)}.
- make sure the position of the candidate macroblock is not out of the frame.
- find the coordinate of the vector V where the MAD is minimum.
- while (V is not at the centre of LDSP) do
- the position of the minimum MAD becomes the centre of the new search of LDSP.
- Compute MAD between MBl and three or five neighbouring points that complete the LDSP of the new centre.
- End do
- compute the MAD between MBl and 4 search points of a small diamond search pattern (SDSP), {(±1, 0), (0, ±1)}.
- find the coordinates of the vector where the MAD is minimum at SDSP and stop the search.

Output

- Motion vectors for all MBls ; Number of search points ; Time of process.

**Figure 4.8: Pseudo code of DS**

### 4.1.2 Predictive Search

Predictive search technique is a lossy block matching algorithm that exploits the correlation between the current MBl and its neighbouring MBl. It utilises the motion information in the spatial and/or temporal neighbouring MBl. The predicted MV can be obtained by selecting one of the previously-coded neighbouring MVs; for example, the predictors can be the MVs of the MBls on the left, top, and top right, as shown in Figure 4.9, or the MV of the collocated MBl in the previous frame, as shown in Figure 4.10, and in the previous two frames.

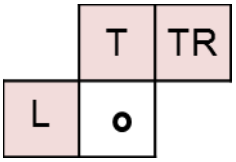


**Figure 4.9: Current MBl with the predictor MV of top (T), left (L) and top right (TR) MBls**

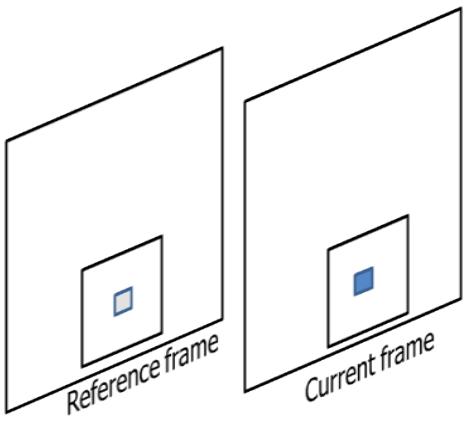


**Figure 4.10: Current MBl and the collocated MBl in the previous frame (adapted from [Huang, 2006])**

The Motion Vector Predictor (MVP) is utilised in two ways:

1. The difference between the current motion vector and the MVP, which is called motion vector difference, is transmitted instead of the current MV itself. The MVP in this case is the median of three candidate predictors, which are the

motion vectors of the three neighbouring MBls, as illustrated in Figure 4.9 [Al-Mualla et al., 2002].

2. The MVP forms an initial estimate of current MV. This type is a fast motion estimation algorithm that has low computational complexity with acceptable performance. It can effectively reduce the search points and hence the computation by exploiting the target macroblock that is likely to belong to the area of the neighbouring MVs, and the initial search starts directly in this area. The MVP could be one or more of the previously-coded neighbouring MVs, or their average MVs as in Figure 4.9,. Note that additional memory for storing the neighbouring MVs is needed in this method [Ezhilarasan and Thambidurai, 2008; Chalidabhongse and Kuo, 1997; Richardson, 2010].

This technique is used in the Adaptive Rood Pattern Search (ARPS) algorithm [Nie and Ma, 2002], Joint Adaptive Block Matching Search (JABMS) algorithm, Unsymmetrical Multi-Hexagon search (UMHexagonS) [Yi et al., 2005], and simplified block matching algorithm for fast motion estimation [Ananthashayana and Pushpa, 2009].

**Adaptive Rood Pattern Search (ARPS) Algorithm**

The ARPS algorithm [Nie and Ma, 2002] based on the MPEG-4 Verification Model (VM) [ISO/IEC, 1999] showed a speed 2-3 times faster and maintained a fairly similar performance than that of the DS [Zhao et al., 2008]. ARPS uses a predictive search technique to form an initial estimate of finding the global minimum point. This relates to the fact that, if the MBl around the current block moves in a particular direction, then there is a high probability that the current MBl will also have a similar motion vector. Moreover, the step size search pattern of this algorithm is changeable according to the motion vector predicted behaviour. This technique depends on the DS technique, which uses two different types of fixed patterns, the Large Search Pattern (LSP) and the Small Search Pattern (SSP), as shown in Figure 4.11. In addition, the motion vector predicted (MVP) of this algorithm is the coded motion vector of the immediate left MBl, which means one neighbouring MV needs to be recorded. This MVP is utilised to pre-determine the motion behaviour of the current MBl and to define the most suitable step size to perform efficient ME. The steps of this algorithm are as follows:

Step 1: determine the step size that refers to the distance between the centre and any vertex points in the LSP. If $x$ and $y$ are the horizontal and vertical components of the MVP, respectively, then the step size will be the maximum absolute value of these components determined as follows [Nie and Ma, 2002]:

$$step\ size = Max\{|x|, |y|\} \tag{4.1}$$

For the MBl on the left side of the frame, the step size will be fixed as 2 pixels.

Step 2: the matching macroblock is searched first within the search points of LSP plus the search point indicated by the MVP, as shown in Figure 4.12. The point that has the least MAD becomes the origin for subsequent search steps. The new search centre directly moves to an area where there is a high probability of finding the global minimum, and the new search pattern is changed to a SSP, as shown in Figure 4.11.

Step 3: the matching MBl found in the current step will be re-positioned as the new search centre of the next search if it is not already at the centre of the search pattern. This process will be repeated until the matching MBl stays at the centre of the SSP.

Figure 4.13 shows the pseudo code of ARPS.

A further development of this algorithm is called Adaptive Rood Pattern-Zero Motion Prejudgment (ARP-ZMP), which can be achieved by checking for zero motion prejudgment in which, if the SAD between the current MBl and the MBl at the same location in the reference frame (i.e., the centre of the current search window) is less than a predefined threshold, then the search is stopped and the MV will be zero [Nie and Ma, 2002].

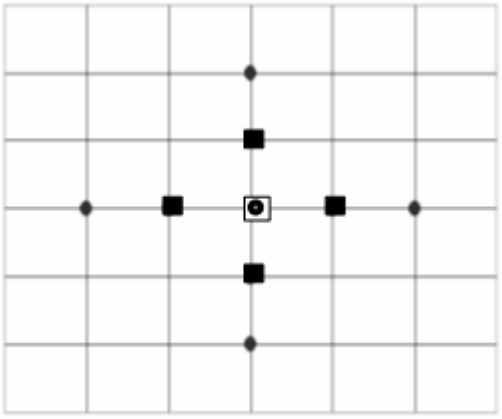

**Figure 4.11: The solid circle points (●) are the LSP and the squares (■) are the SSP for ARPS**

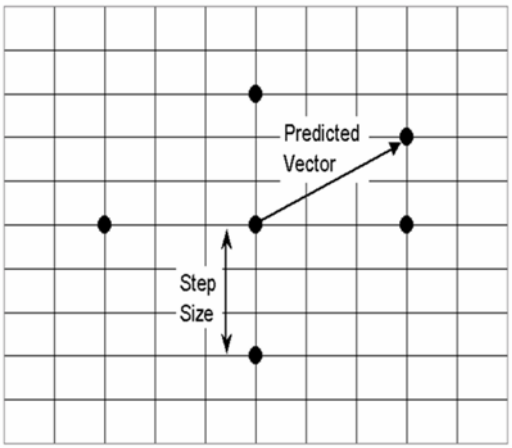


**Figure 4.12: Adaptive Rood Pattern Search [Nie and Ma, 2002]**

Input

- Convert video to frames and convert to greyscale
- Read frames
- Let frame I be the reference frame of frame I+2
- Divided the frames into macroblocks of size N×N.
- Let the search window of maximum stepsize =P

Find the motion vectors for each macroblock by using ARPS motion estimation

- For each macroblock MBl in frame I+2
- If MBl on the left side of the frame, then stepsize = 2
- Else stepzise $= Max\{|x|, |y|\}$, where x,y are the components of the MV for the previous left MBl.
- Compute the MAD between MBl and 5 or 6 search points of large search pattern (LSP) which are {(±stepsize,0), (0, ±stepsize), (0,0) }and if stepsize ≠2 then add the search point (x,y).
- make sure the position of the candidate macroblock is not out of the frame.
- the position of the minimum MAD becomes the centre of the new search pattern which is small search pattern SSP of 4 points {(±1, 0), (0, ±1)}.
- find the coordinate of the vector V where the MAD is minimum.
- if( V is not at the centre of SSP) then let V be the centre of new search and repeat SSP till V become the centre of the SSP
- else find the coordinates of V and stop the search.

Output

- Motion vectors for all MBls ; Number of search points ; Time of process.

**Figure 4.13: Pseudo code of ARPS**

### 4.1.3 Hierarchical or Multiresolution Search

Hierarchical search exploits the correlation between different resolution levels that represent the same image, which is shown in Figure 4.14 [Song and Ra, 1998]. It uses a multiresolution structure (also known as a pyramid structure) that has different image resolutions with smaller image size at the coarser level. The multiresolution structure is constructed either with simple subsampling or filtering.

Hierarchical search is based on the idea of performing motion estimation at each level successively. Thus, motion estimation is first applied at the lowest resolution level to

obtain an estimate of motion vector. This MV is then passed to the next higher resolution level as an initial estimate. Motion estimation at the higher resolution level is then used to refine this initial estimate. This process is repeated until the highest resolution level is reached. Typically, a two- or three-level hierarchical search is adopted. To reduce the complexity of calculating BDMs, small MBls are used for block matching algorithm at lower resolution levels. Moreover, smaller search ranges are used at higher-resolution levels, since motion estimation starts from a good initial estimate. This reduces the number of locations to be searched. Therefore, more levels can save the amount of computation required, but it has the disadvantage of possibly being trapped in a local minimum because, when the subsampling or filtering is applied to an image, some important details will be lost. In spite of this, multiresolution technique has been regarded as one of the most efficient methods in BMA and it is adopted in applications with very large frames and search areas [Song and Ra, 1998; Cai et al., 2009; Al-Mualla et al., 2002; Nie and Ma, 2002; Huang et al., 2006].

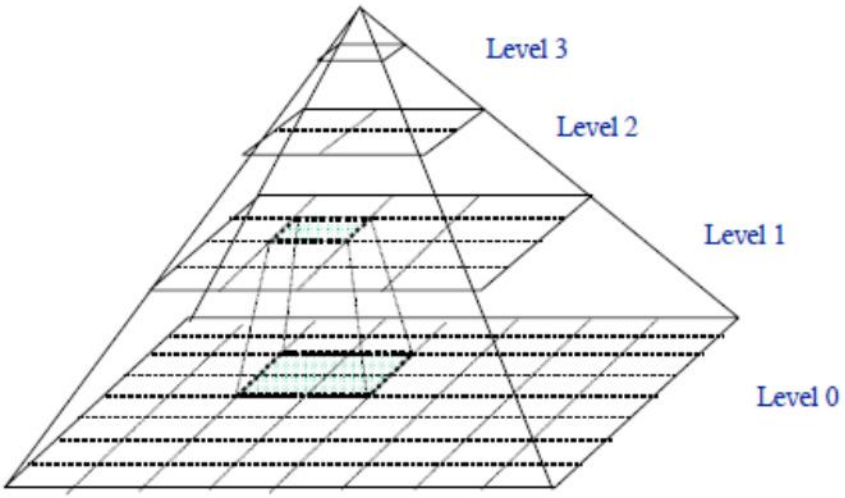


**Figure 4.14: Hierarchical motion estimation using a mean pyramid of three levels [Lin et al., 1998]**

### 4.1.4 Subsampled Pixels on Matching Error Computation

The previous three groups of BMAs can reduce the computation of ME by limiting the number of search locations. This category reduces the complexity of the BDM by decreasing the number of MBl pixels in current and candidate MBls to speed up ME. In

homogeneous areas, neighbouring pixels have high correlation and hence subsampling for these areas can be done without search quality regression. However, in highly textured areas the subsampling will be less accurate. Therefore, this category does not guarantee to find the best match, hence it is lossy BMA even when checking all search area locations. Koga et al used in their work [Koga et al., 1981] a uniform subsampling pattern that performs 2:1 pixel subsampling in both horizontal and vertical directions. As a result, the total computation can be reduced by a factor of 4, as shown in Figure 4.15. Liu and Zaccarin in their work [Liu and Zaccarin, 1993] have used a non-uniform subsampling pattern.

Figure 4.16 shows a block of 8 × 8 pixels with each pixel labelled $a, b, c,$ and $d$ in a regular pattern. If only the pixels of the pattern that consists of all the $\boldsymbol{a}$ pixels are used for block matching, then the computation is reduced by a factor of 4**.** To reduce the drawback that ¾ of the pixels do not enter into the matching computation, all four subsampling patterns are used in a specific alternating manner, as illustrated in Figure 4.16.

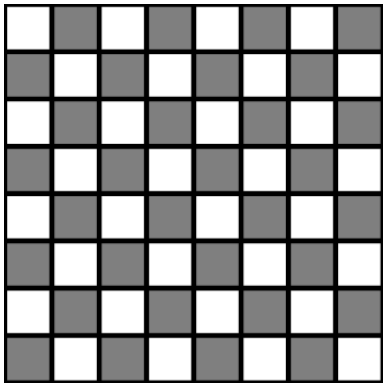

**Figure 4.15: Uniform subsampling pattern 2:1 [Alzoubi and Pan, 2007]**

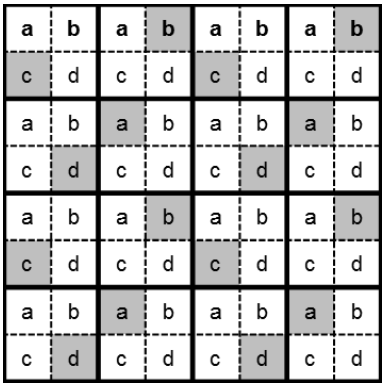


**Figure 4.16: Non-uniform subsampling pattern 4:1 [Liu and Zaccarin, 1993]**

To enhance the quality of a non-uniform subsampling, Yui-Lam and Wan-Chi [Yui-Lam and Wan-Chi, 1996] changed the number of pixels in the subsampling pattern according to block details. That is, for shade MBls fewer pixels are used and more pixels are involved for high-activity MBls. Such a computation reduction method can be

incorporated into other BMAs to achieve higher computational gain, as in [Alzoubi and Pan, 2007].

### 4.1.5 Bitwidth Reduction

In a luminance frame, each pixel is represented with 8 bits resolution. This search technique reduces the original 8 bits resolution to less bits width in order to reduce the hardware cost and power consumption and then applies normal ME search strategies. The first algorithm proposed in this category was Bit-Plane Matching (BPM), which indicates whether a pixel is edge or not [Jian et al., 1995]. The MBl mean is used as the threshold to satisfy a One–Bit Transformation (1BT), and the bit plane of an image frame is constructed in the form of:

$$B(i,j) = \begin{cases} 1 & if\ I(i,j) \geq t_{bm} \\ 0, & otherwise \end{cases} \tag{4.2}$$

where $t_{bm}$ is the threshold value that is set equal to the MBl mean, $I(i,j)$ shows the $(i,j)th$ pixel of the image frame and $B(i,j)$ shows the corresponding bit-plane value.

The other common transformation maps a frame of multi-valued pixels to a frame of binary-valued pixels by comparing the original frame with their multi-bandpass filtered versions to construct *1BT* representations [Natarajan et al., 1997]. Each frame *I* is filtered with a *17 ×17 kernel K* which is given as in equation 4.3. The filtered frame $I_F$ is compared with the original frame *I* to create a one-bit frame *B*, as in equation 4.4 [Erturk, 2007].

$$K(i,j) = \begin{cases} 1/25 & i,j \in [0,4,8,12,16] \\ 0, & otherwise \end{cases} \tag{4.3}$$

$$B(i,j) = \begin{cases} 1 & if\ I(i,j) \geq I_F(i,j) \\ 0, & otherwise \end{cases} \tag{4.4}$$

where $I_F(i,j)$ is the filtered form of the image frame $I(i,j)$.

To find the best matching MBl for the current MBl, a full search can be used. The error between current and candidate MBls will be calculated as the Number of Non-Matching Points (NNMP), which is measured by the exclusive-or (XOR) operation as follows [Erturk, 2007]:

$$NNMP(m,n) = \frac{1}{M \times N}\sum_{i=1}^{M}\sum_{j=1}^{N}\left(B^{t}(i,j) \oplus B^{t-1}(i+m, j+n)\right) \tag{4.5}$$

$$-s \leq m,n \leq s-1$$

where $(m,n)$ shows the candidate displacement, $B^{t}(i,j)$ and $B^{t-1}(i,j)$ are the one-bit planes for the current and reference frame, respectively, $s$ determines the search range, and $\oplus$ is the XOR operation [Mizuki et al., 1996].

In Erturk and Erturk (2005), a Two-Bit Transformation (*2BT*) was proposed to improve motion estimation accuracy compared with 1BT. The first bit plane of 2BT is constructed using the mean value ($\mu = E[I_{tw}]$) of the threshold window surrounding the current MBl. The second bit plane is constructed using the square root of the variance value ($\sigma^2 = E[I_{tw}^2] - E^2[I_{tw}]$) as follows:

$$B_1(i,j) = \begin{cases} 1 & if\ I(i,j) \geq \mu \\ 0, & otherwise \end{cases} \tag{4.6}$$

$$B_2(i,j) = \begin{cases} 1 & if\ I(i,j) \geq \mu + \sigma\ \ or\ \ I(i,j) \leq \mu - \sigma \\ 0, & otherwise \end{cases} \tag{4.7}$$

where $B_1(i,j)$ and $B_2(i,j)$ represent the 2BT, while the number of non-matching points is defined as:

$$NNMP(m,n) = \frac{1}{M \times N}\sum_{i=1}^{M}\sum_{j=1}^{N}\{B_1^t(i,j) \oplus B_1^{t-1}(i+m,j+n)\} \| \{B_2^t(i,j) \oplus B_2^{t-1}(i+m,j+n)\} \quad (4.8)$$

$$-s \leq m,n \leq s-1$$

where $(m,n)$ shows the candidate displacement, $B_{1,2}^t(i,j)$ and $B_{1,2}^{t-1}$ are the two-bit planes for the current and reference frame, respectively, $s$ represents the search range, and $\oplus$ is the XOR operation. The operation$\|$ denotes the Boolean OR operation.

Some other algorithms were proposed to enhance and modify the 2BT as in [Demir and Erturk, 2007] and [Nam-Joon et al, 2009]. All these algorithms save hardware costs and power consumption but are run at the risk of losing too much quality and hence they are classified as lossy block matching algorithms.

## 4.2 Lossless Block Matching Algorithms (Fast Full Search)

In this section lossless block matching algorithms will be discussed. A lossless algorithm attempts to improve the time to determine the matching MBl without affecting the quality of the FS. However, many studies have indicated that the quality of the produced compressed videos is not as good as that of the ones produced by FS [Huang et al., 2006]. Usually, the ideas of this category are borrowed from the fast search of Vector Quantisation (VQ) [Chang-Da and Gray, 1985].

### 4.2.1 Partial Distortion Elimination (PDE) Algorithm

This algorithm is the earliest algorithm in this category that has been widely used to reduce the computational complexity efficiently. It is employed in the FS algorithms in H.263 and H.264 [Kim Jong-Nam and Choi Tae-Sun, 2000; Lin Chen-Fu and Leou Jin-Jang, 2005]. It uses the halfway-stop technique in the BDM calculation. In other words,

the partial sum of matching distortion between current MBl and candidate MBl is stopped as soon as the matching distortion exceeds the current minimum distortion, meaning that the remaining computation is avoided. The conventional top-to-bottom kth partial SAD matching scan is determined as follows:

$$\sum_{i=1}^{k}\sum_{j=1}^{N}\left|C(i,j) - R\left(i+v_x, j+v_y\right)\right| \quad , \; k = 1, 2, \ldots N \tag{4.9}$$

where $N$ represents MBl size, C and R are the current and candidate MBls. If $k$ is smaller than $N$ and the summation exceeds the current $SAD_{min}$, then the remaining summation can quit and move to the next candidate MBl. Figure 4.17 shows the pseudo code of PDE.

Input

- Convert video to frames and convert to greyscale
- Read frames
- Let frame I be the reference frame of frame I+2
- Divided the frames into macroblocks of size N×N.
- Let the search window of maximum stepsize =P

Find the motion vectors for each macroblock by using PDE motion estimation

- For each macroblock MBl in frame I+2.
- compute the SAD between current MBl and the candidate macroblocks in centre of the search windows.
- Put the $SAD_{min}$=SAD.
- For the next search point R, let Sum=0
- compute SAD between the first line of MBl and R add the result to SUM
- While (SUM <= $SAD_{min}$) do
- compute SAD between the next line of MBl and R add the result to SUM
- end do
- let $SAD_{min}$ = MINIMUM ( $SAD_{min}$, SUM)
- go to the next search point and repeat the process till complete the search window points
- find the coordinates of the vector where $SAD_{min}$.

Output

- Motion vectors for all MBls ; Number of search points ; Time of process.

**Figure 4.15: Pseudo code of PDE**

The speed-up problem in this algorithm depends on: (1) **fast searching**, that is, how fast the global minimum in a given search range is detected; (2) **fast matching error**, that is, how to stop the calculation of the matching error early in the comparison process, which means finding the $k$ value in equation (4.9) faster to stop the partial sum.

The *fast searching* can be satisfied by applying the PDE algorithm with a spiral-ordered search starting at the centre of the search area since the best match location is usually centre-biased, as shown in section 4.1, then going outward in a spiral design. This was employed in Telenor's H.263 codec [Al-Mualla et al., 2002].

The *fast matching* can be satisfied by eliminating the average number of rows examined per MBl as well as the operations required. PDE employs SAD as a BDM to avoid more multiplication when calculating the matching error using MSE and others. Moreover, instead of the ordinary top-to-bottom matching scan, there are different scanning orders that improve performance of block matching. Kim et al. proposed various types of matching scan [Kim Jong-Nam and Choi Tae-Sun, 2000; Kim Jong-Nam et al., 2002; Jong-Nam et al., 2001] depending on the relationship between block matching error and the spatial complexity of the reference MBl, which is based on the concept of representative pixels. That is, the representative pixels are examined earlier than other pixels to detect the impossible candidates faster and reject them to obtain the reduction of computation in the block-matching algorithm. This algorithm is called *adaptive matching scan algorithm based on gradient magnitude*. It utilises four directions: top-to-bottom, bottom-to-top, left-to-right, right-to-left. It uses gradient magnitude to measure the image complexity due to performance and computational complexity. In general, the gradient points in the direction of the maximum increase of a function. The gradient magnitude *G* can be calculated as follows:

$$|G[f(x,y)]| \approx |G_x| + |G_y| \approx |f(x,y) - f(x+1,y)| + |f(x,y) - f(x,y+1)| \qquad (4.10)$$

The gradient magnitudes are calculated in four 8×8 sub-blocks of the candidate MBl, as shown in Figure 4.18, and then make a sum of gradient magnitudes in sub-blocks {(1),(2),(3),(4)}, which are in four cases: *(1)+(2), (3)+(4), (1)+(3), (2)+(4)*. The maximum value of these sums points to the direction of matching scan; for example, the direction of matching scan is from top-to-bottom when the sum of gradient magnitudes *(1)* and *(2)* is maximum, as shown in Figure 4.18, which describes this algorithm. The sub-block may be 4×4, i.e. there are 16 sub-blocks as in Jong-Nam et al. (2001). The matching scan order will also be according to the local complexity of the sub-block.

If the matching scan order is well arranged then the probability to eliminate the average number of rows examined increases.

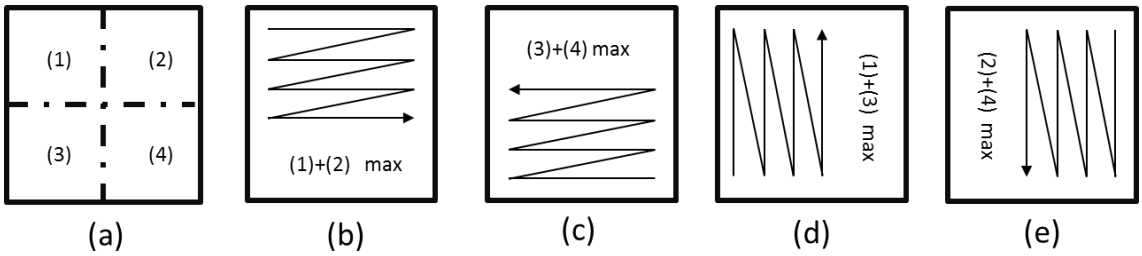


**Figure 4.18: Adaptive matching scan based on representative pixels: (a) gradient magnitudes of sub-block division, (b) (top-to-bottom) matching scan when (1)+(2) is maximum, (c) bottom-to top matching scan when (3)+(4) is maximum, (d) left-to right when (1)+(3) is maximum, (e) right-to left when (2)+(4) is maximum [Kim Jong-Nam and Choi Tae-Sun, 2000]**

However, these algorithms are not effective since decreasing the number of checking rows does not necessarily lead to enhancing the real time needed, because a lot of add/subtract operation is required per MBl to compute the gradient magnitude in order to decide the matching order, which may render it unsuitable for real-time video coding systems. Therefore, three low complexity scanning orders were proposed by Grecos et al. (2004) which show improvements of ¼ operation count ratio and show an increase in the speed-up ratio of 45 times on average as compared with an adaptive matching scan algorithm based on gradient magnitude. Unlike the *adaptive matching scan algorithm,* two of Grecos et al.'s algorithms – *spiralling inward scanning order* and *alternating spiralling inward scanning order* – used fixed order of SAD computation between current and reference MBls to eliminate unsuitable predictors in the reference frame. These algorithms are based on the idea that the sides of the MBl could represent the most information. Therefore, the representative pixels are examined earlier than other pixels without pre-processing, by computing the SAD value between pixels located on the sides of the squares of decreasing size inside the current and reference macroblocks, as shown in Figure 4.19, in order to reject impossible candidate predictors faster than the conventional top-to-bottom scan. The fixed direction scanning of the *spiralling inward scanning order* starts from top-horizontal and ends in left-vertical (Figure 4.19); it may increase computations since the complexity of candidate MBl could be in any vertical or horizontal sides. If a candidate MBl should be rejected on the basis of left-vertical SAD information then it has to wait until three sides of SAD computations are completed. For this reason, the *alternating spiralling inward scanning order* was

designed to reject the candidate MBl on the basis of horizontal and vertical SAD information, as shown in Figure 4.19 (b).

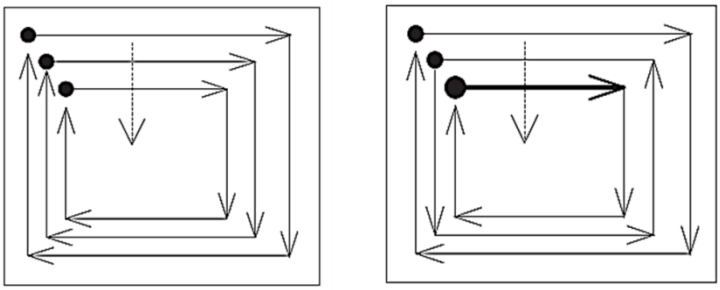


**Figure 4.19: (a) spiralling inward scanning order, (b) alternating spiralling inward scanning order [Grecos et al., 2004]**

The last algorithm of Grecos et al.'s, which is *horizontal/vertical scanning order*, utilises very limited pre-processing to avoid increasing the real time needed for computation and hence losing the benefit of computational reduction that happened with the *adaptive matching scan algorithm*. It determines the scanning order by examining only the SAD between the boundary rows and columns of the current and candidate MBls. The scanning direction will be the direction of the maximal SAD.

### 4.2.2 Successive Elimination Algorithm (SEA)

The SEA [Li and Salari, 1995] eliminates impossible candidate MBl by checking if the absolute difference between the summation of current MBl pixels and the summation of candidate MBl pixels is larger than the updated minimum SAD; if it is, then this candidate MBl should be rejected. Thus, a large part of unnecessary computation for impossible candidate MBls can be avoided. This algorithm is based on the triangular mathematical inequality given by:

$$\left|\sum_i x_i\right| \leq \sum_i |x_i| \tag{4.11}$$

where $x_i$ are arbitrary real numbers. Appling this inequality to the SAD achieves:

$$\left|\sum_{i=1}^{N}\sum_{j=1}^{N} C(i,j) - \sum_{i=1}^{N}\sum_{j=1}^{N} R(i+x,j+y)\right| = \left|\sum_{i=1}^{N}\sum_{j=1}^{N} C(i,j) - R(i+x,j+y)\right| \leq \sum_{i=1}^{N}\sum_{j=1}^{N} |C(i,j) - R(i+x,j+y)| \tag{4.12}$$

where $C(i,j)$ is the pixel value of current MBl at the position $(i,j)$ and R$(i+x,j+y)$ is the pixel value of reference frame with the vector $(x,y)$, which are within the search range $[-p,p]$. In other words, the previous inequality can be written as:

$$|SC - SR(x,y)| \leq SAD(C,R,x,y) \tag{4.13}$$

where $SC$ is the summation of current MBl and $SR(x,y)$is the summation of candidate MBl at the vector $(x,y)$. If $SAD_{min}(x_0,y_0)$ is the current updated minimum SAD at the search location $(x_0,y_0)$, then to achieve better match MBl at the location $(x,y)$ the SAD should be less than $SAD_{min}$ , that is $SAD(x,y) \leq SAD_{min}(x_0,y_0)$. This will substitute in (4.13) to get: $|SC - SR(x,y)| \leq SAD_{min}(x_0,y_0)$. This means that a MBl at location $(x,y)$ can be immediately skipped from the search if:

$$|SC - SR(x,y)| \geq SAD_{min}(x_0,y_0) \tag{4.14}$$

While, if the difference $|\mathrm{SC} - \mathrm{SR(x,y)}|$ is smaller than $SAD_{min}(x_0,y_0)$, then the candidate MBl is elected to calculate SAD between these two MBls and the new SAD becomes $SAD_{min}$. Since the candidate MBls are overlapping then the two horizontal neighbouring candidate MBls $SR(x,y)$ and $SR(x+1,y)$ are also overlapping and they share N−1 columns. Therefore, subtracting the sum of the first column of MBl $SR(x,y)$ and adding the sum of the last column in MBl $SR(x+1,y)$ will improve the block matching computation. A similar procedure can be used for vertical neighbouring candidate MBls.

Note that, similar to PDE, if the global minimum in a given search range is detected at the initial search, then SEA will be faster [Essannouni et al., 2006; Huang et al., 2006].

Various algorithms have been introduced to enhance SEA [Soo-Mok et al., 2000; Jung et al., 2002; Hwal-Suk et al., 2008; Man-Yau and Wan-Chi, 2006].

## 4.3 Chapter Summary

The FS algorithm is the simplest, but the most computation-intensive BMA, which exhaustively tests all the search locations for the best matching macroblock within the search area. Fast block matching algorithms have been developed to reduce the huge computational complexity of FS. Various methods and techniques have been proposed for fast BMA search; some of them have been adopted in video coding standards. Similar to all video and image compression techniques, fast block matching algorithms can be classified into lossy and lossless categories. Lossy BMAs can achieve more compression ratio and faster processes than FS by sacrificing the quality of the compressed video whereas lossless BMAs have the specific requirement to preserve the quality of the video. There are various lossy and lossless BMAs. Lossy BMAs can be classified into: Fixed Set of Search Patterns, Predictive Search, Hierarchical or Multiresolution Search, Subsampled Pixels on Matching Error Computation, and Bit-width Reduction, while lossless BMAs include PDE algorithm and SEA. Some of these categories have been used in this thesis to propose and develop novel techniques that enhance both lossless and lossy BMAs process, as will be discussed in Chapter 5.

# CHAPTER 5: ENHANCED FAST BLOCK MATCHING MOTION ESTIMATION

This chapter discusses novel techniques proposed to enhance both lossless block matching algorithms and lossy block matching algorithms processes. The general motion in any video frame is usually coherent; that is, if the macroblocks around the current macroblock move in a particular direction then there is a high probability that the current macroblock will also have the same direction. Therefore, the research work in this thesis used the mean value of two motion vectors of the previous neighbouring macroblocks to predict the first step of the search process in different techniques depending on the algorithm. The neighbouring macroblocks are chosen as the top and left macroblocks.

As shown in the previous chapter, the fast full search Partial Distortion Elimination (PDE) algorithm has been widely used to reduce the computational complexity efficiently. It utilises a halfway-stop technique in the Block Distortion Measure (BDM) calculation. The performance problem in this algorithm depends on fast searching; that is, how fast global minimum in a given search range is detected as well as how fast matching error can calculate the matching error on a candidate Macroblock (MBl). The novel proposed techniques attempt to capture the global minimum in the first search by using the predictor Motion Vectors (MVs); therefore, all the proposed algorithms will use the PDE to enhance and improve the time needed for processing. Moreover, PDE technique was applied to the existing fast block matching algorithm to improve the time needed for processing without affecting the quality.

This chapter is divided into four sections: section 1 introduces the novel method of lossless block matching algorithms, which is called Fast Computations of Full Search Block Matching Motion Estimation (FCsFS). The purpose of this method is to decrease the computational time required to determine the matching macroblock of the full search while keeping the resolution of the predicted frames the same as the full search. This is performed by using the motion vector of two previous neighbouring MBls – the up and left – to determine the search window using the mean values. The correlation between current and neighbouring MBls increase the probability that the global

minimum is detected in the new search window, therefore applying the PDE algorithm will speed up the search processing.

Sections 2 and 3 propose two novel techniques of lossy block matching algorithms. These two novel methods use three types of fast block matching algorithm: fixed set of search patterns, predictive search and PDE algorithm. The aim of these algorithms is to improve the fast block matching motion estimation by decreasing both computational time required to determine the matching macroblock and the residual prediction error between current frames and compensated frames. The first algorithm is called Mean Predictive Block Matching (MPBM) and the second algorithm is called Enhanced Mean Predictive Block Matching Algorithm (EMPBM). The chapter summary is provided in section 4.

## 5.1 Fast Computations of Full Search (FCsFS) Block Matching Motion Estimation

As seen in Chapter 4, various scanning orders in both searching and matching have improved performance in the full search block matching algorithm that uses the halfway-stop technique. Some of these have used various types of matching scan between current and candidate MBls, depending on the spatial complexity of the reference MBl [Kim Jong-Nam and Choi Tae-Sun, 2000; Kim Jong-Nam et al., 2002; Jong-Nam et al., 2001]. It has been proven that some of these algorithms are not effective since decreasing the number of checking rows does not necessarily lead to enhancing the real time needed for processing a full search, because many add/subtract operations are required per MBl to compute the gradient magnitude of MBls in order to decide the matching order, which refers to a state that is unsuitable for real-time video coding systems.

The proposed algorithm FCsFS is one of the lossless block matching algorithms that attempts to avoid this problem. The purpose of the proposed method is to decrease the computational time required to determine the matching macroblocks of full search while keeping the resolution of the predicted frames the same as the resolution obtained from full search. This is performed by using two predictors, which are the motion vector of

the two previous neighbouring MBls, the up ($MV_A$) and the left ($MV_L$), as shown in Figure 5.1.

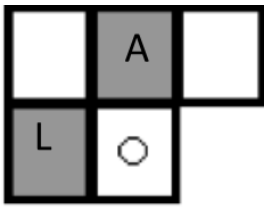


**Figure 5.1: Position of the two predictive macroblocks**

The purpose of using these predictors is to get the global matching MBl faster than using a single previous neighbour. Furthermore, the selection of these predictors will avoid unnecessary computations arising from choosing three previous neighbouring MBls. The neighbours may move to different directions; therefore, these MVs are used to determine the new search window depending on the mean of its components. That is, the search range of the new search windows will be the mean of $x$-components and $y$-components of $MV_A$ and $MV_L$, respectively, as follows:

$$w = round\left(abs\left(\frac{x_A + x_L}{2}\right)\right)$$
$$h = round\left(abs\left(\frac{y_A + y_L}{2}\right)\right) \quad (5.1)$$

where : $x_A$ and $x_L$ are the $x$-components of $MV_A$ and $MV_L$, respectively.

$y_A$ and $y_L$ are the $y$-components of $MV_A$ and $MV_L$, respectively.

The current MBls are searched for the reference image using 'first the search range of $\pm w$ in the $x$-axis and $\pm h$ in the $y$-axis' instead of using the fixed search range of $\pm p$ in both of them, as seen in Figure 5.2.

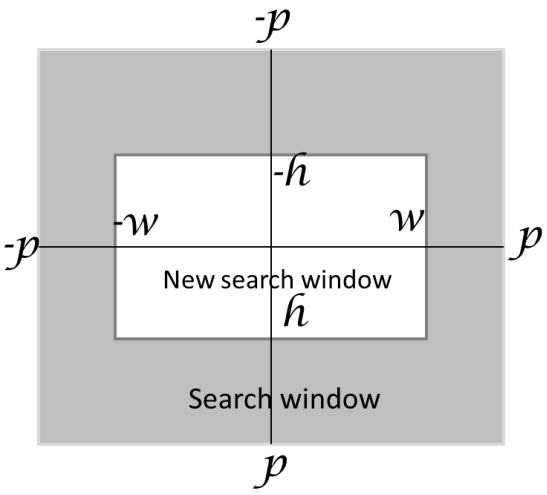


**Figure 5.2: The default search window of maximum step size p and the new search window of maximum step size h in the x-axis and w in the y-axis**

Meanwhile, there is a high correlation between neighbouring MBls therefore the global matching MBl has a probability to be in the new search window of maximum step size w in the x-axis and h in the y-axis. Hence, applying the PDE algorithm will speed up the search process. This search will stop if the error between the matching MBl obtained from this search window range and the current MBl is less than the threshold value (N×N, for the MBlsize=N). Then the rest of the default search window will not need to be completed. Otherwise, the rest of the default search window will be completed. The threshold will be computed as the number of pixels of the MBl since one degree difference for each pixel will not affect the matching MBls. Figure 5.3 shows the block diagram of the proposed algorithm FCFS and its pseudo code is illustrated in Figure 5.4.

The simulation results indicate that the FCsFS technique reduces the search time of the macroblock matching, and keeps the resolution same as full search.

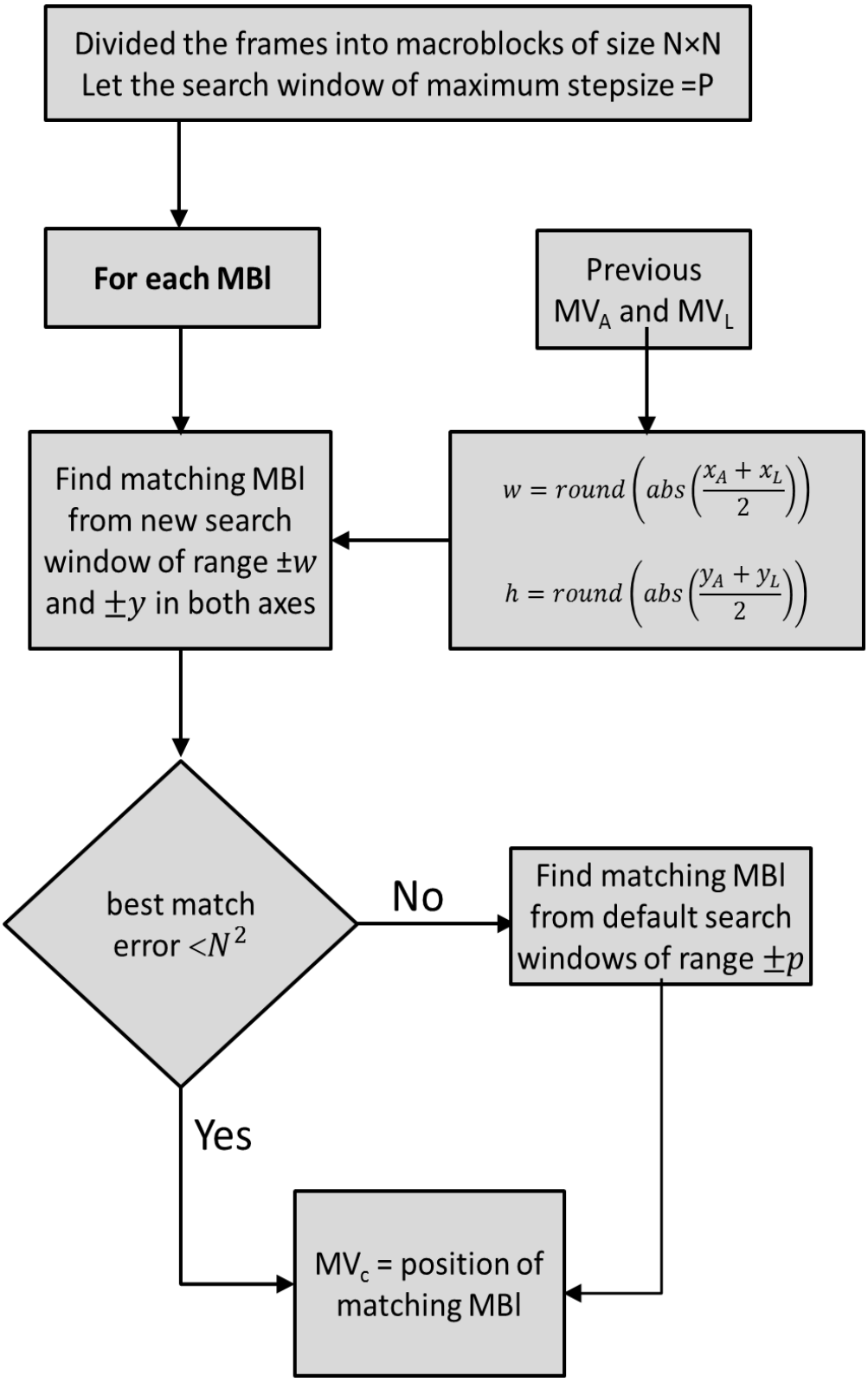


**Figure 5.3: The diagram of the proposed FCsFS to get the motion vector of the current MBl**

Input

- Convert video to frames and convert to greyscale
- Read frames
- Let frame I be the reference frame of frame I+2
- Divided the frames into macroblocks of size N×N.
- Let the search window of maximum stepsize =P

Find the motion vectors for each macroblock by using FCsFS motion estimation

- For each macroblock MBl in frame I+2.
- compute the SAD between current MBl and the candidate macroblocks in centre of the search windows.
- Put the $SAD_{min}$=SAD.
- if MBl is on the top-left corner then PDE will be apply
- else for the previous above MV $(x_A, y_A)$ and the left MV $(x_L, y_L)$, let $w = round(abs((x_A + x_L)/2))$, $h = round(abs((y_A + y_L)/2))$
- let the new search window of maximum stepsize= w in the $x$-axis and h in the $y$-axis
- For the next search point R in the new search window, let Sum=0
- compute SAD between the first line of the pixels for MBl and R add the result to SUM
- While (SUM <= $SAD_{min}$) do
- compute SAD between the next line of the pixels MBl and R add the result on SUM
- end do
- let $SAD_{min}$ = MINIMUM ( $SAD_{min}$, SUM)
- go to the next search point in the new window and repeat the process till complete the search points of the new window.
- If $SAD_{min} <= N^2$ then the search is complete
- Else the rest of the default search window will be completed by the same way
- find the coordinates of the vector that where $SAD_{min}$.

Output

- Motion vectors for all MBls ; Number of search points ; Time of process.

**Figure 5.4: Pseudo code of FCsFS**

## 5.2 Mean Predictive Block Matching (MPBM)

In this section, a novel algorithm in lossy block matching algorithms is proposed [Ahmed et al., 2011b]. The novel technique has improved the fast block matching algorithms by combining three types: predictive search technique, fixed set of search patterns, and PDE algorithm.

The first type, predictive search technique, utilises the motion information of two previous spatial neighbouring MBls, left and above, as shown in Figure 5.1, in order to form an initial estimate of current MV. As shown in the previous section, since the motion of neighbouring MBls is coherent then using these predictors increase the probability of determining the global matching MBl by avoiding different directions motion that regards to use one previous neighbour. Moreover these two predictors will avoid unnecessary computations required from selecting three previous neighbouring MBls. The maximum of the mean $x$ and $y$ components for the two predictor MVs will be used to determine the step size.

The second type, fixed set of search patterns, as in ARPS [Nie and Ma, 2002] and DS techniques [Shan and Kai-Kuang, 1997], uses two different types of fixed patterns, the Large Search Pattern (LSP) and the Small Search Pattern (SSP), as shown in Chapter 4. Moreover, the first step search includes the MVs of two previous neighbouring MBls with the LSP. The step size will be used to determine the position of the LSP in the first step. Therefore, seven positions are examined in this step. To avoid unnecessary computations, this technique utilises a pre-defined threshold value for the error between the current macroblock and the matching macroblock that has been determined from the first step. If the error is less than the threshold value ($MBlsize \times MBlsize$), the SSP will not be needed, and hence the computations will be reduced.

The last type, the PDE algorithm, has been used to improve the computation time. It is used to stop the partial sum of matching distortion between current macroblock and candidate macroblock as soon as the matching distortion exceeds the current minimum distortion.

The following explains the steps involved in the proposed technique:

**Step 1:** compute the sum of absolute differences ($SAD_{centre}$) between the current MBl of size N×N and the MBl at the same location in the reference frame (i.e. the centre of the current search window). In this case, if the sum of absolute differences ($SAD_{centre}$) is less than a pre-defined threshold value ($N \log_2 N$), this means that there will be no motion and the search process will be terminated.

**Step 2:** if the macroblock MBl is in the high left corner, then only 5 points of LSP {(±stepsize,0), (0, ± stepsize), (0,0)} will be searched first; otherwise, the above motion vector ($MV_A$) and left motion vector ($MV_L$) will be added to the first search and used to predicate the step size as follows:

$$L_x = round\left(abs\left(\frac{x_A + x_L}{2}\right)\right)$$

$$L_y = round\left(abs\left(\frac{y_A + y_L}{2}\right)\right) \tag{5.2}$$

where : $x_A$ and $x_L$ are the $x$-components of $MV_A$ and $MV_L$, respectively.

$y_A$ and $y_L$ are the $y$-components of $MV_A$ and $MV_L$, respectively.

In this case step size = max{ $L_x$ , $L_y$}.

**Step 3:** the matching macroblock is then searched using the PDE algorithm within LSP points {(±stepsize,0), (0, ± stepsize), (0,0)} and the following vectors {($MV_A$), ($MV_L$ )}, as shown in Figure 5.5. That is, if the current SAD value exceeds the previous SAD then the computation will be stopped and will jump to the next position; otherwise, all pixels will be completed and go to the next search point repeat the same process till complete all LSP points.

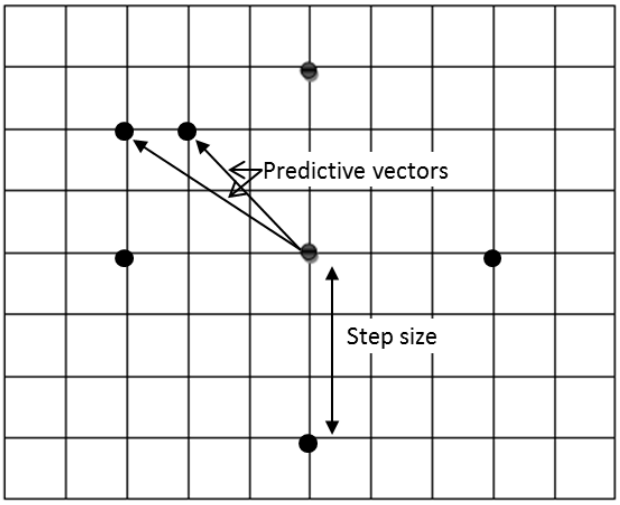


**Figure 5.5: The solid circle points (●) are the first step search in MPBM, which is the Large Search Pattern (LSP) and the two predictive vectors**

**Step 4:** if the error between the current MBl and the matching MBl from previous search pattern in step 3 is less than the pre-defined threshold ($N \times N$), value then the process will be stopped and the matching MBl will give the motion vector. Otherwise, the position of the matching macroblock in step 3 becomes the centre of the new search and the SSP of four points {(±1, 0), (0, ±1)} will be checked as shown in Figure 4.8 in the ARPS algorithm. If the matching macroblock stays in the centre then the computation will be ended; otherwise, the same process will be repeated until the matching macroblock reaches the centre. The matching centre will give the motion vector.

Figure 5.6 illustrates a block diagram of the proposed fast block matching algorithm MPBM, while Figure 5.7 shows the pseudo code of MPBM. The simulation results indicated that the ratio between PSNR of compensated frames generated by the novel algorithms and the time needed for computation gives better results in comparison to the benchmarked algorithms.

Since the initial search depends on two neighbouring MBls, therefore the first step search has a probability of containing the global minimum MBl and hence the time should be enhanced. Also, the MPBM algorithm does not have a limited number of search steps. Therefore, for all motion activity video sequences, MPBM algorithm dose

not trapped into a local minimum point and the global minimum can be founded with more accurately than other algorithms.

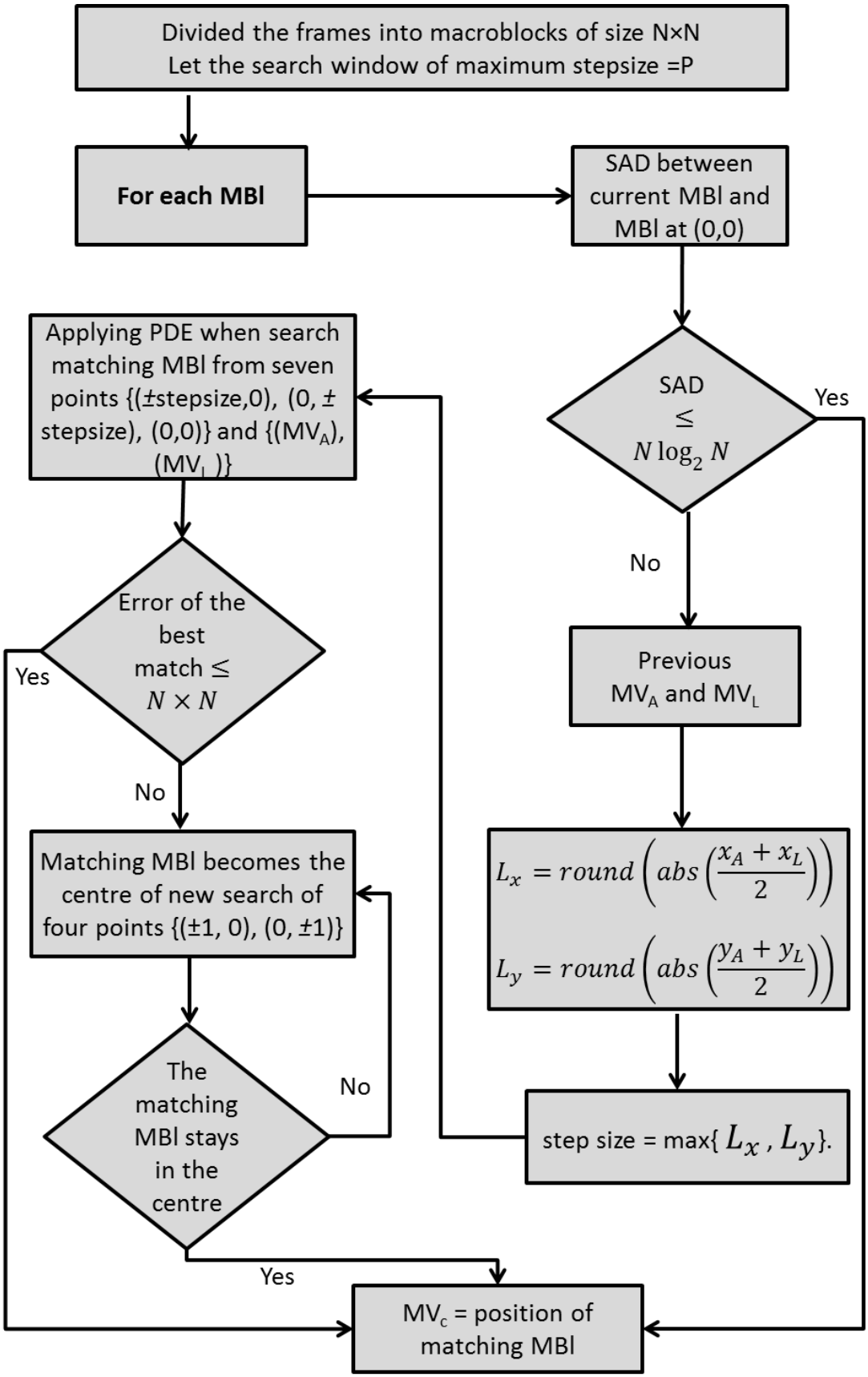


**Figure 5.6: The diagram of the MPBM algorithm**

Input

- Convert video to frames and convert to greyscale
- Read frames
- Let frame I be the reference frame of frame I+2
- Divided the frames into macroblocks of size N×N.
- Let the search window of maximum stepsize =P

Find the motion vectors for each macroblock by using MPBM motion estimation

- For each macroblock MBl in frame I+2.
- compute the SAD between current MBl and the candidate macroblocks in centre of the search windows.
- If SAD < = $N \log_2 N$ then the centre will be the matching MBl and the search is stop
- Else put the $SAD_{min}$=SAD.
- if MBl is on the top-left corner then let stepsize=2 and the search points will be the large search pattern (LSP) of only five points {(±stepsize,0), (0, ±stepsize), (0,0)}
- else add the previous above MV $(x_A, y_A)$ and the left MV $(x_L, y_L)$, to the LSP and let $L_x = round(abs((x_A + x_L)/2))$, $L_y = round(abs((y_A + y_L)/2))$, let stepsize = max{ $L_x$ , $L_y$}
- For the first search point R, let Sum=0
- compute SAD between the first line pixels of MBl and R add the result to SUM
- While (SUM < = $SAD_{min}$) do
- compute SAD between the next line pixels of MBl and R add the result on SUM
- end do
- let $SAD_{min}$ = MINIMUM ( $SAD_{min}$, SUM)
- go to the next search point in the search pattern and repeat the process till complete the search points.
- make sure the position of the candidate macroblock is not out of the frame.
- If $SAD_{min}$ < = $N^2$ then the search is complete
- Else the position of the $SAD_{min}$ becomes the centre of the new search pattern which is small search pattern SSP of 4 points {(±1, 0), (0, ±1)}.
- Make sure that don't calculate the same points again that were calculate in the previous search.
- make sure the position of the candidate macroblock is not out of the frame.
- If the position not at the centre of SSP then let it be the centre of new search and repeat SSP till it become the centre of the SSP
- find the coordinates of the vector that where $SAD_{min}$.

Output

- Motion vectors for all MBls ; Number of search points ; Time of process.

**Figure 5.7: Pseudo code of MPBM**

## 5.3 Enhanced Mean Predictive Block Matching Algorithm (EMPBM) Using Edge Detection

Enhanced Mean Predictive Block Matching Algorithm is a new technique proposed to decrease the computations of the previous fast block matching algorithm Mean Predictive Block Matching algorithm [Ahmed et al., 2012]. In order to find the matching macroblock for the current macroblock from the previous frame, this technique classifies the current macroblock into shade and edge. The shade macroblock has a probability to move in the same direction as its neighbouring macroblocks. This will lead to search only the motion vectors of the neighbouring macroblocks and ignore other motion vectors that were utilised in the first search step of the Mean Predictive Block Matching algorithm. For edge macroblock, the proposed technique will use the same approach that was used in the Mean Predictive Block Matching algorithm.

Edge information can be described as a straight line across the macroblock with a sharp change of intensity in the spatial domain [Ali Al-Fayadh, 2009]. A fixed small size *4×4* macroblock is utilised to achieve good subjective quality. Therefore, this technique can be useful for small MBls in variable block-size motion estimation. In order to avoid more computations in the existing edge detection methods, the absolute value approach has been used. The idea is to use the absolute value between the summation values of the vertical halves of the macroblock and the absolute value of the difference between the summation values of the horizontal halves, as shown in Figure 5.8.

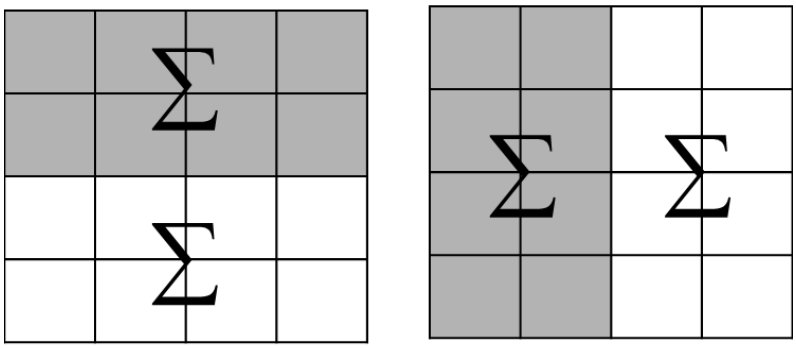


**Figure 5.8: Vertical halves and horizontal halves for 4×4 MBls**

When the sum of theses difference is less than a threshold value $T$ [ $(T = 4 * N^2); N = 4$], the macroblock is classified as shade; otherwise, the macroblock will be classified as edge, as follows:

Let $B = \{b_{ij};\ 1 \leq i, j \leq 4\}$ represent a 4×4 frame macroblock. In this case, $b_{ij}$is a grey level pixel value corresponding to position (*i*, *j*) of row *i* and column j in the image block *B*. The discrete gradients of the macroblock *B* in the x and in the y directions are determined as follows:

$$G_x = \left| \sum_{i=1}^{2} \sum_{j=1}^{4} b_{ij} - \sum_{i=3}^{4} \sum_{j=1}^{4} b_{ij} \right| \tag{5.3}$$

$$G_y = \left| \sum_{i=1}^{4} \sum_{j=1}^{2} b_{ij} - \sum_{i=1}^{4} \sum_{j=3}^{4} b_{ij} \right|$$

The gradient magnitude is defined by:

$$G = G_x + G_y \tag{5.4}$$

If the gradient magnitude *G* in Equation (5.3) of the macroblock *B* is smaller than threshold $T$ [ $T = (2N)^2; N = 4$], then it is considered that the macroblock contains no significant gradient and it is classified as a shade macroblock; otherwise, it will be classified as an edge macroblock.

The shade macroblock has a high probability to move in the same direction of its neighbouring macroblock. This fact has been used in MPBM to decrease the search points as follows:

**Step 1:** as in MPBM, compute the sum of absolute differences (SADcentre) between the current macroblock and the macroblock at the same location in the reference frame (i.e. the centre of the current search window). In this case, if the sum of absolute differences (SADcentre) is less than a pre-defined threshold value ($N \log_2 N$; *N=4*) this means that there will be no motion and the process will be determined.

**Step 2:** use gradient magnitude to classify the current MBl. For shade macroblocks, only the above motion vector ($MV_A$) and left motion vector ($MV_L$) will be tested, while for the edge macroblock the LSP search points will be tested, as shown in Figure 5.5.

**Step 3:** the PDE algorithm will be applied. That is, if the current SAD value exceeds the previous SAD then the computation will be stopped.

**Step 4:** if the error of the matching macroblock from previous steps is less than the pre-defined threshold value then the process will be stopped and the matching macroblock will give the motion vector. Otherwise, the matching macroblock will become the centre of the new search. If the matching macroblock stays in the centre then the computation will be ended; otherwise, the same process will be repeated until the matching macroblock reaches the centre.

The simulation results of this algorithm show improvement in computational complexity compared with the MPBM while trying to keep or enhance the resolution of compensated frames.

Figure 5.9 shows the block diagram of the proposed EMPBM algorithm and its pseudo code is illustrated in Figure 5.10.

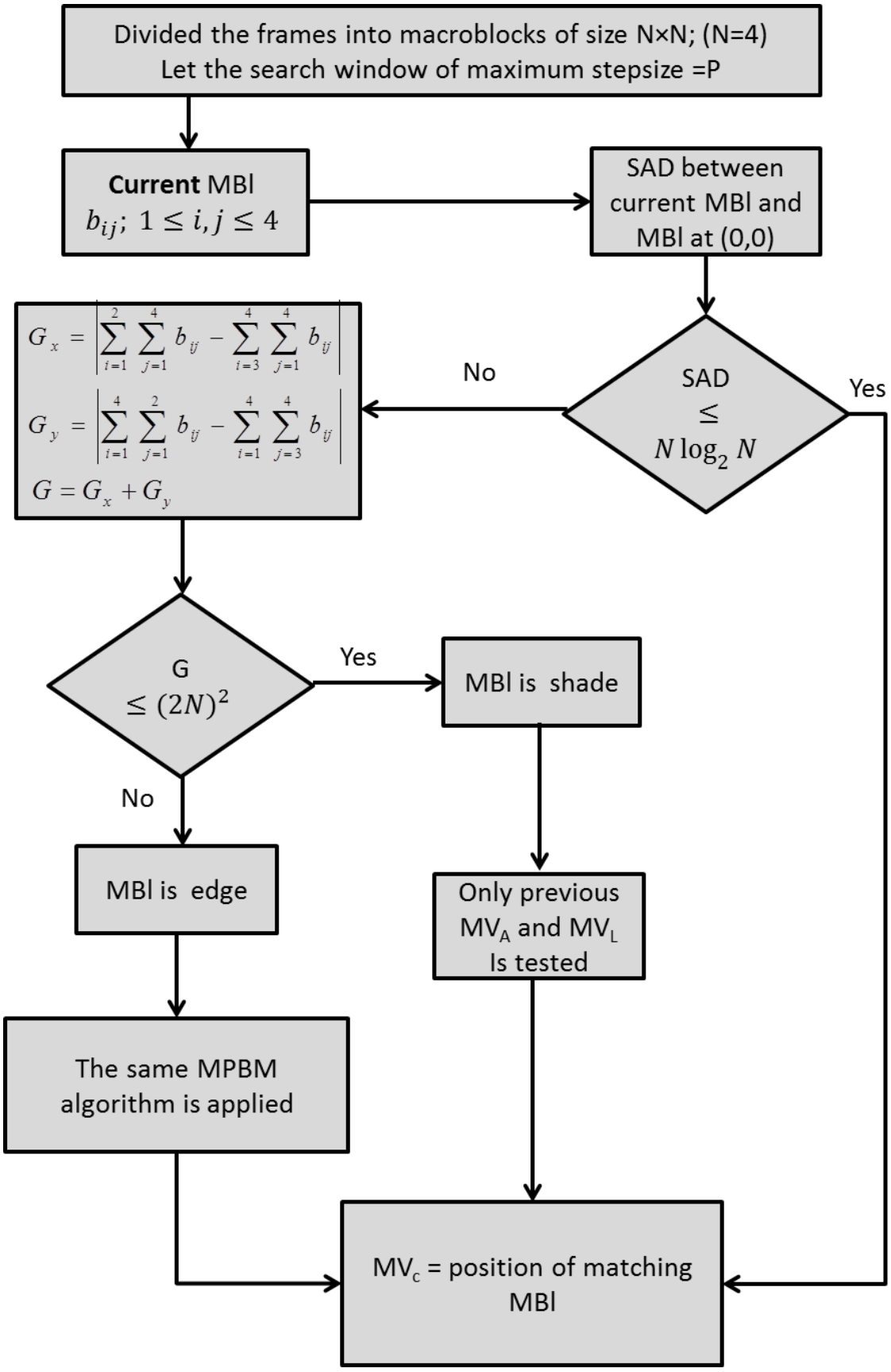


**Figure 5.9: The diagram of the EMPBM algorithm**

Input

- Convert video to frames and convert to greyscale
- Read frames
- Let frame I be the reference frame of frame I+2
- Divided the frames into macroblocks of size N×N; N=4.
- Let the search window of maximum stepsize =P

Find the motion vectors for each macroblock by using EMPBM motion estimation

- For each macroblock MBl in frame I+2.
- Compute the SAD between current MBl and the candidate macroblocks in centre of the search windows.
- If SAD <= $N \log_2 N$ then the centre will be the matching MBl and the search is stop.
- Else if MBl is on the top-left corner then let stepsize=2 and the search points will be the large search pattern (LSP) of only five points {(±stepsize,0), (0, ±stepsize), (0,0)}
- Else Put the $SAD_{min}$=SAD, the gradient magnitude G of current MBl $B = \{b_{ij};\ 1 <= i,j <= 4\}$, Let $G_y = abs(sum(sum(bij(\mathrm{i} = 1{:}2, \mathrm{j} = 1{:}4))) - sum(sum(bij(\mathrm{i} = 3{:}4, \mathrm{j} = 1{:}4))))$; Let $G_y = abs(sum(sum(bij(i = 1{:}4, j = 1{:}2\ ))) - sum(sum(bij(i = 1{:}4, j = 3{:}4))))$; Let $G = G_x + G_y$
- If $G <= (2 * N)^2$; $N = 4$, then B is shade then then only previous above MV $(x_A, y_A)$ and the left MV $(x_L, y_L)$ will be candidate and the search is stop.
- else add the previous above MV $(x_A, y_A)$ and the left MV $(x_L, y_L)$, to the LSP and let $L_x = round(abs((x_A + x_L)/2))$, $L_y = round(abs((y_A + y_L)/2))$, let stepsize = max{ $L_x$ , $L_y$}
- For the first search point R, let Sum=0
- compute SAD between the first line pixels of B and R add the result to SUM
- While (SUM <= $SAD_{min}$) do
- compute SAD between the next line pixels of B and R add the result on SUM
- end do.
- let $SAD_{min}$ = MINIMUM ( $SAD_{min}$, SUM)
- go to the next search point in the search pattern and repeat the process till complete the search points.
- make sure the position of the candidate macroblock is not out of the frame.
- If $SAD_{min}$ <= $N^2$ then the search is complete
- Else the position of the $SAD_{min}$ becomes the centre of the new search pattern which is small search pattern SSP of 4 points {(±1, 0), (0, ±1)}.
- Make sure that don't calculate the same points again that were calculate in the previous search.
- make sure the position of the candidate macroblock is not out of the frame.
- If the position not at the centre of SSP then let it be the centre of new search and repeat SSP till it become the centre of the SSP
- find the coordinates of the vector that where $SAD_{min}$.

Output

- Motion vectors for all MBls ; Number of search points ; Time of process.

**Figure 5.10: Pseudo code of EMPBM**

## 5.4 Chapter Summary

This chapter presented novel techniques in both the lossless block matching algorithms process and lossy block matching algorithms process. The improvements in these processes were achieved by:

(1) using two previous neighbours, the above and left MBls, to predict the first step of the search process and to determine the global matching MBl faster than using one previous neighbour. Furthermore, avoiding unnecessary computations comes from choosing three previous neighbouring MBls. It all goes back to the fact that these two neighbours MBl may be moved to different directions; therefore, the proposed algorithms will use the mean of the MVs as a starting point with a different style depending on the algorithm.

(2) using the PDE algorithm enhanced and improved the time needed for processing since the proposed techniques try to catch the global minimum MBl in the first search by using the predictor MVs which improve the performance of PDE.

The proposed technique of lossless block matching algorithms is Fast Computations of Full Search Block Matching Motion Estimation, which decreases the computational time required to determine the matching macroblock of the full search while keeping the resolution of the predicted frames the same as the one obtained from full search. This is determined by using the predictive search technique to predict the new search window and the partial distortion elimination algorithm to decrease the search time. It also completes the original search windows if the error between the matching MBl from the new search windows and the current MBl is not small enough.

The improvement of lossy block matching algorithms was illustrated by two other proposed techniques: Mean Predictive Block Matching (MPBM) and Enhanced Mean Predictive Block Matching Algorithm (EMPBM). The first technique combine three types of fast block matching algorithm: predictive search technique, fixed set of search patterns, and partial distortion elimination algorithm, while the second technique is trying to improve the first one by classifying the current macroblock into *shade* and *edge*. The shade macroblock has a high probability to move in the same direction as its neighbouring macroblocks. This will lead to test only the motion vectors of the neighbouring macroblocks and ignore other motion vectors that were utilised in the first

search step of the MPBM algorithm. For the edge macroblock, the proposed technique will use the same approach that was used in the MPBM algorithm.

# CHAPTER 6: EXPERIMENTAL RESULTS AND ANALYSIS

This chapter presents the experimental results for the proposed algorithms to enhance fast block matching estimation.

The performance of the proposed algorithms is evaluated using speed of search to get the matching Macroblocks (MBls) and the efficiency of keeping the RPE between the current frame and its prediction the same as for the full search technique. The results are benchmarked with standard fast block matching algorithms. Table 6.1 shows a brief comparison between these algorithms.

**Table 6.1: Comparison between the novel algorithms and the standard block matching algorithms.**

| Motion Estimation Algorithm | Complexity | Advantage | Disadvantage |
|---|---|---|---|
| **ES** | $(2p+1)^2$ | Best picture quality | Very high computational cost |
| **PDE** | $(2p+1)^2$ | Same as ES quality with less computation than ES | Very high computational cost |
| **Proposed FCsFS** | | Picture quality is similar to ES and less computation than PDE | Very high computational cost |
| **TSS** | $[1+8*\log_2(p+1)]$ | Less complexity, Than ES | can't detect small motion |
| **NTSS** | $[1+8*\log_2(p+1)]$ +8 | For small motion video the complexity is less than TSS | For high motion video the complexity is higher than TSS |
| **SESTSS** | Maximum $[6*\log_2(p+1)]$ | Less complexity than TSS | The quality can be reduce in some videos |
| **DS** | 9+4n | For small and high motion video the complexity is less than NTSS | The quality is not as NTSS |
| **ARPS** | 6+ 4n | Similar quality as DS and less complexity | Need memory to store previous predicted MBl |
| **Proposed MPBM** | 7+4n | Better quality than ARPS and less complexity | Need memory to store two previous predicted MBls |
| **Proposed EMPBM** | For shade MBl : 3<br>For edge MBl : 7+4n | Less complexity than MPBM | The size MBl should be small for edge detection process |

This chapter is organised as follows. Section 6.1 includes selected video sequences, benchmarked algorithms, measure tools, software and hardware - those are used to determine the performance of the proposed techniques. Section 6.2 illustrates the experimental result and analysis of the novel technique for the lossless block matching algorithms' process, (FCsFS). The simulation results and analysis for the novel techniques of lossy block matching algorithms' process are shown in sections 6.3, (MPBM) and section 6.4, (EMPBM). While section 6.5 discusses the results of applying the PDE technique to the Diamond Search, which is called Enhanced Diamond Search (EDS), and New Three Step Search, which is called Enhanced New Three Step Search (ENTSS), hence compares the results with MPBM, section 6.4 gives the simulation results of the Enhanced Mean Predictive Block Matching Algorithm (EMPBM). The chapter summary will be provided in section 6.6.

## 6.1 Framework Evaluation

The performance of the proposed techniques is benchmarked with the well-known standard algorithms. The FCsFS algorithm is benchmarked with FS and PDE. The two novel techniques in the lossy block matching algorithms' process MPBM and EMPBM algorithms are evaluated by benchmarking with the FS, DS, NTSS, FSS, SESTSS, and ARPS, whose search strategies and patterns are described in Chapter 4. PDEDS and PDENTSS are compared with DS and NTSS respectively.

The simulation results of these techniques are determined using Matlab 2009 software with an 'Intel (R) Core(TM)i3 CPU M330@2.13 GHz 2.13 GHz' process.

The experimental results of all proposed techniques were conducted on the luminance component for 50 frames of six popular video sequences from [National Science Foundation, 2011]. Three of them are CIF format (Common Intermediate Format) video sequences (i.e. 352×288 pixels, 30fps), which are "News" (Figure 6.1), "Stefan" (Figure 6.2), and "Coastguard" (Figure 6.3). The remaining videos are QCIF format (Quarter-CIF) video sequences (i.e. 176×144 pixels, 30fps), which are "Claire" ( Figure 6.4), "Akiyo" (Figure 6.5), and "Carphone" (Figure 6.6).

These selected video sequences have various motion activities. "Akiyo" and "Claire" have low motion activity; "News" and "Carphone" have medium motion activity; while

"Coastguard" and "Stefan" have high motion activity. These video sequences have been used in this thesis to study the performance of the proposed techniques.

To avoid unreasonable results that can be obtained from the high correlation between successive frames, all the proposed and benchmarked algorithms have used two-steps backward frame as a reference frame, which means that if the current frame is I then the reference frame is *I-2*.

Four measuring tools have been used to determine the performance of the proposed techniques. Two of them are used to measure the speed search of these algorithms, which are *the processing average time* in seconds and *the average number* of search points required to get the motion vectors. In order to assess the quality of the predicted frames or compensated frames generated by the proposed algorithms, two measuring tools are used, which are the MSE and the PSNR:

$$MSE(f,\hat{f}) = \frac{1}{M \times N}\sum_{i=1}^{M}\sum_{j=1}^{N}\left(f(i,j) - \hat{f}(i,j)\right)^2 \tag{6.1}$$

where M and N are the horizontal and vertical dimensions of the frame, respectively, and $f(i,j)$ and $\hat{f}(i,j)$ are the pixels values at location $(i,j)$ of the original and predicted frames, respectively.

And

$$PSNR((f,\hat{f}) = 10\ log_{10}\left(\frac{(f_{max})^2}{MSE}\right) \tag{6.2}$$

where $f_{max}$ is the maximum possible pixel value which is used here: 255 for an 8-bit resolution.

It should be noted that the MAD and PSNR between the original and the compensated frames are measured by computing the MAD and PSNR for each frame with their compensated frames separately and then calculating their arithmetic mean.

Moreover, the statistical figures those give for frame-by-frame comparison of PSNR, MAD and number of search points per MBl, are illustrated using selected frames of the

video sequences for all proposed algorithms to be clear figures instead of using the whole 50 frames.


1st frame 50th frame

**Figure 6.1: News (CIF)**


1st frame 50th frame

**Figure 6.2: Stefan (CIF)**

1st frame 50th frame

**Figure 6.3: Coastguard (CIF)**

1st frame 50th frame

**Figure 6.4: Claire (QCIF)**

1st frame 50th frame

**Figure 6.5: Akiya (QCIF)**

1st frame 50th frame

**Figure 6.6: Carphone (QCIF)**

## 6.2 Simulation Results of FCFS

Simulations were carried out to test the performance of the proposed FCFS. The size of each MBl will be *16×16* for all the selected video sequences and the current MBls are searched for the reference image using a search range of ±7 for the original search windows.

The simulation results for FCsFS are benchmarked with the simulation results for FS and PDE. The computational complexity is measured using: (1) the average number of search points required to get each motion vector, as shown in Table 6.2 and (2) the time required for these algorithms, since applying PDE improves the computational time without access to the number of search points; therefore, the time needed for processing has been used to evaluate the performance of the proposed algorithm, which is shown in. Table 6.4 and Table 6.5 show the simulation results for the mean MAD and the mean PSNR respectively, for the proposed and benchmarked techniques.

**Table 6.2: Average number of search points per MBl of size 16 ×16**

| Sequence | Format | FS | PDE | FCFS |
|---|---|---|---|---|
| **Claire** | QCIF | 184.56 | 184.6 | 48.98 |
| **Akiyo** | QCIF | 184.56 | 184.6 | 46.2 |
| **Carphone** | QCIF | 184.56 | 184.6 | 170.2 |
| **News** | CIF | 204.28 | 204.3 | 121.6 |
| **Stefan** | CIF | 204.28 | 204.3 | 204.3 |
| **Coastguard** | CIF | 204.28 | 204.3 | 204.3 |

**Table 6.3: The simulation results of the average time in seconds needed to process 50 frames**

| Sequence | Format | FS | PDE | FCsFS |
|---|---|---|---|---|
| **Claire** | QCIF | 0.351 | 0.18 | 0.06 |
| **Akiyo** | QCIF | 0.334 | 0.11 | 0.01 |
| **Carphone** | QCIF | 0.336 | 0.18 | 0.15 |
| **News** | CIF | 1.492 | 0.65 | 0.38 |
| **Stefan** | CIF | 1.464 | 1.09 | 0.88 |
| **Coastguard** | CIF | 1.485 | 1.19 | 1.03 |

**Table 6.4: The simulation results of mean MAD for 50 frames**

| Sequence | Format | ES | PDE | FCsFS |
|---|---|---|---|---|
| **Claire** | QCIF | 1.13 | 1.13 | 1.13 |
| **Akiyo** | QCIF | 0.81 | 0.81 | 0.81 |
| **Carphone** | QCIF | 3.42 | 3.42 | 3.42 |
| **News** | CIF | 1.59 | 1.59 | 1.59 |
| **Stefan** | CIF | 11.6 | 11.6 | 11.6 |
| **Coastguard** | CIF | 7.91 | 7.91 | 7.91 |

**Table 6.5: The simulation results of mean PSNR for 50 frames**

| Sequence | Format | FS | PDE | FCsFS |
|---|---|---|---|---|
| **Claire** | QCIF | 38.94 | 38.94 | 38.94 |
| **Akiyo** | QCIF | 39.61 | 39.61 | 39.61 |
| **Carphone** | QCIF | 30.82 | 30.82 | 30.81 |
| **News** | CIF | 33.48 | 33.48 | 33.47 |
| **Stefan** | CIF | 22.16 | 22.16 | 22.16 |
| **Coastguard** | CIF | 26.19 | 26.19 | 26.19 |

All codes are implemented in Matlab, hence it takes a long time to process the condition statements. Nevertheless, the experimental results show that the proposed technique reduces the search time of the macroblock matching, while keeping the resolution of the predicted frames the same as the one predicted using the full search algorithm. Also, it could be noted that the performance of the proposed FCsFS algorithm is more effective if the video sequences have lower motion activity and vice versa. This is due to using two previous neighbours to predict the dimension of the new search window which has a high probability to contain the global matching MBl. Furthermore, for high motion activity video sequences “Stefan” and “Coastguard”, the number of search points in the FCsFS is the same as FS and PDE but with enhancement in the processing time. Figure 6.7 and Figure 6.8 show the frame-by-frame comparison of the average number of search points per MBl using the PSNR and MAD quality measures for low motion activity video sequences of 23 frames “Claire” and “Akiyo”, respectively; while the frame-by-frame comparisons for the medium motion activity video sequences of 23 frames of “News” and “Carphone” are shown in Figure 6.9 and Figure 6.10, respectively. For medium motion activity video sequences of 23 frames of

"Coastguard" and "Stefan"; the frame-by-frame comparisons are illustrated in Figure 6.11 and Figure 6.12, respectively.

For each video sequence, the visual images illustrated from Figure 6.13 to Figure 6.18 to describe the performance of the proposed technique at frame 50 and its predicted frame from reference frame 48 using the block matching motion estimation FS, PDE and the proposed FCsFS.

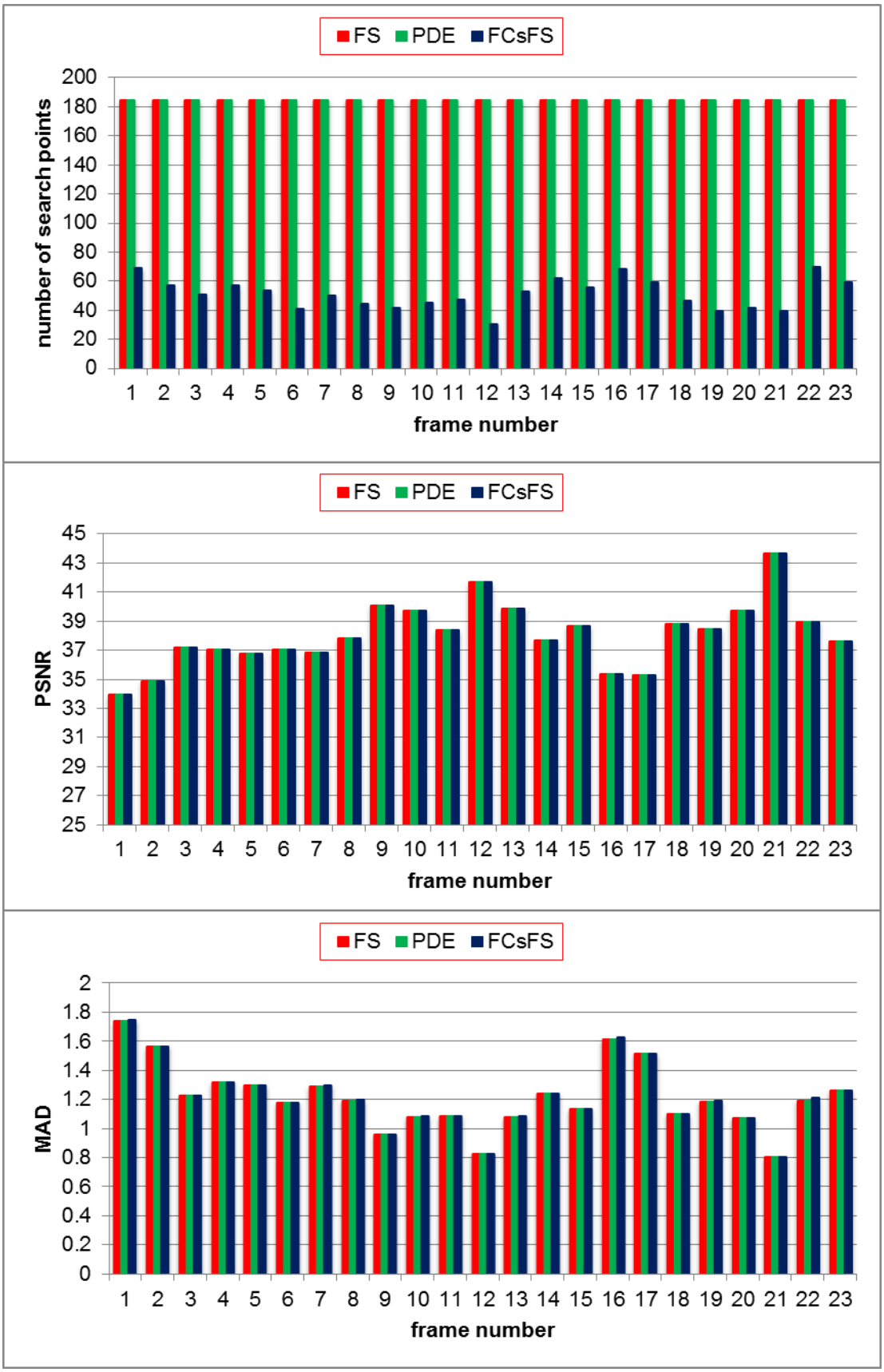


**Figure 6.7: Average number of search points per MBl, PSNR performance and MAD of FCsFS, FS and PDE in "Claire" video sequence of 23 frames**

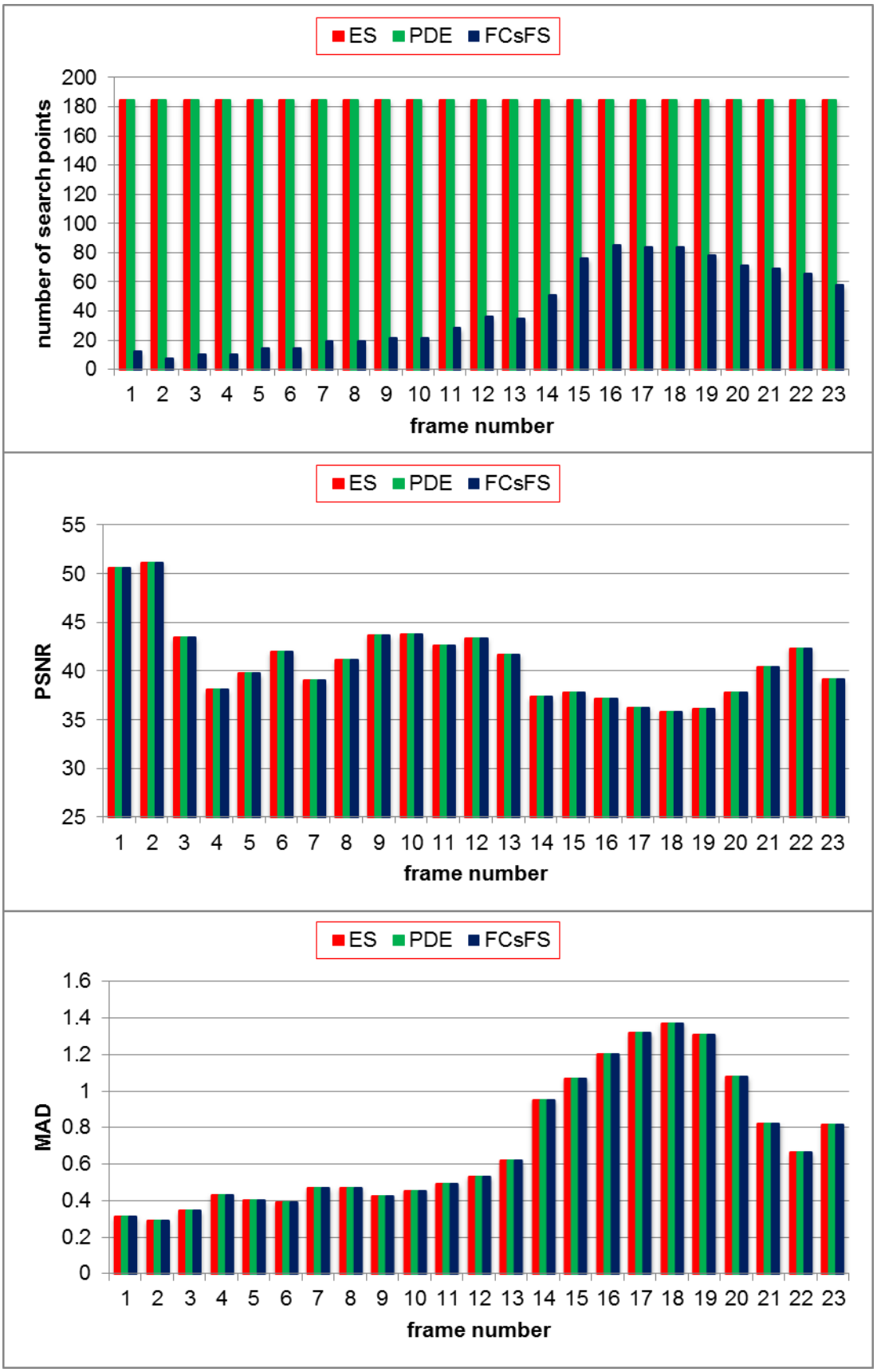


**Figure 6.8: Average number of search points per MBl, PSNR performance and MAD of FCsFS, FS and PDE in "Akiyo" video sequence of 23 frames**

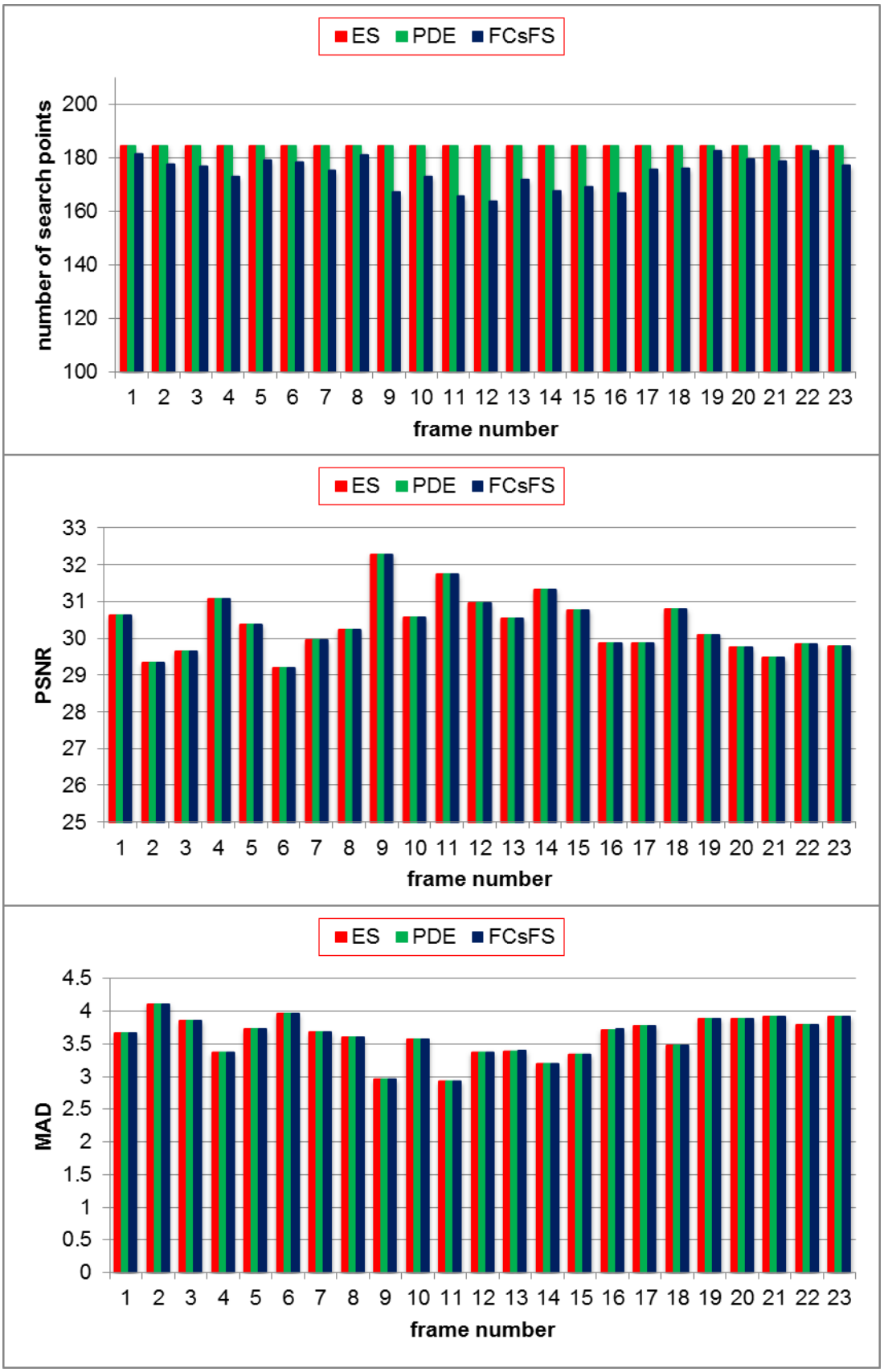


**Figure 6.9: Average number of search points per MBl, PSNR performance and MAD of FCsFS, FS and PDE in "Carphone" video sequence of 23 frames**

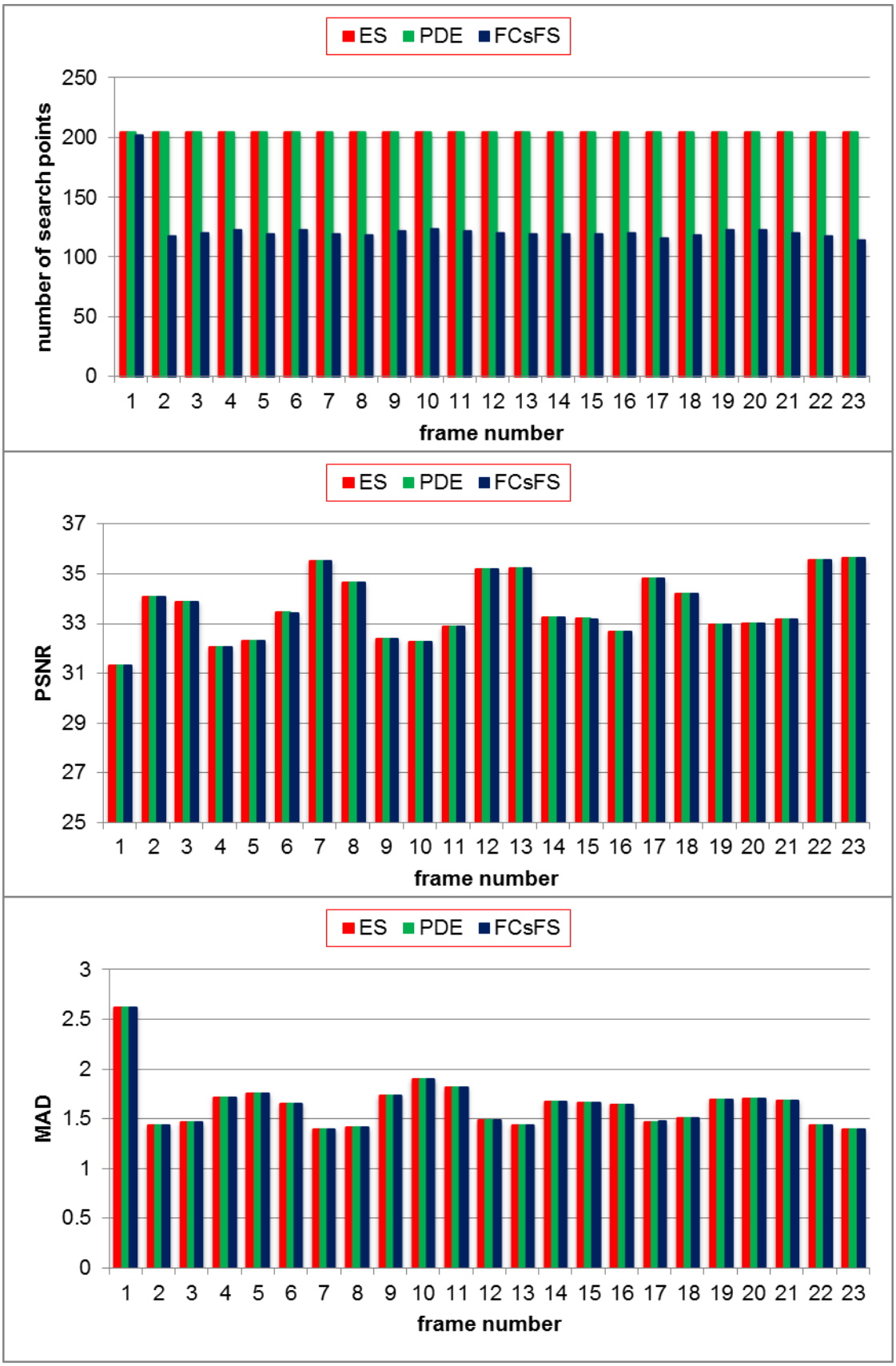


**Figure 6.10: Average number of search points per MBl, PSNR performance and MAD of FCsFS, FS and PDE in "News" video sequence of 23 frames**

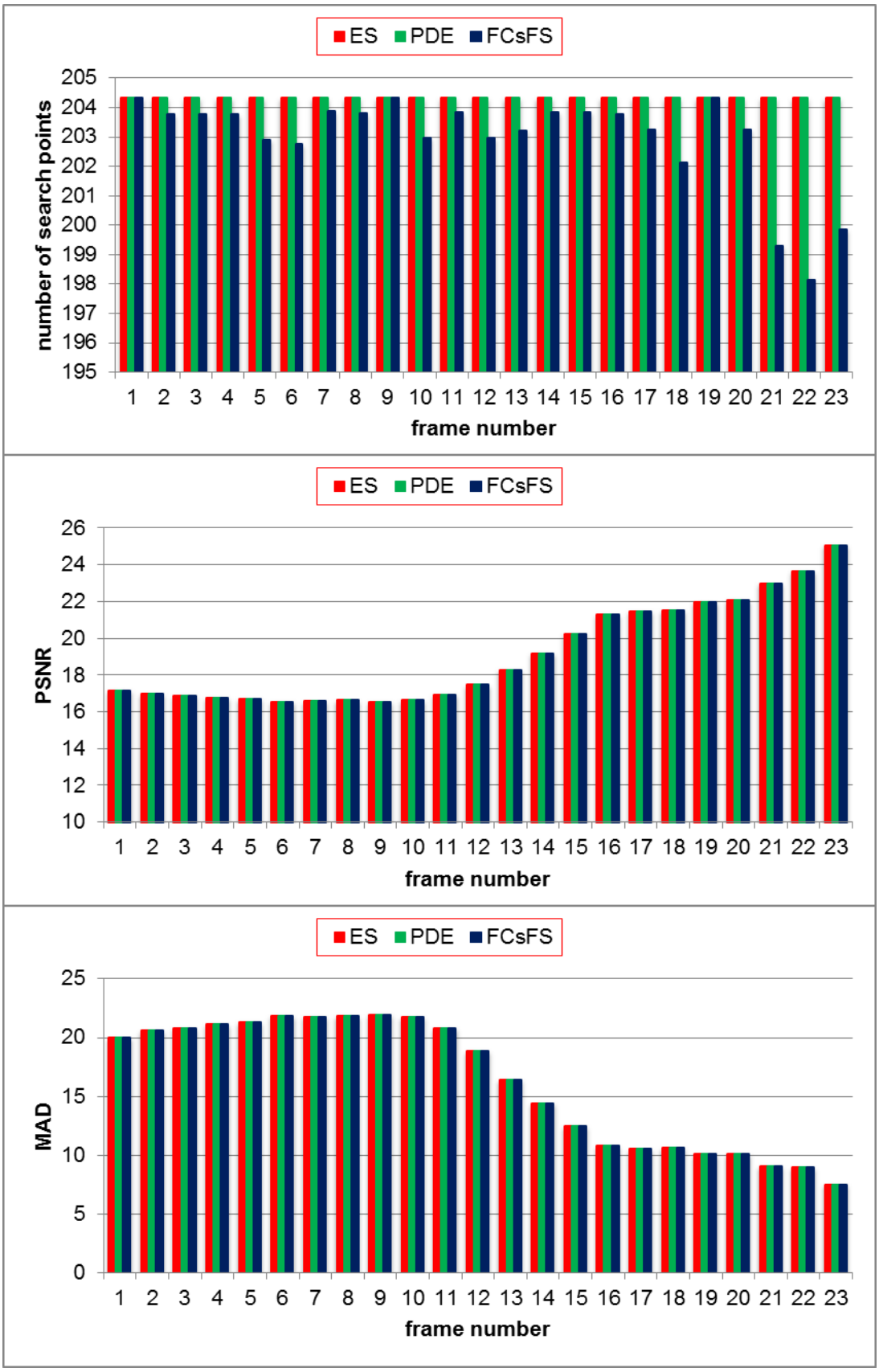


**Figure 6.11: Average number of search points per MBl, PSNR performance and MAD of FCsFS, FS and PDE in "Stefan" video sequence of 23 frames**

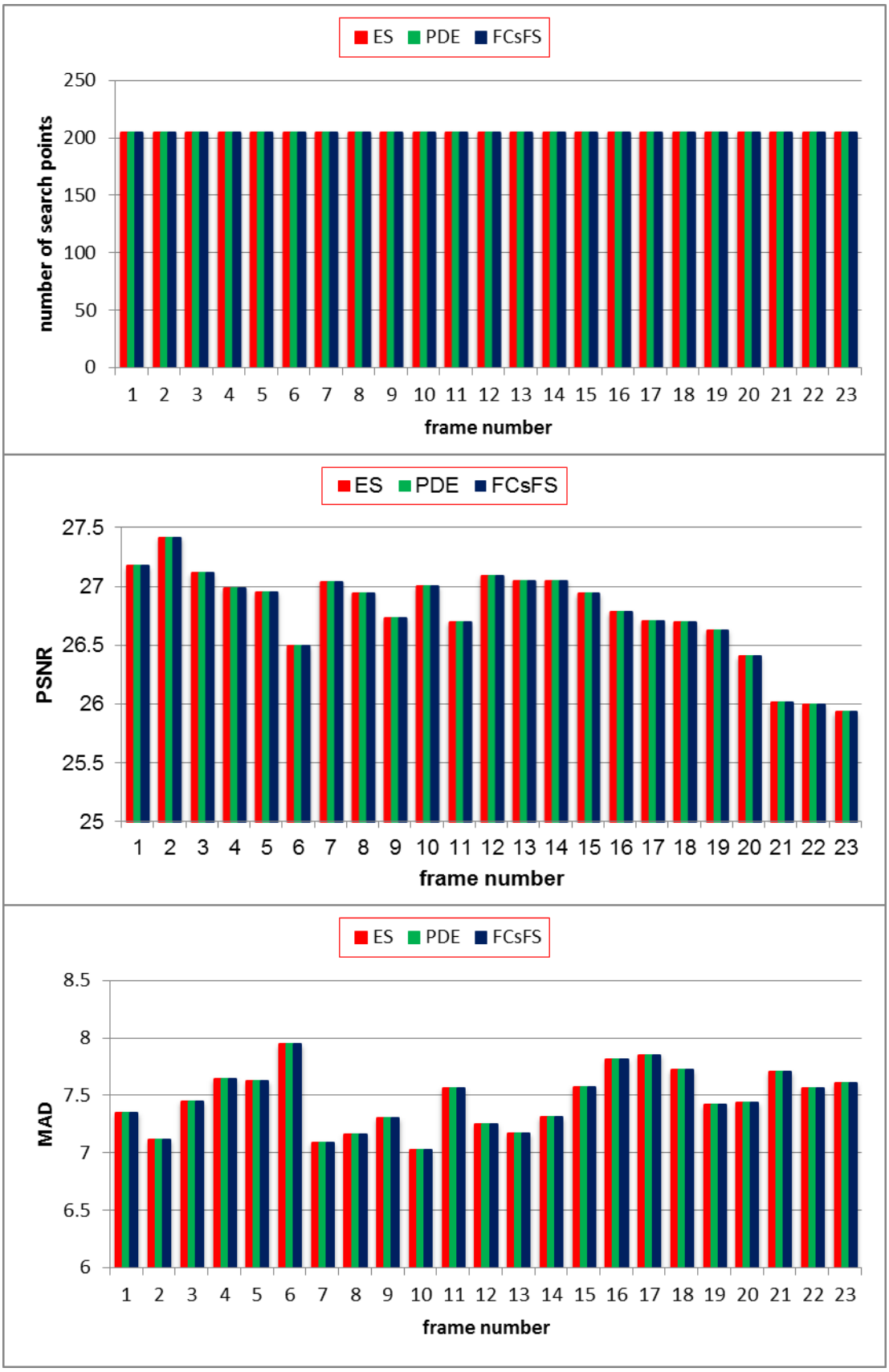


**Figure 6.12: Average number of search points per MBl, PSNR performance and MAD of FCsFS, FS and PDE in "Coastguard" video sequence of 23 frames**

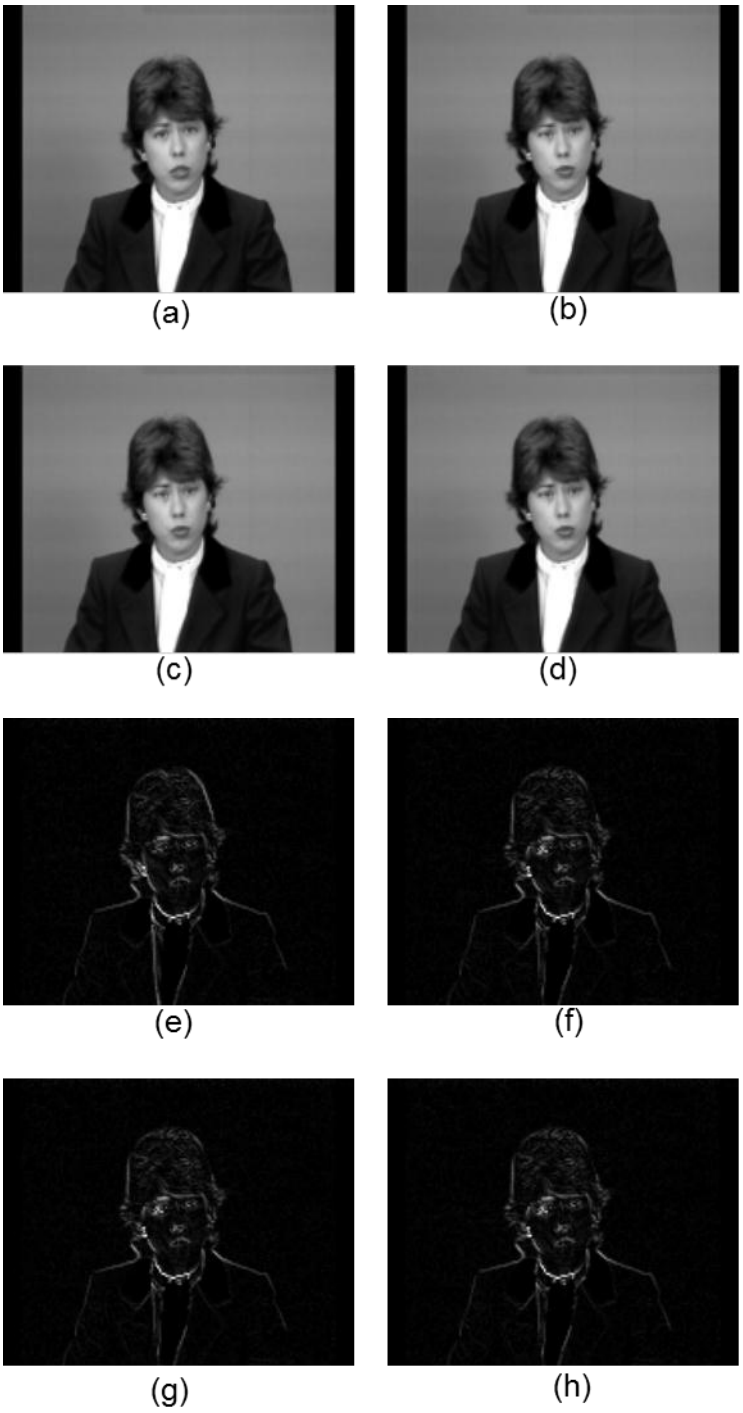


**Figure 6.13: (a) Frame 50 of "Claire" (b) predicted frame using FS, (c) predicted frame using PDE, (d) predicted frame using FCsFS, (e) the difference error between frame 50 and its reference frame 48, (f) the difference error between frame 50 and its predicted frame using FS, (g) the difference error between frame 50 and its predicted frame using PDE, (h) the difference error between frame 50 and its predicted frame using the proposed FCsFS**

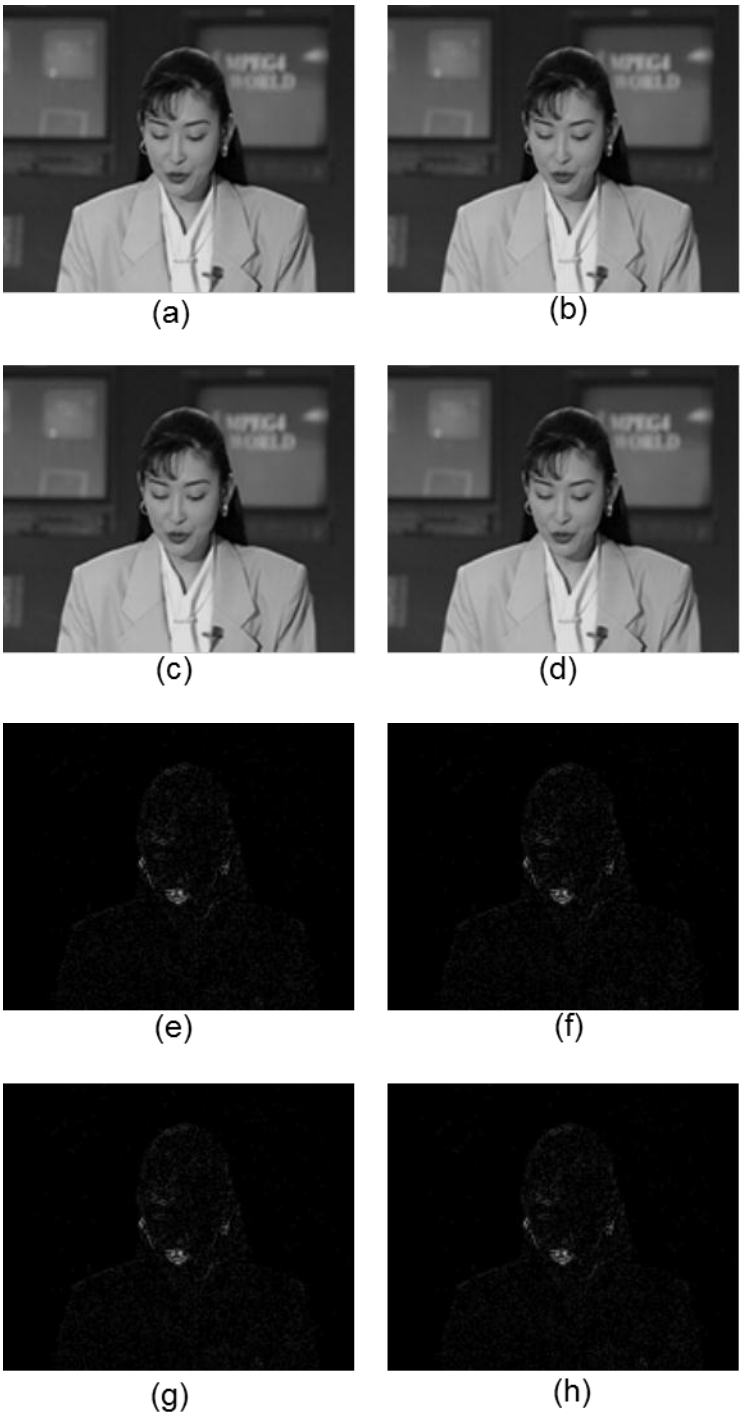


**Figure 6.14: (a) Frame 50 of "Akiyo" (b) predicted frame using FS, (c) predicted frame using PDE, (d) predicted frame using FCsFS, (e) the difference error between frames 50 and its reference frame 48, (f) the difference error between frame 50 and its predicted frame using FS, (g) the difference error between frame 50 and its predicted frame using PDE, (h) the difference error between frame 50 and its predicted frame using the proposed FCsFS**

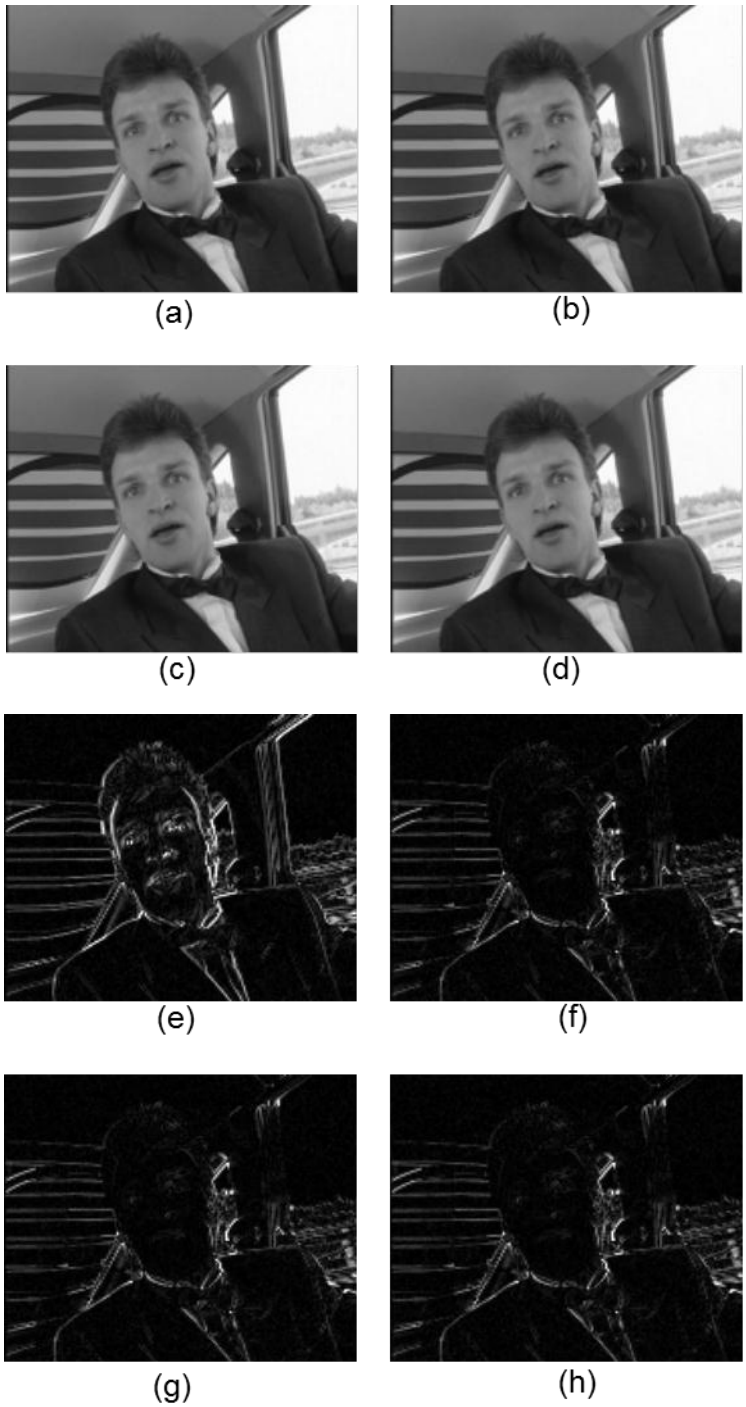


**Figure 6.15: (a) Frame 50 of "Carphone" (b) predicted frame using FS, (c) predicted frame using PDE, (d) predicted frame using FCsFS, (e) the difference error between frame 50 and its reference frame 48, (f) the difference error between frame 50 and its predicted frame using FS, (g) the difference error between frame 50 and its predicted frame using PDE, (h) the difference error between frame 50 and its predicted frame using the proposed FCsFS**

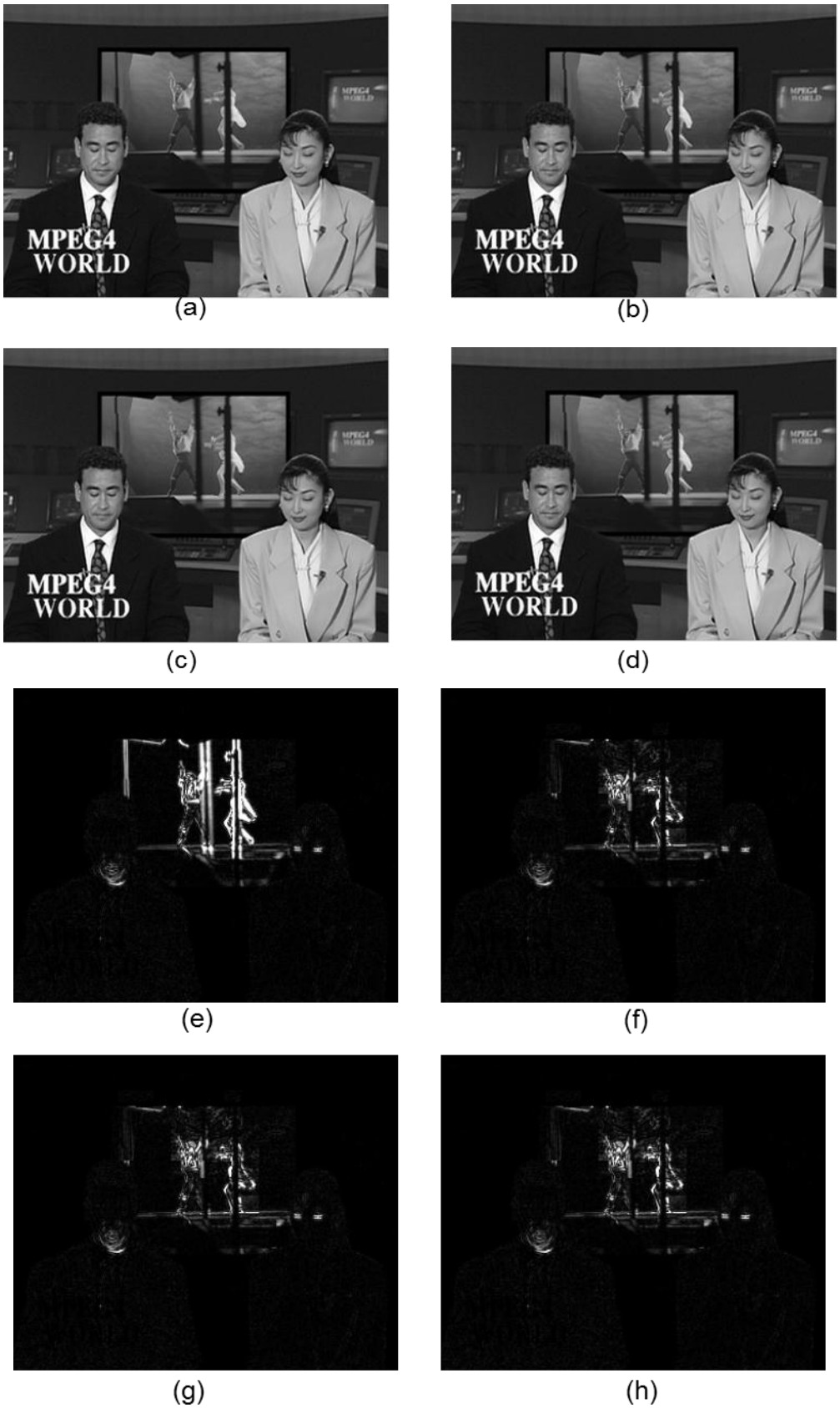


**Figure 6.16: (a) Frame 50 of "News" (b) predicted frame using FS, (c) predicted frame using PDE, (d) predicted frame using FCSFS, (e) the difference error between frame 50 and its reference frame 48, (f) the difference error between frame 50 and its predicted frame using FS, (g) the difference error between frame 50 and its predicted frame using PDE, (h) the difference error between frame 50 and its predicted frame using the proposed FCSFS**

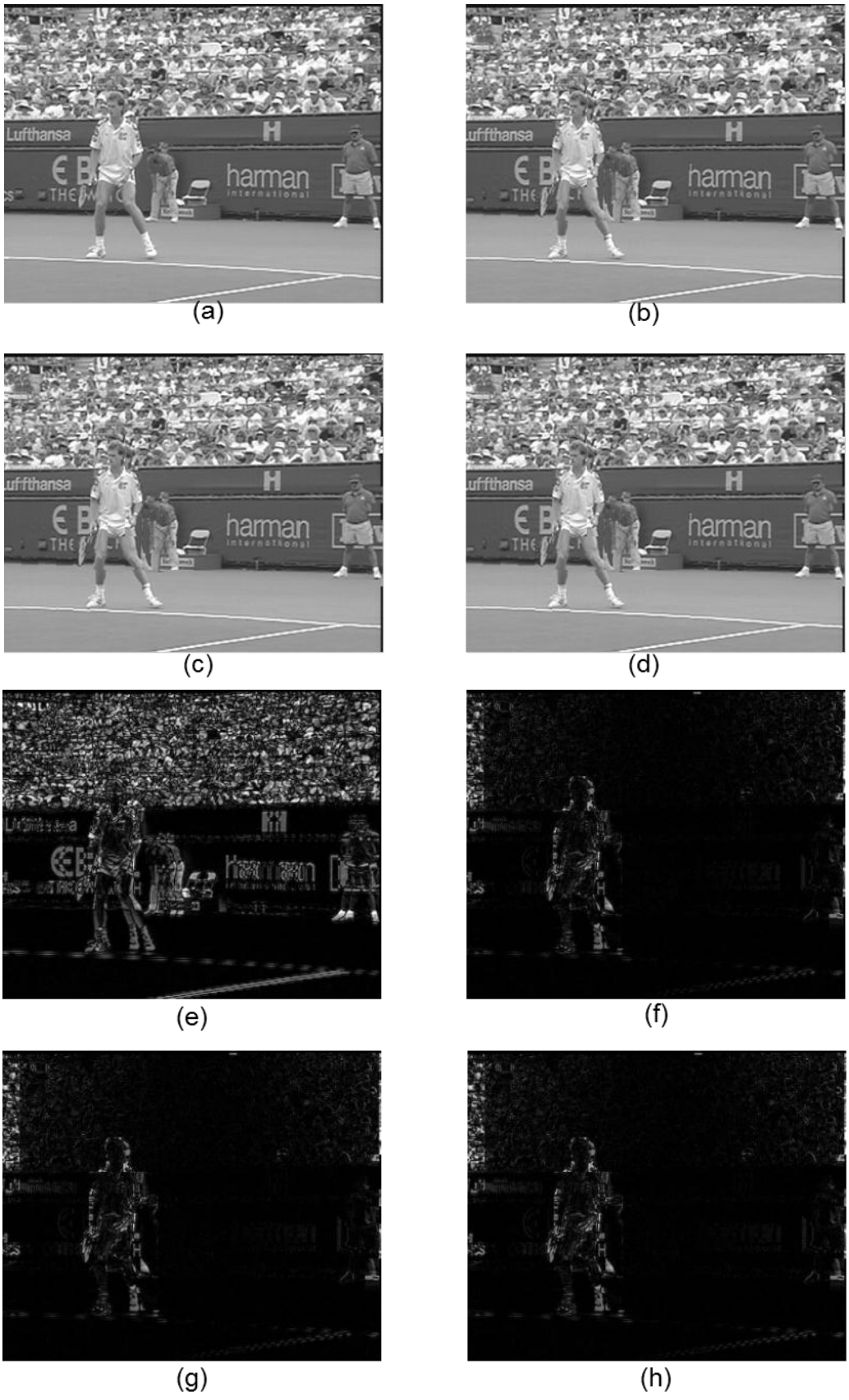


**Figure 6.17: (a) Frame 50 of "Stefan" (b) predicted frame using FS, (c) predicted frame using PDE, (d) predicted frame using FCSFS, (e) the difference error between frame 50 and its reference frame 48, (f) the difference error between frame 50 and its predicted frame using FS, (g) the difference error between frame 50 and its predicted frame using PDE, (h) the difference error between frame 50 and its predicted frame using the proposed FCSFS**

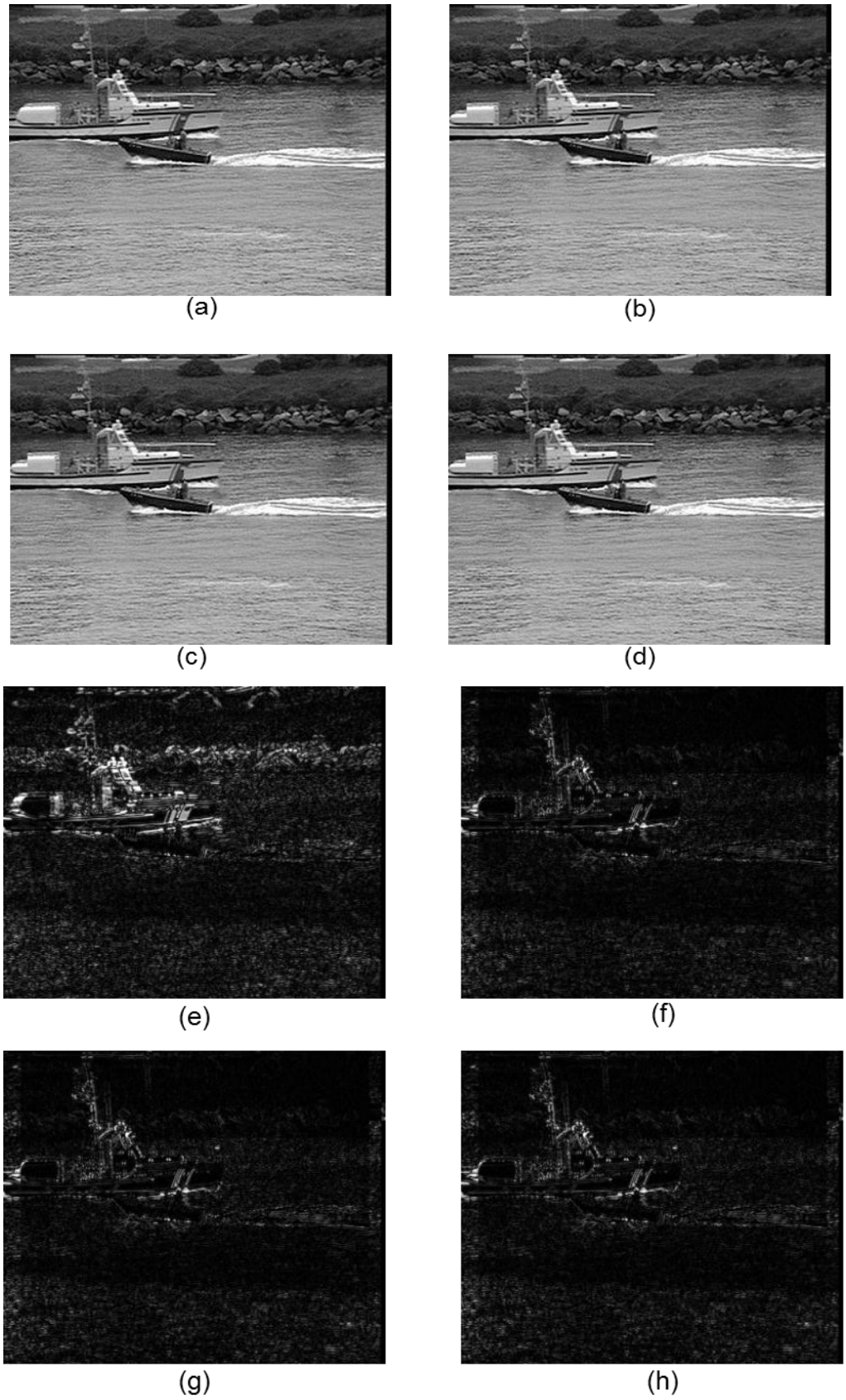


**Figure 6.18: (a) Frame 50 of "Coastguard" (b) predicted frame using FS, (c) predicted frame using PDE, (d) predicted frame using FCSFS, (e) the difference error between frame 50 and its reference frame 48, (f) the difference error between frame 50 and its predicted frame using FS, (g) the difference error between frame 50 and its predicted frame using PDE, (h) the difference error between frame 50 and its predicted frame using the proposed FCSFS**

## 6.3 Simulation Results of Mean Predictive Block Matching Algorithm (MPBM)

The performance of the novel technique MPBM is benchmarked with six standard fast block matching algorithms, which are FS, DS, NTSS, 4SS, SESTSS, and ARPS. The size of each MBl will be 16 ×16 for all the selected video sequences and the current MBls are searched for the reference image using a search range of ±7. The SAD and MAD are used as the Block Distortion Measures.

The simulation results indicated that the proposed algorithm (MPBM) shows improvement in the computational complexity; also, it attempts to keep or reduce the error between current and compensated frames.

The results of the computational complexity measured by the average number of search points required to get each motion vector are shown in Table 6.6. Moreover, the processing time of these algorithms should be computed for the performance when applying the PDE algorithm. The time required for these algorithms is shown in Table 6.7. The resolution of the predicted frames that is built by the proposed and benchmarked algorithms is explained by mean MAD, which is shown in Table 6.8, and mean PSNR, which is shown in Table 6.9.

**Table 6.6: Average number of search points per MBl of size 16 ×16**

| Sequence | Format | FS | DS | NTSS | 4SS | SESTSS | ARPS | MPBM |
|---|---|---|---|---|---|---|---|---|
| **Claire** | QCIF | 184.6 | 11.63 | 15.09 | 14.77 | 16.13 | 5.191 | 2.128 |
| **Akiyo** | QCIF | 184.6 | 11.46 | 14.76 | 14.67 | 16.2 | 4.958 | 1.938 |
| **Carphone** | QCIF | 184.6 | 13.76 | 17.71 | 16.12 | 15.73 | 7.74 | 7.06 |
| **News** | CIF | 204.3 | 13.1 | 17.07 | 16.38 | 16.92 | 6.058 | 3.889 |
| **Stefan** | CIF | 204.3 | 17.69 | 22.56 | 19.05 | 16.11 | 9.641 | 9.619 |
| **Coastguard** | CIF | 204.3 | 19.08 | 27.26 | 19.91 | 16.52 | 9.474 | 8.952 |

**Table 6.7: The simulation results of average time in seconds needed to process 50 frames**

| Sequence | Format | FS | DS | NTSS | 4SS | SESTSS | ARPS | MPBM |
|---|---|---|---|---|---|---|---|---|
| **Claire** | QCIF | 0.351 | 0.037 | 0.031 | 0.031 | 0.037 | 0.025 | **0.015** |
| **Akiyo** | QCIF | 0.354 | 0.036 | 0.031 | 0.031 | 0.037 | 0.023 | **0.006** |
| **Carphone** | QCIF | 0.338 | 0.039 | 0.036 | 0.032 | 0.035 | 0.031 | **0.033** |
| **News** | CIF | 1.539 | 0.161 | 0.142 | 0.136 | 0.151 | 0.112 | **0.079** |
| **Stefan** | CIF | 1.537 | 0.267 | 0.232 | 0.174 | 0.15 | 0.158 | **0.139** |
| **Coastguard** | CIF | 1.551 | 0.263 | 0.235 | 0.178 | 0.15 | 0.152 | **0.14** |

**Table 6.8: The simulation results of mean MAD for 50 frames**

| Sequence | Format | FS | DS | NTSS | 4SS | SESTSS | ARPS | MPBM |
|---|---|---|---|---|---|---|---|---|
| **Calire** | QCIF | 1.13 | 1.13 | 1.13 | 1.13 | 1.14 | 1.13 | **1.13** |
| **Akiyo** | QCIF | 0.81 | 0.81 | 0.81 | 0.81 | 0.81 | 0.81 | **0.81** |
| **Carphone** | QCIF | 3.42 | 3.47 | 3.47 | 3.6 | 3.8 | 3.51 | **3.49** |
| **News** | CIF | 1.59 | 1.6 | 1.6 | 1.61 | 1.61 | 1.61 | **1.6** |
| **Stefan** | CIF | 11.6 | 12.6 | 12.1 | 12.6 | 13.3 | 12.1 | **11.9** |
| **Coastguard** | CIF | 7.91 | 8.05 | 7.99 | 8.02 | 8.3 | 7.99 | **7.94** |

**Table 6.9: The simulation results of mean PSNR for 50 frames**

| Sequence | Format | FS | DS | NTSS | 4SS | SESTSS | ARPS | MPBM |
|---|---|---|---|---|---|---|---|---|
| **Calire** | QCIF | 38.94 | 38.94 | 38.94 | 38.92 | 38.89 | 38.94 | **38.94** |
| **Akiyo** | QCIF | 39.61 | 39.61 | 39.61 | 39.61 | 39.61 | 39.61 | **39.61** |
| **Carphone** | QCIF | 30.82 | 30.69 | 30.7 | 30.4 | 30.1 | 30.58 | **30.6** |
| **News** | CIF | 33.77 | 33.45 | 33.63 | 33.42 | 33.19 | 33.39 | **33.56** |
| **Stefan** | CIF | 22.16 | 21.49 | 21.81 | 21.51 | 21.04 | 21.82 | **21.93** |
| **Coastguard** | CIF | 26.19 | 25.98 | 26.05 | 26.02 | 25.6 | 26.05 | **26.11** |

**Table 6.10: The ratio between PSNR and processing time**

| Sequence | Format | FS | DS | NTSS | 4SS | SESTSS | ARPS | MPBM |
|---|---|---|---|---|---|---|---|---|
| **Claire** | QCIF | 110.94 | 1052.4 | 1256.1 | 1255.5 | 1051.1 | 1557.6 | **2596** |
| **Akiyo** | QCIF | 111.89 | 1100.3 | 1277.7 | 1277.7 | 1070.5 | 1722.2 | **6601.7** |
| **Carphone** | QCIF | 91.183 | 786.92 | 852.78 | 950 | 860 | 986.45 | **927.27** |
| **News** | CIF | 21.943 | 207.76 | 236.83 | 245.74 | 219.8 | 298.13 | **424.81** |
| **Stefan** | CIF | 14.418 | 80.487 | 94.009 | 123.62 | 140.27 | 138.1 | **157.77** |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Coastguard** | CIF | 16.886 | 98.783 | 110.85 | 146.18 | 170.67 | 171.38 | **186.5** |

Similar to previous technique these codes have been implemented in Matlab and the simulation results indicated that the proposed algorithm (MPBM) shows improvement in the computational complexity, and it tries to keep or reduce the error between current and compensated frames benchmarked with the other algorithms.

For low motion activity video sequences, the resolution of the predicted frame (Table 6.8 and Table 6.9) is close to the ones predicted by full search and there is enhancement in the computational complexity; while for the medium and high motion activity video sequences, the improvement of computational complexity and the resolution of the predicted frame are acceptable compared with other fast block matching algorithms. The "Carphone" video sequence has less average number of search points (Table 6.6) but the average time (Table 6.7) is not the lowest; this is due to the condition statements which take a long time to process in Matlab. Moreover, it could be noticed in Table 6.10 that the ratio between PSNR and time needed for computation of the proposed algorithm gives the best results in comparison to the benchmarked algorithms.

To introduce a more clear expression for this performance, Figure 6.19 to Figure 6.24 show the frame-by-frame comparison of the average number of search points per MBl, PSNR performance and MAD of MPBM, FS, DS and ARPS for 23 frames of the tested videos, respectively.

For each video sequence, the visual images illustrated from Figure 6.25 to Figure 6.30 describe the performance of the proposed technique at frame 50 and its predicted frame from reference frame 48 using the block matching motion estimation DS, ARPS and the proposed MPBM.

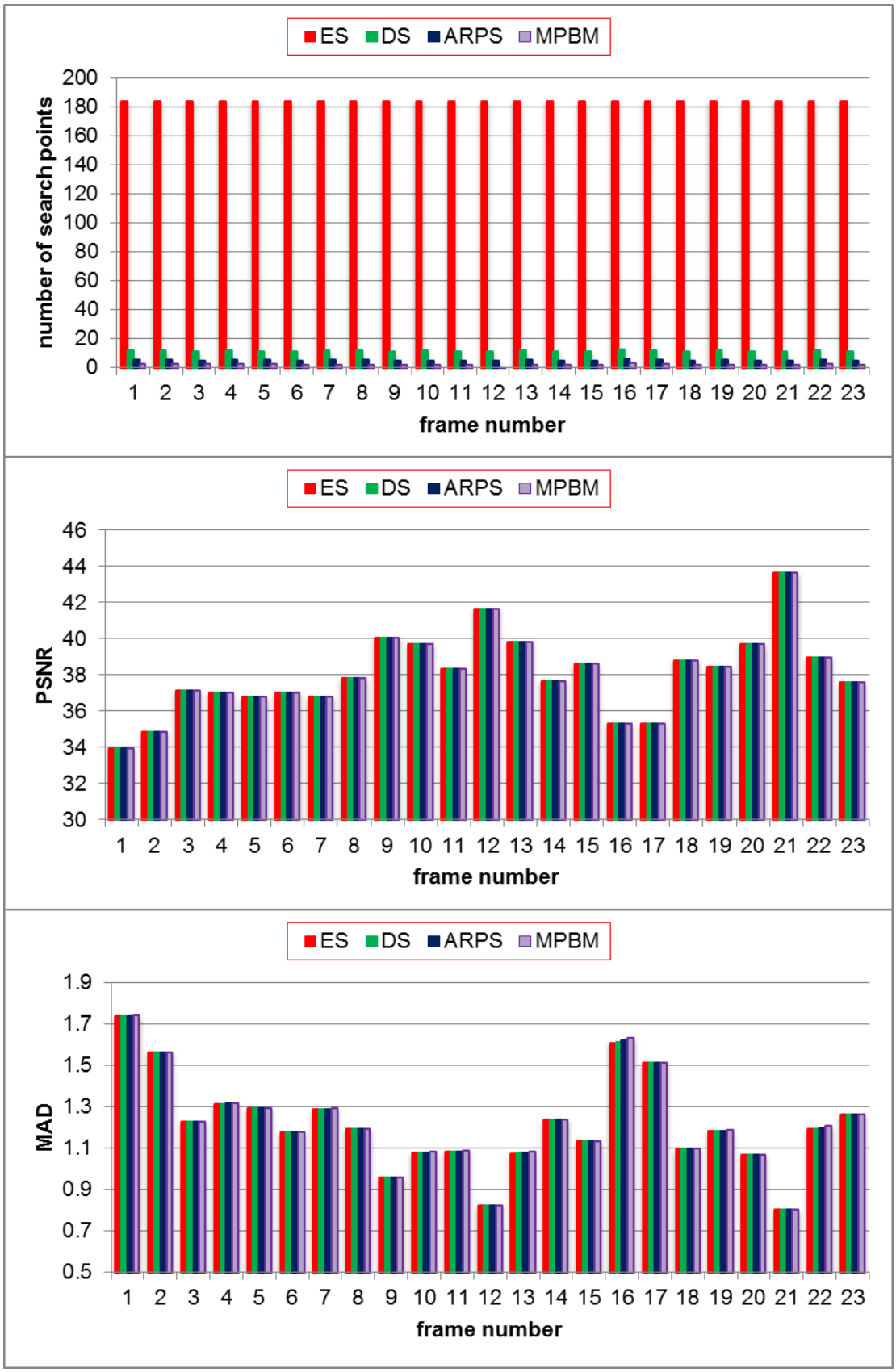


**Figure 6.19: Average number of search points per MBl, PSNR performance and MAD of MPBM and different search algorithms in "Claire" video sequence of 23 frames**

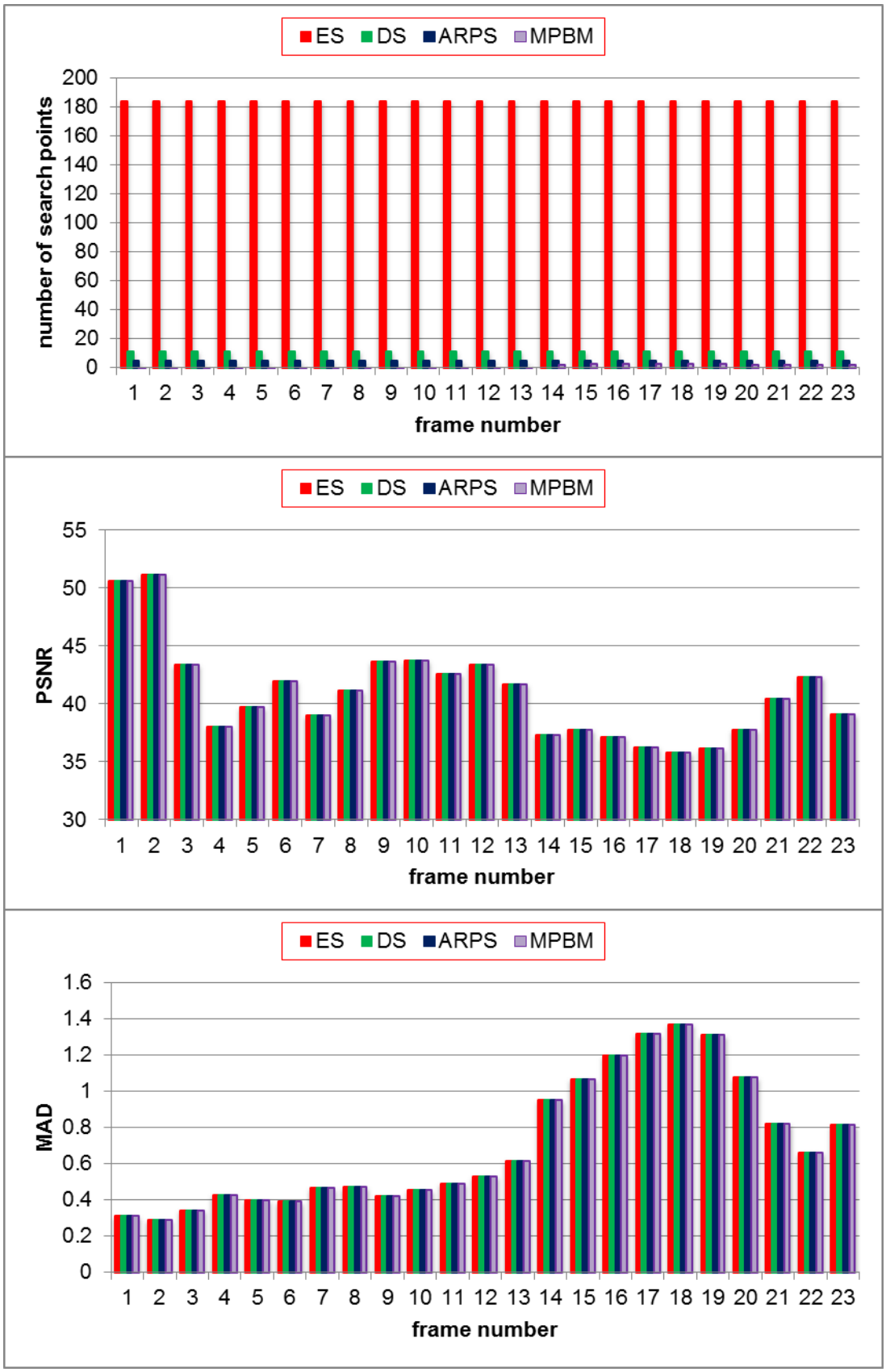


**Figure 6.20: Average number of search points per MBl, PSNR performance and MAD of MPBM and different search algorithms in "Akiyo" video sequence of 23 frames**

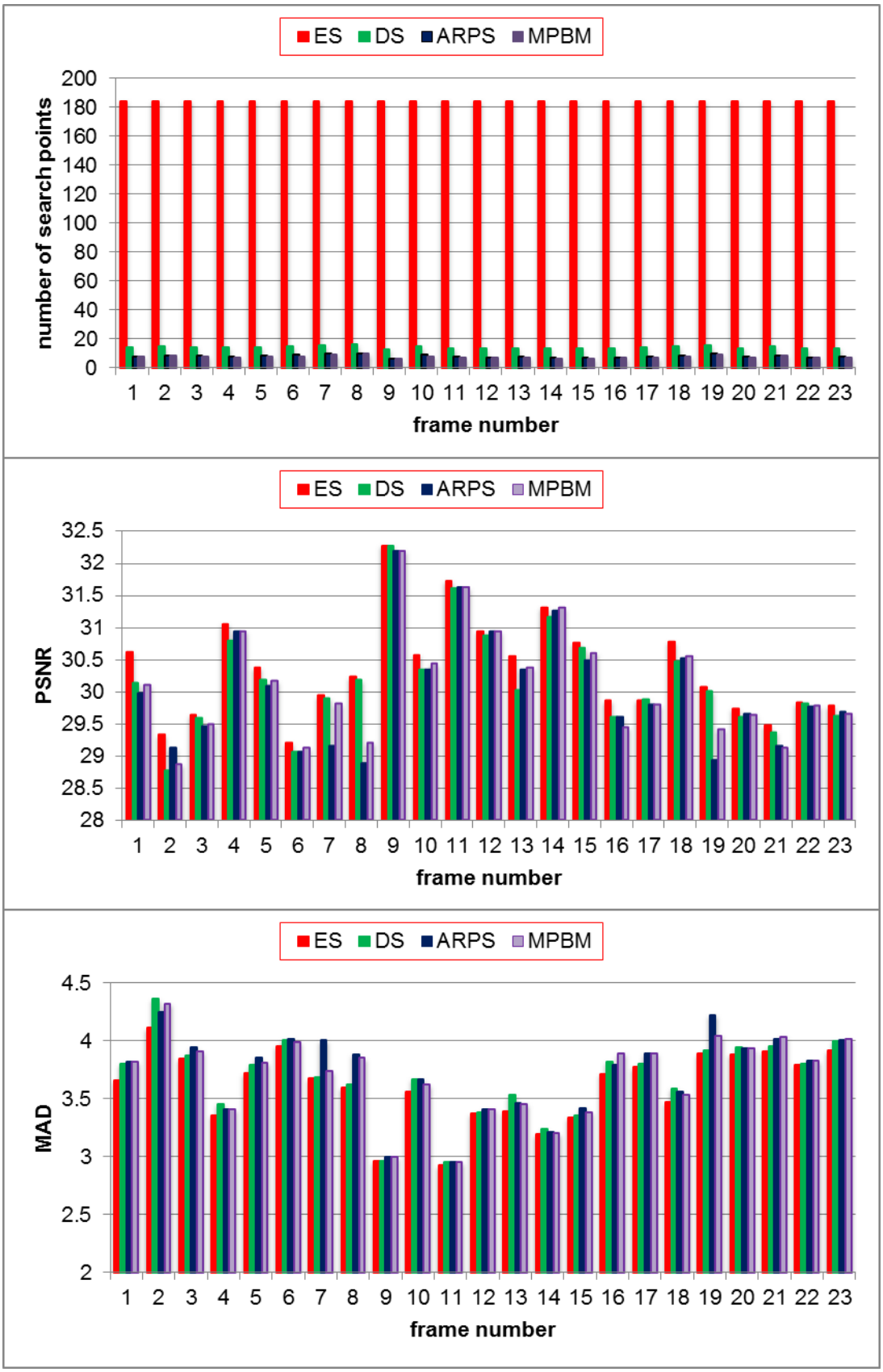


**Figure 6.21: Average number of search points per MBl, PSNR performance and MAD of MPBM and different search algorithms in "Carphone" video sequence of 23 frames**

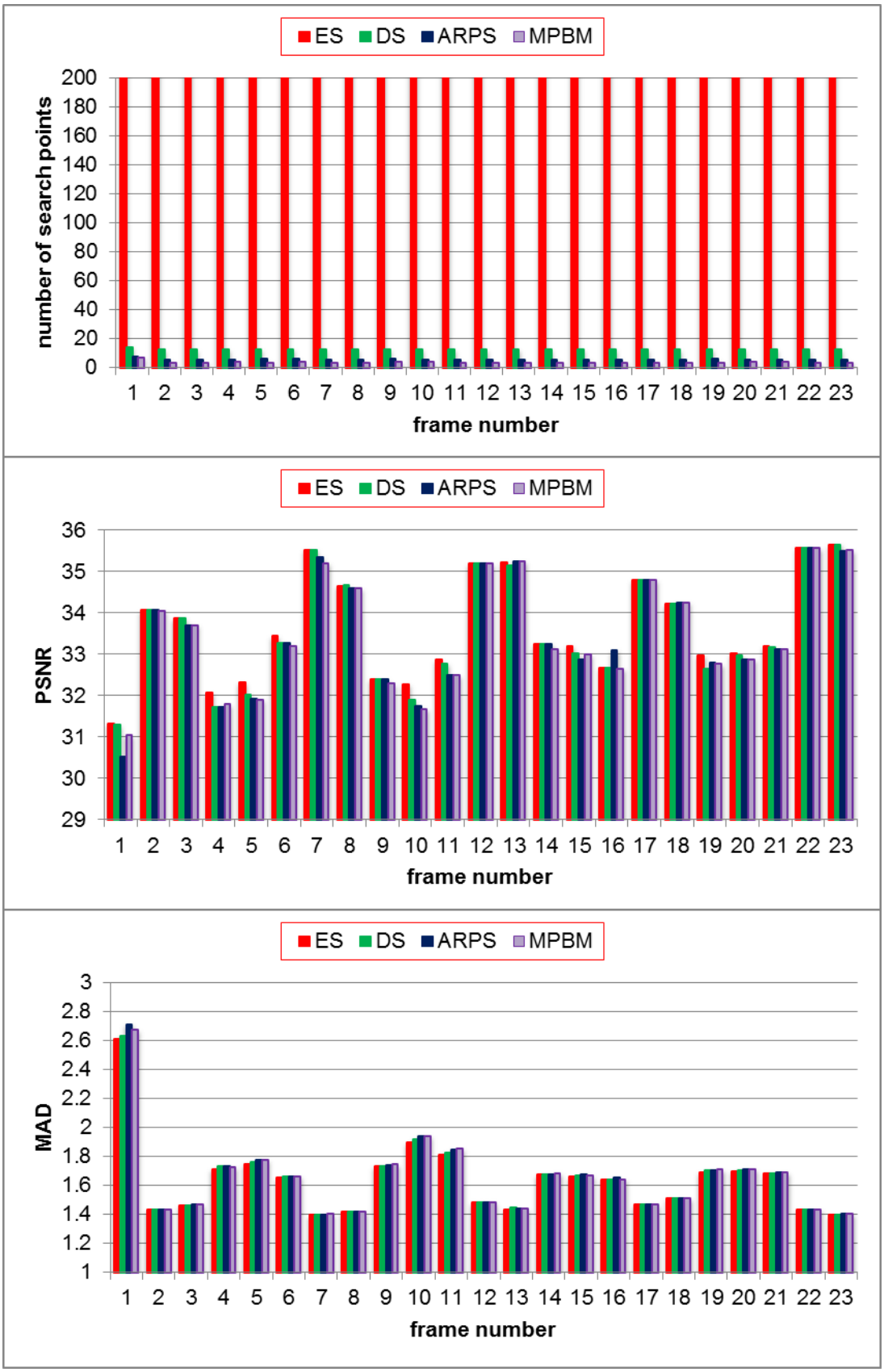


**Figure 6.22: Average number of search points per MBl, PSNR performance and MAD of MPBM and different search algorithms in "News" video sequence of 23 frames**

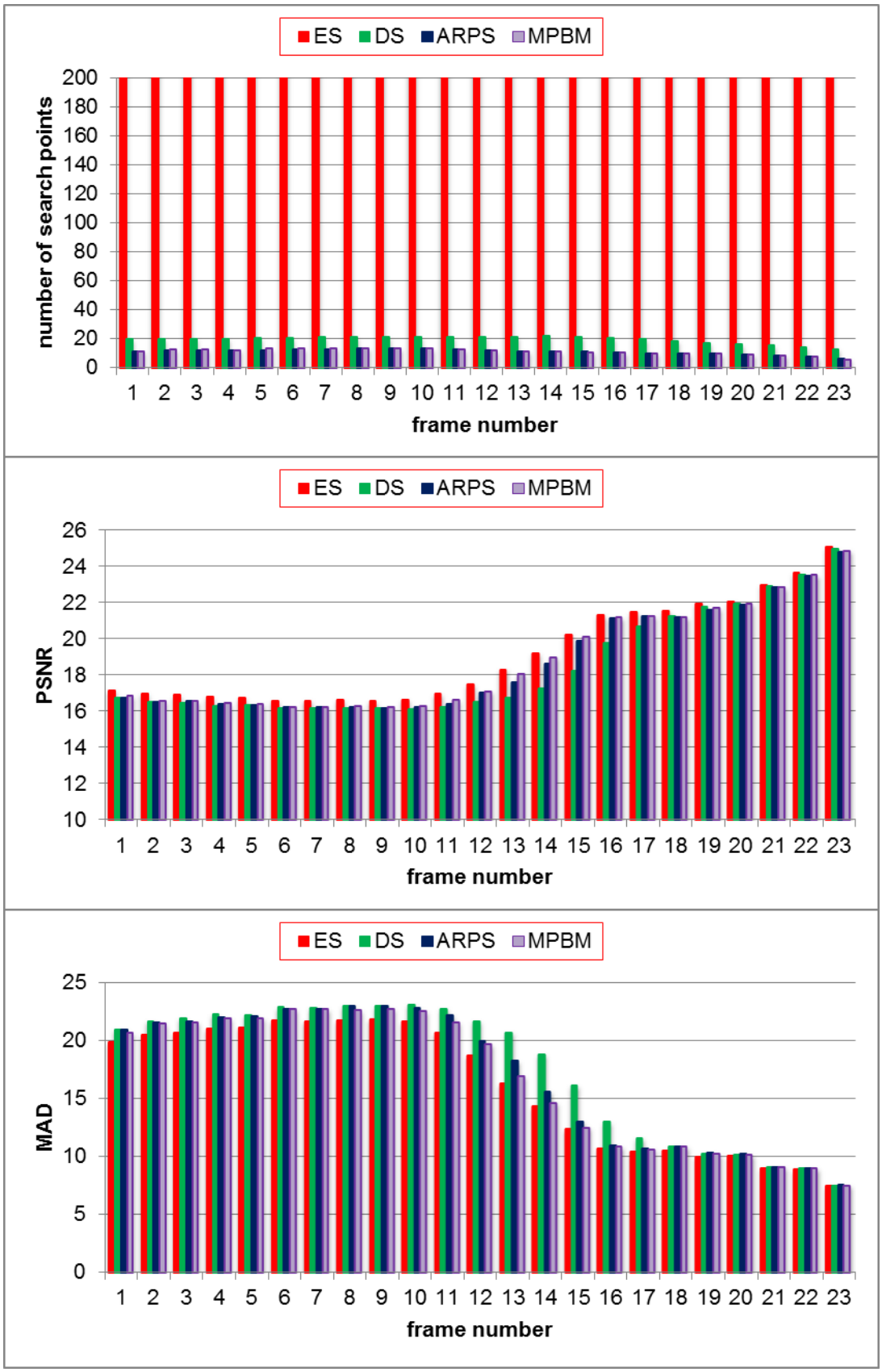


**Figure 6.23: Average number of search points per MBl, PSNR performance and MAD of MPBM and different search algorithms in "Stefan" video sequence of 23 frames**

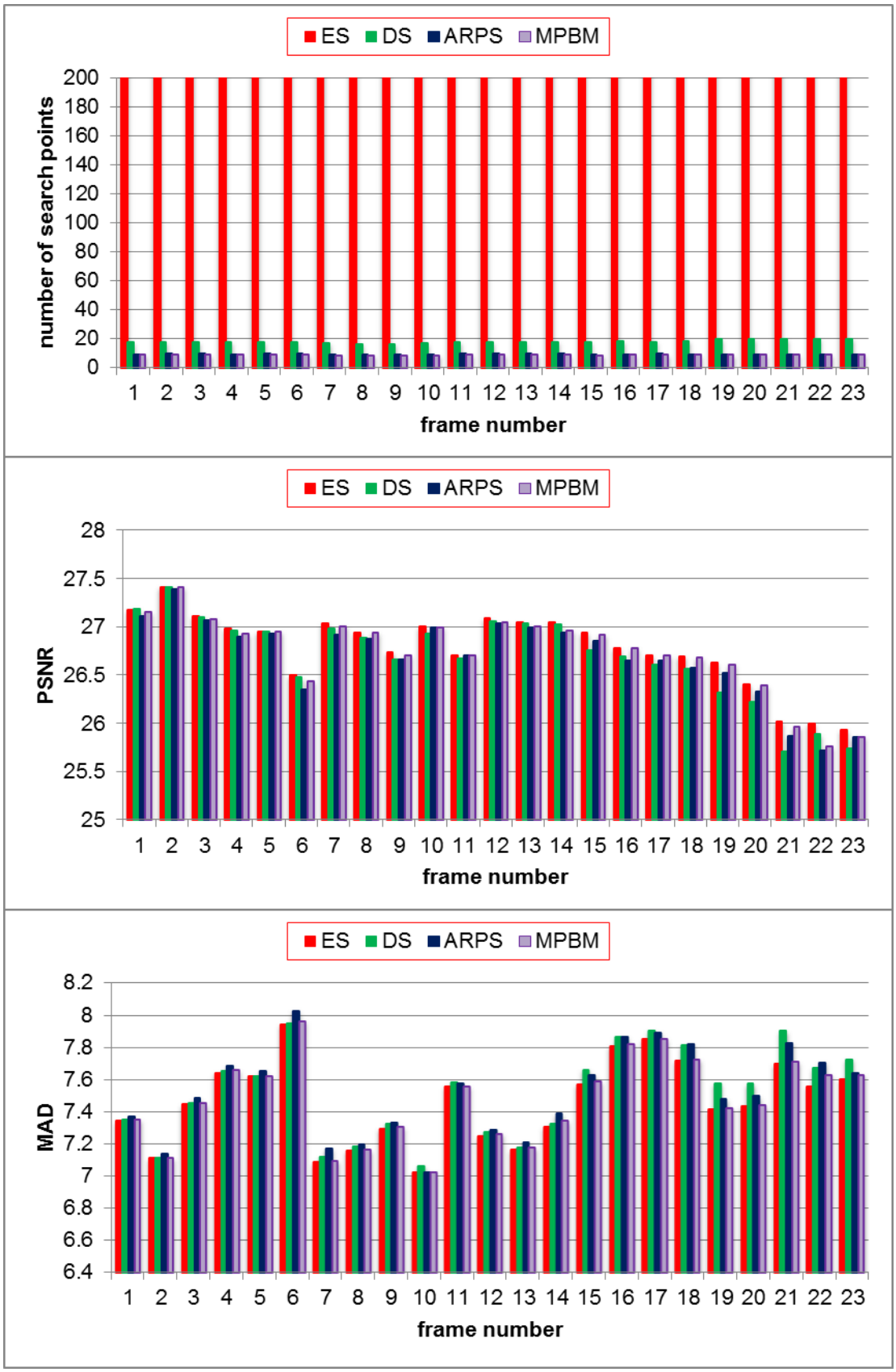


**Figure 6.24: Average number of search points per MBl, PSNR performance and MAD of MPBM and different search algorithms in "Coastguard" video sequence of 23 frames**


**Figure 6.25: (a) Frame 50 of "Claire", (b) predicted frame using DS, (c) predicted frame using ARPS, (d) predicted frame using MPBM, (e) the difference error between frame 50 and its reference frame 48, (f) the difference error between frame 50 and its predicted frame using DS, (g) the difference error between frame 50 and its predicted frame using ARPS and (h) the difference error between frame 50 and its predicted frame using the proposed MPBM**

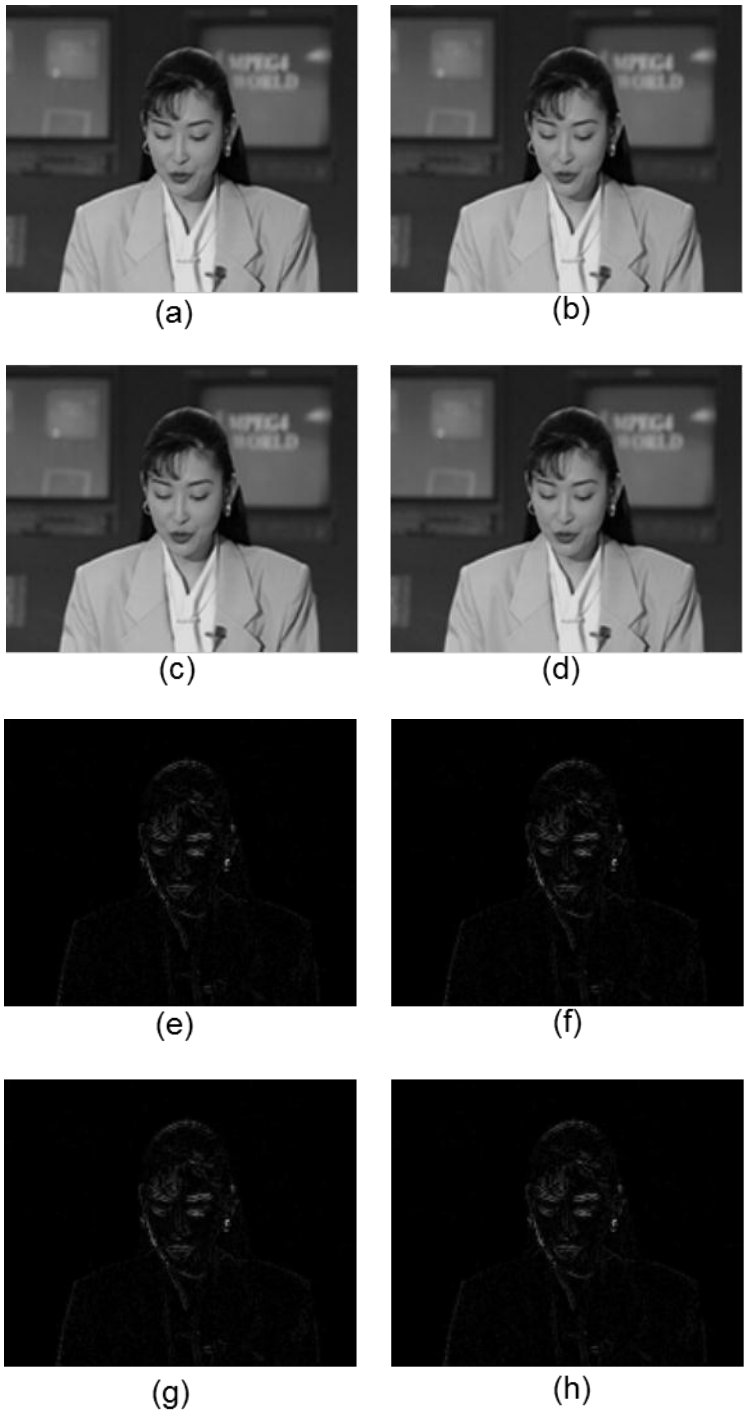


**Figure 6.26: (a) Frame 50 of "Akiyo", (b) predicted frame using DS, (c) predicted frame using ARPS, (d) predicted frame using MPBM, (e) the difference error between frame 50 and its reference frame 48, (f) the difference error between frame 50 and its predicted frame using DS, (g) the difference error between frame 50 and its predicted frame using ARPS and (h) the difference error between frame 50 and its predicted frame using the proposed MPBM**

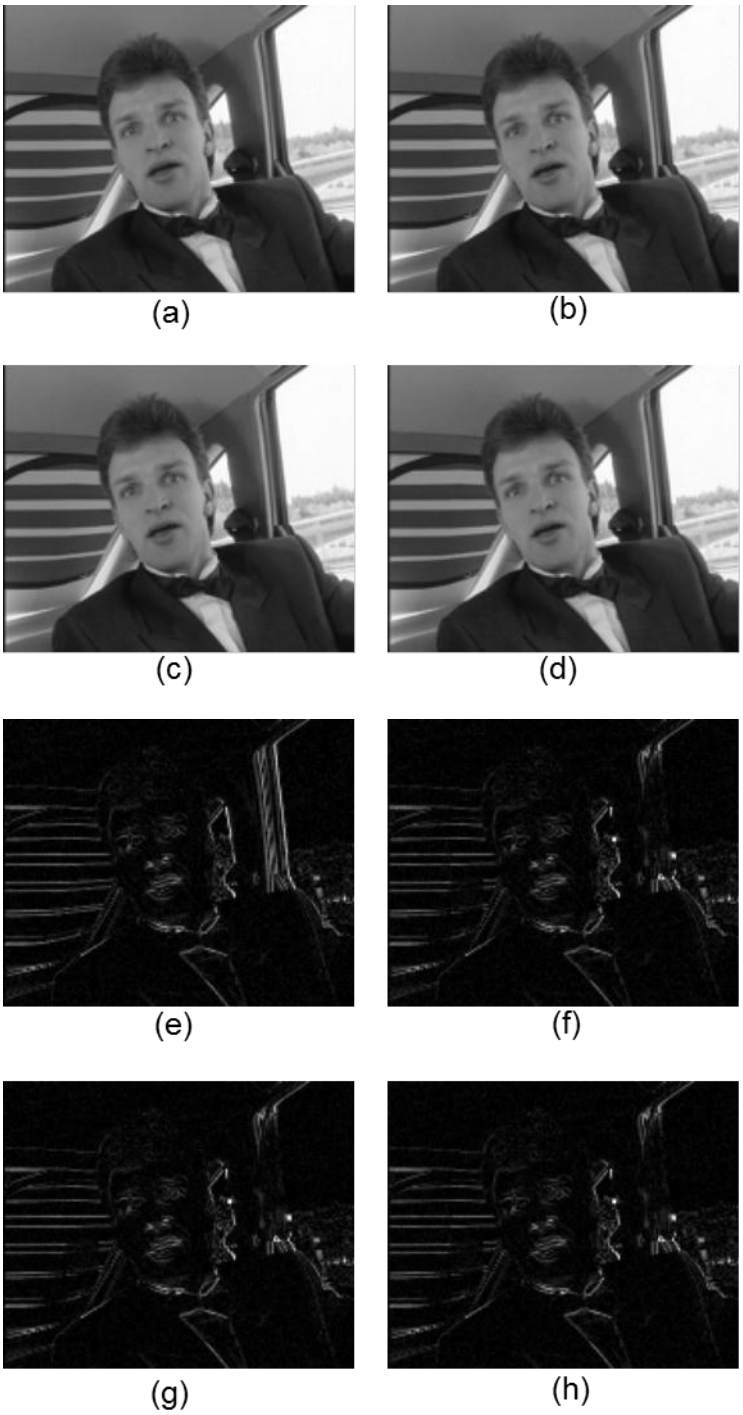


**Figure 6.27: (a) Frame 50 of "Carphone", (b) predicted frame using DS, (c) predicted frame using ARPS, (d) predicted frame using MPBM, (e) the difference error between frame 50 and its reference frame 48, (f) the difference error between frame 50 and its predicted frame using DS, (g) the difference error between frame 50 and its predicted frame using ARPS and (h) the difference error between frame 50 and its predicted frame using the proposed MPBM**

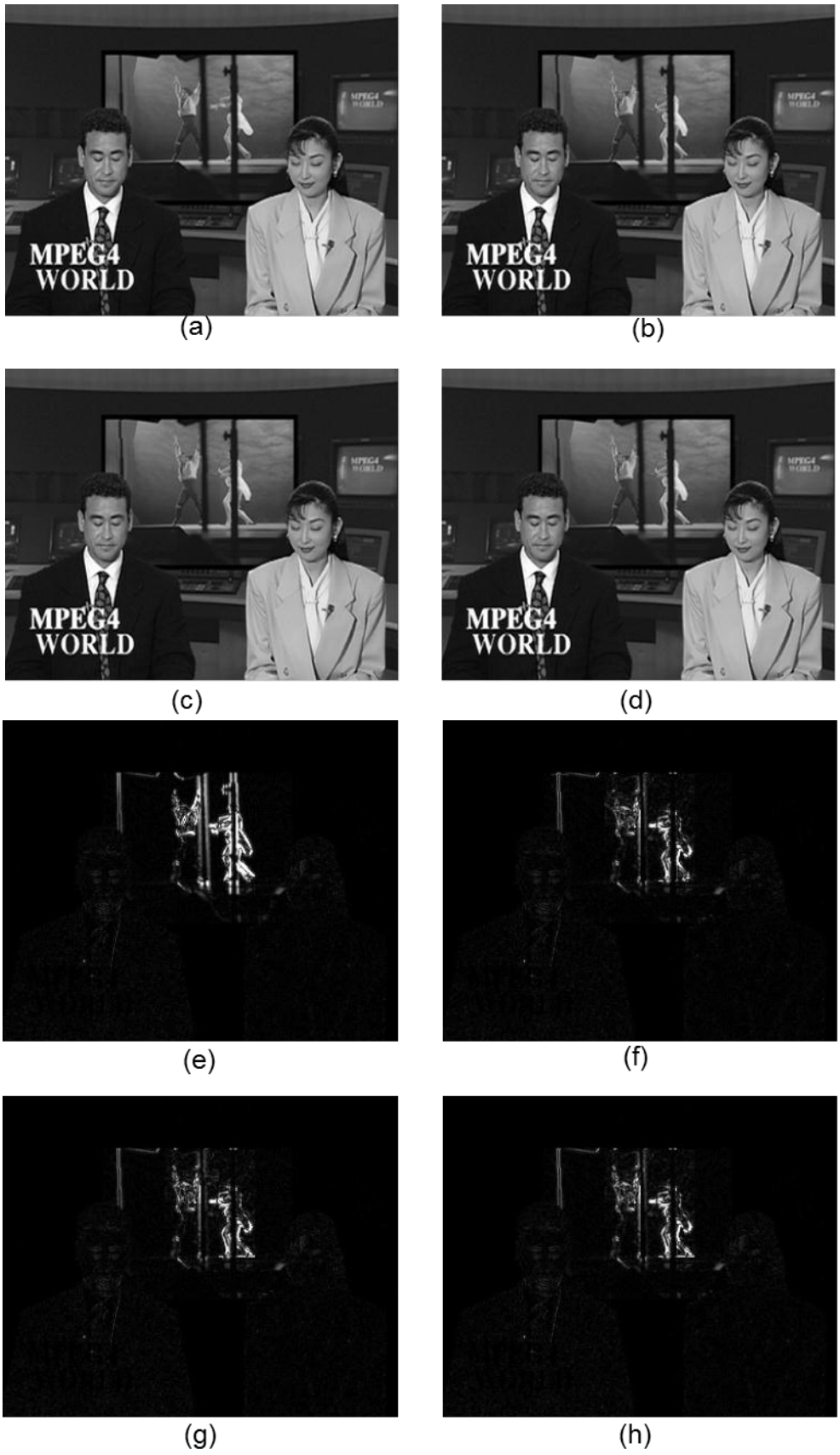


**Figure 6.28: (a) Frame 50 of "News", (b) predicted frame using DS, (c) predicted frame using ARPS, (d) predicted frame using MPBM, (e) the difference error between frame 50 and its reference frame 48, (f) the difference error between frame 50 and its predicted frame using DS, (g) the difference error between frame 50 and its predicted frame using ARPS and (h) the difference error between frame 50 and its predicted frame using the proposed MPBM**

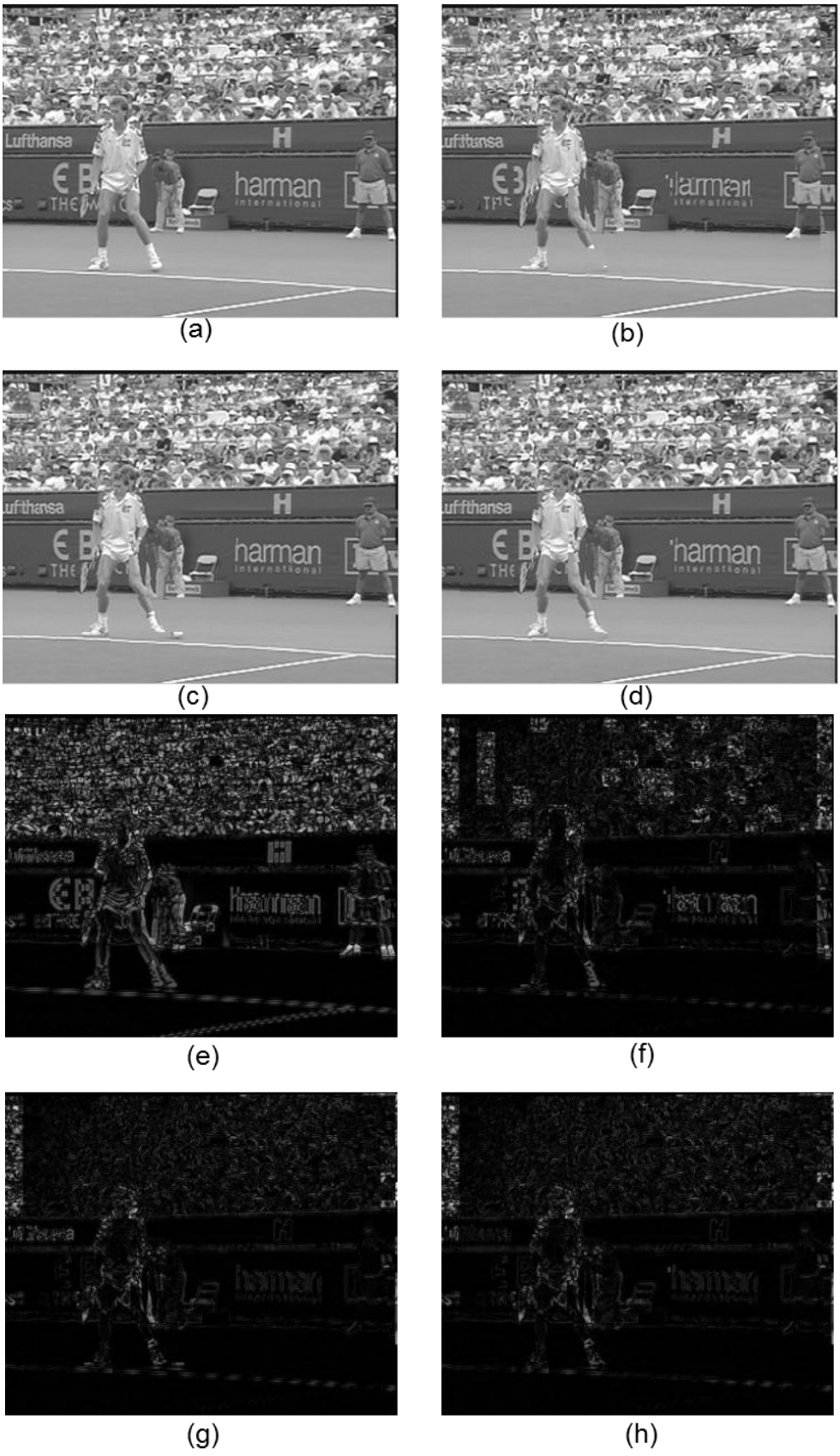


**Figure 6.29: (a) Frame 50 of "Stefan", (b) predicted frame using DS, (c) predicted frame using ARPS, (d) predicted frame using MPBM, (e) the difference error between frame 50 and its reference frame48, (f) the difference error between frame 50 and its predicted frame using DS, (g) the difference error between frame 50 and its predicted frame using ARPS and (h) the difference error between frame 50 and its predicted frame using the proposed MPBM**

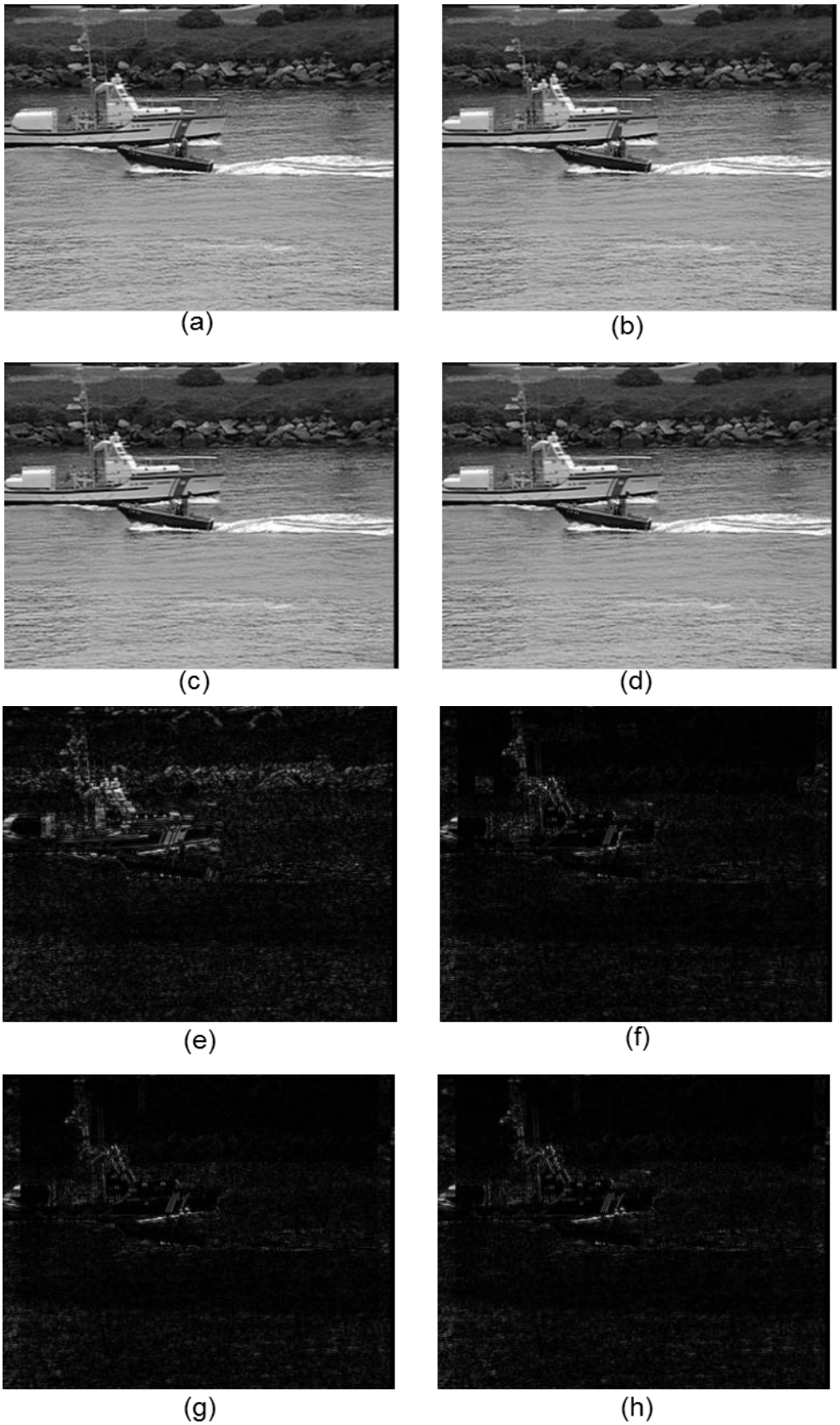


**Figure 6.30: (a) Frame 50 of "Coastguard", (b) predicted frame using DS, (c) predicted frame using ARPS, (d) predicted frame using MPBM, (e) the difference error between frame 50 and its reference frame 48, (f) the difference error between frame 50 and its predicted frame using DS, (g) the difference error between frame 50 and its predicted frame using ARPS and (h) the difference error between frame 50 and its predicted frame using the proposed MPBM**

## 6.4 Simulation Results of Applying Partial Distortion Elimination Technique to Existing Fast Block Matching Estimation

This section shows the simulation results of applying PDE to some of the existing fast block matching estimation techniques including Diamond Search and New Three Step Search, which are called PDE Diamond Search (PDEDS) and PDE New Three Step Search (PDENTSS) respectively [Ahmed et al., 2011a]. This has been done to enhance the time needed for processing without affecting the resolution of the predicted frames that have been built by these algorithms. The time needed to process these new techniques and the MPBM algorithm are shown in Table 6.11 while mean PSNR is shown in Table 6.12.

**Table 6.11: The simulation results of average time in seconds needed to process 50 frames**

| Sequence | Format | DS | PDEDS | NTSS | PDENTSS | MPBM |
|---|---|---|---|---|---|---|
| **Claire** | QCIF | 0.04 | 0.04 | 0.032 | 0.021 | 0.015 |
| **Akiyo** | QCIF | 0.03 | 0.03 | 0.029 | 0.011 | 0.006 |
| **Carphone** | QCIF | 0.04 | 0.05 | 0.035 | 0.027 | 0.033 |
| **News** | CIF | 0.16 | 0.14 | 0.137 | 0.075 | 0.079 |
| **Stefan** | CIF | 0.25 | 0.34 | 0.221 | 0.203 | 0.139 |
| **Coastguard** | CIF | 0.25 | 0.33 | 0.230 | 0.2 | 0.14 |

**Table 6.12: The simulation results of mean PSNR for 50 frames**

| Sequence | Format | DS | PDEDS | NTSS | PDENTSS | MPBM |
|---|---|---|---|---|---|---|
| **Claire** | QCIF | 38.94 | 38.94 | 38.94 | 38.94 | 38.94 |
| **Akiyo** | QCIF | 39.61 | 39.61 | 39.61 | 39.61 | 39.61 |
| **Carphone** | QCIF | 30.69 | 30.69 | 30.7 | 30.7 | 30.6 |
| **News** | CIF | 33.45 | 33.45 | 33.63 | 33.63 | 33.56 |
| **Stefan** | CIF | 21.49 | 21.49 | 21.81 | 21.81 | 21.93 |
| **Coastguard** | CIF | 25.98 | 25.98 | 26.05 | 26.05 | 26.11 |

As can be noted from Table 6-10, PDE enhanced the processing time when used for NTSS, and had an approximately similar processing time when applied to DS. This is due to the condition statements used in the PDE algorithm to stop the research early and hence enhance the time; however, if the global minimum matching MBl is not detected early in the search, this will lead to longer processing time. On the other hand, the proposed MPBM algorithm provides the best time and resolution values in comparison to PDEDS and PDENTSS for slow and fast motion activity video sequences, as demonstrated in Table 6.11 and Table 6.12.

## 6.5 Enhanced Mean Predictive Block Matching Algorithm (EMPBM)

This section illustrates the performance of the EMPBM technique with the MPBM and the six standard algorithms as shown in section 6.3.Video frames are divided into *4×4* MBls since the edge detection method required *4×4* MBl to work effectively. The same search range of ±7 is utilised. The SAD and MAD are both used as the BDMs.

The results of the computational complexity measured by the average number of search points required to detect each motion vector are shown in Table 6.13 and the average time needed for processing is shown in Table 6.14. The simulation results of mean of MAD and mean PSNR are explained in Table 6.15 and Table 6.16, respectively.

The simulation results of this algorithm show improvement in computational complexity when compared with the MPBM. Also, the resolution of the predicted frame using EMPBMA is nearly the same as for the one using MPBMA.

These results show that the motion activity of video sequences did not affect the computational complexity of the proposed algorithm or the resolution of the predicted frames in comparison to MPBM. This is due to the similarity between these two algorithms.

Figure 6.31 to Figure 6.36 illustrate the frame-by-frame comparison of average number of search points per MBl, PSNR performance and MAD of EMPBM, MPBM, ES, and ARPS for 23 frames of the tested videos, respectively.

For each video sequence, the visual images illustrated from Figure 6.37 to Figure 6.42 describe the performance of the proposed technique at frame 50and its predicted frame

from reference frame 48 using the block matching motion estimation EMPBM, MPBM, ES, and ARPS. In each figure, image (a) represents frame 50 while images (b), (c) and (d) represent the prediction of frame 50 from frame 48 as a reference frame by using FS, MPBM and EMPBM, respectively.

**Table 6.13: Average number of search points per MBl of size 4 ×4**

| Sequence | Format | FS | DS | NTSS | 4SS | SESTSS | ARPS | MPBM | EMPBM |
|---|---|---|---|---|---|---|---|---|---|
| **Claire** | QCIF | 210.1 | 15.64 | 19.2 | 18.3 | 16.74 | 8.188 | 2.49 | 1.95 |
| **Akiyo** | QCIF | 210.1 | 12.76 | 16.66 | 16.51 | 17.5 | 5.195 | 1.86 | 1.74 |
| **Carphone** | QCIF | 210.1 | 16.22 | 21.16 | 19.02 | 16.66 | 8.655 | 7.3 | 6.3 |
| **News** | CIF | 217.49 | 13.99 | 18.4 | 17.58 | 17.43 | 6.373 | 3.72 | 3.21 |
| **Stefan** | CIF | 217.49 | 18.18 | 24.25 | 20.49 | 16.44 | 10.18 | 9.67 | 8.54 |
| **Coastguard** | CIF | 217.49 | 19.06 | 27.97 | 20.92 | 16.66 | 10.65 | 10.4 | 9.25 |

**Table 6.14: The simulation results of average time in seconds needed to process 50 frames**

| Sequence | Format | FS | DS | NTSS | 4SS | SESTSS | ARPS | MPBM | EMPBM |
|---|---|---|---|---|---|---|---|---|---|
| **Claire** | QCIF | 3.32 | 0.55 | 0.36 | 0.35 | 0.34 | 0.44 | 0.16 | 0.12 |
| **Akiyo** | QCIF | 3.23 | 0.38 | 0.29 | 0.29 | 0.34 | 0.27 | 0.07 | 0.05 |
| **Carphone** | QCIF | 3.22 | 0.49 | 0.37 | 0.34 | 0.33 | 0.4 | 0.32 | 0.3 |
| **News** | CIF | 13.3 | 1.7 | 1.3 | 1.25 | 1.33 | 1.31 | 0.74 | 0.69 |
| **Stefan** | CIF | 13.2 | 2.51 | 1.91 | 1.57 | 1.3 | 1.77 | 1.56 | 1.53 |
| **Coastguard** | CIF | 13.5 | 2.5 | 2.11 | 1.62 | 1.34 | 1.96 | 1.86 | 1.85 |

**Table 6.15: The simulation results of mean MAD for 50 frames**

| Sequence | Format | FS | DS | NTSS | 4SS | SESTSS | ARPS | MPBM | EMPBM |
|---|---|---|---|---|---|---|---|---|---|
| **Claire** | QCIF | 0.91 | 0.96 | 0.952 | 0.984 | 1.02 | 0.97 | 1.01 | 1.02 |
| **Akiyo** | QCIF | 0.69 | 0.71 | 0.7 | 0.73 | 0.75 | 0.71 | 0.71 | 0.71 |
| **Carphone** | QCIF | 2.39 | 2.61 | 2.586 | 2.794 | 16.7 | 2.68 | 2.64 | 2.65 |
| **News** | CIF | 1.09 | 1.17 | 1.18 | 1.207 | 1.28 | 1.21 | 1.19 | 1.21 |
| **Stefan** | CIF | 7.08 | 9.32 | 8.541 | 9.447 | 10.5 | 8.61 | 8.27 | 8.31 |
| **Coastguard** | CIF | 5.63 | 7.17 | 6.44 | 7.031 | 7.37 | 6.47 | 6.33 | 6.33 |

**Table 6.16: The simulation results of mean PSNR for 50 frames**

| **Sequence** | Format | FS | DS | NTSS | 4SS | SESTSS | ARPS | MPBM | EMPBM |
|---|---|---|---|---|---|---|---|---|---|
| **Claire** | QCIF | 40.61 | 40.34 | 40.43 | 39.83 | 39.2 | 40.3 | 40.3 | 40.3 |
| **Akiyo** | QCIF | 41.73 | 41.41 | 41.49 | 40.93 | 40.44 | 41.4 | 41.4 | 41.4 |
| **Carphone** | QCIF | 34.12 | 33.35 | 33.51 | 32.73 | 31.82 | 33 | 33.2 | 33.2 |
| **News** | CIF | 38.21 | 37.09 | 37.18 | 36.87 | 35.86 | 36.7 | 37 | 36.8 |
| **Stefan** | CIF | 26.26 | 23.52 | 24.71 | 23.73 | 22.74 | 24.6 | 25 | 25 |
| **Coastguard** | CIF | 29.46 | 27.05 | 28.03 | 27.24 | 26.71 | 28.1 | 28.3 | 28.4 |

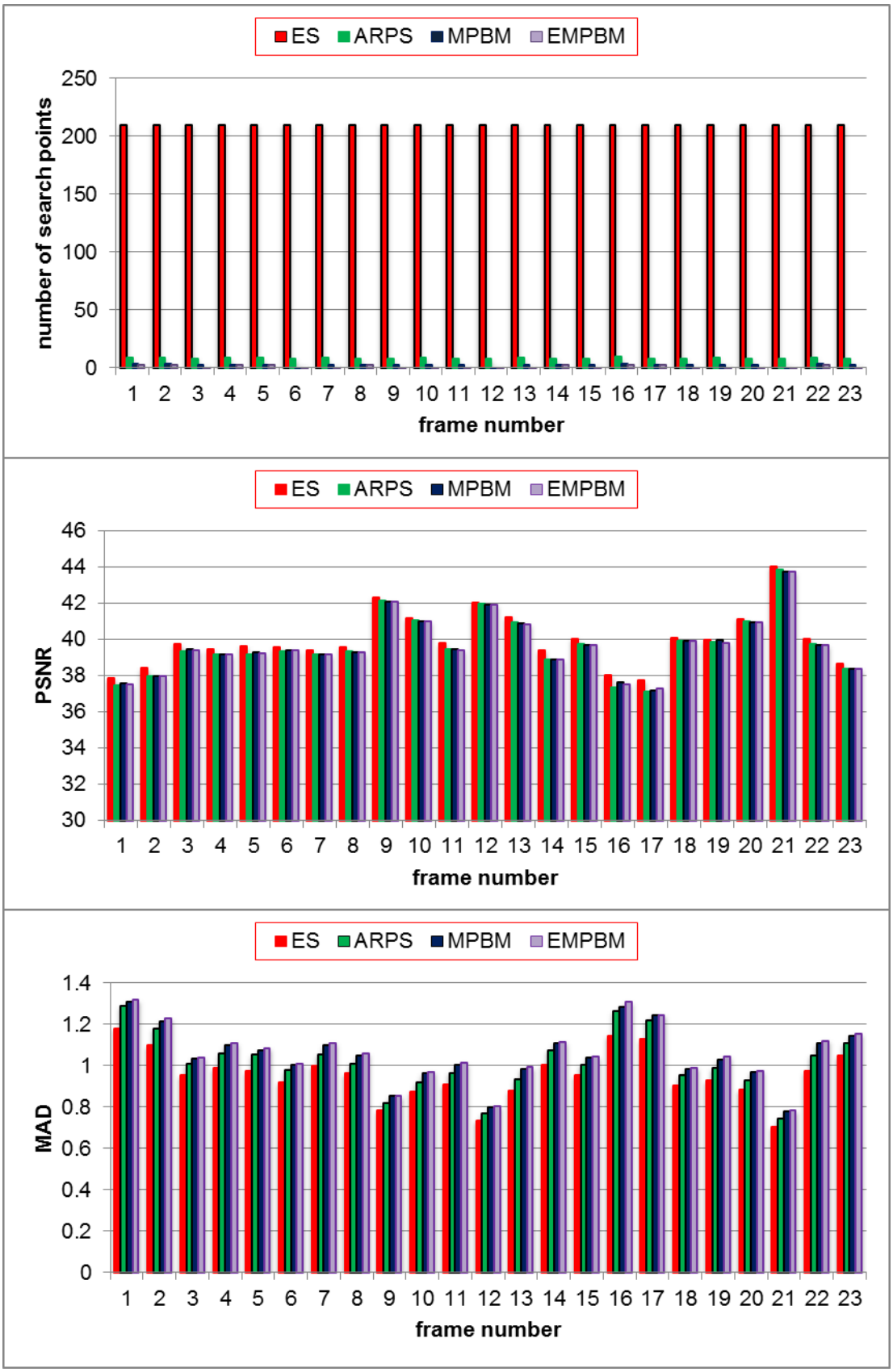


**Figure 6.31: Average number of search points per MBl, PSNR performance and MAD of EMPBM , MPBM, ES, and ARPS in "Claire" video sequence of 23 frames**

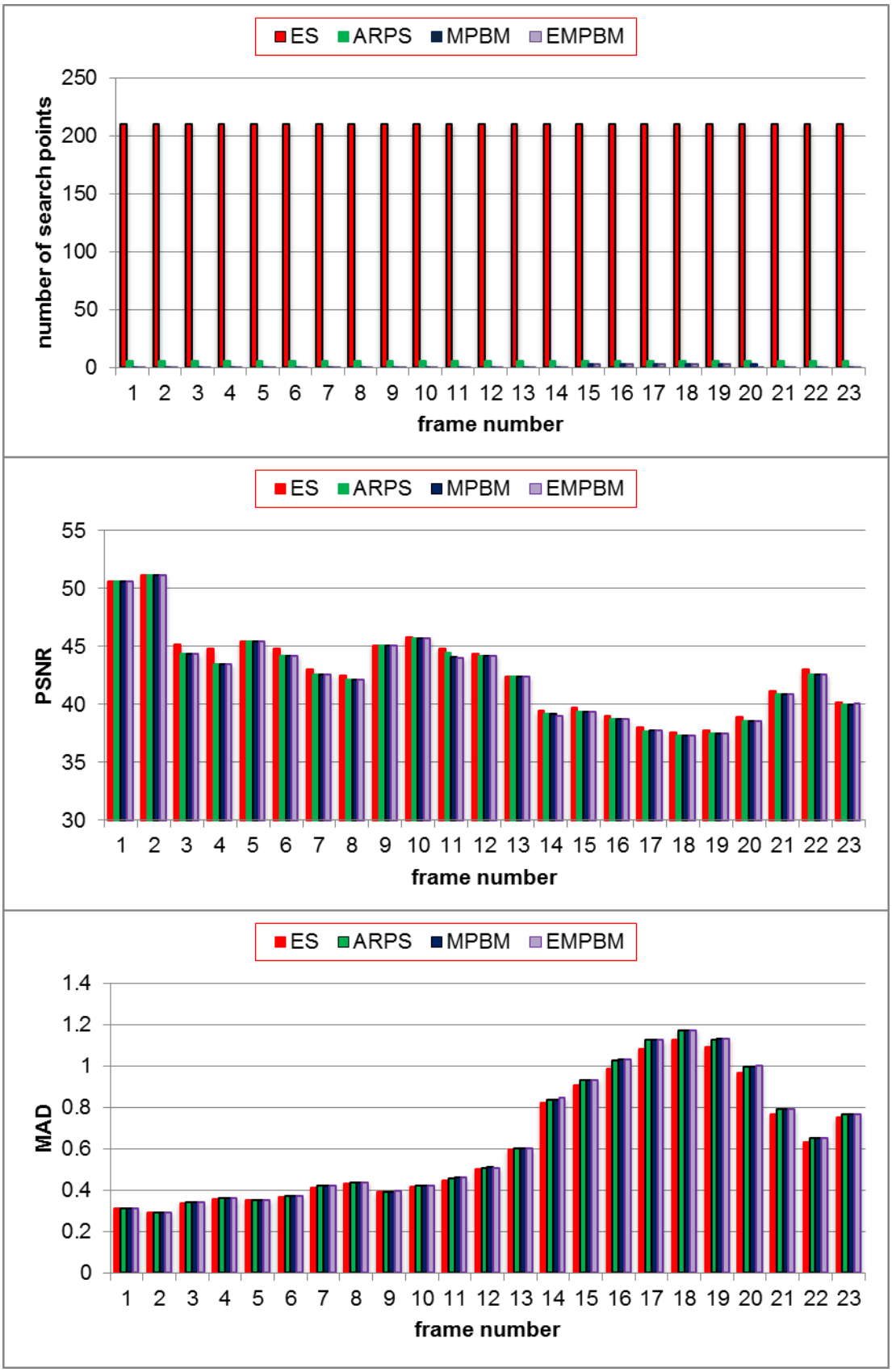


**Figure 6.32: Average number of search points per MBl, PSNR performance and MAD of EMPBM , MPBM, ES, and ARPS in "Akiyo" video sequence of 23 frames**

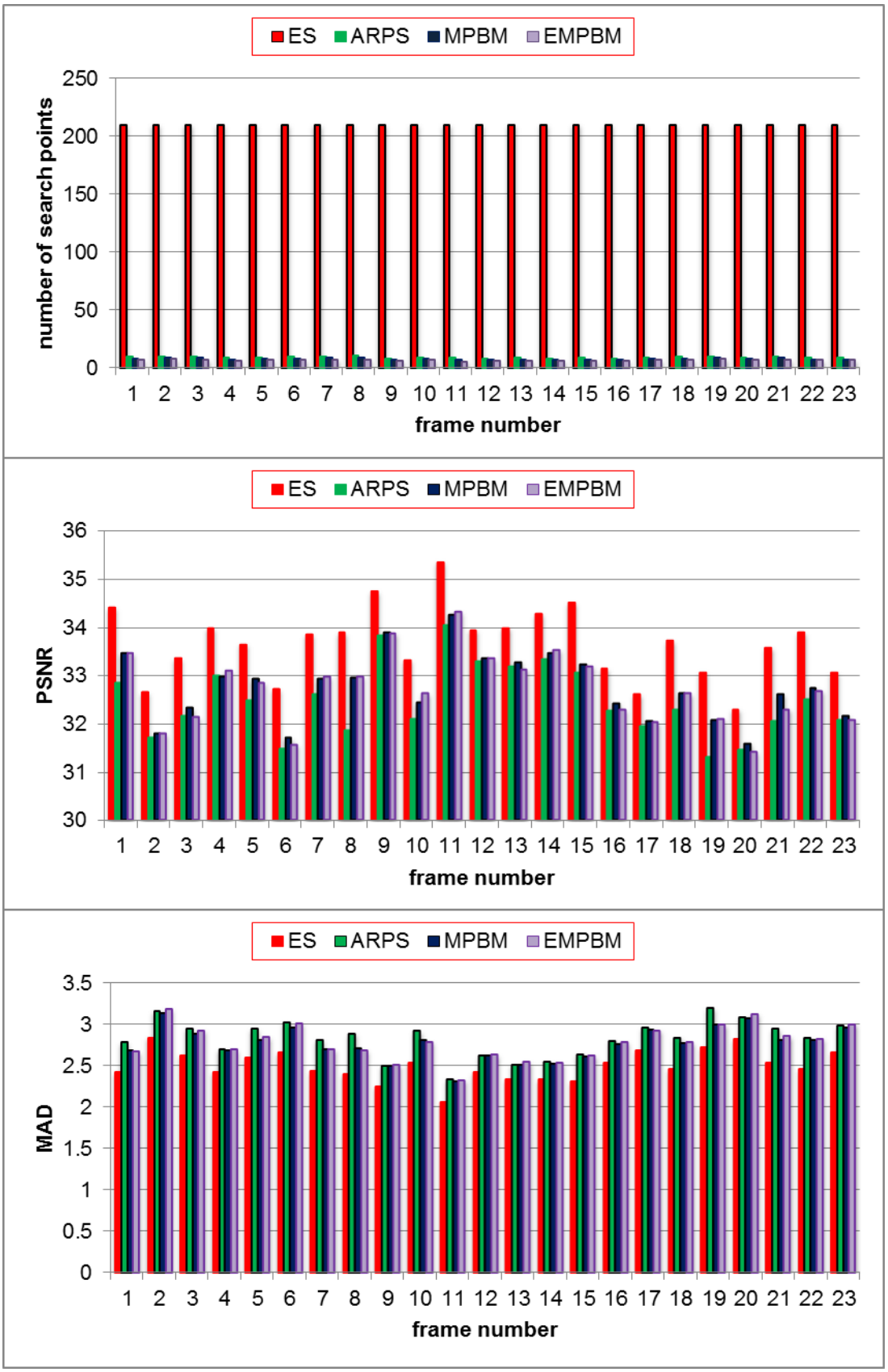


**Figure 6.33: Average number of search points per MBl, PSNR performance and MAD of EMPBM , MPBM, ES, and ARPS in "Carphone" video sequence of 23 frames**

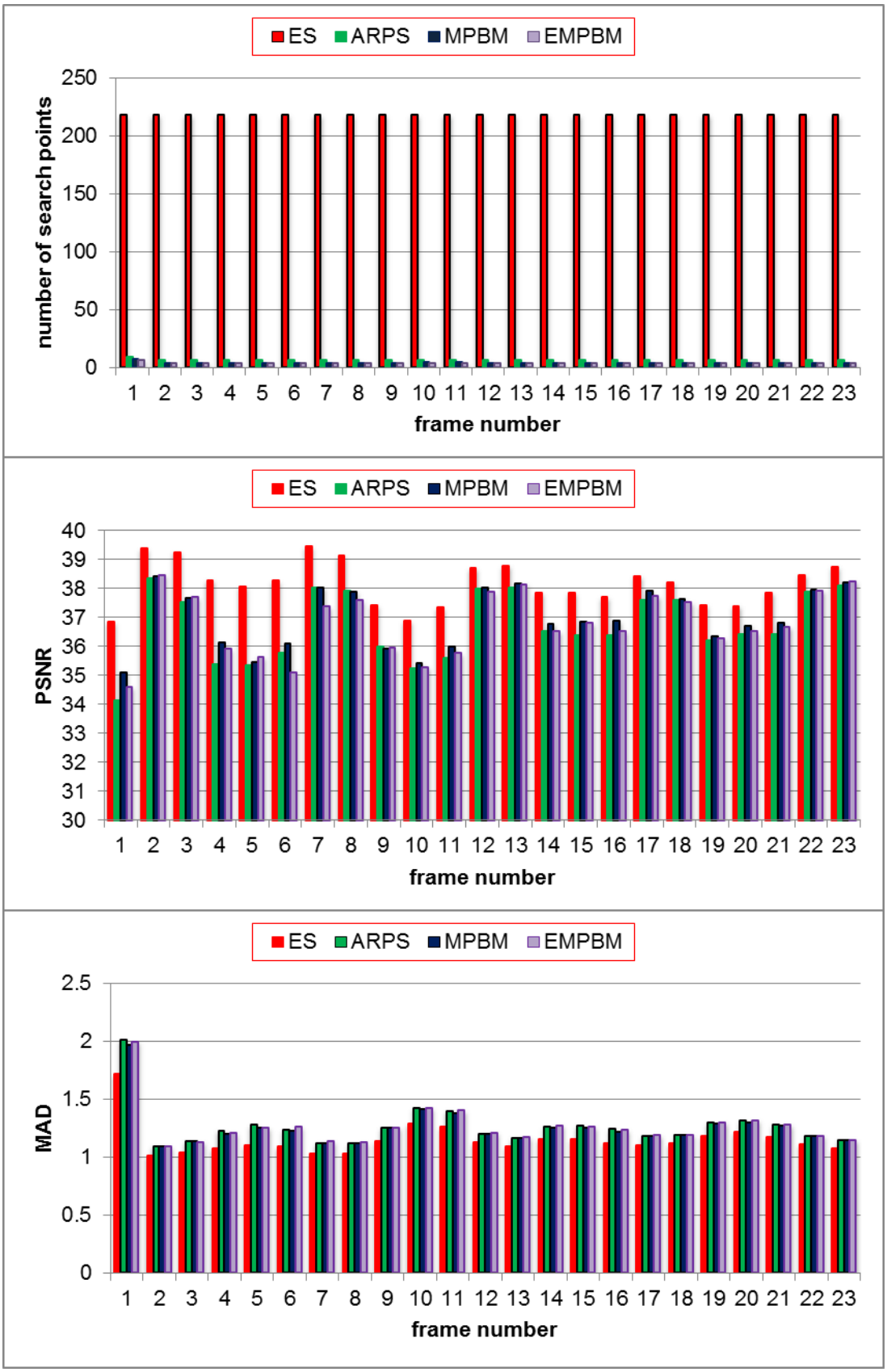


**Figure 6.34: Average number of search points per MBl, PSNR performance and MAD of EMPBM , MPBM, ES, and ARPS in "News" video sequence of 23 frames**

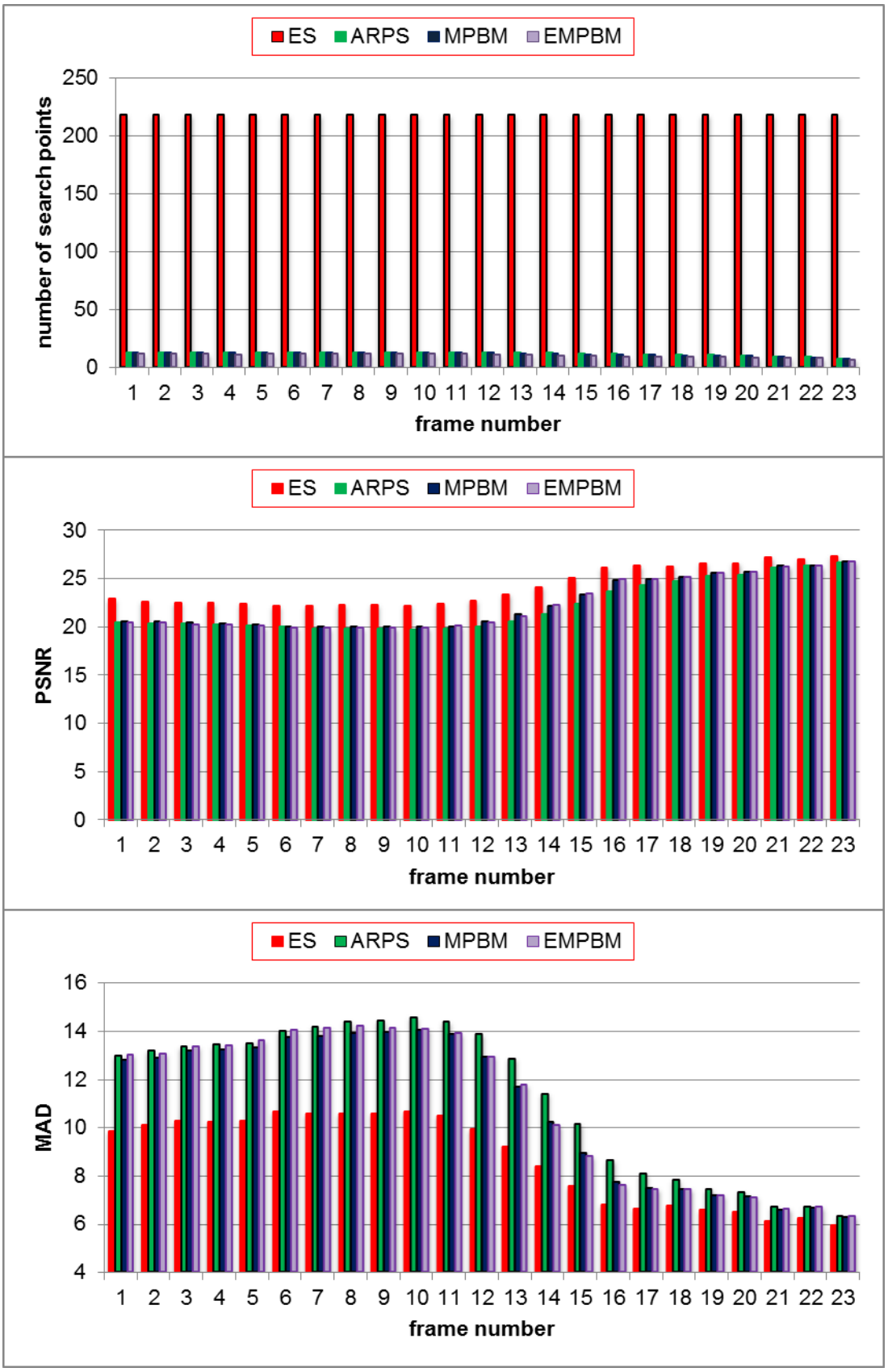


**Figure 6.35: Average number of search points per MBl, PSNR performance and MAD of EMPBM , MPBM, ES, and ARPS in "Stefan" video sequence of 23 frames**

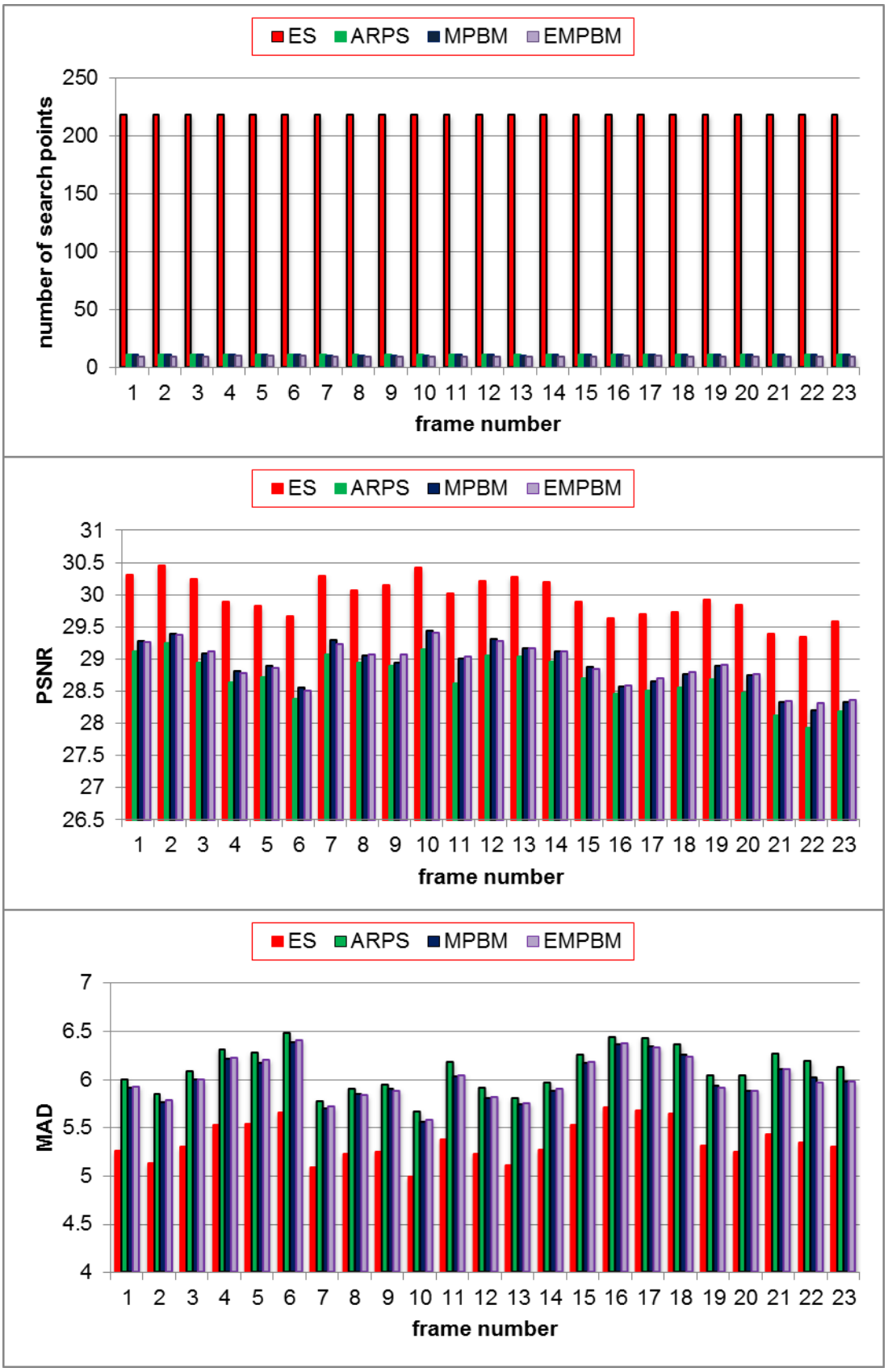


**Figure 6.36: Average number of search points per MBl, PSNR performance and MAD of EMPBM , MPBM, ES, and ARPS in "Coastguard" video sequence of 23 frames**

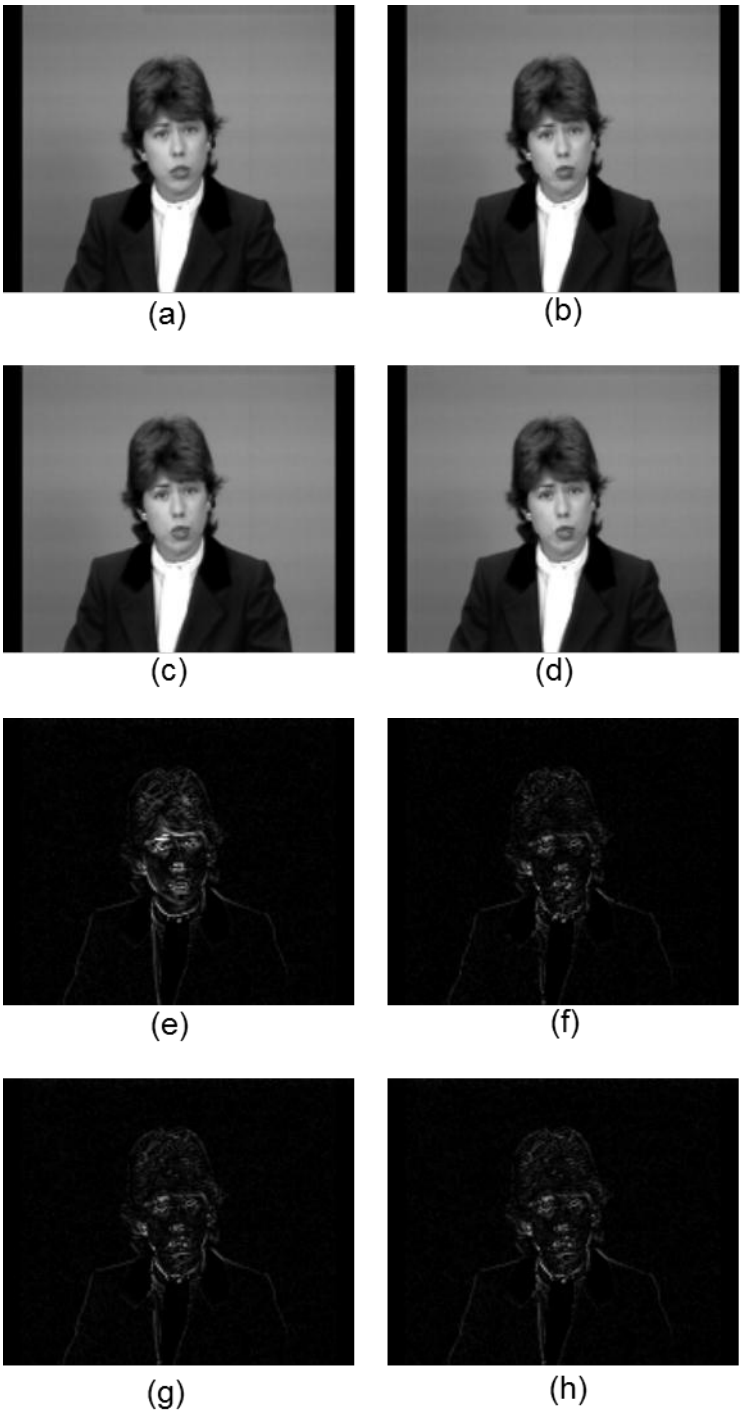


**Figure 6.37: MBl size 4×4 (a) Frame 50 of "Claire", (b) predicted frame using FS, (c) predicted frame using MPBM, (d) predicted frame using EMPBM, (e) the difference error between frame 50 and its reference frame 48, (f) the difference error between frame 50 and its predicted frame using FS, (g) the difference error between frame 50 and its predicted frame using MPBM, (h) the difference error between frame 50 and its predicted frame using the proposed EMPBM**

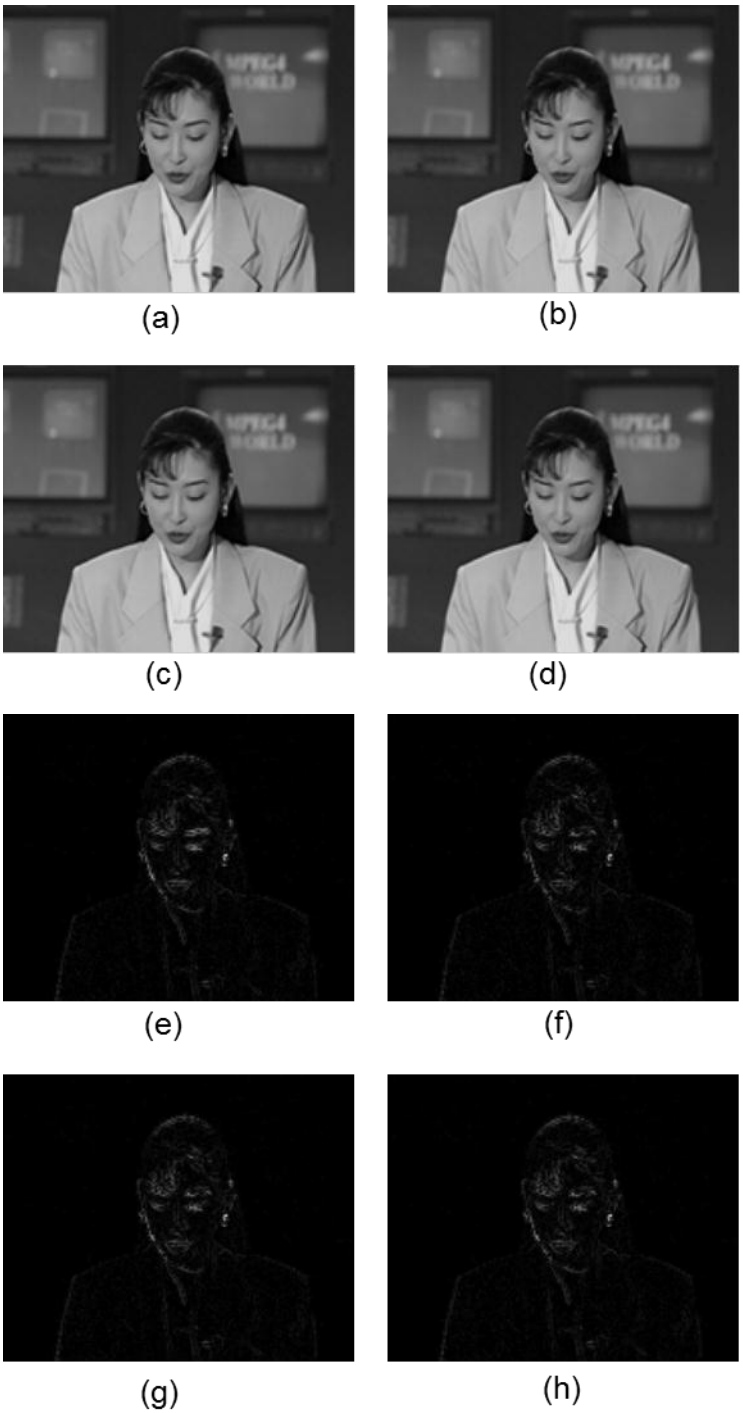


**Figure 6.38: MBl size 4×4 (a) Frame 50 of "Akiyo", (b) predicted frame using FS, (c) predicted frame using MPBM, (d) predicted frame using EMPBM, (e) the difference error between frame 50 and its reference frame 48, (f) the difference error between frame 50 and its predicted frame using FS, (g) the difference error between frame 50 and its predicted frame using MPBM, (h) the difference error between frame 50 and its predicted frame using the proposed EMPBM**

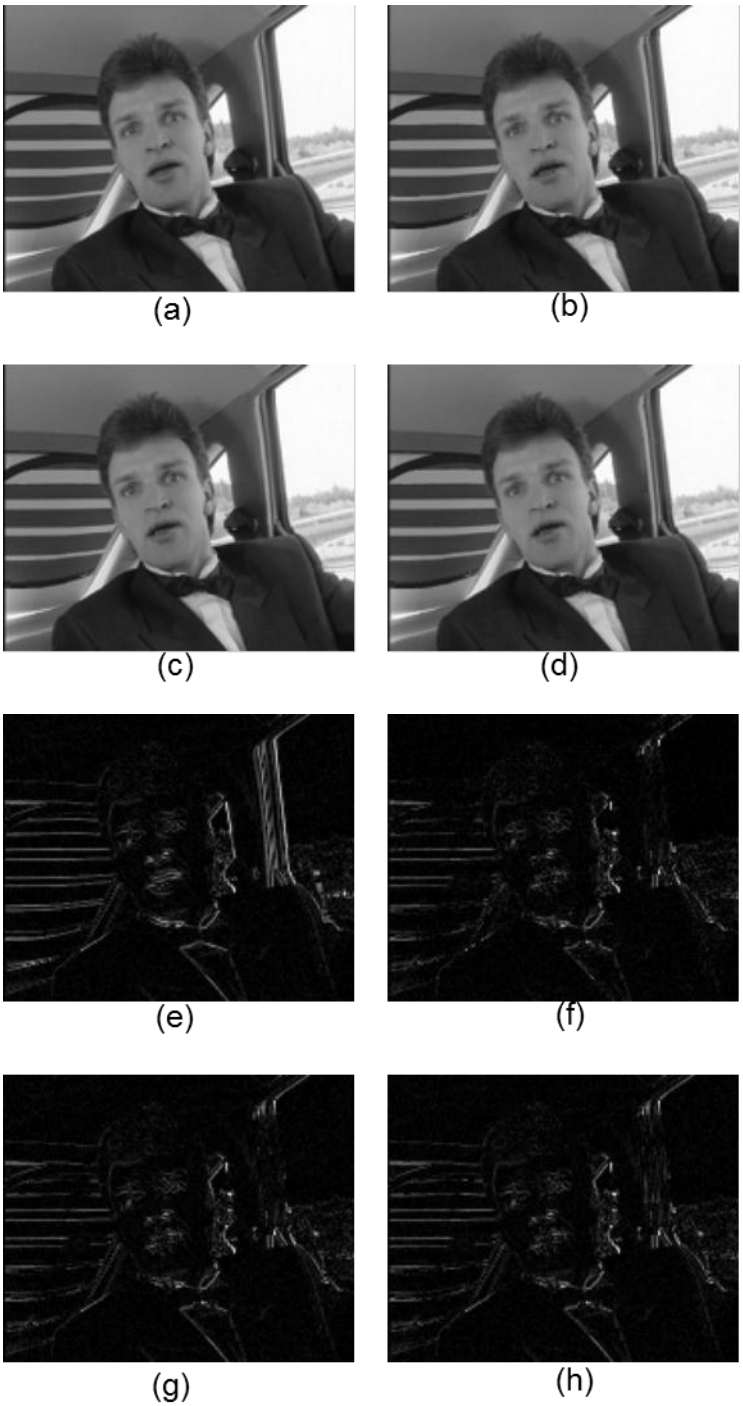


**Figure 6.39: MBl size 4×4 (a) Frame 50 of "Carphone", (b) predicted frame using FS, (c) predicted frame using MPBM, (d) predicted frame using EMPBM, (e) the difference error between frame 50 and its reference frame 48, (f) the difference error between frame 50 and its predicted frame using FS, (g) the difference error between frame 50 and its predicted frame using MPBM, (h) the difference error between frame 50 and its predicted frame using the proposed EMPBM**

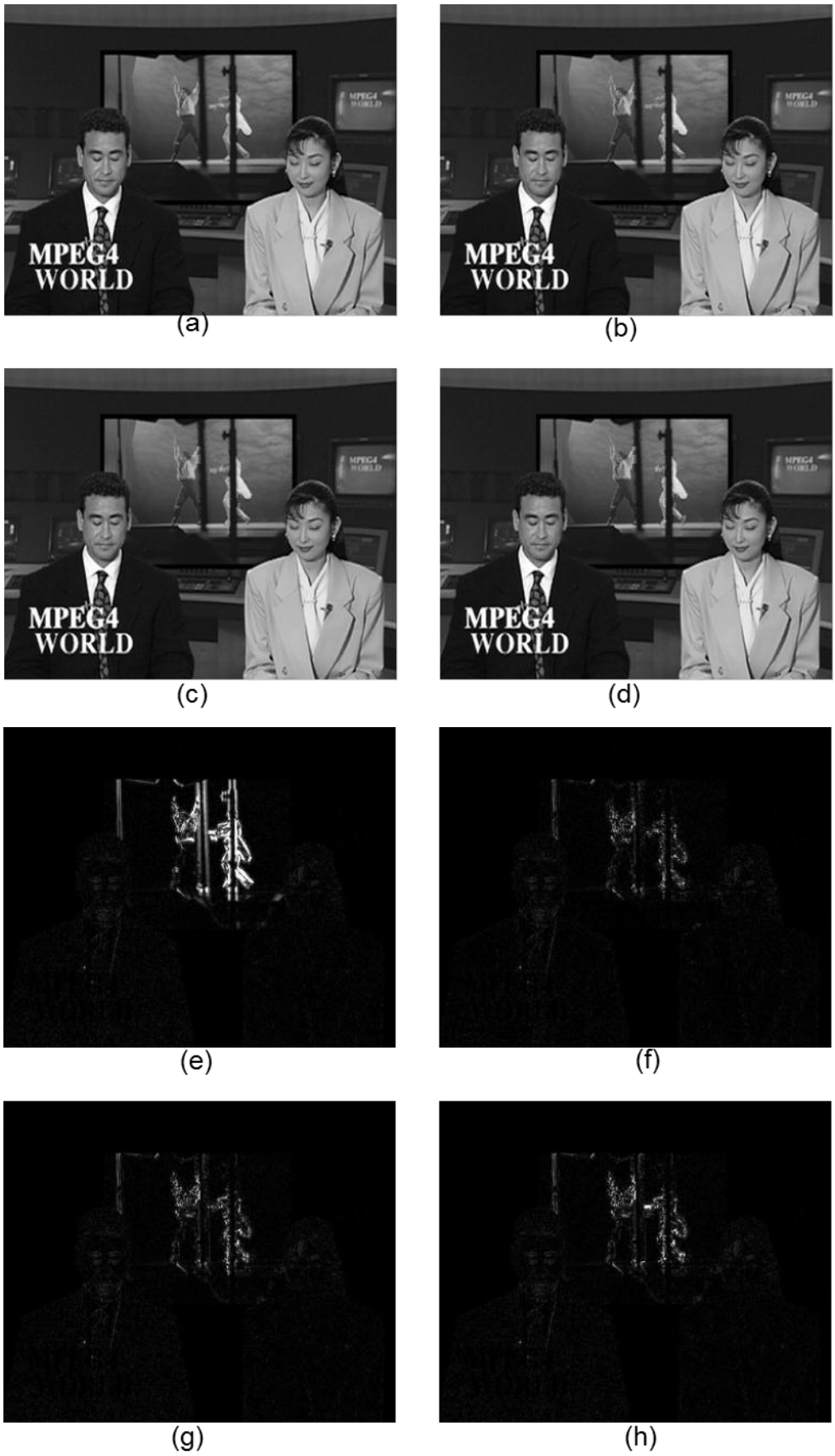


**Figure 6.40: MBl size 4×4 (a) Frame 50 of "News", (b) predicted frame using FS, (c) predicted frame using MPBM, (d) predicted frame using EMPBM, (e) the difference error between frame 50 and its reference frame 48, (f) the difference error between frame 50 and its predicted frame using FS, (g) the difference error between frame 50 and its predicted frame using MPBM, (h) the difference error between frame 50 and its predicted frame using the proposed EMPBM**

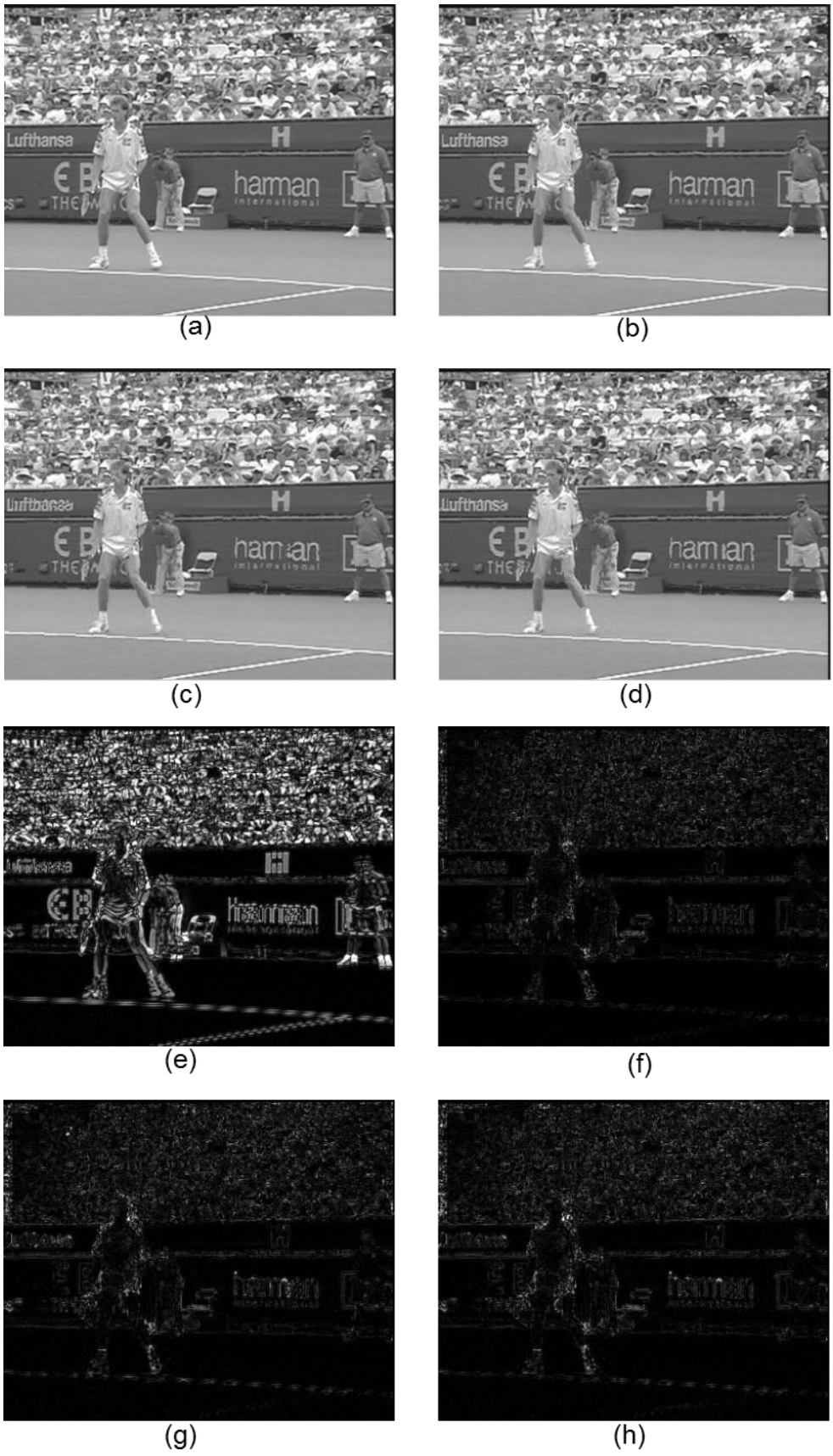


**Figure 6.41: MBl size 4×4 (a) Frame 50 of "Stefan", (b) predicted frame using FS, (c) predicted frame using MPBM, (d) predicted frame using EMPBM, (e) the difference error between frame 50 and its reference frame 48, (f) the difference error between frame 50 and its predicted frame using FS, (g) the difference error between frame 50 and its predicted frame using MPBM, (h) the difference error between frame 50 and its predicted frame using the proposed EMPBM**

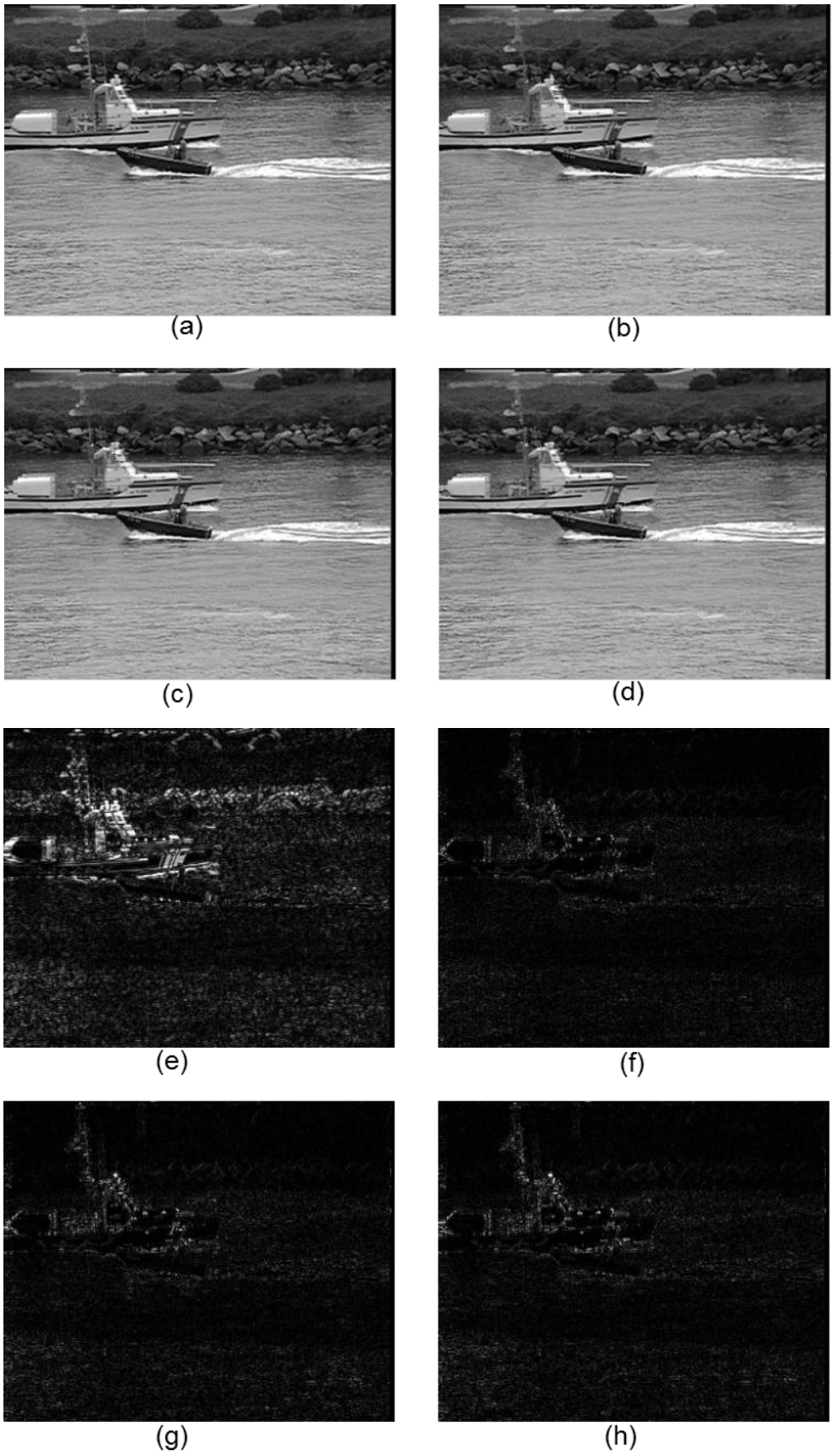


**Figure 6.42: MBl size 4×4 (a) Frame 50 of "Coastguard", (b) predicted frame using FS, (c) predicted frame using MPBM, (d) predicted frame using EMPBM, (e) the difference error between frame 50 and its reference frame 48, (f) the difference error between frame 50**

**and its predicted frame using FS, (g) the difference error between frame 50 and its predicted frame using MPBM, (h) the difference error between frame 50 and its predicted frame using the proposed EMPBM**

## 6.6 Chapter Summary

This chapter introduced the simulation results for the proposed algorithms. The simulations indicate that, for lossless BMA, the novel technique Fast Computations of Full Search Block Matching Motion Estimation reduces the search time of the macroblock matching, while keeping the resolution of the predicted frames the same as the one predicted using full search. Moreover, this technique is more effective if the video sequences have lower motion activity and vice versa. This is due to using two previous neighbours to predict the dimension of the new search window which has a high probability to contain the global matching MBl.

For lossy BMA, the simulation results indicated that the Mean Predictive Block Matching Algorithm shows improvement in the computational complexity; also, it tries to keep or reduce the error between the current and compensated frames when benchmarked with the standard BMA. For low motion activity video sequences, the resolution of the predicted frame is close to the ones predicted by full search and there is enhancement in the computational complexity; while for medium and high motion activity video sequences, the improvement of the computational complexity and the resolution of the predicted frame are acceptable in comparison with other fast block matching algorithms. Moreover, the simulation result of applying partial distortion elimination to two selected standard algorithms, which are DS and NTSS, indicated that the proposed techniques EDS and ENTSS improve the processing time needed without affecting the resolution of the predicted frames that have been built by these algorithms. Also, these algorithms show improvement for the medium motion activity video sequences in comparison to MPBM, but the resolution of the predicted frames built by these algorithms is not as much as for the one built by MPBM; while, for the low and high motion activity video sequences, MPBM still gives the best results.

Finally, the simulations of the Enhanced Mean Predictive Block Matching algorithm indicate that using edge detection could improve computational complexity when compared with the MPBM. Also, it should be noted that the resolution of compensated

frames built by the proposed technique attempts to be the same as the one built by MPBM or is sometimes enhanced. The motion activity of video sequences did not affect the computational complexity of the proposed algorithm and the resolution of the predicted frames built by it.

# CHAPTER 7: CONCLUSIONS AND FUTURE WORK

The main idea of video compression techniques is to remove the redundant information that exists in video sequences in order to be stored or transmitted. Inter-frame encoding is the main coding tool for removing temporal redundancy in video sequences. In inter-frame encoding the current frame can be predicted from the reference frames. Motion estimation is the technique used to estimate the motion of the moving objects from one location in the current frame to another in the reference frame. Block Matching Algorithm (BMA) is a practical and widely used method to carry out frame prediction. It is the most computationally intensive part in video compression. Therefore, decreasing this complexity has caught the attention of many researchers. Various techniques of Fast Block Matching algorithms (FBMAs) that reduce the huge computational complexity are reviewed in this thesis. These techniques are classified into lossy and lossless block matching algorithms.

The aim of this research work is to develop novel algorithms for the purpose of improving the computational complexity of FBMA in comparison to the existing FBMA.

In this chapter, the conclusions about this research work including the contributions and future research directions will be demonstrated.

## 7.1 Research Contributions

This thesis makes a number of research contributions related to fast block matching algorithms. Novel algorithms were developed to improve the computational complexity of both lossless and lossy block matching algorithms. Key contributions of this research work can be summarised as:

- Using the mean value of two motion vectors which are the above and the left neighbouring macroblocks: the proposed video compression techniques take advantage of the fact that the general motion in any video frame is usually coherent [Barjatya, 2004]. This coherent nature of the video frames dictates a probability of a macroblock having the same direction of motion as the macroblocks surrounding it. Therefore, two previous neighbouring MBls (above

and left) have been used to predict the first step of the search process. The aim of using these neighbouring MBls is to speed up the process of finding the global matching MBl and to avoid unnecessary computations related to choosing three previous neighbouring MBls. To aid their initial calculations, the proposed techniques use the mean value of the motion vectors of these macroblocks.

- The Partial Distortion Elimination algorithm is used to reduce the search time: using the predictor MVs led to increasing the probability of finding the global minimum in the first search. Hence, the Partial Distortion Elimination algorithm is used to enhance and improve the time needed for processing.
  Also, the Partial Distortion Elimination algorithm technique has been used to improve the time needed for processing two standard fast block matching algorithms without affecting the quality of the compensated frames.

- For the lossless BMA, the performance of the proposed Fast Computations of Full Search is evaluated using the initial calculation to determine the new search window. The new search window will contain the global minimum, hence, applying the Partial Distortion Elimination algorithm speeds up the search process. Moreover, the rest of the main search windows will be ignored when the error of the matching macroblock from this search is small.

- For the lossy BMA, two novel techniques, Mean Predictive Block Matching and Enhanced Mean Predictive Block Matching algorithms, are illustrated. The first technique combines three types of fast block matching algorithm: predictive search technique, fixed set of search patterns, and partial distortion elimination algorithm. This algorithm uses previous neighbouring macroblocks to determine the initial step size search pattern. Seven positions will be examined in the first step and five positions later in which the partial distortion elimination algorithm is applied.
  The second technique attempts to improve the Mean Predictive Block Matching algorithm by classifying the current macroblock into *shade* and *edge*. The shade macroblock has a probability to move in the same direction as its neighbouring macroblocks. This has led to examining only the motion vectors of the neighbouring macroblocks and ignoring other motion vectors that were utilised

in the first search step of the Mean Predictive Block Matching algorithm. For the edge macroblock, the proposed technique uses the same approach that was used in the Mean Predictive Block Matching algorithm.

- The edge detection technique used to classify MBls has been built in as simple a way as possible to avoid more computations. In spite of this algorithm making an improvement to MPBM, the performance of this technique needs to be compared with the existing one.
- The simulation results of various video sequence types indicated that the novel techniques showed improved results in comparison to the benchmarked lossless and lossy block matching algorithms. This improvement is measured in terms of the processing time for lossless block matching algorithm; while, for lossy block matching algorithms, the novel techniques decrease both the average number of search points required per macroblock for the videos and the residual prediction error in comparison to the standard fixed set of search pattern of block matching algorithms.

## 7.2 Future Research Directions

This section considers a number of possible future directions to improve the performance of the proposed techniques and extend their application. The research work achieved for this thesis could be continued by investigating the following items:

- The efficiency of the proposed architecture is determined by using the mean value of the two motion vectors for the above and left previous neighbouring macroblocks. This process has been designed to support the initial search, hence improving the computational complexity of BMAs. One of the possible areas of future research is related to the use of this process to enhance the performance of the existing fast BMAs and then compare their efficiency. Moreover, since the VBSME has become the default of video coding standards, therefore, the efficiency will be more effective if the best algorithm determines the MVs using the VBSME instead of the FBSME that was used in this research work.

- There are two outputs from ME and MCP: RPE, which is the difference between the current frame and the predicted frame, and the MVs. These outputs are sent to the decoder to reconstruct MCP. The efficient compression could be satisfied by decreasing the RPE, which is another direction for future research. This could be achieved by rotating the best matching MBl in different directions if the error between current and best matching MBl is more than the threshold. The best matching MBl will be rotated with a degree of ±10º, ±20º, ±30º, and ±45º and can be compared with the current MBl. Each angle is represented by corresponding symbol. The symbol that represents the best matching rotated MBls should be sent to the decoder with the MV of the current MBl. This method can be useful by decreasing the transmitted error; hence high compression ratio will be achieved. This could also be used to enhance the resolution of the decompressed frame.

- In the proposed algorithms, each pixel in a luminance frame is represented with eight-bit resolution. To represent pixels with a single bit-plane, a one-bit transform (1BT) bit plane could be used [Jian et al., 1995]. It uses the mean of MBl as a threshold value to indicate whether a pixel is edge or not, as follows:

$$B(i,j) = \begin{cases} 1 & if\ I(i,j) \geq t_{bm} \\ 0, & otherwise \end{cases}$$

where $t_{bm}$ is the threshold value that is set equal to the MBl mean, $I(i,j)$ shows the $(i,j)th$ pixel of the image frame and $B(i,j)$ shows the corresponding bit-plane value.

Moreover, the error between current and candidate MBls will be calculated as the number of non-matching points (NNMP), which is measured by the exclusive-or (XOR) operation instead of MAD or SAD as in equation (4.5). A suggested research direction could be the idea of using a single bit-plane with the proposed fast BMAs as a search pattern instead of FS, to enhance the computational time of the BMAs.

- Using existing edge detection algorithms such as Canny edge detection or Sobel edge detection [Sharifi et al., 2002] to classify the MBls in EMPBM and compare the results.

# REFERENCES

AHMED, Z., HUSSAIN, A. J. & AL-JUMEILY, D. (4-6 July 2011). "Mean Predictive Block Matching [MPBM]". *IEEE in 3rd European Workshop On Visual Information Processing (EUVIP2011).*

AHMED, Z., HUSSAIN, A. J. & AL-JUMEILY, D. (2011a). "Enhanced Computation Time for Fast Block Matching Algorithm". *In Proc. IEEE Developments in E-systems Engineering (DeSE)*; pp.289-293.

AHMED, Z., HUSSAIN, A. J. & AL-JUMEILY, D. (2011b). "Mean Predictive Block Matching (MPBM) for fast block-matching motion estimation". *In Proc. IEEE 3rd European Workshop on Visual Information Processing (EUVIP)*; pp.67-72.

AHMED, Z., HUSSAIN, A. J. & AL-JUMEILY, D. (2012). "Edge detection for fast block-matching motion estimation to enhance Mean Predictive Block Matching algorithm". *In Proc. IEEE International Symposium on Innovations in Intelligent Systems and Applications (INISTA)*; pp.1-5.

AKRAM, M. & IZQUIERDO, E. (2010). "A Multi-Pattern Search Algorithm for Block Motion Estimation in Video Coding". *In Proc. IEEE 12th International Asia-Pacific Web Conference (APWEB)*; pp.407-410.

AL-MUALLA, M. E., CANAGARAJAH, C. N. & BULL, D. R. (2002). "Video Coding for Mobile Communications: Efficiency, Complexity and Resilience*"*: *Academic Press.*

ALI AL-FAYADH, A. J. H., PAULO LISBOA, AND DHIYA AL-JUMEILY (2009). "Novel hybrid classified vector quantization using discrete cosine transform for image compression ". *Journal of Electronic Imaging;* Vol.18(2).

ALZOUBI, H. & PAN, W. D. (2007). "Efficient Global Motion Estimation using Fixed and Random Subsampling Patterns". *IEEE International Conference on Image Processing ICIP 2007*; Vol.1; pp.I477- I480.

ANANTHASHAYANA, V. K. & PUSHPA, M. K. (2009). "Joint Adaptive Block Matching Search (JABMS) Algorithm for Motion Estimation". *International Journal of Recent Trends in Engineering;* Vol.2(2); pp.212-216.

BARJATYA, A. (2004). "Block Matching Algorithms for Motion Estimation". *Final Project Paper, DIP 6620.*

BARJATYA, A. ( DIP 6620 Spring 2004). "Block Matching Algorithms for Motion Estimation".

BHASKARAN, V. & KONSTANTINIDES, K. (1997). "Image and Video Compression Standards: Algorithms and Architecture*", 2nd* edition: *Kluwer Academic Publishers.*

BHATTACHARYYA, S. S. & DEPRETTERE, E. F. (2010). "Handbook of signal processing systems": *Springer*.

BOVIK, A. (2010). "Handbook of Image and Video Processing", *2nd* edition: *Academic Press*.

BOVIK, A. C. (2009). "Chapter 1 - Introduction to Digital Video Processing". *In: The Essential Guide to Video Processing*, 2nd edition; Boston: Academic Press.

BROSS, B., HAN, W. J., OHM, J. R., SULLIVAN, G. J. & WEINGAND, T. (2012). "High Efficiency Video Coding (HEVC), text specification draft 6". *Doc. JCTVC-H1003, Joint Collaborative Team on Video Coding (JCT-VC) of ITU-T VCEG and ISO/IEC MPEG, April;* Vol.21.

CAI, C., ZENG, H. & MITRA, S. (2009). "Fast motion estimation for H.264". *Signal Processing: Image Communication*.

CE, Z., XIAO, L., CHAU, L. & LAI-MAN, P. (2004). "Enhanced hexagonal search for fast block motion estimation". *IEEE Transactions on Circuits and Systems for Video Technology;* Vol.14(10); pp.1210-1214.

CHALIDABHONGSE, J. & KUO, C. C. J. (1997). "Fast motion vector estimation using multiresolution-spatio-temporal correlations". *IEEE Transactions on Circuits and Systems for Video Technology;* Vol.7(3); pp.477-488.

CHANG-DA, B. & GRAY, R. (1985). "An Improvement of the Minimum Distortion Encoding Algorithm for Vector Quantization". *IEEE Transactions on Communications;* Vol.33(10); pp.1132-1133.

CHANYUL, K. (2010). "*Complexity adaptation in video encoders for power limited platforms*". Dublin City University.

CHAO-FENG, T., YEN-TAI, L. & MENG-JE, L. (2012). "A VLSI architecture for three-step search with variable block size motion vector". *In Proc. IEEE 1st Global Conference on Consumer Electronics (GCCE)*; pp.628-631.

CHEUNG, C.-H. & PO, L.-M. (2002). "A novel cross-diamond search algorithm for fast block motion estimation". *IEEE Trans. Circuits Syst. Video Technol;* Vol.12(12); pp.1168- 1177.

EMARKETER (2013). "*Mobile, Video Drive Up Digital Ad Investment in the UK*". [Online]. Available: http://www.emarketer.com/Article/Mobile-Video-Drive-Up-Digital-Ad-Investment-UK/1010097; [Accessed 12.09. 2013].

ERTURK, S. (2007). "Multiplication-Free One-Bit Transform for Low-Complexity Block-Based Motion Estimation". *IEEE Signal Processing Letters;* Vol.14(2); pp.109-112.

ESSANNOUNI, F., THAMI, R. O. H., SALAM, A. & ABOUTAJDINE, D. (2006). "An efficient fast full search block matching algorithm using FFT algorithms". *IJCSNS International Journal of Computer Science a 130 nd Network Security;* Vol.6(3); pp.130-133.

EZHILARASAN, M. & THAMBIDURAI, P. (2008). "Simplified Block Matching Algorithm for Fast Motion Estimation in Video Compression ". *Journal of Computer Science;* Vol.4(4); pp.282-289.

GOEL, S. & BAYOUMI, M. A. (2006). "Multi-Path Search Algorithm for Block-Based Motion Estimation". *In Proc. IEEE International Conference on Image Processing*; pp.2373-2376.

GONZALEZ, R., WOODS, R. & EDDINS, S. (2009). "Digital Image processing using MATLAB*", 2nd* edition: *Gatesmark Publishing*.

GRECOS, C., SAPARON, A. & CHOULIARAS, V. (2004). "Three novel low complexity scanning orders for MPEG-2 full search motion estimation". *Real-Time Imaging;* Vol.10(1); pp.53-65.

HORN, B. & SCHUNCK, B. (1981). "Determining Optical Flow". *ARTIFICAL INTELLIGENCE;* Vol.17; pp.185-203.

HUANG, D.-Y. (2005). "*A XviD-based Video Codec for Computer Animation"*. MSC, National Central University.

HUANG, S.-Y. (2006). "Adaptive computation-aware scheme for software-based predictive block motion estimation". *Journal of Visual Communication and Image Representation;* Vol.17(4); pp.767-782.

HUANG, Y.-W., CHEN, C.-Y., TSAI, C.-H., SHEN, C.-F. & CHEN, L.-G. (2006). "Survey on Block Matching Motion Estimation Algorithms and Architectures with New Results". *The Journal of VLSI Signal Processing;* Vol.42(3); pp.297-320.

HUI-YU, H. & SHIH-HSU, C. (2011). "Block motion estimation based on search pattern and predictor". *In Proc. IEEE Symposium on Computational Intelligence for Multimedia, Signal and Vision Processing (CIMSIVP)*; pp.47-51.

HWAL-SUK, L., JIK-HAN, J. & DONG-JO, P. (2008). "An effective successive elimination algorithm for fast optimal block-matching motion estimation". *In Proc. IEEE 15th International Conference on Image Processing (ICIP)*; pp.1984-1987.

ISO/IEC (1993). "Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s – Part 2: video". *ISO/IEC 11172-2*.

ISO/IEC (1996). "Information technology – Generic coding of moving pictures and associated audio – Part 2: video". *ISO/IEC 13818-2*.

ISO/IEC (1999). "Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s". *ISO/IEC 11172-3*.

ITU-T & ISO/IEC (2003). "Advanced Video Coding for Generic Audiovisual Services". *H.264, MPEG, 14496-10*.

JAE-YONG, K. & SUNG-BONG, Y. (1999). "An efficient hybrid search algorithm for fast block matching in video coding". *Proceedings of the IEEE the Region 10 Conference TENCON 99*; Vol.1; pp.112-115.

JAIN, J. & JAIN, A. (1981). "Displacement Measurement and Its Application in Interframe Image Coding". *IEEE Transactions on Communications;* Vol.29(12); pp.1799-1808.

JIAN, F., KWOK-TUNG, L., MEHRPOUR, H. & KARBOWIAK, A. E. (1995). "Adaptive block matching motion estimation algorithm using bit-plane matching". *In Proc. IEEE International Conference on Image Processing*; Vol.3; pp.496-499.

JIANHUA, L. & LIOU, M. L. (1997). "A simple and efficient search algorithm for block-matching motion estimation". *IEEE Transactions on Circuits and Systems for Video Technology;* Vol.7(2); pp.429-433.

JIZHENG, X., FENG, W. & WENJUN, Z. (2009). "Intra-Predictive Transforms for Block-Based Image Coding". *Signal Processing, IEEE Transactions on;* Vol.57(8); pp.3030-3040.

JONG-NAM, K., SUNG-CHEAL, B. & BYUNG-HA, A. (2001). "Fast Full Search Motion Estimation Algorithm Using various Matching Scans in Video Coding". *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews;* Vol.31(4); pp.540-548.

JONG-NAM, K. & TAE-SUN, C. (1998). "A fast three-step search algorithm with minimum checking points using unimodal error surface assumption". *IEEE Transactions on Consumer Electronics;* Vol.44(3); pp.638-648.

JUNG, S. M., SHIN, S. C., BAIK, H. & PARK, M. S. (2002). "Efficient multilevel successive elimination algorithms for block matching motion estimation". *IEE Proceedings -Vision, Image and Signal Processing;* Vol.149(2); pp.73-84.

KIM, C. (2010). "*Complexity Adaptation in Video Encoders for Power Limited Platforms"*. PhD, Dublin City University.

KIM JONG-NAM, BYUN SUNG-CHEAL, KIM YONG-HOON & AHN BYUNG-HA (2002). "Fast full search motion estimation algorithm using early detection of impossible candidate vectors". *IEEE Transactions on Signal Processing;* Vol.50(9); pp.2355-2365.

KIM JONG-NAM & CHOI TAE-SUN (2000). "A fast full-search motion-estimation algorithm using representative pixels and adaptive matching scan". *IEEE Transactions on Circuits and Systems for Video Technology;* Vol.10(7); pp.1040-1048.

KOGA, T., ILINUMA, K., HIRANO, A., IIJIMA, Y. & Y.ISHIGURO (1981). "Motion compensated interframe coding for video conferencing". *National Telecommum Conference*; pp.531–535.

KOU, W. (1995). "Digital Image Compression: Algorithms and Standards": *Kluwer Academic Publishers*.

KRATZ, S. & BALLAGAS, R. (2007). "Gesture recognition using motion estimation on mobile phones". *Proc. of 3rd International Workshop on Pervasive Mobile Interaction Devices (PERMID'07)*.

LAI-MAN, P. & WING-CHUNG, M. (1996). "A novel four-step search algorithm for fast block motion estimation". *IEEE Transactions on Circuits and Systems for Video Technology;* Vol.6; pp. 313–317.

LEONTARIS, A., COSMAN, P. C. & TOURAPIS, A. M. (2009). "Multiple Reference Motion Compensation: A Tutorial Introduction and Survey". *Found. Trends Signal Process;* Vol.2(4); pp.247-364.

LI LIU, COHEN, R., SUN, H., VETRO, A. & ZHUANG, X. (2010 ). "New Techniques for Next Generation Video Coding". *In Proc. IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*; pp.111 - 116

LI, W. & SALARI, E. (1995). "Successive elimination algorithm for motion estimation". *IEEE Transactions on Image Processing;* Vol.4(1); pp.105-107.

LIN, C.-W., CHANG, Y.-J. & CHEN, Y.-C. (1998). "Hierarchical Motion Estimation Algorithm Based on Pyramidal Successive Elimination". *International Computer Symposium*; pp.41-44.

LIN CHEN-FU & LEOU JIN-JANG (2005). "An adaptive fast full search motion estimation algorithm for H.264". *In Proc. IEEE International Symposium on Circuits and Systems ISCAS*; Vol.2; pp.1493-1496.

LIU, B. & ZACCARIN, A. (1993). "New fast algorithms for the estimation of block motion vectors". *IEEE Transactions on Circuits and Systems for Video Technology;* Vol.3(2); pp.148-157.

MAN-YAU, C. & WAN-CHI, S. (2006). "New results on exhaustive search algorithm for motion estimation using adaptive partial distortion search and successive elimination algorithm". *In Proc. IEEE International Symposium on Circuits and Systems ISCAS*; pp.3977-3981.

MARPE, D., WIEGAND, T. & SULLIVAN, G. J. (2006). "The H.264/MPEG4 advanced video coding standard and its applications". *Communications Magazine, IEEE;* Vol.44(8); pp.134-143.

METKAR, S. & TALBAR, S. (2010). "Fast motion estimation using modified orthogonal search algorithm for video compression". *Signal, Image and Video Processing;* Vol.4; pp.123–128.

MITRA, S. & ACHARYA, T. (2007). "Gesture Recognition: A Survey". *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, ;* Vol.37(3); pp.311-324.

MIZUKI, M. M., DESAI, U. Y., MASAKI, I. & CHANDRAKASAN, A. (1996). "A binary block matching architecture with reduced power consumption and silicon area requirement". *In Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing ICASSP*; Vol.6; pp.3248-3251.

MOERITZ, S. & DIEPOLD, K. (2004). "Understanding MPEG 4: Technology and Business Insights*"*: *Focal Press*.

MOGUS, F. A., XINYING, L. & LEI, W. (2010). "Evaluation of the performance of motion Estimation algorithms in video coding". *In Proc. IEEE 2nd International Conference on Information Science and Engineering (ICISE)*; pp.3693-3696.

MR. P. VIJAYKUMAR, A. K., SIDHARTH BHATIA (2011). "Latest Trends, Applications and Innovations in Motion Estimation Research". Vol.2 (7).

NATARAJAN, B., BHASKARAN, V. & KONSTANTINIDES, K. (1997). "Low-complexity block-based motion estimation via one-bit transforms". *IEEE Transactions on Circuits and Systems for Video Technology;* Vol.7(4); pp.702-706.

NATIONAL SCIENCE FOUNDATION (2011). "*Video Trace Library"*. [Online]. Available: http://trace.eas.asu.edu/yuv/index.html; [Accessed 2.2 2011].

NIE, Y. & MA, K.-K. (2002). "Adaptive rood pattern search for fast block-matching motion estimation ". *IEEE Trans on Image Processing;* Vol.11(12); pp.1442-1448.

NIGHTINGALE, J., QI, W. & GRECOS, C. (2012). "HEVStream: A Framework for Streaming and Evaluation of High Efficiency Video Coding (HEVC) Content in Loss-prone Networks". *IEEE Transactions on Consumer Electronics;* Vol.58(2); pp.404-412.

OHM, J. & SULLIVAN, G. J. (2013). "High Efficiency Video Coding: The Next Frontier in Video Compression [Standards in a Nutshell]". *IEEE Signal Processing Magazine;* Vol.30(1); pp.152-158.

PEREIRA, F. C. & EBRAHIMI, T. (2002). "The MPEG-4 Book*"*: *Prentice Hall PTR*.

PRASANTHA, H. S., SHASHIDHARA, H. L. & BALASUBRAMANYA MURTHY, K. N. (2007). "Image Compression Using SVD". *In Proc. IEEE International Conference on Conference on Computational Intelligence and Multimedia Applications*; Vol.3; pp.143-145.

PU, I. M. (2005). "Fundamental Data Compression*"*: *Butterworth-Heinemann*.

PURI, A., HANG, H. M. & SCHILLING, D. (1987). "An efficient block-matching algorithm for motion-compensated coding". *IEEE International Conference on Acoustics, Speech, and Signal Processing;* Vol.12; pp.1063-1066.

REOXIANG, L., BING, Z. & LIOU, M. L. (1994). "A new three-step search algorithm for block motion estimation". *IEEE Transactions on Circuits and Systems for Video Technology;* Vol.4(4); pp.438-442.

RICHARDSON, I. (2003). "H.264 and MPEG-4 Video Compression: Video Coding for Next Generation Multimedia": *Wiley*.

RICHARDSON, I. E. G. (2010). "The H.264 advanced video compression standard", *2nd* edition; UK: *John Wiley & Sons Inc*.

RUIZ, G. A. & MICHELL, J. A. (2011). "An efficient VLSI processor chip for variable block size integer motion estimation in H.264/AVC". *Signal Processing: Image Communication;* Vol.26(6); pp.289-303.

SAYOOD, K. (2006). "Introduction to Data Compression", *3rd* edition: *Morgan Kaufmann*.

SHAN, Z. & KAI-KUANG, M. (1997). "A new diamond search algorithm for fast block matching motion estimation". *In Proc. IEEE International Conference on Information, Communications and Signal Processing (ICICS)*; Vol.1; pp.292-296.

SHARIFI, M., FATHY, M. & TAYEFEH MAHMOUDI, M. (2002). "A classified and comparative study of edge detection algorithms". *International Conference on Information Technology: Coding and Computing, 2002.*; pp.117-120.

SONG, B. C. & RA, J. B. (1998). "A hierarchical block matching algorithm using partial distortion measure". *In Proc. SPIE Visual Communications and Image Processing '98*; Vol.3309; pp.88-95.

SOO-MOK, J., SUNG-CHUL, S., HYUNKI, B. & MYONG-SOON, P. (2000). "Nobel successive elimination algorithms for the estimation of motion vectors". *In Proc. IEEE International Symposium on Multimedia Software Engineering*; pp.332-335.

SRINIVASAN, R. & RAO, K. R. (1985). "Predictive coding based on efficient motion estimation". *IEEE Transactions on Communications;* Vol.33(8); pp.888–896.

SULLIVAN, G., TOPIWALA, P. & LUTHRA, A. (2004). "The H.264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions". *SPIE conference on Applications of Digital Image Processing XXVII*.

SULLIVAN, G. J. & WIEGAND, T. (2005). "Video Compression - From Concepts to the H.264/AVC Standard". *Proceedings of the IEEE;* Vol.93(1); pp.18-31.

TIAN, L., LI, S., AHN, J., CHU, D., HAN, R., LV, Q. & MISHRA, S. (2013). "Understanding User Behavior at Scale in a Mobile Video Chat Application". *In Proc. UbiComp '13 ACM International Joint Conference on Pervasive and Ubiquitous Computing*; pp.647-656.

TURAGA, D. & CHEN, T. (2001). "I/P Frame Selection Using Classification Based Mode Decision". *International Conference on Image Processing ICIP*; pp.550-553.

V.K.ANANTHASHAYANA & PUSHPA.M.K (2009). " joint adaptive block matching search algorithm,". *World Academy of Science, Engineering and Technology* Vol.56; pp.225-229.

VANNE, J. (2011). "*Design and Implementation of Configurable Motion Estimation Architecture for Video Encoding"*. PhD, Tampere University of Technology

WAGGONER, B. (2002). "Compression for Great Digital Video: Power Tips, Techniques, and Common Sense*"*: *CMP*.

WIEGAND, T., SULLIVAN, G. J., BJONTEGAARD, G. & LUTHRA, A. (2003). "Overview of the H.264/AVC video coding standard". *IEEE Transactions on Circuits and Systems for Video Technology;* Vol.13(7); pp.560-576.

WIEN, M. (2003). "Variable block-size transforms for H.264/AVC". *IEEE Transactions on Circuits and Systems for Video Technology;* Vol.13(7); pp.604-613.

XIAOQUAN, Y. & NAM, L. (2005). "Rapid block-matching motion estimation using modified diamond search algorithm". *In Proc. IEEE International Symposium on Circuits and Systems ISCAS*; Vol.6; pp.5489-5492.

XIONG, X., SONG, Y. & AKOGLU, A. (2011). "Architecture design of variable block size motion estimation for full and fast search algorithms in H.264/AVC". *Computers & Electrical Engineering;* Vol.37(3); pp.285-299.

XUAN, J. & LAP-PUI, C. (2004). "An efficient three-step search algorithm for block motion estimation". *IEEE Transactions on Multimedia;* Vol.6(3); pp.435-438.

YI, X., ZHANG, J., LING, N. & SHANG, W. (2005). "Improved and simplified fast motion estimation for JM (JVT-P021)". *Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG (ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q.6) 16th Meeting: Poznan, Poland*.

YOUTUBE (2013). "*YouTube fact sheet"*. [Online]. Available: http://www.youtube.com/yt/press/statistics.html; [Accessed 10.09. 2013].

YU, L. & PENG WANG, J. (2010). "Review of the current and future technologies for video compression". *Journal of Zhejiang University - Science C;* Vol.11(1); pp.1-13.

YUI-LAM, C. & WAN-CHI, S. (1996). "New adaptive pixel decimation for block motion vector estimation". *IEEE Transactions on Circuits and Systems for Video Technology;* Vol.6(1); pp.113-118.

ZHAO, H., YU, X.-B., SUN, J.-H., SUN, C. & CONG, H.-Z. (2008). "An Enhanced Adaptive Rood Pattern Search Algorithm for Fast Block-Matching Motion Estimation". *Congress on Image and Signal Processing;* Vol.1; pp.416-420.

ZHU, S. & MA, K.-K. (2000). "A new diamond search algorithm for fast block-matching motion estimation". *IEEE Transactions on Image Processing;* Vol.9(2); pp.287-290.