

**BLOCK MOTION ESTIMATION USING DIRECTIONAL ADAPTIVE
SEARCH WINDOW**

By

NOR FARHANA BT. FADLY CHEW

FINAL PROJECT REPORT

**Submitted to the Electrical & Electronics Engineering Programme
in Partial Fulfillment of the Requirements
for the Degree
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)**

**Universiti Teknologi Petronas
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan**

© Copyright 2006

by

Nor Farhana Bt. Fadly Chew, 2006

CERTIFICATION OF APPROVAL

BLOCK MOTION ESTIMATION USING DIRECTIONAL ADAPTIVE SEARCH WINDOW

by

Nor Farhana Bt. Fadly Chew

A project dissertation submitted to the
Electrical & Electronics Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Approved:

Ms. Nasreen Badruddin
Project Supervisor

Approved:




Assoc. Prof. Dr. Varun Jeoti
Project Co-Supervisor

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK

December 2006

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.



Nor Farhana Bt. Fadly Chew

ABSTRACT

Motion estimation (ME) is the exploitation of similarities between adjacent frames in a video sequence by eliminating temporal redundancy, and is an essential part of the H.264 and other video compression standards. However, it introduces an increase of computation complexity resulting in longer execution time. Thus, adaptive motion estimation for H.264 is proposed in order to reduce the execution time while giving better PSNR performance. The algorithm determines the amount of motion in each block and classifies them as low, medium and high motion. From the magnitude and direction of the x and y motion vector components, the search window (search range) is dynamically adjusted. For high motion, the search range is set to be the maximum value and vice versa. The results show that execution time could be reduced to almost half (50%) of the conventional method since the number of search points and computations decrease in the range of 40% to 60%. Furthermore, the method gives a better image quality for video sequence with uniform motion and negligible PSNR loss in others. By introducing early termination in the adaptive motion estimation, the number of computation could be reduced even further since the search process is terminated immediately certain criteria are satisfied. By using Option 2 for early termination, the search point computation and PSNR is reduced with average 1.3% and 1.027% from the adaptive motion estimation without the early termination process.

ACKNOWLEDGEMENTS

First and foremost, the Author would like to thank God, for giving His precious time to the Author to work on Final Year Project. Not forgotten to Electrical and Electronics Engineering Department of Universiti Teknologi Petronas (UTP) for the chance given to me to carry out this project.

This dissertation could not have been written without Ms. Nasreen Badruddin who not only served as my supervisor but also encouraged and challenged me throughout my academic program as well as this project. She patiently guided me through the dissertation process, never accepting less than my best efforts.

A special thank is dedicated to my co-supervisor, Assoc. Prof. Dr. Varun Jeoti for sharing his idea and thought in this project. Also an appreciation to Ms Siti Hawa Tahir (EE FYP Technician) for helping me throughout the project process.

Finally, I would like to thank my mother and father, who gave me the opportunity and the spirit to educate myself. Mom, Dad, you are the best. Special thank to Yulnaikey Mohd. Yusoff for his continuous support. He was always being there to listen and encourage me to face with problems. To all my friends, all ideas, advices, support and compassions to me are completely appreciated.

November 01, 2006
Universiti Teknologi Petronas

TABLE OF CONTENTS

LIST OF TABLES	ix
LIST OF FIGURES.....	x
CHAPTER 1 INTRODUCTION	1
1.1 Background of Study.....	1
1.2 Problem Statement	1
1.2.1 Problem Identification	1
1.2.2 Significance of the Project.....	2
1.3 Objectives and Scope of Study	3
1.3.1 The Relevancy of the Project.....	3
1.3.2 Feasibility of the Project within Scope and Time Frame.....	3
CHAPTER 2 LITERATURE REVIEW	4
2.1 Video Resolution.....	4
2.2 H.264.....	5
2.2.1 Inter-prediction	5
2.2.2 Block Matching Method	6
2.2.3 Matching Criteria.....	7
2.2.4 Block Size	7
2.2.5 Search Window.....	8
2.2.6 Search Method	8
2.2.7 Video Quality.....	8
2.2.8 Computation Complexity.....	9
2.2.9 Early Search Termination.....	10
CHAPTER 3 METHODOLOGY / PROJECT WORK	11
3.1 Procedure Identification	11
3.1.1 Current Frame Analysis	12
3.1.2 Adaptive Search Window and Search Range Decision.....	13
3.1.3 Early Termination	15
3.1.4 Directive Search Process	21
3.1.5 Predicted image reconstruction.....	22
3.1.6 Computation Count.....	22
3.1.7 PSNR Determination	22

3.2 Summary of algorithm	23
3.3 Tools Required	24
CHAPTER 4 RESULTS AND DISCUSSION	26
4.1 Motion Vector Estimation	26
4.2 Adaptive Search Window (Search Range)	27
4.3 Comparison between Non-adaptive and Adaptive Search Window	28
4.4 Adaptive Motion Estimation with Early Termination	32
CHAPTER 5 CONCLUSION AND RECOMMENDATION	36
CHAPTER 6 REFERENCES	37
APPENDICES	39
Appendix A Gantt Chart	40
Appendix B Window Size	42
Appendix C Calculate PSNR (loopPSNR.m)	46
Appendix D Adaptive Motion Estimation (NEWADAPTIVE.m)	47
Appendix E Non-Adaptive Motion Estimation (NEWFS.m)	55
Appendix F Adaptive Motion Estimation With Early Termination (NEWADAPTIVE_ET.m)	59
Appendix G Split .YUV And .QCIF Video (extract.m)	68

LIST OF TABLES

Table 1 Video Resolutions (pixel).....	4
Table 2 Direction of the motion based on sign of motion vector component.	13
Table 3 Search Range for horizontal direction (mv_x).....	15
Table 4 Search Range for Vertical Direction (mv_y).....	15
Table 5 Percentage of stationary and non-stationary blocks.	16
Table 6 Statistical data for stationary SAD.	17
Table 7 Statistical data for non-stationary SAD.....	17
Table 8 Distribution result for <i>Claire.qcif</i>	18
Table 9 Distribution result for <i>Container.qcif</i>	18
Table 10 Distribution result for <i>News.qcif</i>	18
Table 11 Distribution result for <i>Hall Monitor.qcif</i>	19
Table 12 Matlab Program File.....	24
Table 13 Raw video.....	25
Table 14 Total number of computation for 30 frames of <i>Hall Monitor.qcif</i>	29
Table 15 Total number of search point for 30 frames of <i>Hall Monitor.qcif</i>	29
Table 16 The average number of search points.....	30
Table 17 The average PSNR comparison.....	31
Table 18 Comparison of the average search point and PSNR using various algorithms for “Container.qcif”.	32
Table 19 Comparison of the average search point and PSNR using various algorithms for “Claire.qcif”.	32
Table 20 Comparison of the average search point and PSNR using various algorithms for “News.qcif”.....	33
Table 21 Comparison of the average search point and PSNR using various algorithms for “Hall Monitor.qcif”.	33
Table 22 Comparison of the percentage reduction of search point for the proposed algorithms.	33
Table 23 Comparison of the percentage reduction of PSNR for the proposed algorithms.	34

LIST OF FIGURES

Figure 1 Block diagram of H.264 encoder. ^[9]	5
Figure 2 Block matching. ^[6]	6
Figure 3 Search Region	8
Figure 4 Procedure of motion estimation.....	11
Figure 5 Search window size based on motion vector.....	14
Figure 6 Cumulative distribution function SAD for <i>Claire.qcif</i>	19
Figure 7 Cumulative distribution function SAD for <i>Container.qcif</i>	20
Figure 8 Cumulative distribution function SAD for <i>News.qcif</i>	20
Figure 9 Cumulative distribution function SAD for <i>Hall Monitor.qcif</i>	20
Figure 10 Directive Search Process	21
Figure 11 Flow diagram of the proposed algorithm.....	23
Figure 12 Motion vector field for <i>Coastguard.qcif</i> at frame 152.....	26
Figure 13 Search ranges R1, R2, R3 and R4 for adaptive search range.	27
Figure 14 Number of search point for “Hall Monitor.qcif”.	28
Figure 15 Percentage of search point difference between adaptive and non-adaptive	29
Figure 16 The average PSNR comparisons for several videos.	30
Figure 17 Comparison of percentage search point reduction for proposed algorithm.	34
Figure 18 Comparison of percentage PSNR reduction for proposed algorithm.	35

CHAPTER 1

INTRODUCTION

1.1 Background of Study

Video compression is a conversion of the digital video data to another format that requires lesser bits (digital) so that the data can be transmitted and stored efficiently. Video data contains spatial and temporal redundancy. Video compression is done by discarding information that is indiscernible to the viewer. Spatial redundancy is reduced by using intra-frame compression where the compression process is done within single frame. On the other hand, temporal redundancy is reduced by encoding only the differences between frames which is also known as inter-frame compression. This task involves various techniques including motion estimation and motion compensation. Motion estimation is an essential part of the H.264 and other video compression standards where it eliminates temporal redundancy between adjacent frames since only the first frame and the differences between frames are encoded. There are various video compression standards that have been designed such as MPEG-1, MPEG-2, MPEG-4, H.261 and H.263. H.264 is the latest video coding standard introduced in 2003. The H.264 adopted various new techniques such as multi reference frames, intra block prediction and in-loop deblocking filter to achieve better visual quality thus increase the complexity of encoder.

1.2 Problem Statement

1.2.1 Problem Identification

Motion estimation is the most computationally intensive component of the encoder since it consumes about 60% to 80% of the total encoding time. Thus, this limits its practical application in the real-time video coding. The H.261 and H.263 video

compression standards, use exhaustive search algorithm (ESA) with a fixed block size of 16×16 and a fixed search window (SW). Ideally, the search window parameter is 7 corresponding to a window of size 15×15 . In [1], a performance comparison of block matching techniques in motion estimation was done for various types of video motion. For a small video motion, the blocks seem not to move at all. Therefore, it is found that optimum result is obtained by using small block and window size. Conversely, for large motion, it is better to use large block and window size. Thus, an algorithm that could adaptively change the search window depending on the motion type is the main concern in order to avoid wasting time in computing the search point especially for a small motion video. For instance; a window size is highly dependant with a search range (SR) where bigger size of SW will correspond to more search points computation.

Early termination is introduced to reduce the computations further since the search process is terminated immediately for stationary blocks. In [10], [11] and [12], fixed threshold value is applied in the early termination process for all video sequence. For large motion video, the thresholding did not contributes much in reducing the computation however, for low motion video, larger threshold lead to low video quality. Therefore, a directional adaptive search window (DASW) for H.264 with early termination algorithm is proposed in order to reduce the execution time with better PSNR performance.

1.2.2 Significance of the Project

Video compression is important in the telecommunication and multimedia are where bandwidth is a valuable commodity. Thus, the study of motion estimation has become extremely important in order to model a current frame as accurate as possible with less computational complexity to save the bandwidth usage.

1.3 Objectives and Scope of Study

The main objective of this project is to design an intelligent algorithm which will dynamically adjust the search window (search range) in the motion estimation process based on the predicted motion vector in a video sequence. The scopes and assumptions for this project are:

- Investigates the motion estimation part of encoder only. Other part such as entropy encoding, quantization and intra-frame prediction are not covered.
- Only the search window size (search range) is changed. The block size is fixed to 16×16 .
- Since only inter-frame prediction is performed, the motion estimation is limited to frames which have almost similar content (no abrupt changes) and uniform motion.

1.3.1 *The Relevancy of the Project*

The development of the adaptive motion estimation for H.264 by using the search method is very significant since this new design will give the encoder the ability to select the best or near optimum search range.

1.3.2 *Feasibility of the Project within Scope and Time Frame*

The project starts with the study of several motion estimation techniques focusing on the blocked-matching method. Research is done as the methods to predict the motion vectors and to determine the amount of motion contained in a particular frame. A program that dynamically adjusts the motion estimation parameters (search method, block size, window size or the search range) based on the predicted motion is created. In the first semester, simulation is done on the directional adaptive search window algorithm using MATLAB. Later in the second semester, the algorithm is enhanced by integrating early termination technique. Finally, a thoughtful analysis is conducted on the result obtained based on the performance measurement of the new adaptive motion estimation technique. **Appendix A** shows the Gantt chart for the whole semester.

CHAPTER 2

LITERATURE REVIEW

2.1 Video Resolution

The size of video image is measured in pixels for digital video. A higher resolution and sharper images is produced as the number of pixels contains increase. The standard resolution of YUV video in pixel is indicated by Common Intermediate Format (CIF). Other standard video resolutions are listed in *Table 1*.

Table 1 Video Resolutions (pixel)

Video Formats	Luminance Resolution	Chrominance Resolution
SQCIF	128 x 96	64 x 48
QCIF	176 x 144	88 x 72
CIF	352 x 288	176 x 144
4 CIF	704 x 576	352 x 288
16 CIF	1408 x 1152	704 x 576

YUV defined color space consists of one luminance or luma component (Y) and two chrominance or chroma components (U and V). The luminance indicates the brightness of the color, while the chrominance components determine the color. A grey-scale image is produced from a Y component without the U and V components. Since human eyes are more sensitive to brightness compared to color, many video compression formats eliminate some information of chrominance to reduce the number of computation without affecting the perceived video quality.

2.2 H.264

H.264 or MPEG-4 Part 10 that was also known as MPEG-4 AVC (Advanced Video Coding) is the latest video compression standard established by the ITU-T Video Coding Experts Group (VCEG) together with the ISO/IEC Moving Picture Experts Group (MPEG) [2]. Recently, it has gained more attention due to its high coding efficiency compared with the previous successive standards. The enhanced compression and perceptual quality of H.264 is obtained from motion estimation, which minimizes the temporal redundancies [3]. *Figure 1* illustrates the block diagram of the H.264 encoder where the ME and motion compensation (MC) take place in the inter-prediction mode.

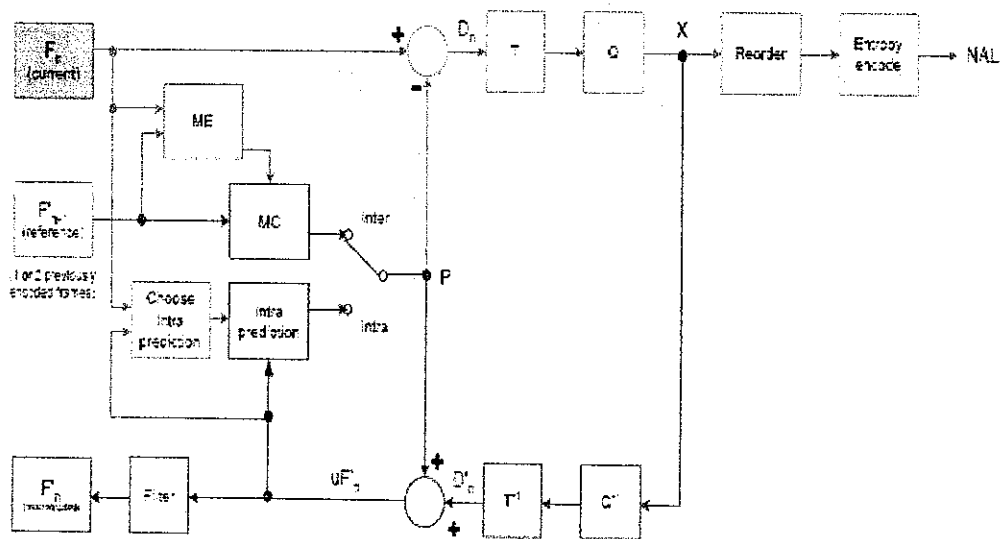


Figure 1 Block diagram of H.264 encoder. [9]

2.2.1 Inter-prediction

Inter-prediction mode creates a prediction model from one or more previously encoded video frames by shifting the samples in the reference frame [9]. A motion vector that denotes the displacement of the best matching block in the reference frame with respect to the block in the current frame is determined in this mode.

2.2.2 Block Matching Method

Theoretically, motion estimation is the process of predicting the current frame based on the available data in a reference frame by obtaining the motion vector [4] [5]. Practically, the block matching method is the simplest and most popular technique of estimating the motion vector by searching the best matched block in the reference frame. It is a combination of four different parameters namely matching criteria, block size, search window (search range) and the search method [1].

Figure 2 illustrates the process of block-matching algorithm [6]. Each frame is divided with equally square size non-overlapped block (16x16 pixels). The block in the current frame is compared with the block at the same position in the reference frame. The reference block is at the centre of the search window (indicated by the dashed lines). Each block in the current frame is compared to a block in the reference frame within the search window using the prediction error measure (MAD, SAD, or MSE). The best matching block which has the minimum amount of error is selected (shown by the thick line in the reference frame) [4] to give the motion vector (MV). MV is the displacement of the best matched block from the original block, indicated by the arrow pointing to the upper-left of the frame.

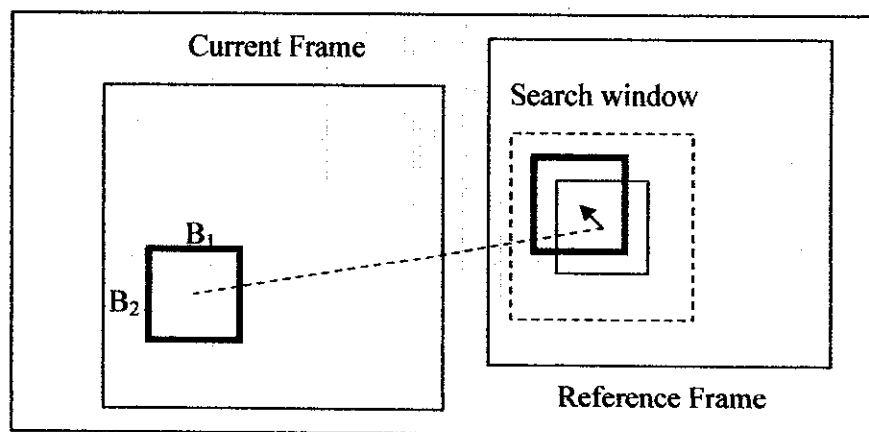


Figure 2 Block matching. [6]

2.2.3 Matching Criteria

There are various criteria that the motion estimation scheme can use to find a close match between two blocks. These include the minimum sum of absolute error (SAE) or sum of absolute differences (SAD), the minimum mean absolute difference (MAD), the minimum mean square error (MSE), and the maximum cross-correlation. However, the most popular methods are MAD and SAD since they are computationally less complex as compared to others. The energy difference between predicted/current block I_p and reference block I_r at coordinate (x, y) is measured as:

$$MAD(mv_x, mv_y) = \frac{1}{B_1 B_2} \sum_{(x,y) \in B} |I_p(x, y, k) - I_r(x + mv_x, y + mv_y, k + 1)| \quad (1)$$

$$MSE(mv_x, mv_y) = \frac{1}{B_1 B_2} \sum_{(x,y) \in B} |I_p(x, y, k) - I_r(x + mv_x, y + mv_y, k + 1)|^2 \quad (2)$$

where B_1 and B_2 denote the size of block ($B_1 \times B_2$), for a candidate of motion vectors (mv_x, mv_y) at frame k . The estimate value for motion vector (mv_x, mv_y) is taken from the coordinate block of the minimum MSE or MAD.

2.2.4 Block Size

The block size is defines as $B_1 \times B_2$ centered at the position (x, y) where for a square block, $B_1 = B_2$. Normally, the block size used is 16×16 or 8×8 . Using a small block size, results in more motion vectors to be computed but a more accurate prediction is produced.

2.2.5 Search Window

The search window is limited to $(B_1 + 2R) \times (B_2 + 2R)$, where R is a predetermined integer. This search window shown in *Figure 3* restricts the region to find the best-matched block and its corresponding motion vector in the reference frame. The larger the search window, the bigger region for motion estimation, thus more computation needs to be done. In general, the search window is restricted to lower the execution time associated with block matching.

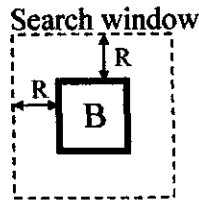


Figure 3 Search Region

2.2.6 Search Method

The search process is conducted over a predetermined search region and it can be modified to suit the needs of particular algorithm [4]. Two types of popular search method are full-search (exhaustive) and fast-search (Three-Step Search, Logarithmic Search, Cross-Search). Full-search is the most accurate one since it searches over all possible blocks within the search window to find optimal motion vector.

2.2.7 Video Quality

Video quality is measured using PSNR matrix. The PSNR also known as peak signal-to-noise ratio is the ratio of signal power to the noise power. PSNR is always measured in decibel units (dB). Normally, PSNR is used to measure the reconstructed image quality after performing video compression compression. Larger value of PSNR corresponds to better video/image quality. The PSNR is defined via using the mean squared error (MSE) where the difference between two $m \times n$ monochromes images I and K is calculated. The MSE is defined as:

$$MSE(mv_x, mv_y) = \frac{1}{B_1 B_2} \sum_{(x,y) \in B} |I_c(x, y, k) - I_o(x + mv_x, y + mv_y, k + 1)|^2 \quad (3)$$

where I_c is the current image that experience image compression (noisy) and I_o is the original image. The PSNR is defined as:

$$PSNR = 10 \cdot \log_{10} \left(\frac{255^2}{MSE} \right) = 20 \cdot \log_{10} \left(\frac{255}{\sqrt{MSE}} \right) \quad (4)$$

2.2.8 Computation Complexity

The computation complexity determines the number of operation counts for one particular frame. Theoretically, the sooner the search point is found, the less number of computations and to the encoder execution time is reduced. Currently, the computation complexity is one of the important metrics that researchers work on in estimating the encoder performance. It is desirable to minimize the complexity. The computation for each block is as follows:

$$\text{Block computation} = (2R + 1)^2 \times B^2 \quad (5)$$

The total entire frame computation is defined as:

$$\text{Frame computation} = (m \times n) \times (2R + 1)^2 \quad (6)$$

The total search point per block is:

$$\text{Search point per block} = (2R + 1)^2 \quad (7)$$

The total number of search point per frame is:

$$\text{Search point per frame} = \frac{(m \times n)}{B^2} (2R + 1)^2 \quad (8)$$

2.2.9 Early Search Termination

Low motion video-sequences tend to have more stationary blocks than non-stationary block. A stationary block is a block that has MV equal to (0, 0), and other blocks than that is considered as non-stationary block. Therefore, an early termination algorithm is introduced which immediately terminates the motion estimation process after certain criteria are satisfied. As a result, the computational cost and execution time could be reduced for zero motion blocks. A larger threshold value results in less computation costs are perform, but the larger degradation of video quality.

CHAPTER 3

METHODOLOGY / PROJECT WORK

3.1 Procedure Identification

The motion estimation procedure will be based on the following project design:

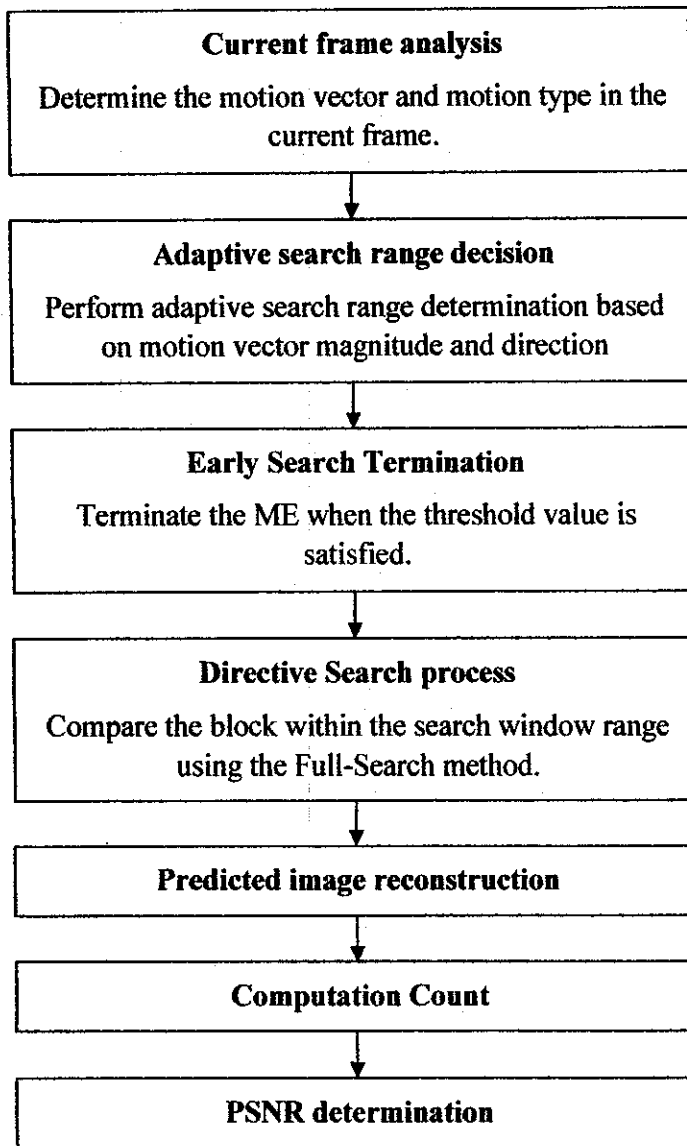


Figure 4 Procedure of motion estimation.

3.1.1 Current Frame Analysis

Generally, the videos are classified into three categories which are *low*, *medium*, and *high* motion. The motion categories are determined hierarchically at the frame level and only the information gathered from the first predicted frames is used which are the motion vectors (mv_x, mv_y) .

In this project, the motion vector is determined by applying the Sum of Absolute Difference (SAD) equation. Initially, the block size and search range were used are fixed to 16 and 7. The SAD equation is given as:

$$SAD(mv_x, mv_y) = \sum_{(x,y) \in B} |I_p(x, y, k) - I_r(x + mv_x, y + mv_y, k + 1)| \quad (5)$$

where mv_x and mv_y indicate the candidate x and y motion vector respectively. The MV chosen is the one which minimized the SAD. The criterion that has been set to classify the three categories of motion ω , in one block is given as:

$$\omega = \begin{cases} \text{Low} & mv_x = 0, mv_y = 0 \\ \text{Medium} & 0 < |mv_x| \leq 4, 0 < |mv_y| \leq 4 \\ \text{High} & |mv_x| > 4, |mv_y| > 4 \end{cases} \quad (6)$$

where $|mv_x|$ and $|mv_y|$ are the magnitudes of their respective associated motion vector $MV(mv_x, mv_y)$ in a predicted frame. The sign of the motion vectors component indicates the direction of the motion in the current frame as:

$mv_x = \text{positive}$, direction = right (\rightarrow)

$mv_x = \text{negative}$, direction = left (\leftarrow)

$mv_y = \text{positive}$, direction = downwards (\downarrow)

$mv_y = \text{negative}$, direction = upwards (\uparrow)

Table 2 Direction of the motion based on sign of motion vector component.

Sign of mv_x	Sign of mv_y	Direction of motion
0	0	no direction
0	+	↓
0	-	↑
+	0	→
-	0	←
+	+	↘
+	-	↗
-	+	↙
-	-	↖

3.1.2 Adaptive Search Window and Search Range Decision

Based on the category of motion and the direction of respective motion vector, the optimum size of the search window is changed by adjusting the search range parameters. The search range is derived based on the predicted blocks motion vector where the motion vectors are classified as small (low motion), medium (medium motion) and large (high motion). The proposed adaptive search window (search range) algorithm on each block is described as follows:

Perform the adaptive search range

if ($\omega == \text{low}$)

search range = sr_{min}

else if ($\omega == \text{medium}$)

search range = $sr_{min} + (sr_{max} - sr_{min})/4$

For this project, the value of sr_{min} is set equal to 4 and sr_{max} is set equal to 16. The reason is for a small motion video, blocks tend to move less than 4 pixels from its initial position. Meanwhile for large motion the blocks can move as far as the size of the blocks 16 pixels. It means that the blocks are totally shifted from its original position. Therefore, the search range is defined as follows:

- i. **Small** : $\omega = \text{low}$, $mv_x = 0$, $mv_y = 0$;
Search range, $R = 4$
- ii. **Medium**: $\omega = \text{medium}$, $0 < |mv_x| \leq 4$, $0 < |mv_y| \leq 4$;
Search range, $R = 7$
- iii. **Large**: $\omega = \text{high}$, $|mv_x| > 4$, $|mv_y| > 4$;
Search range, $R = 16$

The adaptive search window's orientation also changed according to the direction of the motion vector component. The search window is now represented by 4 parameters i.e. R_1 , R_2 , R_3 and R_4 . The values of R_1 , R_2 , R_3 and R_4 are dynamically adjusted based on the magnitude and direction of motion vector.

For example, in *Figure 5*; if the motion vector (3,2) is large in the right direction (mv_x) and downward direction (mv_y), thus it is effectively to have larger value of R_2 and R_4 compare to R_1 and R_3 .

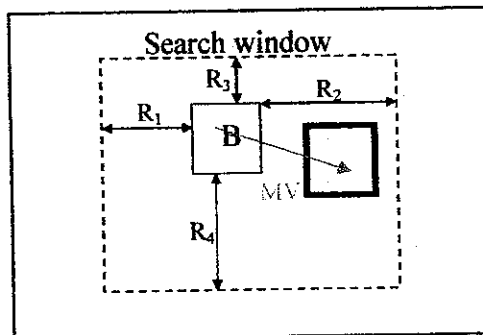


Figure 5 Search window size based on motion vector.

Table 3 Search Range for horizontal direction (mv_x).

Motion Category	Sign	R_1	R_2
low	0	4	4
medium	+	4	7
medium	-	7	4
high	+	4	16
high	-	16	4
default	0	7	7

Table 4 Search Range for Vertical Direction (mv_y)

Motion Category	Sign	R_3	R_4
low	0	4	4
medium	+	4	7
medium	-	7	4
high	+	4	16
high	-	16	4
default	0	7	7

Appendix B shows all possible of search window sizes estimated from equation $(B_1 + R_1 + R_2) \times (B_2 + R_3 + R_4)$.

3.1.3 Early Termination

An early termination process is employed in order to reduce the computation cost by terminated the search process immediately without examining the other points after some criterion is met. This method exploits the characteristics of stationary blocks where the amount of SAD for stationary blocks theoretically is smaller than non-stationary blocks at zero motion position (0, 0). A stationary block is a zero motion block that has $MV = (0, 0)$, while moving block that has $MV \neq (0,0)$ is considered as non-stationary block Thus, this method is only applicable for low motion video sequences since it has a lot of stationary blocks compared to non-stationary ones.

Table 6 Statistical data for stationary SAD.

Parameters	News	Claire	Hall Monitor	Container
Min	103	34	148	110
Max	3224	1302	3060	1022
Mean	463.5783	183.5165	440.8316	224.6354
Median	273	122	337	169.5
Std	524.4001	219.1274	375.3875	155.827

Table 7 Statistical data for non-stationary SAD.

Parameters	News	Claire	Hall Monitor	Container
Min	156	75	313	111
Max	4022	3419	361	138
Mean	728.8125	1637	333.75	121.3333
Median	386.5	1691	330.5	115
Std	1115	1152.8	20.1556	14.5717

The normalized value for probability distribution function, $F(x \leq n)$ of stationary SAD is examined. This normalization process is done to adjust the differences among data that have various ranges and sources in order to create common basis for comparison purposes. The results percentage of stationary (P1) and non-stationary (P2) blocks that have SAD value under the x value are indicated in *Table 8, 9, 10* and *11*. The normalized value is calculated by using the equation:

$$\text{Normalized Value (NV)} = \frac{F(x \leq n) - \min F(x)}{\max F(x) - \min F(x)} \quad (7)$$

Table 8 Distribution result for *Claire.qcif*

<i>n</i>	Stationary			Non-Stationary			Normalized
	$F(x \leq n)$	<i>x</i>	P1 %	$x1 \leq x$	$F1(x1)$	P2 %	Value (NV)
0.8	0.7912	171	72.71	113	0.25	2	0.108
0.7	0.6923	149	63.62	113	0.25	2	0.0907
0.6	0.5934	136	54.53	113	0.25	2	0.0804
0.5	0.4945	121	45.44	113	0.25	2	0.0694

Table 9 Distribution result for *Container.qcif*

<i>n</i>	Stationary			Non-Stationary			Normalized
	$F(x \leq n)$	<i>x</i>	%	$x1 \leq x$	$F1(x1)$	%	Value (NV)
0.8	0.7917	236	76.77	138	1	3.03	0.1382
0.7	0.6979	191	67.68	138	1	3.03	0.0808
0.6	0.5833	179	56.56	138	1	3.03	0.0757
0.5	0.5	169	48.49	138	1	3.03	0.0647

Table 10 Distribution result for *News.qcif*

<i>n</i>	Stationary			Non-Stationary			Normalized
	$F(x \leq n)$	<i>x</i>	%	$x1 \leq x$	$F1(x1)$	%	Value (NV)
0.8	0.7952	583	66.67	453	0.8125	13.13	0.1538
0.7	0.6988	464	58.59	453	0.8125	13.13	0.1157
0.6	0.5904	333	49.5	209	0.375	6.06	0.0737
0.5	0.494	252	41.42	209	0.375	6.06	0.0477

Table 11 Distribution result for *Hall Monitor.qcif*

<i>n</i>	Stationary			Non-Stationary			Normalized Value (NV)
	$F(x \leq n)$	<i>x</i>	%	$x1 \leq x$	$F1(x1)$	%	
0.8	0.8	474	76.77	361	1	4.04	0.112
0.7	0.6947	406	66.66	361	1	4.04	0.0886
0.6	0.5895	365	56.57	361	1	4.04	0.0745
0.5	0.4947	336	47.47	334	0.75	3.03	0.0646

Through the cumulative distribution function plot, the distributions of the SAD for stationary and non-stationary data value are clearly pictured. *Figure 6 to Figure 9* illustrate that the plots of empirical cumulative distribution function of the stationary and non-stationary SAD for the 4 video sequences. This plot is used to examine the distribution of a sample data. From the figures, it is shown that a low motion video has the steepest slope compared to high motion video sequence in the stationary block plots. This is because most of the sample data falls at the lower value range.

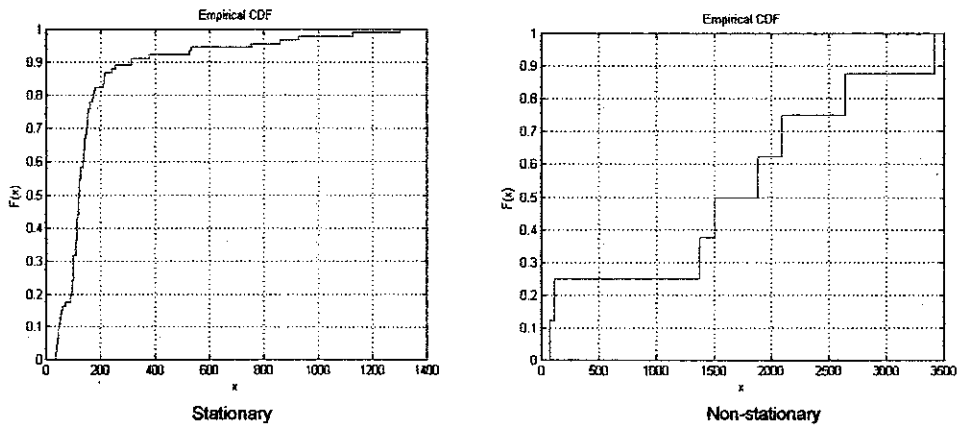


Figure 6 Cumulative distribution function SAD for *Claire.qcif*

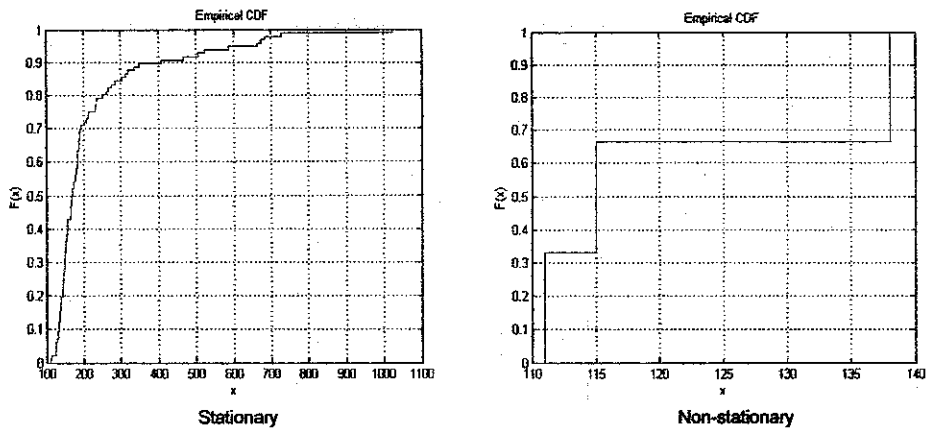


Figure 7 Cumulative distribution function SAD for *Container.qcif*

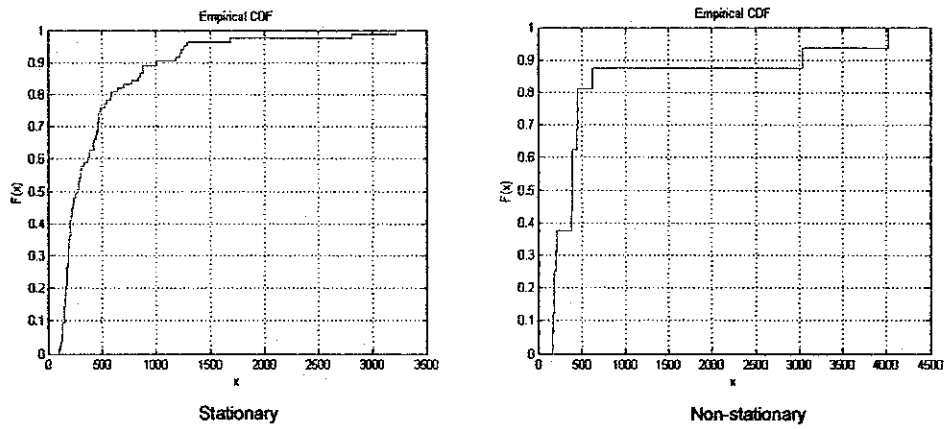


Figure 8 Cumulative distribution function SAD for *News.qcif*

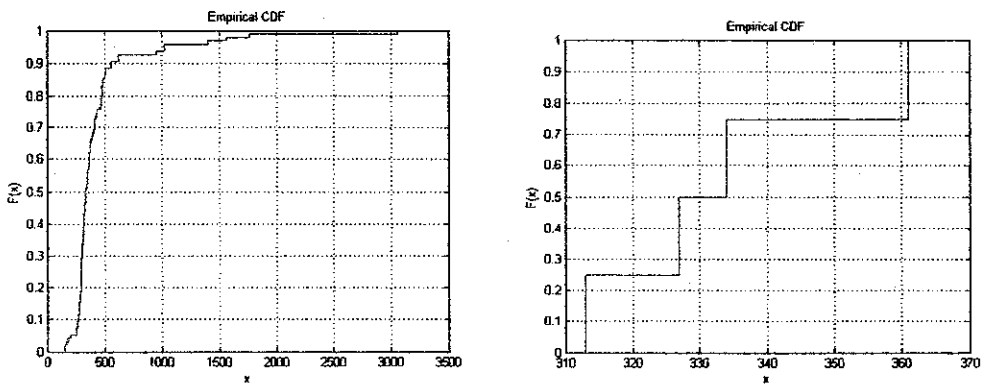


Figure 9 Cumulative distribution function SAD for *Hall Monitor.qcif*

In Option 1, the value of x at $F(x \leq 0.6)$ is chosen as threshold value, it is found that less than 10% of the blocks which is the non-stationary blocks will be misinterpreted as part of stationary blocks. The average of normalized value at $F(x \leq 0.6)$ for these four video is calculated to be equal to 0.076 and is set as fixed parameter to adaptively calculate the threshold value. The threshold value (TH_SAD) is calculated using the expression:

$$TH_SAD = 0.076(\max F(x) - \min F(x)) + \min F(x) \quad (8)$$

On the other hand, in Option 2 for x at $F(x \leq 0.8)$, less than 15% of the blocks will be misinterpreted. The calculated average of normalized value at $F(x \leq 0.8)$ for these four video is equal to 0.128. The threshold value (TH_SAD) is calculated using the expression:

$$TH_SAD = 0.128(\max F(x) - \min F(x)) + \min F(x) \quad (9)$$

3.1.4 Directive Search Process

The motion estimation (ME) was done using the same method as in *subsection 3.1.1* by applying the SAD to predict the motion vector and displacement. The block size used is fixed to 16×16 but the search range is varied depending on the amount of motion vector. In sequence, this process was continued from the previous part, where the last predicted frame was taken as the new reference frame for the next frame prediction. A directive search process is done which started from the centre of search window. The process then next proceeded from the most interest region (area that has biggest search range) to less interest region as in *Figure 10*.

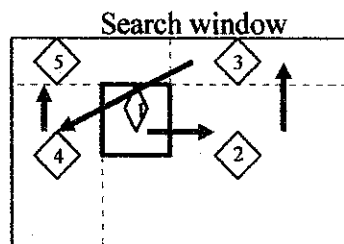


Figure 10 Directive Search Process

3.1.5 Predicted image reconstruction

The filtered predicted image was saved in Bitmap format replacing the current frame to be used for the next prediction.

3.1.6 Computation Count

For this adaptive search range motion estimation, the computation count varies between different block and different frames depending on the size of search range (R_1 , R_2 , R_3 and R_4 parameters). The computation count is one of the parameters in estimating the encoder performance. The smaller the search range, lesser the computation count, thus the execution time is reduced. The computation count for an entire frame is calculated using this formula:

$$\text{Frame computation} = \text{Height} \times \text{Width} \times (R_1 + R_2 + 1) \times (R_3 + R_4 + 1) \quad (10)$$

Meanwhile, the total search point count for entire frame indicates by:

$$\text{Search point per frame} = \frac{(\text{Height} \times \text{Width})}{B^2} (R_1 + R_2 + 1)(R_3 + R_4 + 1) \quad (11)$$

3.1.7 PSNR Determination

Peak-Signal-to-Noise Ratio (PSNR) is another matrix that is used to evaluate the encoder performance in term of reconstructed image quality produced. The PSNR is measured as in (4). Appendix C shows the MATLAB codes for calculating the PSNR of reconstructed frame.

3.2 Summary of algorithm

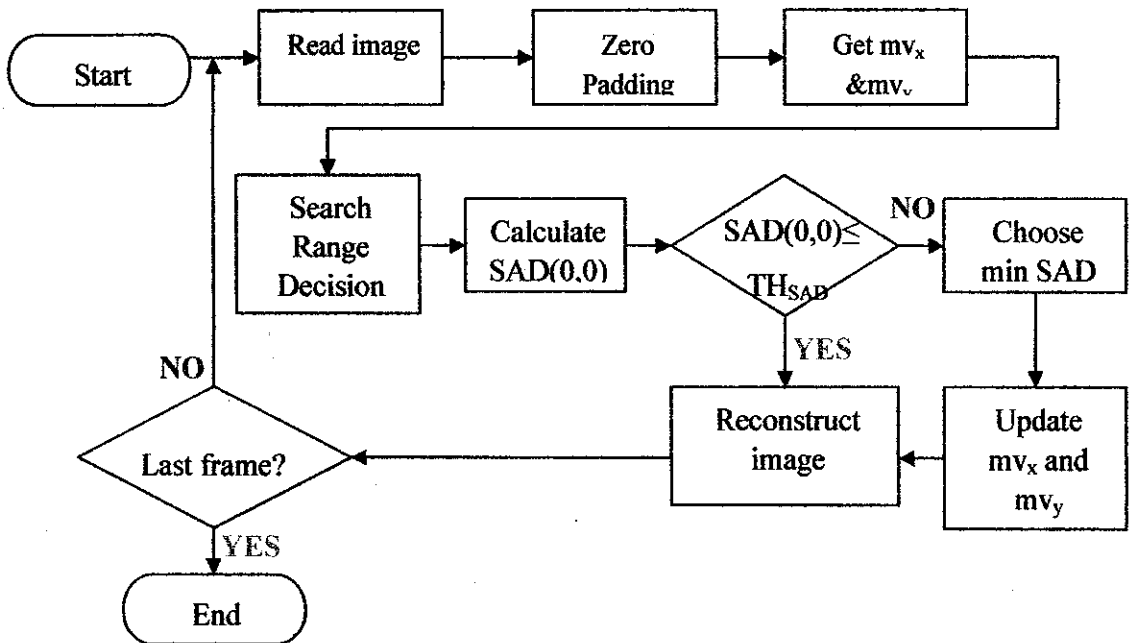


Figure 11 Flow diagram of the proposed algorithm.

Figure 11 shows the flow diagram of the adaptive search range motion estimation for the program in Appendix D. The proposed work as follows:

Step 1) Initially, the mv_x and mv_y are considered as default value.

Step 2) Read the reference and current image.

Step 3) Pad the border of reference image with 0.

Step 4) Get the mv_x and mv_y for all blocks.

Step 5) Perform the search range decision for R_1 , R_2 , R_3 , and R_4 .

Step 6) Apply the adaptive search range in calculating SAD.

Step 7) Check if SAD (0, 0) greater than TH_{SAD} . Else, terminate the ME process and go to step 10.

Step 8) Check MV predictors and choose the minimum SAD as the new mv_x and mv_y .

Step 9) Update the new mv_x and mv_y .

Step 10) Reconstruct the image and predict for the next frame.

3.3 Tools Required








Throughout this project, the simulation is done using mathematics software called MATLAB. Therefore, knowledge in that software is essential in order to developed programs for this project. MATLAB is a powerful mathematical tool that allows matrix computation, calculation and plotting data. It consists of several toolboxes including image processing block set toolbox. *Table 12* lists name of the programs have been developed using MATLAB for this project.

Table 12 Matlab Program File

M-file Name	Functions
loopPSNR.m	Calculates the PSNR between the compressed image and original image shown in Appendix C .
NEWADAPTIVE.m	Calculates SAD, performs adaptive search range motion estimation shown in Appendix D .
NEWFS.m	Calculates SAD and performs motion estimation using non-adaptive search range (fixed search range) and reconstruct the predicted image shown in Appendix E .
NEWADAPTIVE_ET.m	Calculates SAD, do adaptive search range motion estimation with early termination shown in Appendix F .
extract.m	Splits the Y, U and V component from .YUV or .QCIF video shown in Appendix G .

Various uncompressed raw video sequences are used in this project. The luminance (Y) and chrominance (U and V) components are split using the codes `extract.m` and `SPLIT.m` as in **Appendix F**. However, only the Y component is used in the motion estimation part. *Table 13* contains the videos and the respective 30 frame sequence that been used for prediction.

Table 13 Raw video

Video Name	Sample Image	Frame Number
Claire.qcif		1-30
Carphone.qcif		1-30
Container.qcif		1-30
News.qcif		1-30
Foreman.yuv		31-60
Coastguard.yuv		151-180
Hall_monitor.qcif		151-180

CHAPTER 4

RESULTS AND DISCUSSION

RESULTS AND DISCUSSION

4.1 Motion Vector Estimation



Figure 12 Motion vector field for *Coastguard.qcif* at frame 152.

The proposed directional adaptive search window method helps to dynamically select the search range and orientation of search window. Initially, the motion estimation was done in the predicted frame 152 with fixed block size of 16 and search range of 7. The motion vector determined is used to decide the search range for next frame prediction based on the design criteria stated in *Chapter 3*. The motion vectors are updated for each block computation in every frame.

Figure 12 illustrates the motion vector field in the first predicted frames based on frame 151 as reference for *Coastguard.qcif* that has uniform amount of motion and

4.3 Comparison between Non-adaptive and Adaptive Search Window

Non-adaptive search window is a conventional algorithm where the search range is $R=7$ for all blocks in the predicted frame. **Appendix E** shows the program for the non-adaptive search window. Using the adaptive search window motion estimation, the size of search range is varied depending on the motion vectors predicted from the frame. As a result, the number of computation differs from one frame to another. *Figure 14* illustrates the number of search point for both adaptive and non-adaptive methods. The number of search point for non-adaptive is consistent throughout all frames which are 22,275/frame. Thus, this contributes to higher execution time for the encoder.

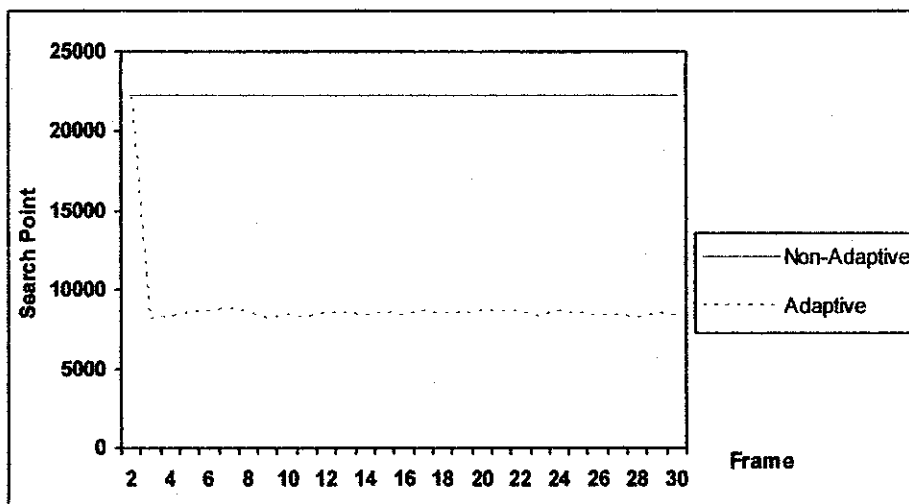


Figure 14 Number of search point for “Hall Monitor.qcif”.

For the adaptive search window, the number of search point is only equal to 22,275/frame during the initial state due to fixed search range. After the motion vectors are determined from the first predicted frame, the search range is dynamically adjusted for each block in each respective frame. The number of search point dropped rapidly to the range between 8,000/frame to 10,000/frame. Therefore, more time is saved for video coding by using this adaptive algorithm.

Table 14 Total number of computation for 30 frames of *Hall Monitor.qcif*

Parameters	Total Computation for 30 frames
Non-Adaptive	165,369,600
Adaptive	66,546,432

Table 15 Total number of search point for 30 frames of *Hall Monitor.qcif*

Parameters	Total Search Point for 30 frames
Non-Adaptive	645,975
Adaptive	259,947

Table 14 shows the total number of computation for 30 sequence frames of *Hall Monitor.qcif*. As can be seen, there are a lot of difference in number of computation for both methods which is of about 59.76%. It is more than half of the total computation for non-adaptive search window motion estimation. Meanwhile, Table 15 shows that number of search point is reduced up to 59.76% from the conventional method. Therefore, it proves that the number of computation is directly proportional to the search point. As a consequence, the adaptive search window can reduce the execution time about 50% from the non-adaptive method.

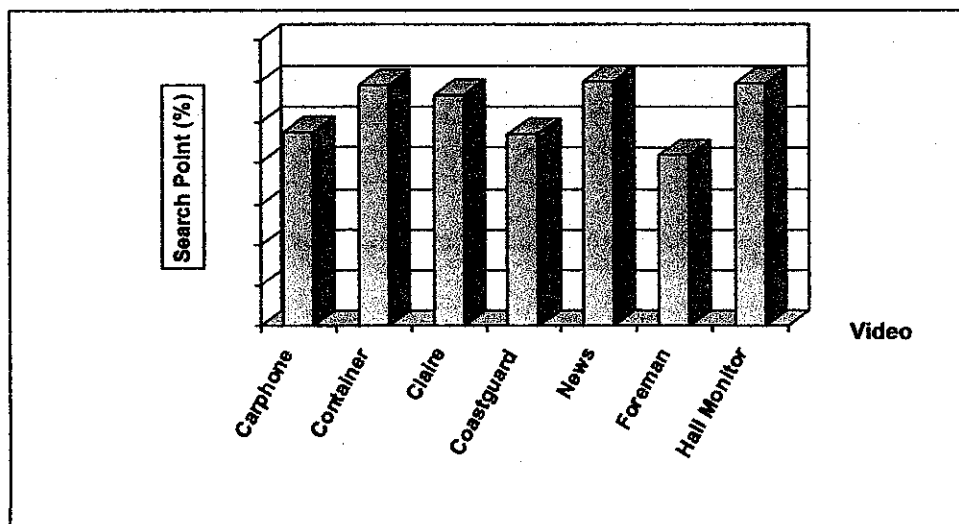


Figure 15 Percentage of search point difference between adaptive and non-adaptive

Table 16 The average number of search points.

Video Sequence	Number of Search Point		Search Point Difference	Percentage Difference (%)
	Non- adaptive	Adaptive		
claire	22,275	9639.93103	12635.069	56.72
carphone	22,275	11603.7931	10671.2069	47.91
container	22,275	9084.10345	13190.8966	59.22
news	22,275	8872.13793	13402.8621	60.17
foreman	22,275	12931.4483	9343.55172	41.95
coastguard	22,275	11729.7931	10545.2069	47.34
hall	22,275	8963.68966	13311.3103	59.76

Figure 15 illustrates the percentage reduction of search point for the proposed search window algorithm compared to the conventional algorithm. The results show that number of computation is decreased by 40% to 60% based on the amount of motion content for the whole frames. Small amount of motion such as *Claire* and *Container* have higher percentage difference compared to high motion such as *Foreman* and *Carphone*. For instance, the average execution time is reduced by almost 50%. Table 16 shows the respective values for Figure 15.

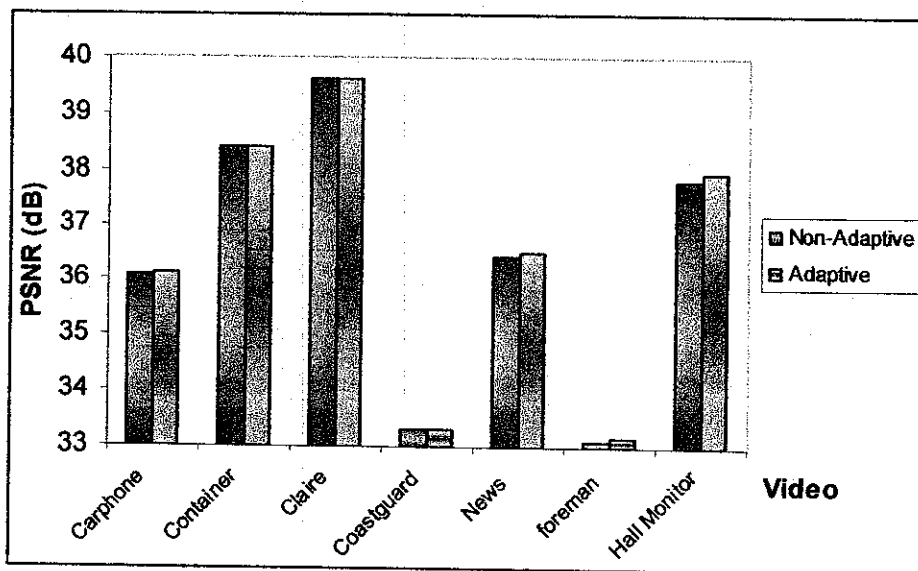


Figure 16 The average PSNR comparisons for several videos.

In *Figure 16*, the average PSNR for seven 30-frame video with uniform motion are analyzed and tabulated in the graph. The graph shows that there is only small degradation in terms of PSNR for *Claire* and *container* video which is about 0.0337% and 0.0141% from the conventional method which is negligible. This is due to the motion vectors throughout the motion estimation process did not have constant direction of motion vector. The direction keeps on changing from frame to frame which is not predictive for the next frame.

The remaining five video sequences namely *coastguard*, *news*, *foreman*, *carphone* and *hall monitor* have larger average PSNR for the adaptive search window motion estimation. *Coastguard* PSNR is increased by 0.00321 dB or 0.00962%, while *Hall monitor* is increased by 0.11962 dB or 0.31625% from the non-adaptive method. This result is expected because these five videos contained almost uniform amount of motion and direction. *Table 17* lists the exact values of the average PSNR comparison used in *Figure 16*.

Table 17 The average PSNR comparison.

Video Sequence	Average PSNR		PSNR Difference (dB)	Percentage Difference (%)
	Non- adaptive	Adaptive		
claire	39.61845	39.60352	-0.0149	-0.0377
carphone	36.07479	36.10183	0.02703	0.07494
container	38.42221	38.41679	-0.0054	-0.0141
news	36.43452	36.51662	0.0821	0.22535
foreman	33.10203	33.18141	0.07938	0.2398
coastguard	33.32583	33.32903	0.00321	0.00962
hall	37.82438	37.944	0.11962	0.31625

4.4 Adaptive Motion Estimation with Early Termination

The comparison results for various algorithms are listed in *Table 18, 19, 20, and 21*. The three types of algorithm that are used for comparison are full search (FS), adaptive search window motion estimation, and adaptive motion estimation with early termination. The early termination contains two options namely Option 1 where the threshold (TH_SAD) is set to be the value of x at $F(x \leq 0.6)$ of the stationary SAD. For Option 2, the value of x at $F(x \leq 0.8)$ of stationary SAD is set as the threshold (TH_SAD) value. The numbers of search points can be reduced further by adapt the early termination in the adaptive motion estimation with only small degradation of video quality as penalty.

Table 18 Comparison of the average search point and PSNR using various algorithms for "Container.qcif".

Carphone	TH_SAD	Average Search Point			Average PSNR (dB)		
		Value	ΔSP	%	Value	$\Delta PSNR$	%
FS	-	22275	0	0	38.42221	0	0
Adaptive	-	9084.103	-13190.9	-59.22	38.41679	-0.0054	-0.0141
Option 1	179.312	8981.379	-13293.6	-59.68	38.38248	-0.0397	-0.1034
Option 2	226.736	8945.586	-13329.4	-59.84	38.37114	-0.0511	-0.1329

Table 19 Comparison of the average search point and PSNR using various algorithms for "Claire.qcif".

Container	TH_SAD	Average Search Point			Average PSNR (dB)		
		Value	ΔSP	%	Value	$\Delta PSNR$	%
FS	-	22275	0	0	39.61845	0	0
Adaptive	-	9639.931	-12635.1	-56.72	39.60352	-0.0149	-0.0377
Option 1	130.368	9472.345	-12802.7	-57.48	39.58255	-0.0359	-0.0906
Option 2	196.304	9012.621	-13262.4	-59.54	39.52238	-0.0961	-0.2425

Table 20 Comparison of the average search point and PSNR using various algorithms for “News.qcif”.

Claire	TH_SAD	Average Search Point			Average PSNR (dB)		
		Value	ΔSP	%	Value	$\Delta PSNR$	%
FS	-	22275	0	0	36.43452	0	0
Adaptive	-	8872.138	-13402.9	-60.17	36.51662	0.0821	0.2254
Option 1	340.196	8796.414	-13478.6	-60.51	36.22741	-0.2071	-0.5684
Option 2	502.488	8757.621	-13517.4	-60.68	35.84514	-0.5894	-1.6176

Table 21 Comparison of the average search point and PSNR using various algorithms for “Hall Monitor.qcif”.

Coastguard	TH_SAD	Average Search Point			Average PSNR (dB)		
		Value	ΔSP	%	Value	$\Delta PSNR$	%
FS	-	22275	0	0	37.82438	0	0
Adaptive	-	8963.69	-13311.3	-59.76	37.944	0.11962	0.3163
Option 1	369.312	8930.28	-13344.7	-59.91	37.41814	-0.4062	-1.074
Option 2	520.736	8681.69	-13593.3	-61.02	36.80031	-1.0241	-2.7074

Table 22 Comparison of the percentage reduction of search point for the proposed algorithms.

Search Point	Adaptive	Option 1	Option 2
Container	-59.22%	-59.68%	-59.84%
Claire	-56.72%	-57.48%	-59.54%
News	-60.17%	-60.51%	-60.68%
Hall Monitor	-59.76%	-59.91%	-61.03%

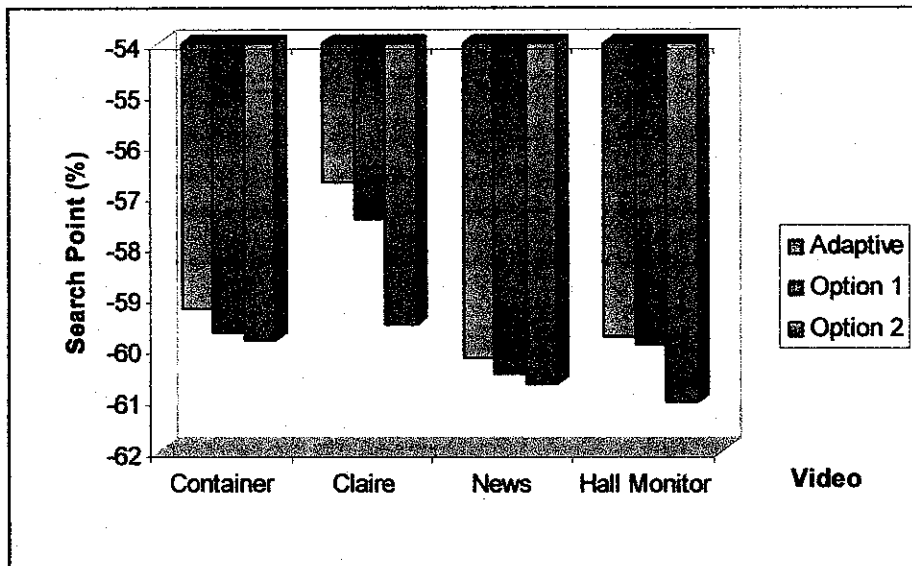


Figure 17 Comparison of percentage search point reduction for proposed algorithm.

Table 22 and Figure 17 summarize the percentage of search point reduction from Table 18 to Table 21. Through the results for all four type videos, the computation of search point shows large reduction mostly using option 2 compare to option 1 since the value of x at $F(x \leq 0.8)$ is larger than $F(x \leq 0.6)$.

Table 23 Comparison of the percentage reduction of PSNR for the proposed algorithms.

Search Point	Adaptive	Option 1	Option 2
Container	-0.0141%	-0.1034%	-0.13292%
Claire	-0.0377%	-0.0906%	-0.24249%
News	0.22535%	-0.5684%	-1.61764%
Hall Monitor	0.31625%	-1.074%	-2.70743%

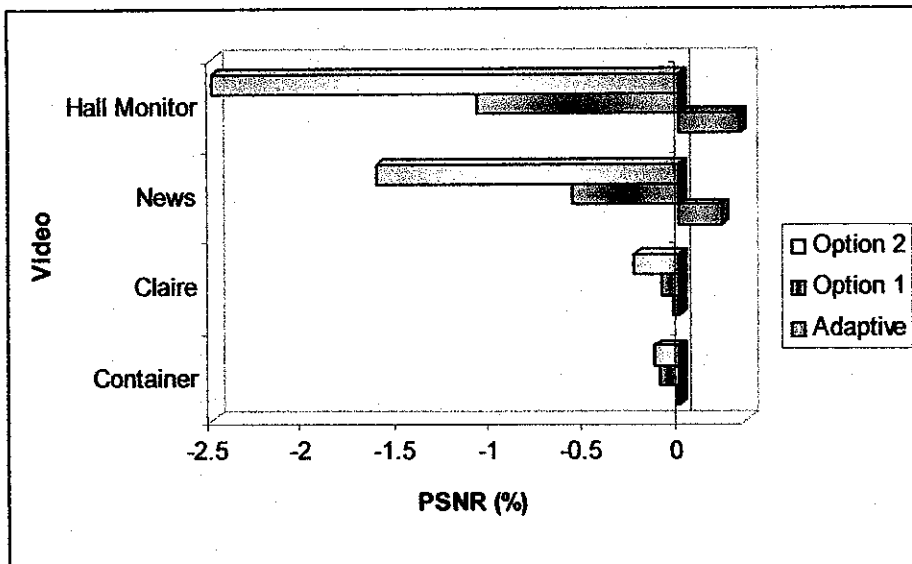


Figure 18 Comparison of percentage PSNR reduction for proposed algorithm.

In contrast, *Table 23* and *Figure 18* summarize the percentage of PSNR reduction from *Table 18* to *Table 21*. Through the results, it is observed that the images have experienced large degradation of quality by using Option 2 compare to Option 1. The reason that certifies the results is there is a trade off between the video quality performance and time saving. The larger the threshold (TH_SAD) value employed in the termination process, the more search points and candidate blocks are skipped. This results in less number of computations, but leads to a larger quality degradation and motion distortion.

The results show that for Option 2 the number of search point is reduced by an average of 1.3% from the algorithm without early termination. Option 1 only gives a reduction of 0.4225%. For example, the PSNR reduction from the adaptive motion estimation without early termination is only 1.027% for Option 2 and 0.3101% for Option 1. Therefore, it can be concluded that a large threshold value results in reducing in number of computation as well as image quality.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

This research is based on designing adaptive motion estimation for H.264 video compression with early termination in order to optimize the performance video compression in term of better PSNR with lesser time of execution. The best motion estimation parameter is determined based on the predicted amount of motion or motion vector in a particular frame. As a result, it shows that there are three categories of motion that can be estimated from the motion vector using the sum of absolute difference (SAD). From the amount of motion vectors, the motion estimation parameter which is the search window is dynamically adjusted based on the decision criteria. It shows that, using adaptive search window motion estimation, the image quality in term of PSNR for five video sequences is better compared to the non-adaptive method. Only small degradation of PSNR resulting from proposed algorithm for two other video sequences (most of 0.038%) which is negligible. Besides that, execution time could be reduced to half (50%) of the non-adaptive method as it reduces the number of search points and computations between the ranges of 40% to 60%. Through the comparison result, by using the early termination the number of computation could be reduced further more with average 1.3% for Option 2. The image has experience degradation of image quality with average of 1.027% from the adaptive motion estimation without the early termination process. This is done by eliminate the search process for stationary block. This proved that the larger the threshold value, the more reduction in term of computation and image quality. For future work, it is recommended to applied the inter/intra mode decision to increase the reconstructed image quality.

CHAPTER 6

REFERENCES

- [1] N. N. Abu Bakar and N. Badruddin, "Performance Comparison of Block Matching Techniques in Motion Estimation for Different Types of Video Sequences", Final Year Project Thesis, Universiti Teknologi PETRONAS, Perak, Malaysia, 2005.
- [2] Wikipedia, "H.264/MPEG-4 AVC", 30 Jan 2006,
http://en.wikipedia/wiki/H.264/MPEG-4_AVC
- [3] A Whatis.com, "H.264", 6 Feb 2006, http://searchsmb.techtarget.com/sDefinition/0,,sid44_gci934039,00.html
- [4] L. Kohout, "Motion Estimation Tutorial", 20 March 1998,
<http://stargate.ecn.purdue.edu/~ips/tutorials/me>
- [5] Richardson, Iain E. G., "Video Codec Design - Developing Image and Video Compression Systems", West Sussex, John Wiley & Sons Ltd, 2002.
- [6] A. Murat Tekalp, "Digital Video Processing", Upper Saddle River, Prentice Hall Signal Processing Series, 1995.
- [7] Y. Liang, I. Ahmad, J. Luo, Y. Sun, and V. Swaminathan, "On Using Hierarchical Motion History for Motion Estimation in H.264/AVC", *IEEE Transactions on Circuit and Systems for Video Technology*, vol. 15, Dec. 2005.
- [8] G. L. LI and M. J. Chen, "Adaptive Search Range Decision and early Termination for Multiple Reference Frame Motion Estimation for H.264", *IEICE Trans. Commun.*, vol. E89-B, Jan. 2006.
- [9] I. E. G. Richardson, "H.264 and MPEG-4 Video Compression", West Sussex, New York: John Wiley & Sons Ltd, 2003.
- [10] W. Ni, B. Guo, G. Ding and L. Yang, "Motion Vector Field and Direction Adaptive Search Technique for Motion Estimation", *Journal of Information & Computational Science* 2:3, September 2005.

[11] T. Xu and W. Chen, "A Fast Adaptive Statistical Genetic Motion Search Algorithm for H.264/AVC", *Proc. Of IEEE International Conference 2006 on Advanced Information Networking and Applications (AINA'06)*, 2006

[12] Z. Yang, J. Bu, C. Chen and X. Li, "Fast Predictive Variable-block Size Motion Estimation for H.264/AVC" in *Proc. Of IEEE International Conference on Multimedia & Expo 2005*, July 2005, pp. 354-357.

APPENDICES

Appendix A Gantt Chart

Appendix B Window Size

Appendix C Calculate PSNR (loopPSNR.m)

Appendix D Adaptive Motion Estimation (NEWADAPTIVE.m)

Appendix E Non-Adaptive Motion Estimation (NEWFS.m)

Appendix F Adaptive Motion Estimation With Early Termination
(NEWADAPTIVE_ET.m)

Appendix G Split .YUV And .QCIF Video (extract.m)

APPENDIX A

Gantt Chart

First Semester of 2 Semester Final Year Project

No. Detail/Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
1 Selection of Project Topic	█														
- Topic assigned to students															
2 Preliminary Research Work	█														
- Study on H.264 and ME															
3 Submission of Preliminary Report			●												
4 Project Work	█														
- Design decision criteria															
- Predict the amount of motion/ motion vector															
- Start on programming in MATLAB															
5 Submission of Progress Report															
								●							
6 Project work continue	█														
- Continue with the MATLAB															
- Analyze and compare the result performance															
7 Submission of Interim Report															
												●			
8 Oral Presentation															
														●	

● Suggested milestone █ Process

Second Semester of 2 Semester Final Year Project

No.	Detail/ Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	Project Work Continue -Research and study on Early Termination														
2	Submission of Progress Report 1		●												
3	Project Work Continue -Set the design criterion -Continue programming in MATLAB														
4	Submission of Progress Report 2						●								
5	Project work continue -Continue with the MATLAB - Analyze and compare the result														
6	Submission of Dissertation Final Draft									●					
7	Submission of Final Report										●				
8	Oral Presentation												●		
9	Submission of Project Dissertation														●

● Suggested milestone ■ Process

APPENDIX B
Window Size

R1	R2	R3	R4	Window Size
4	4	4	4	24 x 24
4	4	4	7	24 x 27
4	4	4	16	24 x 64
4	4	7	4	24 x 27
4	4	7	7	24 x 30
4	4	7	16	24 x 39
4	4	16	4	24 x 36
4	4	16	7	24 x 39
4	4	16	16	24 x 48
4	7	4	4	27 x 24
4	7	4	7	27 x 27
4	7	4	16	27 x 64
4	7	7	4	27 x 27
4	7	7	7	27 x 30
4	7	7	16	27 x 39
4	7	16	4	27 x 36
4	7	16	7	27 x 39
4	7	16	16	27 x 48
4	16	4	4	36 x 24
4	16	4	7	36 x 27
4	16	4	16	36 x 64
4	16	7	4	36 x 27

4	16	7	7	36 x 30
4	16	7	16	36 x 39
4	16	16	4	36 x 36
4	16	16	7	36 x 39
4	16	16	16	36 x 48
7	4	4	4	27 x 24
7	4	4	7	27 x 27
7	4	4	16	27 x 64
7	4	7	4	27 x 27
7	4	7	7	27 x 30
7	4	7	16	27 x 39
7	4	16	4	27 x 36
7	4	16	7	27 x 39
7	4	16	16	27 x 48
7	7	4	4	30 x 24
7	7	4	7	30 x 27
7	7	4	16	30 x 64
7	7	7	4	30 x 27
7	7	7	7	30 x 30
7	7	7	16	30 x 39
7	7	16	4	30 x 36
7	7	16	7	30 x 39
7	7	16	16	30 x 48
7	16	4	4	39 x 24
7	16	4	7	39 x 27
7	16	4	16	39 x 64

7	16	7	4	39 x 27
7	16	7	7	39 x 30
7	16	7	16	39 x 39
7	16	16	4	39 x 36
7	16	16	7	39 x 39
7	16	16	16	39 x 48
16	4	4	4	36 x 24
16	4	4	7	36 x 27
16	4	4	16	36 x 64
16	4	7	4	36 x 27
16	4	7	7	36 x 30
16	4	7	16	36 x 39
16	4	16	4	36 x 36
16	4	16	7	36 x 39
16	4	16	16	36 x 48
16	7	4	4	39 x 24
16	7	4	7	39 x 27
16	7	4	16	39 x 64
16	7	7	4	39 x 27
16	7	7	7	39 x 30
16	7	7	16	39 x 39
16	7	16	4	39 x 36
16	7	16	7	39 x 39
16	7	16	16	39 x 48
16	16	4	4	48 x 24
16	16	4	7	48 x 27

16	16	4	16	48 x 64
16	16	7	4	48 x 27
16	16	7	7	48 x 30
16	16	7	16	48 x 39
16	16	16	4	48 x 36
16	16	16	7	48 x 39
16	16	16	16	48 x 48

APPENDIX C

Calculate PSNR (loopPSNR.m)

```
function [a,b,PSNR]=loopPSNR(name,namereal,f1,f2)
a=0;
for loop = f1:f2
cimg = imread (strcat(num2str(loop),name));
oimg = imread (strcat(num2str(loop),namereal));

[m,n]=size(cimg);

eimg=cimg-oimg;
mse=sum(sum(eimg.^2))/(m*n);

RMSE=sqrt(mse);

PSNR(loop)=20*log10(255/RMSE);
a=a+PSNR(loop);
end
b=a/29
```

APPENDIX D

Adaptive Motion Estimation (NEWADAPTIVE.m)

```
%%Adaptive motion estimation (search range) with adaptive deblocking
%%filter
function [Set_P, calc, num] = NEWADAPTIVE(name, f1, f2)

A=ones(9,11);
B=ones(9,11);

exe=0;
mvx=A*111;
mvy=B*111;

for loop = f1:f2 %start loop frames
    num=loop+2-f1;
    comp =0;
    cmp=0;

    i1 = imread (strcat(num2str(loop-1),name));
    im2 = imread (strcat(num2str(loop), name));
    B = 16;%block size
    sz = B*B;
    R = 7; %search window
    [height,width] = size(i1); %width = row, height = column
    im1 = padarray(i1,[(R) (R)],0,'both');%pad both x and y sides with '0'

    n1=0;
    for i=1:B:height-B+1
        r=i-(n1*B)+n1;

        n2=0;
        for j=1:B:width-B+1 %for every block in the reference and predicted
            image frame (dinstinct)
                c=j-(n2*B)+n2;

                vx = mvx(r,c);
                vy = mvy(r,c);
                s_point =0;
```

```

if vx==0
    R1=4;
    R2=4;

elseif abs(vx)>0 & abs(vx)<=4
    if vx>0
        R1=4;
        R2=7;
    else
        R1=7;
        R2=4;
    end

elseif abs(vx)>4 & abs(vx)<100
    if vx>0
        R1=4;
        R2=16;
    else
        R1=16;
        R2=4;
    end

else
    R1=7;
    R2=7;

end

if vy==0
    R3=4;
    R4=4;
elseif abs(vy)>0 & abs(vy)<=4
    if vy>0
        R3=4;
        R4=7;
    else
        R3=7;
        R4=4;
    end
end

```



```
elseif abs(vy)>4 & abs(vy)<100
```

```
  if vy>0
```

```
    R3=4;
```

```
    R4=16;
```

```
  else
```

```
    R3=16;
```

```
    R4=4;
```

```
  end
```

```
else
```

```
  R3=7;
```

```
  R4=7;
```

```
end
```

```
if R1>R2
```

```
  RA=-R1;
```

```
  RB=R2;
```

```
  d1=-1;
```

```
  d2=1;
```

```
else
```

```
  RA=R2;
```

```
  RB=-R1;
```

```
  d1=1;
```

```
  d2=-1;
```

```
end
```

```
if R3>R4
```

```
  RC=-R3;
```

```
  RD=R4;
```

```
  d3=-1;
```

```
  d4=1;
```

```
else
```

```
  RC=R4;
```

```
  RD=-R3;
```

```
  d3=1;
```

```
  d4=-1;
```

```
end
```

```
SADmin = 256*B*B;
```

```

for k =[0:d3:RC]
    for l =[0:d1:RA]
        SADabsdiff = imabsdiff(im2(i:i+B-1,j:j+B-1),im1(i+k+R3+(R-
R3):i+k+R4+(R-R4)+B-1,j+l+R1+(R-R1):j+l+B+R2+(R-R2)-1));%   im1   will
slide its block by one pixel at a time
        SAD = sum(sum(SADabsdiff));%sum all SAD avg for one row

        if SAD < SADmin
            SADmin = SAD;
            dx=1;
            dy=k;
        end
        s_point = s_point+1;
    end
end

for k =[0:d3:RC]
    for l =[d2:d2:RB]
        SADabsdiff = imabsdiff(im2(i:i+B-1,j:j+B-1),im1(i+k+R3+(R-
R3):i+k+R4+(R-R4)+B-1,j+l+R1+(R-R1):j+l+B+R2+(R-R2)-1));%   im1   will
slide its block by one pixel at a time
        SAD = sum(sum(SADabsdiff));%sum all SAD avg for one row

        if SAD < SADmin
            SADmin = SAD;
            dx=1;
            dy=k;
        end
        s_point = s_point+1;
    end
end

for k =[d4:d4:RD]
    for l =[0:d1:RA]
        SADabsdiff = imabsdiff(im2(i:i+B-1,j:j+B-1),im1(i+k+R3+(R-
R3):i+k+R4+(R-R4)+B-1,j+l+R1+(R-R1):j+l+B+R2+(R-R2)-1));%   im1   will
slide its block by one pixel at a time
        SAD = sum(sum(SADabsdiff));%sum all SAD avg for one row

        if SAD < SADmin
            SADmin = SAD;
            dx=1;

```

```

        dy=k;
    end
    s_point = s_point+1;
end
end

for k =[d4:d4:RD]
    for l =[d2:d2:RB]
        SADabsdiff = imabsdiff(im2(i:i+B-1,j:j+B-1),im1(i+k+R3+(R-
R3):i+k+R4+(R-R4)+B-1,j+l+R1+(R-R1):j+l+B+R2+(R-R2)-1));% im1 will
slide its block by one pixel at a time
        SAD = sum(sum(SADabsdiff));%sum all SAD avg for one row

        if SAD < SADmin
            SADmin = SAD;
            dx=l;
            dy=k;
        end
        s_point = s_point+1;
    end
end

cmp=(R1+R2+1)*(R3+R4+1)*B^2;

%_____writing predicted image_____

imp(i:i+B-1,j:j+B-1)= im1(i+dy+R:i+dy+R+B-1,dx+R+j:j+dx+R+B-1);
    iblk = floor ((i-1)/B+1);
    jblk = floor ((j-1)/B+1);
    mvx(iblk,jblk) = dx;
    mvy(iblk,jblk) = dy; %record the estimated MV

    SAD99(iblk,jblk) = SAD;
    r1(iblk,jblk) = R1;
    r2(iblk,jblk) = R2;
    r3(iblk,jblk) = R3;
    r4(iblk,jblk) = R4;
    SP(iblk,jblk) = s_point;

    SAD1(iblk,jblk)= SADmin;

```

```

n2=n2+1;

    comp=comp+cmp;
end

n1=n1+1;
end
% _____ %
%deblocking filter

inp=inp;
dd=B;

[dimy,dimx]=size(inp);    % Calcolo dimensione immagine

p=0;    % Azzeramento contatore strong filter

% Passaggio su tutte le righe...
for j=dd:dd:dimx-dd
    for i=1:1:dimy
        diff=inp(i,j+1)-inp(i,j);    % x=D-C
        diff1=inp(i,j-1)-inp(i,j);    % x1=B-C
        diff2=inp(i,j+2)-inp(i,j+1);    % x2=E-D
        if ((abs(diff1)<5)&(abs(diff2)<5))    % Strong filter (if
|x1|,|x2|<5)
            inp(i,j-2)=inp(i,j-2)+(diff/8);    % a=A+x/8
            inp(i,j-1)=inp(i,j-1)+(diff/4);    % b=B+x/4
            inp(i,j)=inp(i,j)+(diff/2);    % c=C+x/2
            inp(i,j+1)=inp(i,j+1)-(diff/2);    % d=D-x/2
            inp(i,j+2)=inp(i,j+2)-(diff/4);    % e=E-x/4
            inp(i,j+3)=inp(i,j+3)-(diff/8);    % f=F-x/8
            p=p+1;    % Aumento cont strong filter
        else    % Soft filter
            inp(i,j-1)=inp(i,j-1)+(diff/8);    % b=B+x/8
            inp(i,j)=inp(i,j)+(diff/2);    % c=C+x/2
            inp(i,j+1)=inp(i,j+1)-(diff/2);    % d=D-x/2
            inp(i,j+2)=inp(i,j+2)-(diff/8);    % e=E-x/8

        end;
    end;
end;

```

```

%Passaggio su tutte le colonne...
for i=dd:dd:dimy-dd
    for j=1:1:dimx
        diff=inp(i+1,j)-inp(i,j);           % x=D-C
        diff1=inp(i-1,j)-inp(i,j);         % x1=B-C
        diff2=inp(i+2,j)-inp(i+1,j);       % x2=E-D
        if ((abs(diff1)<5)&(abs(diff2)<5))    % Strong filter (if
|x1|,|x2|<5)
            inp(i-2,j)=inp(i-2,j)+(diff/8); % a=A+x/8
            inp(i-1,j)=inp(i-1,j)+(diff/4); % b=B+x/4
            inp(i,j)=inp(i,j)+(diff/2);     % c=C+x/2
            inp(i+1,j)=inp(i+1,j)-(diff/2); % d=D-x/2
            inp(i+2,j)=inp(i+2,j)-(diff/4); % e=E-x/4
            inp(i+3,j)=inp(i+3,j)-(diff/8); % f=F-x/8
            p=p+1;                           % Aumento cont strong filter
        else                                  % Soft filter
            inp(i-1,j)=inp(i-1,j)+(diff/8); % b=B+x/8
            inp(i,j)=inp(i,j)+(diff/2);     % c=C+x/2
            inp(i+1,j)=inp(i+1,j)-(diff/2); % d=D-x/2
            inp(i+2,j)=inp(i+2,j)-(diff/8); % e=E-x/8
        end;
    end;
end;

inp;

%
-----

SAD1
mvx
mvy

[X,Y] = meshgrid(1:B:width-B+1,1:B:height-B+1); %Set motion vector
coordinates

figure, imshow(inp,[0 255]); title (strcat(num2str(loop),'Predicted
Frame'))
hold on
quiver(X,Y,mvx,mvy)
hold off

SN = strcat(num2str(loop),name);

```

```
imwrite(inp,SN,'bmp') %write predicted image to file
%imwrite(imp,strcat(num2str(loop),'Ynews.Y')) %write predicted image to
file

tot_SP = sum(sum(SP))
Set_P(num) = tot_SP

exe=exe+comp;
calc(num)=comp;
% _____ %
end %end loop frames

totexe=exe
```

APPENDIX E

Non-Adaptive Motion Estimation (NEWFS.m)

```
%%%Full Search Motion Estimation
function [Set_P, calc, num] = NEWFS(name,f1,f2)

exe=0;
%name = 'hall_monitorY.Y';
%f1=152;
%f2=180;

for loop = f1:f2 %start loop frames
    num=loop+2-f1;
    comp =0;
    cmp=0;

    i1 = imread (strcat(num2str(loop-1),name));
    im2 = imread (strcat(num2str(loop), name));
    B = 16;%block size
    sz = B*B;
    R = 7; %search window
    [height,width] = size(i1); %width = row, height = column
    im1 = padarray(i1,[(R) (R)],0,'both');%pad both x and y sides with '0'

    for i=1:B:height-B+1
        for j=1:B:width-B+1 %for every block in the reference and predicted
            image frame (dinstinct)
                s_point =0;

            %j,k= column, i,l = row
            SADmin = 256*B*B;

            for k =[0:R -R:-1]
                for l =[0:R -R:-1]

                    SADabsdiff = imabsdiff(im2(i:i+B-1,j:j+B-1),im1(i+R+k:i+R+k+B-1,j+R+l:j+R+l+B-1));% im1 will slide its block by
                    one pixel at a time
                    SAD = sum(sum(SADabsdiff));%sum all SAD avg for one row
```

```

        if SAD < SADmin
            SADmin = SAD;
            dx=1;
            dy=k;
        end

        s_point = s_point+1;

    end
end

cmp=(R+R+1)*(R+R+1)*B^2;

% _____writing predicted image_____ %

imp(i:i+B-1,j:j+B-1)= im1(i+dy+R:i+dy+R+B-1,dx+R+j:j+dx+R+B-1);
    iblk = floor ((i-1)/B+1);
    jblk = floor ((j-1)/B+1);
    mvx(iblk,jblk) = dx;
    mvy(iblk,jblk) = dy; %record the estimated MV

    SAD99(iblk,jblk) = SAD;
    SP(iblk,jblk) = s_point;
    SAD1(iblk,jblk)= SADmin;

    comp=comp+cmp;
end

end

% _____ %

% _____ %

%deblocking filter

inp=imp;
dd=B;

[dimy,dimx]=size(inp);    % Calcolo dimensione immagine

p=0;    % Azzeramento contatore strong filter

```



```

% Passaggio su tutte le righe...
for j=dd:dd:dimx-dd
    for i=1:1:dimy
        diff=inp(i,j+1)-inp(i,j);           % x=D-C
        diff1=inp(i,j-1)-inp(i,j);         % x1=B-C
        diff2=inp(i,j+2)-inp(i,j+1);       % x2=E-D
        if ((abs(diff1)<5)&(abs(diff2)<5))    % Strong filter (if
|x1|,|x2|<5)
            inp(i,j-2)=inp(i,j-2)+(diff/8); % a=A+x/8
            inp(i,j-1)=inp(i,j-1)+(diff/4); % b=B+x/4
            inp(i,j)=inp(i,j)+(diff/2);     % c=C+x/2
            inp(i,j+1)=inp(i,j+1)-(diff/2); % d=D-x/2
            inp(i,j+2)=inp(i,j+2)-(diff/4); % e=E-x/4
            inp(i,j+3)=inp(i,j+3)-(diff/8); % f=F-x/8
            p=p+1;                           % Aumento cont strong filter
        else                                  % Soft filter
            inp(i,j-1)=inp(i,j-1)+(diff/8); % b=B+x/8
            inp(i,j)=inp(i,j)+(diff/2);     % c=C+x/2
            inp(i,j+1)=inp(i,j+1)-(diff/2); % d=D-x/2
            inp(i,j+2)=inp(i,j+2)-(diff/8); % e=E-x/8

        end;
    end;
end;

%Passaggio su tutte le colonne...
for i=dd:dd:dimy-dd
    for j=1:1:dimx
        diff=inp(i+1,j)-inp(i,j);           % x=D-C
        diff1=inp(i-1,j)-inp(i,j);         % x1=B-C
        diff2=inp(i+2,j)-inp(i+1,j);       % x2=E-D
        if ((abs(diff1)<5)&(abs(diff2)<5))    % Strong filter (if
|x1|,|x2|<5)
            inp(i-2,j)=inp(i-2,j)+(diff/8); % a=A+x/8
            inp(i-1,j)=inp(i-1,j)+(diff/4); % b=B+x/4
            inp(i,j)=inp(i,j)+(diff/2);     % c=C+x/2
            inp(i+1,j)=inp(i+1,j)-(diff/2); % d=D-x/2
            inp(i+2,j)=inp(i+2,j)-(diff/4); % e=E-x/4
            inp(i+3,j)=inp(i+3,j)-(diff/8); % f=F-x/8
            p=p+1;                           % Aumento cont strong filter
        else                                  % Soft filter

```

```

    inp(i-1,j)=inp(i-1,j)+(diff/8); % b=B+x/8
    inp(i,j)=inp(i,j)+(diff/2); % c=C+x/2
    inp(i+1,j)=inp(i+1,j)-(diff/2); % d=D-x/2
    inp(i+2,j)=inp(i+2,j)-(diff/8); % e=E-x/8
end;
end;
end;

inp;

% _____

[X,Y] = meshgrid(1:B:width-B+1,1:B:height-B+1); %Set motion vector
coordinates

figure, imshow(inp,[0 255]); title (strcat(num2str(loop),'Predicted
Frame'))
hold on
quiver(X,Y,mvx,mvy)
hold off

SN = strcat(num2str(loop),name);
imwrite(inp,SN,'bmp') %write predicted image to file
%imwrite(imp,strcat(num2str(loop),'Ynews.Y')) %write predicted image to
file

tot_SP = sum(sum(SP))
Set_P(numa) = tot_SP

exe=exe+comp;
calc(num)=comp;
% _____
_____%
end %end loop frames

totexe=exe

```

APPENDIX F

Adaptive Motion Estimation With Early Termination

(NEWADAPTIVE_ET.m)

```
%%%Adaptive motion estimation (search range) with adaptive deblocking
```

```
%%%filter
```

```
function [Set_P, calc, num] = NEWADAPTIVE_ET(name,f1,f2)
```

```
A=ones(9,11);
```

```
B=ones(9,11);
```

```
exe=0;
```

```
TH_SAD=0;
```

```
TH=0;
```

```
name = 'carphoneY.Y';
```

```
mvx=A*111;
```

```
mvy=B*111;
```

```
SAD3=A*10000;
```

```
f1=2;
```

```
f2=4;
```

```
for loop = f1:f2 %start loop frames
```

```
    num=loop+2-f1;
```

```
    comp =0;
```

```
    cmp=0;
```

```
    i1 = imread (strcat(num2str(loop-1),name));
```

```
    im2 = imread (strcat(num2str(loop), name));
```

```
    B = 16;%block size
```

```
    sz = B*B;
```

```
    R = 7; %search window
```

```
    [height,width] = size(i1); %width = row, height = column
```

```
    im1 = padarray(i1,[(R) (R)],0,'both');%pad both x and y sides with '0'
```

```
    no=0;
```

```
    TH_SAD
```

```

n1=0;
for i=1:B:height-B+1
    r=i-(n1*B)+n1;

    n2=0;

for j=1:B:width-B+1 %for every block in the reference and predicted
image frame (dinstinct)
    c=j-(n2*B)+n2;

    vx = mvx(r,c);
    vy = mvy(r,c);
    s_point =0;

    if vx==0
        R1=4;
        R2=4;

    elseif abs(vx)>0 & abs(vx)<=4

        if vx>0
            R1=4;
            R2=7;
        else
            R1=7;
            R2=4;
        end

    elseif abs(vx)>4 & abs(vx)<100

        if vx>0
            R1=4;
            R2=16;
        else
            R1=16;
            R2=4;
        end

    else

        R1=7;

```

```

    R2=7;

end

if vy==0
    R3=4;
    R4=4;
elseif abs(vy)>0 & abs(vy)<=4

    if vy>0
        R3=4;
        R4=7;
    else
        R3=7;
        R4=4;
    end

elseif abs(vy)>4 & abs(vy)<100

    if vy>0
        R3=4;
        R4=16;
    else
        R3=16;
        R4=4;
    end

else
    R3=7;
    R4=7;
end

if R1>R2
    RA=-R1;
    RB=R2;
    d1=-1;
    d2=1;
else
    RA=R2;
    RB=-R1;
    d1=1;
    d2=-1;
end

```

```

end

if R3>R4
    RC=-R3;
    RD=R4;
    d3=-1;
    d4=1;
else
    RC=R4;
    RD=-R3;
    d3=1;
    d4=-1;
end

%j,k= column, i,l = row
SADmin = 256*B*B;

for k =[0:d3:RC]
    for l =[0:d1:RA]
        if (k==0 & l==0)
            SADabsdiff = imabsdiff(im2(i:i+B-1,j:j+B-1),im1(i+k+R3+(R-
R3):i+k+R4+(R-R4)+B-1,j+l+R1+(R-R1):j+l+B+R2+(R-R2)-1));% im1 will
slide its block by one pixel at a time
            SAD = sum(sum(SADabsdiff));%sum all SAD avg for one row
            SADO = SAD;

            if (SADO<TH_SAD)
                dx=0;
                dy=0;
                s_point = s_point+1;
                break
            end
        else
            SADabsdiff = imabsdiff(im2(i:i+B-1,j:j+B-1),im1(i+k+R3+(R-
R3):i+k+R4+(R-R4)+B-1,j+l+R1+(R-R1):j+l+B+R2+(R-R2)-1));% im1 will
slide its block by one pixel at a time
            SAD = sum(sum(SADabsdiff));%sum all SAD avg for one row
        end

        if SAD < SADmin
            SADmin = SAD;
        end
    end
end

```

```

        dx=1;
        dy=k;
    end

    s_point = s_point+1;

end

if (k==0 & l==0)
    if (SAD0<TH_SAD)
        break
    end
end
end

end

for k =[0:d3:RC]
    for l =[d2:d2:RB]
        SADabsdiff = imabsdiff(im2(i:i+B-1,j:j+B-1),im1(i+k+R3+(R-
R3):i+k+R4+(R-R4)+B-1,j+l+R1+(R-R1):j+l+B+R2+(R-R2)-1));% im1 will
slide its block by one pixel at a time
        SAD = sum(sum(SADabsdiff));%sum all SAD avg for one row

        if SAD < SADmin
            SADmin = SAD;
            dx=1;
            dy=k;
        end
        s_point = s_point+1;
    end
end

end

for k =[d4:d4:RD]
    for l =[0:d1:RA]
        SADabsdiff = imabsdiff(im2(i:i+B-1,j:j+B-1),im1(i+k+R3+(R-
R3):i+k+R4+(R-R4)+B-1,j+l+R1+(R-R1):j+l+B+R2+(R-R2)-1));% im1 will
slide its block by one pixel at a time
        SAD = sum(sum(SADabsdiff));%sum all SAD avg for one row

        if SAD < SADmin
            SADmin = SAD;
            dx=1;
            dy=k;
        end
    end
end

```

```

        end
        s_point = s_point+1;
    end
end

for k =[d4:d4:RD]
    for l =[d2:d2:RB]
        SADabsdiff = imabsdiff(im2(i:i+B-1,j:j+B-1),im1(i+k+R3+(R-
R3):i+k+R4+(R-R4)+B-1,j+1+R1+(R-R1):j+1+B+R2+(R-R2)-1));%   im1   will
slide its block by one pixel at a time
        SAD = sum(sum(SADabsdiff));%sum all SAD avg for one row

        if SAD < SADmin
            SADmin = SAD;
            dx=1;
            dy=k;
        end
        s_point = s_point+1;
    end
end

cmp=(R1+R2+1)*(R3+R4+1)*B^2;

% _____writing predicted image_____ %

imp(i:i+B-1,j:j+B-1)= im1(i+dy+R:i+dy+R+B-1,dx+R+j:j+dx+R+B-1);
    iblk = floor ((i-1)/B+1);
    jblk = floor ((j-1)/B+1);
    mvx(iblk,jblk) = dx;
    mvy(iblk,jblk) = dy; %record the estimated MV

    SAD99(iblk,jblk) = SAD;
    r1(iblk,jblk) = R1;
    r2(iblk,jblk) = R2;
    r3(iblk,jblk) = R3;
    r4(iblk,jblk) = R4;
    SP(iblk,jblk) = s_point;

    SAD1(iblk,jblk)= SADmin;

```



```

n2=n2+1;

if(num == 2)
    if (dx==0 & dy==0)
        no=no+1;
        SADst(no)=SADmin;
        SAD3(iblk,jblk)= SADmin;
    end
end

comp=comp+cmp;
end

n1=n1+1;

end
% _____ %
% _____ %
%deblocking filter

inp=imp;
dd=B;

[dimy,dimx]=size(inp);    % Calcolo dimensione immagine

p=0;    % Azzeramento contatore strong filter

% Passaggio su tutte le righe...
for j=dd:dd:dimx-dd
    for i=1:1:dimy
        diff=inp(i,j+1)-inp(i,j);    % x=D-C
        diff1=inp(i,j-1)-inp(i,j);    % x1=B-C
        diff2=inp(i,j+2)-inp(i,j+1);    % x2=E-D
        if ((abs(diff1)<5)&(abs(diff2)<5))    % Strong filter (if
|x1|,|x2|<5)
            inp(i,j-2)=inp(i,j-2)+(diff/8);    % a=A+x/8
            inp(i,j-1)=inp(i,j-1)+(diff/4);    % b=B+x/4
            inp(i,j)=inp(i,j)+(diff/2);    % c=C+x/2

```

```

    inp(i,j+1)=inp(i,j+1)-(diff/2); % d=D-x/2
    inp(i,j+2)=inp(i,j+2)-(diff/4); % e=E-x/4
    inp(i,j+3)=inp(i,j+3)-(diff/8); % f=F-x/8
    p=p+1; % Aumento cont strong filter
else % Soft filter
    inp(i,j-1)=inp(i,j-1)+(diff/8); % b=B+x/8
    inp(i,j)=inp(i,j)+(diff/2); % c=C+x/2
    inp(i,j+1)=inp(i,j+1)-(diff/2); % d=D-x/2
    inp(i,j+2)=inp(i,j+2)-(diff/8); % e=E-x/8

end;
end;
end;

%Passaggio su tutte le colonne...
for i=dd:dd:dimy-dd
    for j=1:1:dimx
        diff=inp(i+1,j)-inp(i,j); % x=D-C
        diff1=inp(i-1,j)-inp(i,j); % x1=B-C
        diff2=inp(i+2,j)-inp(i+1,j); % x2=E-D
        if ((abs(diff1)<5)&(abs(diff2)<5)) % Strong filter (if
|x1|,|x2|<5)
            inp(i-2,j)=inp(i-2,j)+(diff/8); % a=A+x/8
            inp(i-1,j)=inp(i-1,j)+(diff/4); % b=B+x/4
            inp(i,j)=inp(i,j)+(diff/2); % c=C+x/2
            inp(i+1,j)=inp(i+1,j)-(diff/2); % d=D-x/2
            inp(i+2,j)=inp(i+2,j)-(diff/4); % e=E-x/4
            inp(i+3,j)=inp(i+3,j)-(diff/8); % f=F-x/8
            p=p+1; % Aumento cont strong filter
        else % Soft filter
            inp(i-1,j)=inp(i-1,j)+(diff/8); % b=B+x/8
            inp(i,j)=inp(i,j)+(diff/2); % c=C+x/2
            inp(i+1,j)=inp(i+1,j)-(diff/2); % d=D-x/2
            inp(i+2,j)=inp(i+2,j)-(diff/8); % e=E-x/8

        end;
    end;
end;
end;

inp;

```

```

% _____

if (num==2)

    figure, hist(SADst)
    figure, [H,STATS]=CDFPLOT(SADst)

    if (no/99 >=0.8)
        TH_SAD=(0.128*(max(SADst)-min(SADst)))+min(SADst)
    end

end

[X,Y] = meshgrid(1:B:width-B+1,1:B:height-B+1); %Set motion vector
coordinates

figure, imshow(inp,[0 255]); title (strcat(num2str(loop),'Predicted
Frame'))
hold on
quiver(X,Y,mvx,mvy)
hold off

SN = strcat(num2str(loop),name);
imwrite(inp,SN,'bmp') %write predicted image to file
%imwrite(imp,strcat(num2str(loop),'Ynews.Y')) %write predicted image to
file

tot_SP = sum(sum(SP))
Set_P(num) = tot_SP

exe=exe+comp;
calc(num)=exe;
% _____
_____%
end %end loop frames

totexe=exe

```

APPENDIX G

Split .YUV And .QCIF Video (extract.m)

```
% FUNC_YUV_SPLIT Extracts Y,U,V frames from YUV files
%   modified by NOR FARHANA BT. FADLY CHEW (4 OCTOBER 2006)
%   FLAG = func_YUV_split(FILENAME, NFRAME) extracts the first NFRAME
number
%   of frames from the YUV file specified by FILENAME and separates the
%   Y,U and V frames into separate image files with extensions ".Y",
".U"
%   and ".V" respectively. The function returns FLAG = 1 if extraction
is
%   successful.
%
function A = extract(in_file_name, nFrame)
% Objective: extract YUV components from the CIF 4:2:0 video file, that
is,
% split CIF file into three files, i.e., .Y, .U, .V.
%
% Jing Tian
% Contact: scuteejtian@hotmail.com
% This program is written in 2005 during my postgraduate studying in
% NTU, Singapore.

% Modified by Nasreen Badruddin (eenb@yahoo.com) on 23 February 2006
% Changes are:
% 1) input arguments for input YUV file and number of frames to extract
% 2) saved each Y, U, V component of each frame into a separate file
%   (output filename uses this format:
%   <name of input file>_f<framenum>.<Y/U/V>
% 3) saved each Y,U,V frame as uncompressed bitmap.
% 4) included comments as HELP

% Old code is marked with "%%%"

%%in_file_name = 'test.cif'
```

```

%%nFrame = 10;

[fid1 message]= fopen(in_file_name,'rb');

%Initialize FLAG
FLAG = 0;

if length(strfind(in_file_name, '.yuv')) == 0
    nRow = 288/2;
    nColumn = 352/2;
    %% in_file_name2 = strrep(in_file_name, '.cif', '.Y');
    %% [fid2 message]= fopen(in_file_name2,'w');
    %% in_file_name3 = strrep(in_file_name, '.cif', '.U');
    %% [fid3 message]= fopen(in_file_name3,'w');
    %% in_file_name4 = strrep(in_file_name, '.cif', '.V');
    %% [fid4 message]= fopen(in_file_name4,'w');
else
    nRow = 288/2;
    nColumn = 352/2;
    %% in_file_name2 = strrep(in_file_name, '.qcif', '.Y')
    %%[fid2 message]= fopen(in_file_name2,'w');
    %% in_file_name3 = strrep(in_file_name, '.qcif', '.U');
    %%[fid3 message]= fopen(in_file_name3,'w');
    %% in_file_name4 = strrep(in_file_name, '.qcif', '.V');
    %%[fid4 message]= fopen(in_file_name4,'w');
end

for i = 1: nFrame

%-----
% BEGINNING OF PART 1 OF NEW CODES BY NASREEN B (23/02/06)
%-----

    nFstr=int2str(i);
    if length(strfind(in_file_name, '.yuv')) == 0
        out_file_name_Y = strrep(in_file_name, '.qcif', 'Y');
        out_file_name_Y = strcat(nFstr,out_file_name_Y, '.Y');
        out_file_name_U = strrep(in_file_name, '.qcif', 'U');
        out_file_name_U = strcat(nFstr,out_file_name_U, '.U');
        out_file_name_V = strrep(in_file_name, '.qcif', 'V');
        out_file_name_V = strcat(nFstr,out_file_name_V, '.V');
    else

```

```

    out_file_name_Y = strrep(in_file_name, '.yuv', 'Y');
    out_file_name_Y = strcat(nFstr, out_file_name_Y, '.Y');
    out_file_name_U = strrep(in_file_name, '.yuv', 'U');
    out_file_name_U = strcat(nFstr, out_file_name_U, '.U');
    out_file_name_V = strrep(in_file_name, '.yuv', 'V');
    out_file_name_V = strcat(nFstr, out_file_name_V, '.V');
end

%-----
%           END OF PART 1 OF NEW CODES BY NASREEN B (23/02/06)
%-----

%reading Y component
    img_y = fread(fid1, nRow * nColumn, 'uchar');
    img_y = reshape(img_y, nColumn, nRow);
    img_y = img_y';

%reading U component
    img_u = fread(fid1, nRow * nColumn / 4, 'uchar');
    img_u = reshape(img_u, nColumn/2, nRow/2);
    img_u = img_u';

%reading V component
    img_v = fread(fid1, nRow * nColumn / 4, 'uchar');
    img_v = reshape(img_v, nColumn/2, nRow/2);
    img_v = img_v';

%writing file
    %%count = fwrite(fid2, img_y, 'uint8');
    %%count = fwrite(fid3, img_u, 'uint8');
    %%count = fwrite(fid4, img_v, 'uint8');

%-----
% BEGINNING OF PART 2 OF NEW CODES BY NASREEN B (23/02/06)
%-----

    imwrite(uint8(img_y), out_file_name_Y, 'bmp')
    % imwrite(uint8(img_u), out_file_name_U, 'bmp')
    % imwrite(uint8(img_v), out_file_name_V, 'bmp')

%-----
%           END OF PART 2 OF NEW CODES BY NASREEN B (23/02/06)
%-----

end

```

```
%%fclose(fid1);  
%%fclose(fid2);  
%%fclose(fid3);  
%%fclose(fid4);
```

```
FLAG = 1;
```