A Fast Motion Estimation Algorithm for Mobile Devices

by

HO-JAE LEE

B.Eng., The University of Seoul, 1997

M.Eng., The University of Seoul, 2000

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

THE FACULTY OF GRADUATE STUDIES

(Electrical and Computer Engineering)

THE UNIVERSITY OF BRITISH COLUMBIA

April 2005

Abstract

With increasing availability of camera-equipped cellular telephones, the demands for video transmissions over mobile networks are growing significantly. Third generation mobile networks such as 1xEV-DO support data rates high enough for transmissions of compressed video, but reducing the power consumption of video compression in mobile devices is an ongoing challenge. The large number of computations required for motion estimation is the main cause for this consumption. In this research, we propose novel spatial and spatial-temporal based motion estimation methods, which sort the sums of absolute difference (SAD) according to specific threshold values to facilitate efficient searches for the local optimal motion vectors. The proposed methods drastically reduce the computation costs of the motion estimation process over the existing techniques.

The proposed algorithm using only spatial correlation performs direct subtraction between the current and previous video frames to obtain a difference frame, which is then partitioned into macroblocks of 16x16 (256) pixels with each pixel having 256 grey levels. For each macroblock we compute the SAD by simply adding all grey levels of the 256 pixels. Next, the average and standard deviation of all macroblocks' SADs are calculated and all SADs are sorted in descending order. Based on thresholds calculated from the mean and standard deviation, the sorted SAD values are divided into three groups corresponding to different degrees of motion, and different computationally efficient scanning processes are applied to the respective macroblocks in each group to search for the motion vectors. To further reduce computations, only a subset of each macroblock is scanned. New types of subsets are proposed for this purpose.

The proposed algorithm that uses both spatial and temporal correlations is similar to the above algorithm, except that the temporal correlations are taken into consideration in computing the difference frame, since each macroblock is highly correlated to the corresponding macroblock in the previous frame as well as the surrounding macroblocks in the same frame.

Performance evaluations showed that, for the same picture quality, the proposed algorithms use only 0.5% of the computations of the full search method and are almost 19 times faster than 4SS, the best presently available conventional method. When compared to the most recently published non-conventional motion estimation method, which outperforms all the existing methods, our approach improves the computational speed by 7.5 times.

Table of Contents

List of Tables

List of Figures

Acknowledgements

Most of all, I would like to thank Dr. Nasiopoulos and Dr. Leung for their direction and encouragement in the preparation of this thesis. I also want to thank Karl Olav Lillevold and Robert Danielsen who participated in developing TMN encoder I mainly used in this thesis and released their work to the public. Without their effort and master piece, this work would not have been possible.

The efforts of my thesis examination committee members Dr. Rabab Ward and Dr. Hussein Alnuweiri are gratefully acknowledged.

I want to thank the past member of the Digital Multimedia Lab, Kemal Ugur who gave me an intuition and good references regarding H.264 and H.263. I also would like to thank the present members Qiang Tang and Matthias von dem Knesebeck for their help.

I thank all my friends, Jae-Seop, Byoung-Ryoul, Byoung-Oh, Ki-Hoon, Yo-An, Reza and more who always stand by and support me whenever and wherever I need. I would like to thank my parents and only sister for their continuous support during the past 30 years.

Finally, a special word of thanks must go to my wife Young-Hye and my daughter Soo-Hyun (Florin), without whose enduring support it would have been not possible for me to achieve my goal.

This work was supported by grants from TELUS Mobility and the Advanced Systems Institute of BC, and by the Canadian Natural Sciences and Engineering Research Council under grant CRD 247855-01.

1 Introduction

In the past few years, significant changes in video compression and wireless data communications have stirred an evolution in portable multimedia applications. However, the presence of video in mobile devices has become one of the most difficult challenges for real time high quality applications.

Mobile communication standards such as 3GPP (3rd Generation Partnership Project), are specifically designed to support data communication between mobile and wired-networks or mobile networks only. Despite the existence of standards that support high speed data flow and real time video rates, the computational burden caused by the video compression process remains a technological challenge. Conventional video encoders spend up to 60% of the computational time on motion estimation (ME) [1]. Power consumption of video enabled mobile devices can be reduced by lowering the power consumption of the motion compensation process. There are three main different types of motion estimation algorithms, the pixel level [2], block level [3], and the global level [4] motion estimation. Among those schemes, the block based motion estimation is widely used by MPEG encoders. In this category, Spatial and Temporal Correlation of motion vectors is exploited in order to lower the computational costs over the conventional Full Search (FS) technique which scans the entire searching area [2]. Other conventional, motion estimation algorithms include the Prediction Model Search using Three Step Search (TSS) [5], New Three Step Search (NTSS) [6][7], Four Step Search (4SS) [8][9], Block based Gradient Descent Search (BBGDS) [10]-[12], Diamond Search [13]-[17], Recursive ME(RME) [18][19], The Prediction Search algorithm (PSA) [20][21], 2D Logarithm Search (LOGS) [22], One-at-a-time search (OTS) [23], and Cross Search [24]. Some other methods try

to match the picture quality of FS and still improve the computational speed of estimation search, but the achieved improvement is much less than the above conventional methods [25]-[33].

In most recent studies, more advanced non conventional methods have been developed, including the Prediction Model Four-Step Search (PM4SS) [34], the Adaptive Predicted Direction Search algorithm (APDSA) [35], and the Hybrid Search approach [46]. However, even these approaches have not resulted in speeds that are practical for low power, real-time realization of mobile devices. Hence, there is an immense interest in developing fast and efficient motion compensation techniques which will enable this new market.

We have developed a novel motion estimation method which substantially improves the speed of the motion estimation process over all the existing techniques. Our proposed method makes use of spatial correlation or spatial-temporal correlation among neighbouring macroblocks, macroblock subsets, and a special search scanning pattern to reduce the computational time for finding the motion vectors.

In Chapter 2, we give an overview of existing fast motion estimation algorithms. In Chapter 3, we describe our new method and in Chapter 4 we present the implementation of our algorithm in hardware and software. Performance evaluation results are presented in Chapter 5. The conclusions are presented in Chapter 6.

2 Overview Of Current Motion Estimation Algorithms

2.1 Full Search Motion Estimation Algorithm

The objective of the motion compensation process is to determine the amount of motion on a block by block basis which minimizes the difference between consecutive frames. The Full Search (FS) algorithm, which is one of the most well known methods, is based on an exhaustive testing of all the candidate blocks within the search window, giving the minimum block distortion position that corresponds to the best matching block. This method, however, involves a large number of computations.

Figure 1 illustrates the motion estimation with searching window $\pm$ W. The macroblock in the current video frame is moved through every single point within the searching window in the previous video frame. When the searching range W is 7, the total number of searching points is 225.

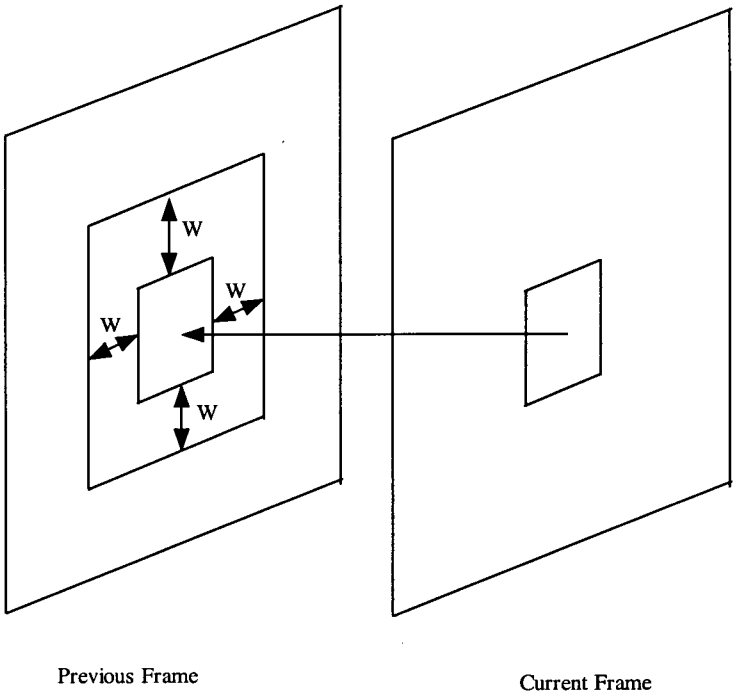


Figure 1 Motion Estimation with Search Window ± W pixels

2.2 Traditional Fast Motion Estimation Algorithms

2.2.1 Three Step Search

In the Three Step Search (TSS) method, eight points around a centre point are tested and the position of minimum distortion becomes the centre point for the next step. For example, figure 2 illustrates one of the searching paths. During the first step, one filled circle at the centre and the 8 filled circles around the centre are tested for minimum distortion. Then this becomes the centre point for the next search step. The new eight points around this centre are located at half the distance of the first step. This process is repeated one more time and the point with the minimum distortion is the final motion vector (MV) [5]. For a maximum displacement of ± 7,

the TSS always needs 25 block matches and this results in just 8% of the computational power needed for FS.

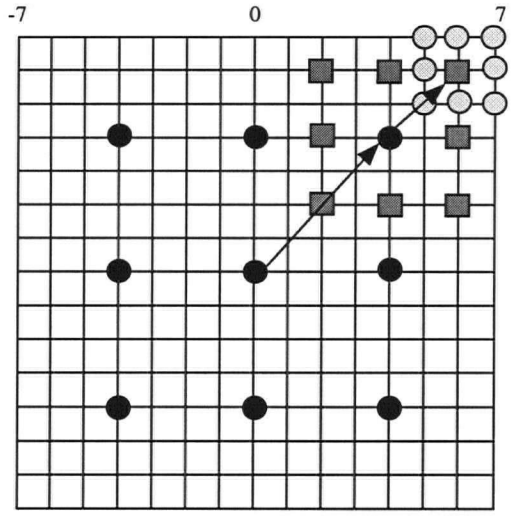


Figure 2 Three Step Searching Path; Filled circles are the points in the first step, Filled squares are the points in the second step, Circles are the points in the third step

2.2.2 New Three Step Search

The New Three Step Search (NTSS) method differs from TSS in assuming a centre biased checking point pattern in its first stage and incorporating a half-way stop technique for low motion (stationary or quasi-stationary) block. In the first stage, in addition to the original checking points in TSS, eight extra points are used, shown as shaded squares in Figure 3. In case the minimum distortion occurs at centre points in the first step, the searching procedure stops and the resulting MV is set to be (0, 0). This completes the search process and it is known as the first step stop. If one of the eight neighbouring points around the centre is the minimum, the searching step will be performed only for the points around the minimum as shown with light-shaded circles in Figure 3. The search stops after this point. This is called the second step stop. If

any of the outer 8 points is the minimum, then the search process is identical to the conventional TSS [6][7].

In the worse case, NTSS requires 33 block matches as compared to 25 matches needed in TSS. However, due to the fact that motion in low bit rate applications such as video conferencing and video phones is very slow and the background is almost stationary, the probability of the first step stop and the second step stop occurring is very high and this can help to speed up the performance of the motion compensation. However, in general, this approach has the same performance as the conventional TSS.

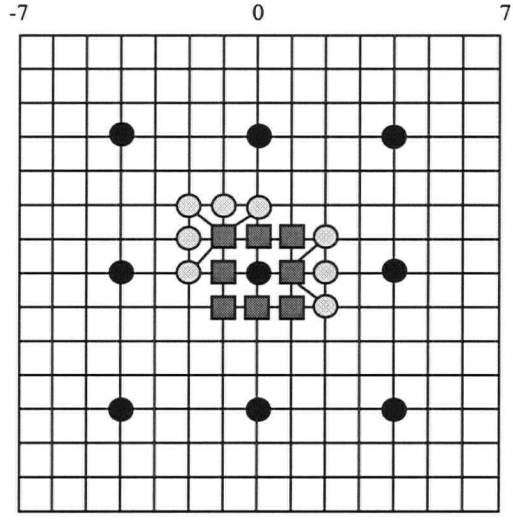


Figure 3 New Three Step Search Path; Filled circles and the filled squares are the points in the first step, Circles are the points in the second step

2.2.3 Four Step Search

The Four Step Search (4SS) also makes use of the centre biased motion distribution characteristic of the video sequences. For maximum motion distributions of ± 7, this algorithm

utilizes a centre biased search pattern with 9 checking points in 5 x 5 window in the first step instead of 9 x 9 window in the TSS or NTSS.

Step 1: A minimum distortion point is found from 9 checking points in 5 x 5 window. In case the minimum distortion point is located on the centre of the 5 x 5 window, go to the fourth step; otherwise go to second step.

Step 2: The 5 x 5 search window is maintained but the centre of the search window is moved to the minimum distortion point from the step 1. In case the minimum distortion point is located on the centre of the 5 x 5 window go to the fourth step; otherwise go to third step.

Step 3: The search pattern is same as Step 2. However, it must go to the fourth step at last.

Step 4: The search window is reduced to 3 x 3 and the direction of the motion vector is determined by the minimum distortion point of the nine checking points.

Figure 4 illustrates one of the search paths. In the worse case, computational requirement of the 4SS is 27 block matches as compared to 33 block matches in the NTSS [8]. Experimental results show that this algorithm requires 8% of computational needs when compared to FS.

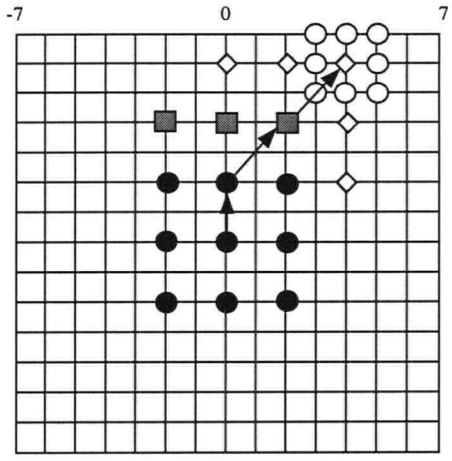


Figure 4 Searching Path for Four Step Search; Filled circles are in the first step, Filled squares are in the second step, Diamonds are in the third step, White circles are in the fourth step

2.2.4 Diamond Search

The Diamond Search (DS) algorithm employs two search patterns as shown in Figure 5. The Large Diamond Shaped Pattern (LDSP) comprises of 9 checking points to compose the diamond shape. The second pattern, called Small Diamond Shaped Pattern (SDSP), comprises of 5 checking points. In the first step, LDSP is repeatedly used until the minimum block distortion occurs at the centre of the pattern. Figure 6 illustrates the overlapping of the checking points. When the minimum distortion is found in the any corners or edges of the diamond pattern, another three or five new checking points are required as shown in Figure 6 (a) and (b), respectively. When the search is switched to SDSP, the minimum distortion is also the final motion vector [14]. Figure 7 illustrates one of the example searching paths and the final MV is set to be (1,-3).

Experimental results show that for the same picture quality when compared to NTSS the DS algorithm reduces computational complexity by approximately 25-30%, which is equivalent to 6% of the computational needs of FS.

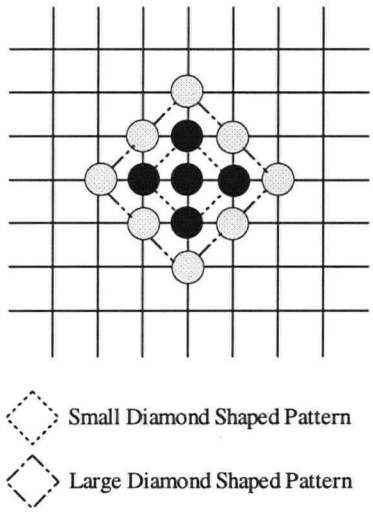


Figure 5 Search pattern for the Diamond Search

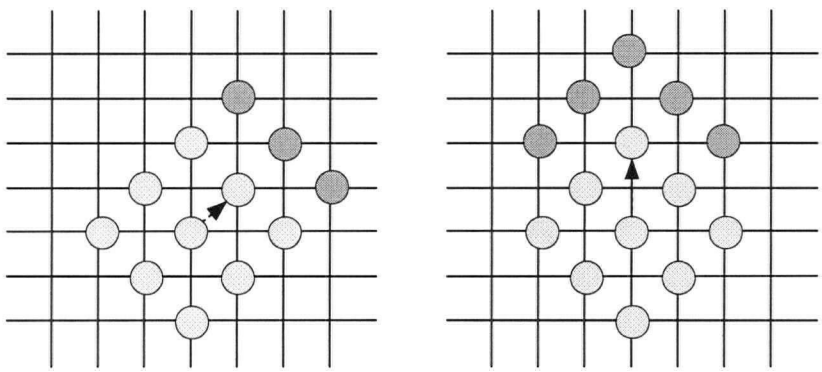


Figure 6 Checking point overlapping when the minimum block distortion is found at (a) one of the corner, (b) one of the edge

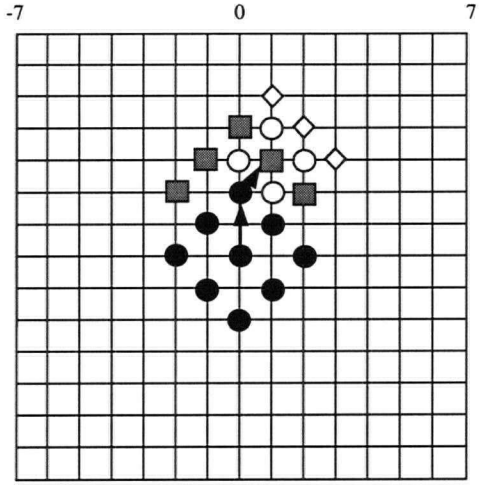


Figure 7 Example of a Search Path for the Diamond Search

2.3 Advanced Fast Motion Estimation Algorithms

2.3.1 Fast Full Search Motion Estimation Algorithm

For the purpose of reducing computational time and preventing miscalculation of motion vectors, fast full search (FFS) motion estimation algorithms are exploited. FFS algorithms eliminate candidate-checking points by using a boundary equation. The main idea of these algorithms is based on the successive elimination algorithm (SEA). SEA uses successive elimination to exclude a large number of possible search points.

Although SEA excludes many search positions, it still finds the best matching block with respect to the used matching criterion. In order to reduce the number of search positions, an inequality is evaluated for every search position, using block properties that can be computed efficiently prior to the actual motion estimation, e.g., the sum of norms [36].

The block-matching error is proportional to the complexity of the reference block with Taylor series expansion [37]. By using a modified form of the series expansion, we can express the matching distortion $d_{t+1}(p)$ in terms of the gradient magnitude of the reference block, as

$$\begin{aligned} d_{t+1}(p) &= \left| f_{t+1}(p) - f_t(p + cmv) \right| \\ &\approx \left| f_t(p + mv) - f_t(p + cmv) \right| \\ &\approx \left| \frac{\partial f_t(p + mv)}{\partial x}(cmvx - mvx) + \frac{\partial f_t(p + mv)}{\partial y}(cmvy - mvy) \right| \\ &\approx \left| \frac{\partial f_{t+1}(p + mv)}{\partial x}(cmvx - mvx) + \frac{\partial f_{t+1}(p + mv)}{\partial y}(cmvy - mvy) \right| \end{aligned} \tag{2.1}$$

where $mv = (mvx, mvy)$ is the motion vector of p position and $cmv = (cmvx, cmvy)$ stands for candidate motion vector corresponding to the matching distortion.

Equation (2.1) indicates that the matching distortion at pixel p is proportional to the gradient magnitude of the reference block in the current frame, which corresponds to the complexity of the image data. This algorithm also adopts the concept of adaptive matching scan which has various scan directions including a top-to-bottom one.

Other types of fast FS use decomposed mean square error (MSE) [38], sum of squared vertical projection [39], and partial distortion elimination [40]. The main advantage of the FSS algorithms is that they achieve the same picture quality as FS. However, the speed improvement obtained by FSS algorithms is only 25% of that of FS, making them significantly slower than other types of fast motion estimation approaches, and not fast enough for real-time implementation for mobile devices.

2.3.2 Rood based Motion Estimation Algorithm

Figure 8 shows the search pattern of this algorithm. The main idea behind the rood search algorithm is that about 80% of the motion vectors are located within a central 5x5 square of the whole macroblock position. It has been shown that over 76% of the motion vectors are distributed within the centre biased rood which is used as the first step by this approach (Figure 8) [41].

Rood based motion estimation differs from diamond search in that a rood-centre-biased search is performed in the first step, and halfway-stop strategy is employed for quasi-stationary or stationary candidate blocks. This process is summarized as follows:

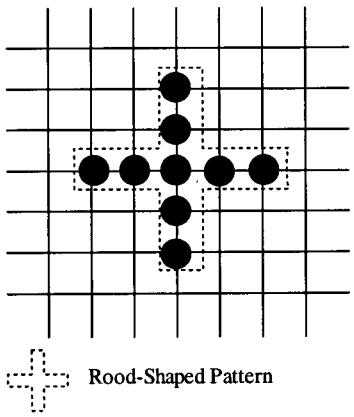


Figure 8 Search pattern for Rood shaped approach

Step 1: The Minimum Block Distortion Measure (BDM) is found from the 9 search points shown in Figure 8. If this minimum occurs at the centre of the rood shaped pattern (RSP), then the search is stopped, otherwise, we continue with step 2.

Step 2: In this case, the DS process is employed. Assume the minimum occurs at the upper point of the rood pattern (see arrow in Figure 9). Then the large diamond search pattern (LDSP) is used to determine the minimum. Since it is expected that the minimum is located at the upper part of the rood shape, only two new points are searched of the possible 9 (new points are shown as shaded squares in Figure 9).

Step 3: A new LDSP is formed by repositioning the minimum BDM found in the previous step as the centre of the LDSP (right hand side shaded square in Figure 9). If the new minimum BDM point is still at the centre of the newly formed LDSP, then go to Step 4, otherwise, repeat this step.

Step 4: With the minimum BDM point in the previous step as the center, a small diamond search pattern (SDSP) is employed at this time. The minimum BDM point resulting from the new 4 candidate points is the final motion vector.

In addition to the above described methods, there are several other modified rood based motion estimation algorithms, such as the Novel small-cross-diamond search algorithm [42], the Adaptive Rood Pattern Search [43] and Improved Adaptive Rood Pattern Search [44]. All of these methods are based on the same concept and yield comparable results to the original rood search algorithm.

Experimental results show that the Rood based Motion Estimation Algorithm improves the search speed by a factor of two when compared with DS. This is equivalent to 4% of the computational complexity of FS. However, the assumption that all motion vectors are centre biased results in some degradation of the image quality compared to that of FS.

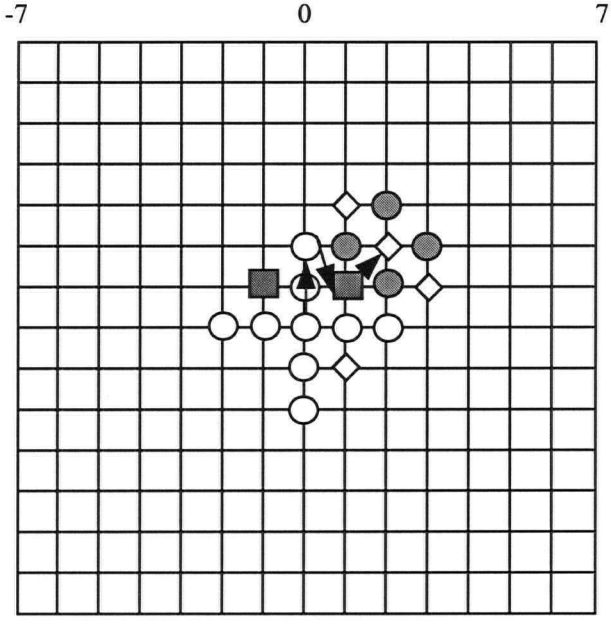


Figure 9 Example of the Search Path for the Rood Pattern Search

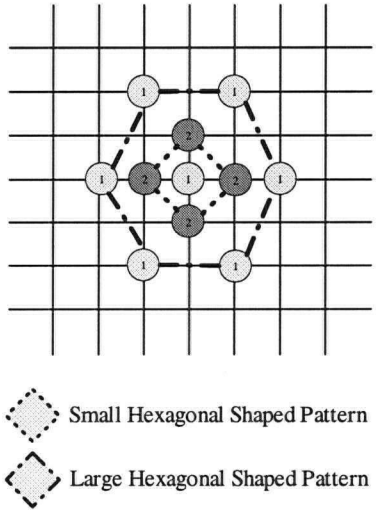


Figure 10 Search pattern for Hexagonal shaped Pattern

Another diamond based search method is the Hexagon based Motion Estimation Algorithm (HEXBS) [45]. This method is similar to DS with the only difference that the large diamond is replaced by a Hexagon (see Figure 10). Consequently, a more circular shaped search pattern is expected, in which minimum numbers of search points are uniformly distributed. The results of this algorithm show that the average speed improvement rate of HEXBS is not as good as that of the diamond search (DS), but the average of the mean of absolute distortion (MAD) is better than that of DS, resulting in better picture quality.

2.3.3 Hybrid Motion Estimation Algorithm

To reduce the number of search points and speed up motion estimation, some algorithms are using several different search patterns to compute motion vectors within one frame. A fast center-biased hybrid search algorithm (FCBHS) uses three different search patterns, the plus (or rood) shaped, X shaped and diamond shaped search patterns [46]. This algorithm can be summarized as follows:

Step 1: The first pattern, which is called plus shaped search pattern, uses five checking points to determine the motion vector at the centre of the search window (See Figure 11 (a)).

Step 2: If the minimum distortion is found on one of the vertexes of the plus pattern, X-Shaped search is performed (See Figure 11 (b)). This search pattern determines whether the motion vector is within (±3, ±3) window around the centre area. If the minimum distortion is still the same as that of the first step, three new points (See circle 22 in Figure 11 (c)) around the minimum distortion position are calculated and the search is stopped.

Step 3: If the minimum distortion occurs in one of the outer points (see circle 21 in Figure 11 (b)), then the third search pattern is employed, which is diamond shaped search

(Figure 11 (d)). The rest of the process follows the conventional DS approach (see Figure 11 (e) and (f)).

Figure 12 illustrates examples of search paths using FCBHS.

This algorithm results in significant speed improvements, requiring only 2.5% of the computational time needed for FS.

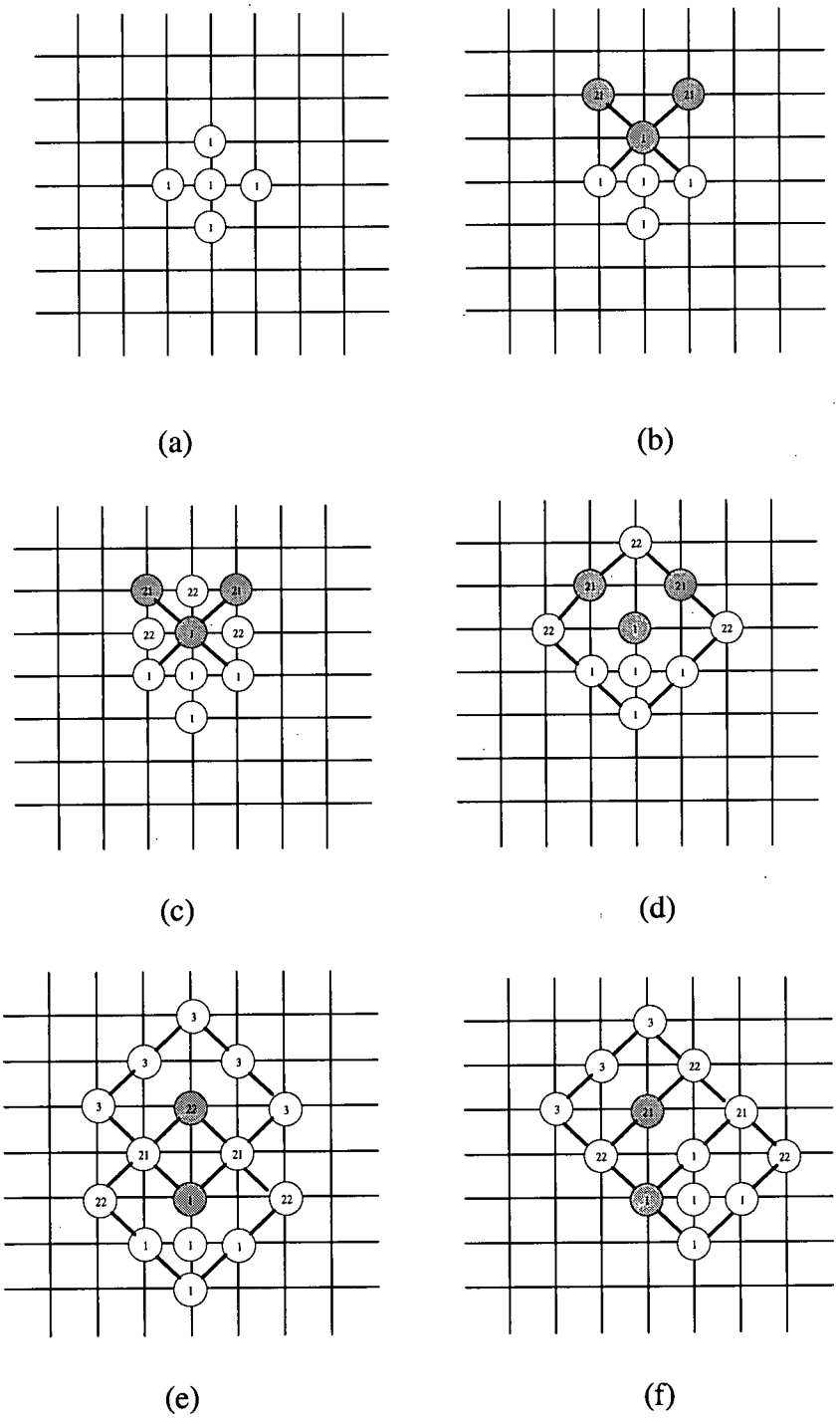


Figure 11 FCBHS search patterns; (a) Plus shaped search pattern; (b) Next step along the vertex of the plus pattern(X shaped search); (c) X's final step; (d) Next step along the face of diamond (X's next step); (e) Next step along the vertex of diamond; (f) Next step of along the face of diamond

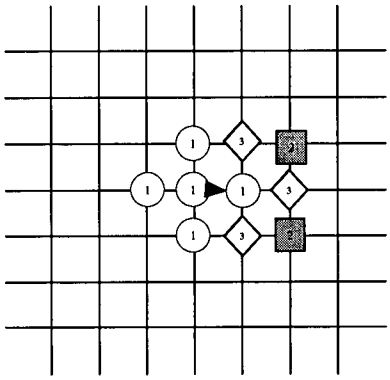


Figure 12 Example of search path using FCBHS, Each number depicts the each search step

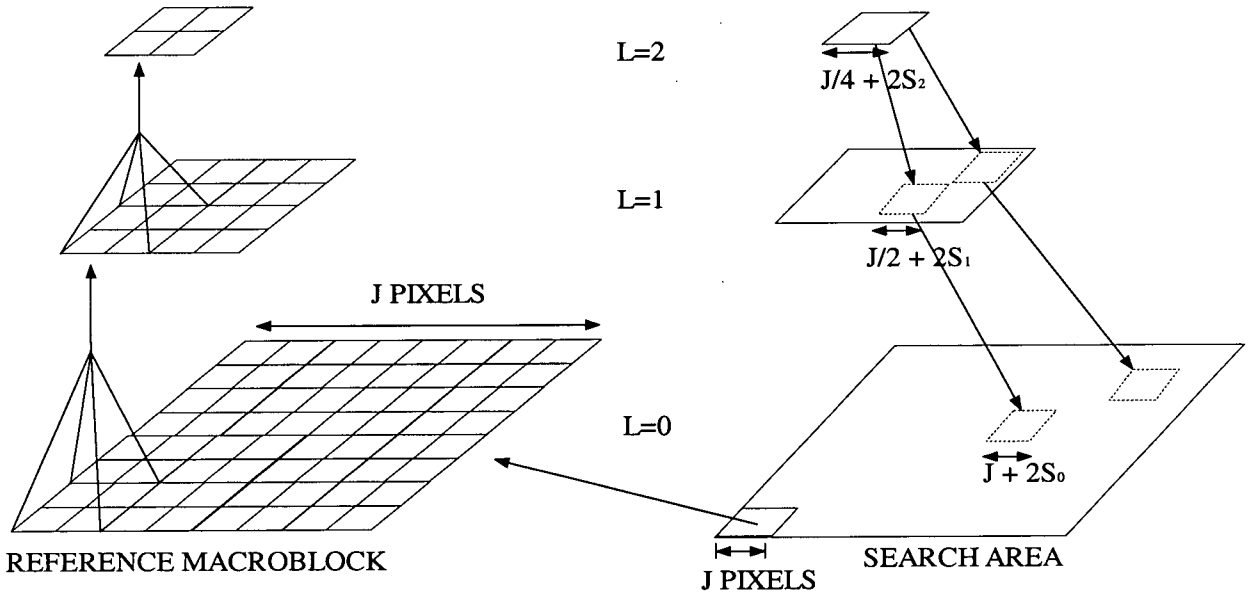


Figure 13 Mean Pyramid generation and multi-resolution motion estimation

2.3.4 Adaptive Motion Estimation Algorithms

Adaptive motion estimation algorithms compute motion vectors by changing search window sizes adaptively, using several search patterns within a frame and finding motion correlation in both spatial and temporal domain.

Content adaptive motion estimation [47] uses a new algorithm, which is called the multi resolution mean pyramid. This method scales computations by adaptively deciding on the number of candidate motion vectors (CMVs) to be passed from each level of the pyramid to the next step. In addition, the number of CMVs is dependent on context complexity. Figure 13 depicts the process of generating the multi resolution mean pyramid and the corresponding motion estimation process. Each pixel on the number of levels is calculated as follows:

$$x_L^k(p,q) = \frac{1}{4}\sum_{u=-1}^{0}\sum_{v=-1}^{0} x_{L-1}^k(2p+u,2q+v) \tag{2.2}$$

$$1 \le L \le 2, 1 \le p \le \frac{N_H}{2^L}, 1 \le q \le \frac{N_V}{2^L}$$

where L denotes the pyramid level, k denotes the frame number, p and q are pixel positions and N_H and N_V denote the horizontal and vertical frame size respectively.

As we move from lower level (L=0) to top level (L=2), the computation involved in determining MV for one MB is decreased by a factor of 4 because the size of each frame is reduced by a factor of 2 in each direction. This also makes the size of MBs in each frame and the search range reduced by a factor of 4. In this algorithm, actual search procedure is performed from top level to bottom level and while passing each level, MVs are refined.

In Motion adaptive search [48], the search window size is changed adaptively. The size of the search window for each macroblock is based on both the motion activity at the macroblock position, which is called local motion activity (LMA), and the overall frame level motion activity, which is called global motion activity (GMA). Two search patterns are used for this algorithm; a diamond search pattern (DSP) and an elastic search pattern (ESP) which is shown in Figure 14. If the motion activity of the present macroblock is smaller than the motion vectors of the neighbouring macroblocks, the motion search is performed around the centre of the search window using DSP. Otherwise, the search is done using ESP. The resulting minimum is chosen as the centre point, around which the search then is performed using DSP.

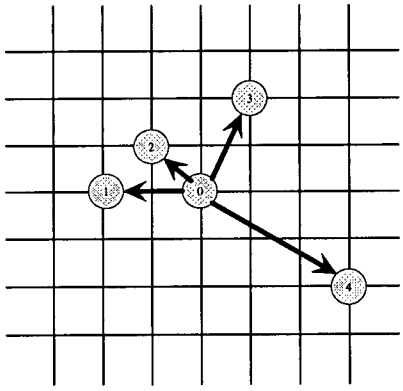


Figure 14 Search pattern for the Elastic search approach

In the Complexity-adaptive search algorithm [49], frame level complexity allocation and block level complexity allocation are used to dynamically adjust to maximize picture quality for a target computational complexity. This process takes advantage of slow and fast motion altering frames. The objective of the frame level complexity control is to determine the target complexity

for each frame in order to keep image quality high and constant. The objective of the block level complexity control is to maintain maximum quality under a target computational complexity.

Diversity-based fast block motion estimation [50] proposed diversity in search strategy (DSS). For instance, it uses 4 decimating patterns as shown in Figure 15 and two search patterns, which are DS and TSS. Pattern A is the decimating pattern consisting of all *a* pixels. Pattern B, C and D are the decimating patterns consisting of *b*, *c* and *d* pixels, respectively. In other words, each pattern consists of quarter size of original macroblock and is reconstructed by choosing each corresponding pixels. DS is used with pattern A and D, and TSS is used with patterns B and C.

When compared to FS and for ± 15 search window, these algorithms need up to 1.4% of computational complexity.

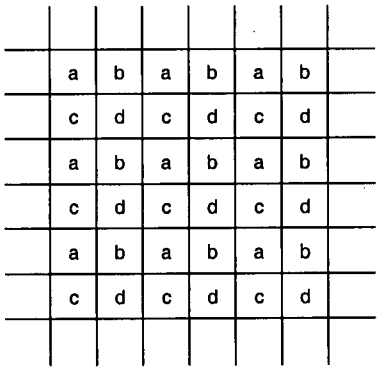


Figure 15 Decimation patterns for Diversity-based fast block motion estimation

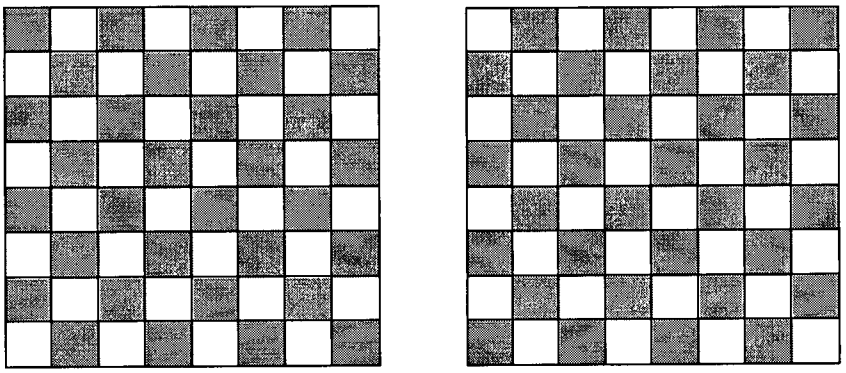
2.3.5 Spatial-Temporal Motion Estimation Algorithm

The temporal correlation of video sequence is usually very high and the absolute difference between two motion vectors which correspond to the same position within ±1 pixels

in two consecutive frames is about 94% of the total in case of 'foreman' sequence [25] . It has also been shown that the MV of a MB is also highly correlated to the MVs of neighbouring MBs, due to spatial correlation [26]. Some motion estimation algorithms use only one of the two correlations and others use both correlations in order to reduce computational time.

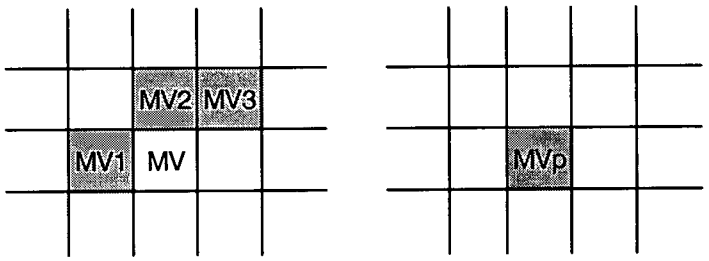
The low power motion estimation algorithm [25] uses two modes to find the MV of the current frame, i.e, the fast MB mode and the normal MV mode. In the fast MB mode, the search operation is performed within the range of [-1, +1] using the MV of the previous frame as the initial search point. Figure 16 shows even sequence frame and odd sequence frame. In this algorithm, shaded MBs and transparent MBs in each frame are not calculated at the same time, but rather done in alternative fashion. For example, for even sequence frame or odd sequence frame, only the shaded MBs are used to find MVs. Therefore, only the half of MBs is checked.

In [26], four previously calculated MVs are used to help determine the value of the present MB as shown in Figure 18. This algorithm divides the search range into three categories according to the flow of adjacent motion vector and the MV of the corresponding MB in the previous frame. The first category is that all MVs are equal, the second is that only spatial MVs are equal, and the last is that some of the MVs are equal or close, or all MVs have different directions. According to the criterion, the search area is decided and at the final step, MV is found when there is no correlation.



(a) (b)

Figure 16 Fast MB mode candidate area (shaded); (a) Even frame; (b) Odd frame



(a) (b)

Figure 17 Comparison block pattern; (a) Current frame; (b) Previous frame

Adaptive motion tracking [27] locates an initial search point by exploiting the correlation of the motion field in both the temporal and spatial domains. An inertial based motion tracking (IBMT) method is used to provide a temporal prediction motion vector while a refined spatial motion prediction (RSMP) is used to find a predictive motion vector in the spatial domain.

Motion estimation for low power video devices [28] and Algorithmic and architectural co-design [29] present the general rules to guide the design of efficient motion estimation

algorithm. Flexible frame-reordering and multi-temporal motion estimation for scalable MPEG encoding in mobile consumer equipment [30] and New flexible motion estimation technique for scalable MPEG encoding using display frame order and multi-temporal references [31] report the three stage display order multi temporal motion estimation algorithm which is based on recursive motion estimation (RME). Low-complexity motion estimation for VLBR video coders [32] proposes new context based BME technique for prediction module and it applies spatial-temporal prediction rules to generate prediction list which contains a set of candidate vectors for the estimation of the motion field.

For a search area of ± 15 pixels, the number of searching points of these algorithms is just 1.2% for FS.

2.3.6 Others

Simplex minimization [51] proposes simplex minimization optimization method to solve block matching motion estimation algorithm. This search method explores directions other than that of the minimum distortion. Efficient multilevel successive elimination algorithms (EMSEA) [52] uses new algorithm based on SEA which is used for FFS. EMSEA is the improved version of multilevel successive elimination algorithm (MSEA) and advanced multilevel successive elimination algorithm (AMSEA).

A Novel Block Motion Estimation Algorithm [53] and Adjustable partial distortion search algorithm [54] use adjustable partial distortion search (APDS) algorithm and this can help to control between the quality of image and the speed of performance. A new lower bound [55] proposes 8 bit partial sums. This idea is from the luminance value of a pixel in frame is of 8 bit. The original block of image is partitioned in a multi-stage multi-candidate algorithm [56] and

through the stages, a number of candidates are eliminated by evaluating the distortion. This algorithm also use spiral search pattern and this can reduce overhead for comparison in most search locations.

3 Proposed New Motion Estimation Algorithm

In this thesis a new and novel motion estimation algorithm is introduced. Our objective is to separate macroblocks into classes with different search requirements. We base this classification on spatial and temporal correlation between macroblocks. The details of our method are described in the following subsections.

3.1 Basic Idea Behind Our Algorithm

3.1.1 Motion Estimation Based on Spatial Correlation

During the first step, the difference between the current and previous frame is calculated by direct subtraction. The resulting difference frames are subdivided into 16x16 macroblocks and the sum of the absolute difference (SAD) for each macroblock is calculated. These SAD values are sorted in descending order and the mean and standard deviation of each macroblock are derived.

Current methods search for the best matching macroblock by scanning the search window from left to right. In our case, the scanning process follows the sorted SAD values. Smaller SAD values indicate that the corresponding motion vector will also be smaller. Therefore, macroblocks that correspond to larger SADs have higher priority and their motion vectors are computed first. Appendix A lists the SAD and MV values of two frames taken from different video streams. We observe that the SAD and MV values are highly correlated with each other and that below a certain SAD value almost all of the MVs are zero. Our objective here was to separate the macroblocks into groups with different search requirements, i.e., those which require more calculations and more accuracy than others.

Although our main goal is to reduce the amount of calculations involved in the motion estimation process, we also need to ensure that we do not compromise the image quality. In order to achieve both objectives, we divide the macroblocks of each frame into three categories as shown in Table 1.

Table 1 List of methods used for each category

	Sorted SAD values	**Used Method for calculating motion vectors**
Category 1	Down to $m + \sigma$	Using any combination of two of the new subsets shown in Figure 18 and 19
Category 2	Between m and $m + \sigma$	Using one of new subset shown in Figure 18 and 19. If motion vectors of several successive MBs are (0,0), the rest of the motion vectors are also set to (0,0).
Category 3	Less than m	Using one of new subset shown in Figure 18 and 19 and taking into account spatial correlation. If motion vectors of several successive MBs are (0,0), the rest of the motion vectors are also set to (0,0).

Several different thresholds were tested in order to find the best classification which results in the least number of motion estimation calculations. Some of these thresholds include a combination of the mean (m) and standard deviation (σ), such as mean+σ/4, mean+σ/2, mean+σ

and mean. Tables 2 to 4 show that, for the same SNR and bit rate, the sum of the mean and standard deviation yields the smallest number of points for all the tested video streams.

Table 2 Total number of searching points

Seq.	Mean+σ	Mean+σ/2	Mean+σ/4	Mean
akiyo	24130	35472	42184	46881
claire	21725	30854	41010	47068
container	24298	41363	43152	45079
grandma	22085	44260	59067	70279
miss_am	24091	34819	42670	49843
news	18775	32299	43237	54661
salesman	18026	35066	48779	64039
silent	17152	25837	34890	49709
suzie	25611	38059	49384	61083

Table 3 Average Bit Rate Obtained

Seq.	Mean+σ (kbps)	Mean+σ/2 (kbps)	Mean+σ/4 (kbps)	Mean (kbps)
akiyo	28.63	28.02	27.76	27.78
claire	27.48	27.21	27.03	26.85
container	44.25	43.51	43.39	43.39
grandma	26.67	25.63	25.41	25.30
miss_am	30.16	28.73	28.33	28.21
news	66.36	65.08	64.23	64.00
salesman	41.09	40.43	40.22	40.25
silent	62.09	60.98	60.18	59.55
suzie	84.67	79.31	76.07	74.15

Table 4 Average SNR of Y component of 100 Frames

Seq.	Mean+σ (dB)	Mean+σ/2 (dB)	Mean+σ/4 (dB)	Mean (dB)
akiyo	34.23	34.31	34.32	34.32
claire	35.98	35.98	36.01	36.03
container	32.49	32.51	32.51	32.51
grandma	33.17	33.20	33.22	33.24
miss_am	36.77	36.92	36.96	36.98
news	32.38	32.40	32.42	32.43
salesman	31.67	31.68	31.69	31.70
silent	32.15	32.17	32.19	32.20
suzie	33.62	33.64	33.73	33.84

We choose this to be the threshold for the first category of macroblocks, which includes SAD values that lie between the highest SAD value and the sum of the mean and standard deviation. Similar performance evaluations led us to choose the mean SAD value as the second threshold, which separates the second from the third category. Thus, the second category includes macroblocks whose corresponding SAD values lie between the mean SAD value and the sum of the mean and the standard deviation. Finally, the third category includes all the macroblocks with SAD values smaller the mean SAD value.

Performance evaluations have shown that in order to maintain high image quality, the search algorithm for the first and second category should not take into consideration any spatial or spatial/temporal correlation between motion vectors.

In addition to classifying the macroblocks into three categories, we derived a scheme that allows us to determine the motion vector using only a subset of the total 256 pixels available in each 16x16 macroblock. The chsosen subset patterns shown in Figures 18 and 19 (shaded blocks) resulted from performance evaluations over a large number of pattern combinations. This process can be expanded to 16x16 macroblocks in a similar fashion.

Performance evaluations have shown that a combination of any two subsets provides enough information for accurate and efficient calculation of motion vectors for macroblocks that belong to the first category. The Diamond Search pattern is used for estimating the motion vectors in this category as well as the other two.

SAD values that are between the sum of the mean and standard deviation and the mean SAD values (i.e., category 2) indicate that there is little or no disparity between the macroblocks of the two consecutive frames. For this category, only one 16 pixel subset is needed for the calculation of the motion vector. Using this macroblock subset reduces the estimation complexity by up to 75%.

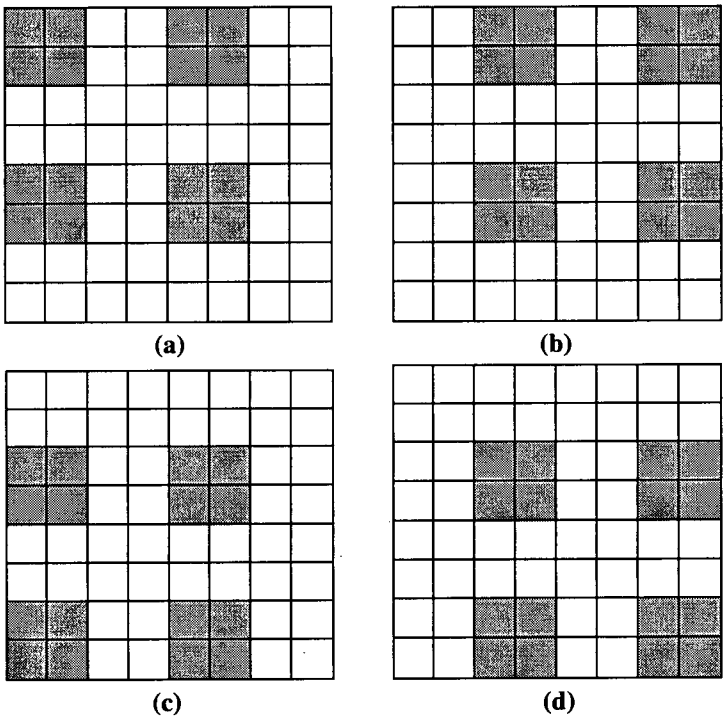


Figure 18 Four types of new subsets for motion estimation

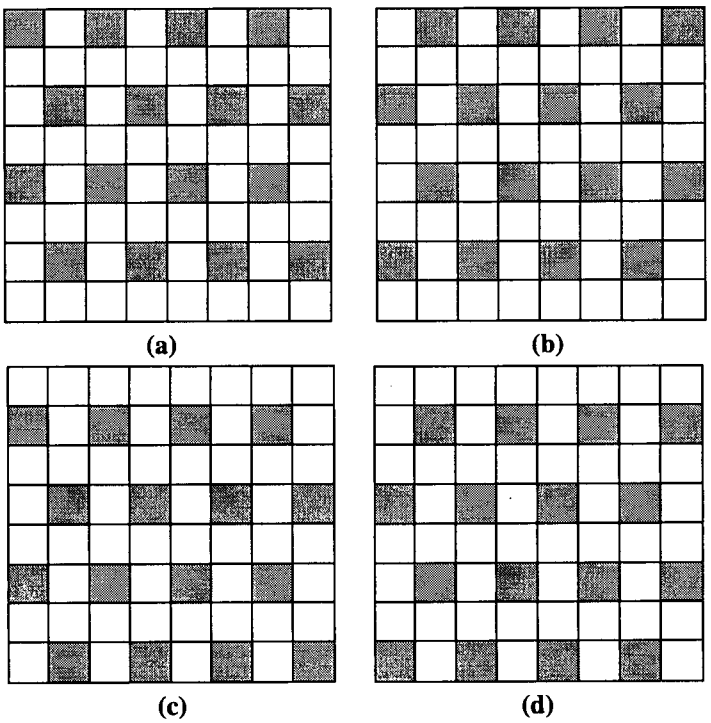


Figure 19 Four types of new subsets for motion estimation

Finally, for category 3, we take the spatial correlation into consideration. Using values of already known motion vectors around the target macroblock (shaded areas in Figure 20), we calculate the corresponding motion vector as the mean of the surrounding motion vectors. This vector becomes the initial search point for the macroblock. This is different from the conventional motion estimation algorithms, which use the center of each macroblock as the initial search point.

For the second and third categories, we further improve the search speed by assuming that if the motion vectors of several successive macroblocks are (0,0), then the rest of the macroblocks are stationary or quasi-stationary and their motion vectors are also (0,0).

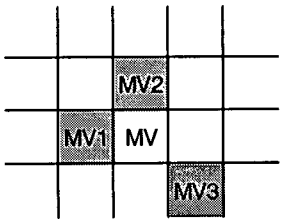


Figure 20 Target vector calculated as the average of already known motion vectors (shaded areas) around it

The following subsection describes in detail our algorithm.

3.1.2 Our basic Algorithm

Figure 21 shows the flowchart of our algorithm.

Our algorithm could be summarized in the following steps;

Step 1: Assume using a QCIF (176x144) frame. If P is the previous frame and C is the present frame, we can calculate the new difference N frame as follows:

$$N = \sum_{i=0}^{i=175} \sum_{j=0}^{j=143} \left| C_{ij} - P_{ij} \right|$$

where P_{ij} and C_{ij} are the luminance values of the (i,j) pixel of the previous and current frames, respectively.

Step 2: Frame N is divided into 16x16 macroblocks. Each macroblock can be indexed by M_{nm} and the sum of absolute difference of the pixels of each macroblock can be represented as SAD_{nm}, where n and m represent rows and columns respectively.

Let the mean value of SAD_{nm} to be $\overline{m}$ and the standard deviation of SAD_{nm} to be σ.

Step 3: Sort SAD_{nm} in descending order and store the value and the order in an 1-dimensional structure array D_i. The array D_i contains the values of SAD_{nm} and the positions of the macroblock (m, n.) Figure 22 illustrates an example of the order of the macroblocks for one frame.

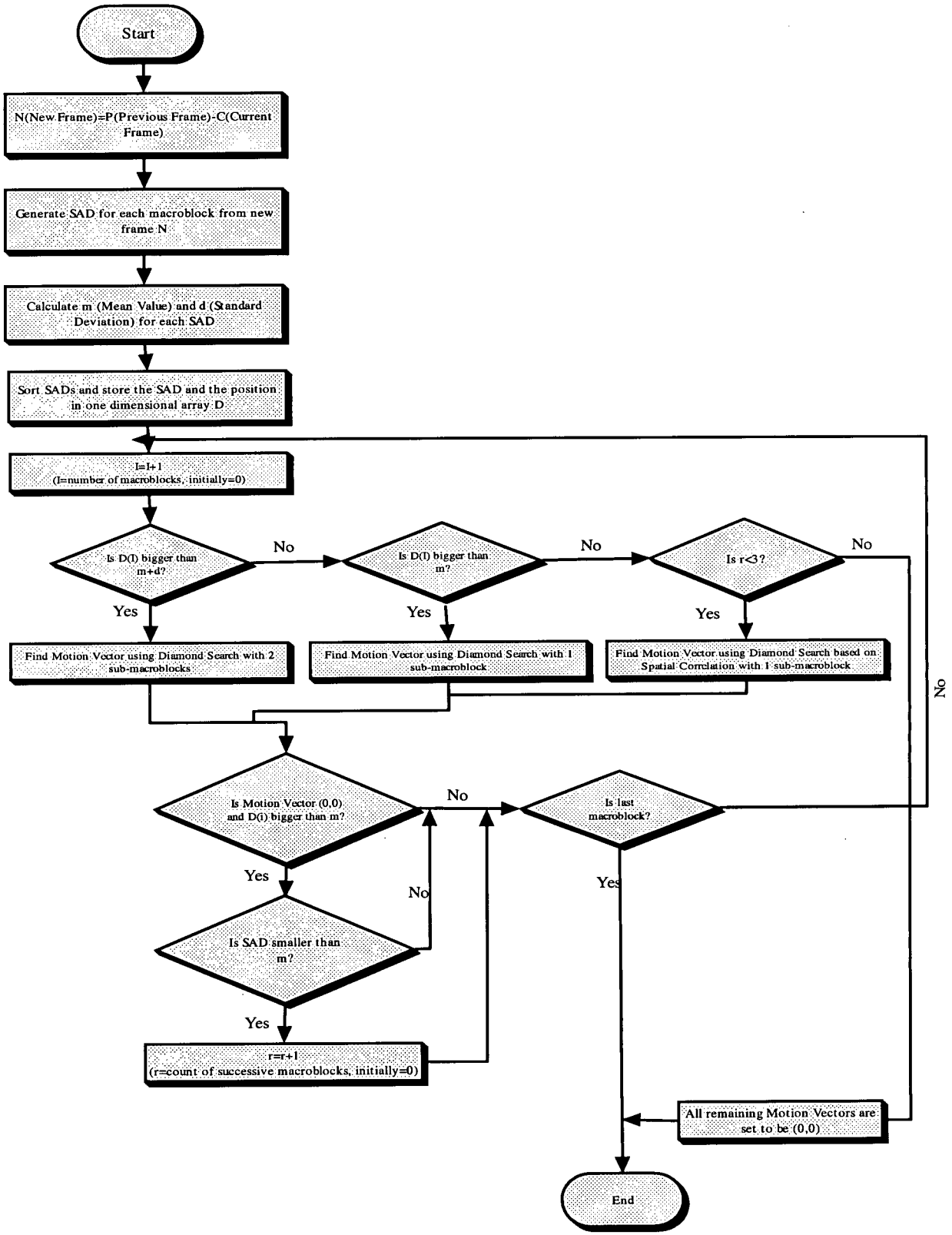


Figure 21 Flowchart for the first proposed motion estimation algorithm

	10	15	27	19	49	48	
	11	20	16	2	4	41	
	30	7	1	6	3	17	
	9	8	5	12	14	13	
	18	21	79	60	54	58	
	79	80	76	31	28	25	

Figure 22 Order of macroblocks

Step 4: Let the motion vector of this frame to be $MV(n,m)$. Initialize the number of repetitions of successive motion vectors to zero (r=0).

$r = 0$

While D_i has SAD values

If $D_i \geq \overline{m} + \sigma$ then

Find the exact motion vector using two of new subsets (Figure 18 and 19) of a macroblock using diamond search (DS) algorithm

Else if $D_i < \overline{m} + \sigma$ and $D_i \geq \overline{m}$ then

Find the exact motion vector using only one of new subset of a macroblock using diamond search (DS) algorithm

If MV =(0,0) and $r > 3$ then

Set rest of MV to (0,0)

Break while loop

Else if MV =(0,0) and $r < 3$

$r = r + 1$

Else

$r = 0$

End if

Else if $D_i < \overline{m}$ then

If neighbouring MV s' values exist then

Predict motion vector using method described in subsection 3.3

End if

Find exact motion vector using one of new subset of a macroblock using a combination of DS and prediction method described in section 3.3.

If MV =(0,0) and $r > 3$ then

Set rest of MVs to (0,0)

Break while loop

Else if MV =(0,0) and $r < 3$

$r = r + 1$

Else

$r = 0$

End if

End if

End While

3.2 Basic Idea Behind New Advanced Algorithm

In the previous section we build our algorithm on spatial correlation in a macroblock. However, it is only natural to try to take advantage of the temporal correlation present in video streams as well.

3.2.1 Motion Estimation Based on Spatial-Temporal Correlation

As in the previous section, sum of the absolute difference (SAD) was calculated for macroblocks resulting from direct subtraction of the present and previous frames. However, in addition to that, we also take advantage of the temporal correlation between two consecutive frames. The motion vectors of the previous frame are used to determine the predicted motion vector of the corresponding macroblocks in the current frame. Then the difference between macroblocks from the current frame and the corresponding search area is calculated taking into consideration the temporal correlation from the previous frame. The new difference frame is subdivided into 16x16 macroblocks and the sum of the absolute difference (SAD) for each macroblock is calculated.

We compare the two SAD values that we obtained for each macroblock and since the smaller one will yield a better approximation for the motion vector, we generate a new SAD array using the smallest of the two values.

The rest of the process for finding the motion vectors is very similar to the one described in the previous section. The SAD array is sorted and the macroblocks are divided into three categories. For the first two categories, the motion vector is calculated in a similar manner as in the previous section.

However, in this case, i.e., for category 2 and 3, when the SAD value is smaller than the sum of the mean and the standard deviation, if several successive motion vectors are (0, 0), we can assume that the rest of macroblocks are stationary or quasi-stationary and for this reason their motion vectors are also set to (0,0). In addition to the above, for the same two categories, temporal correlation between motion vectors is used to reduce motion estimation calculations. For the same categories, if several successive motion vectors are the same as the corresponding vectors from the previous frame, the rest of the macroblock vectors are set to have the values from the previous frame as well.

For category 3, when the SAD is smaller than the mean, the initial search point is set as explained in Section 3.3 and the same process as in previous case is used.

3.2.2 New Advanced Algorithm

Figure 23 shows the flowchart of new algorithm.

Step 1: Assume using a QCIF (176x144) frame. If P is the previous frame and C is the present frame, we can calculate the new difference N_{nc} frame as follows:

$$N_{nc} = \sum_{i=0}^{i=175} \sum_{j=0}^{j=143} \left| C_{ij} - P_{ij} \right|$$

If there are previous motion vectors, we can generate another frame as follow,

$$N_c = \sum_{i=0}^{i=175} \sum_{j=0}^{j=143} \left| C_{i,j} - P_{i-x,j-y} \right|$$

where P_{ij} and C_{ij} are the luminance values of the (i,j) pixel of the previous and current frames, respectively.

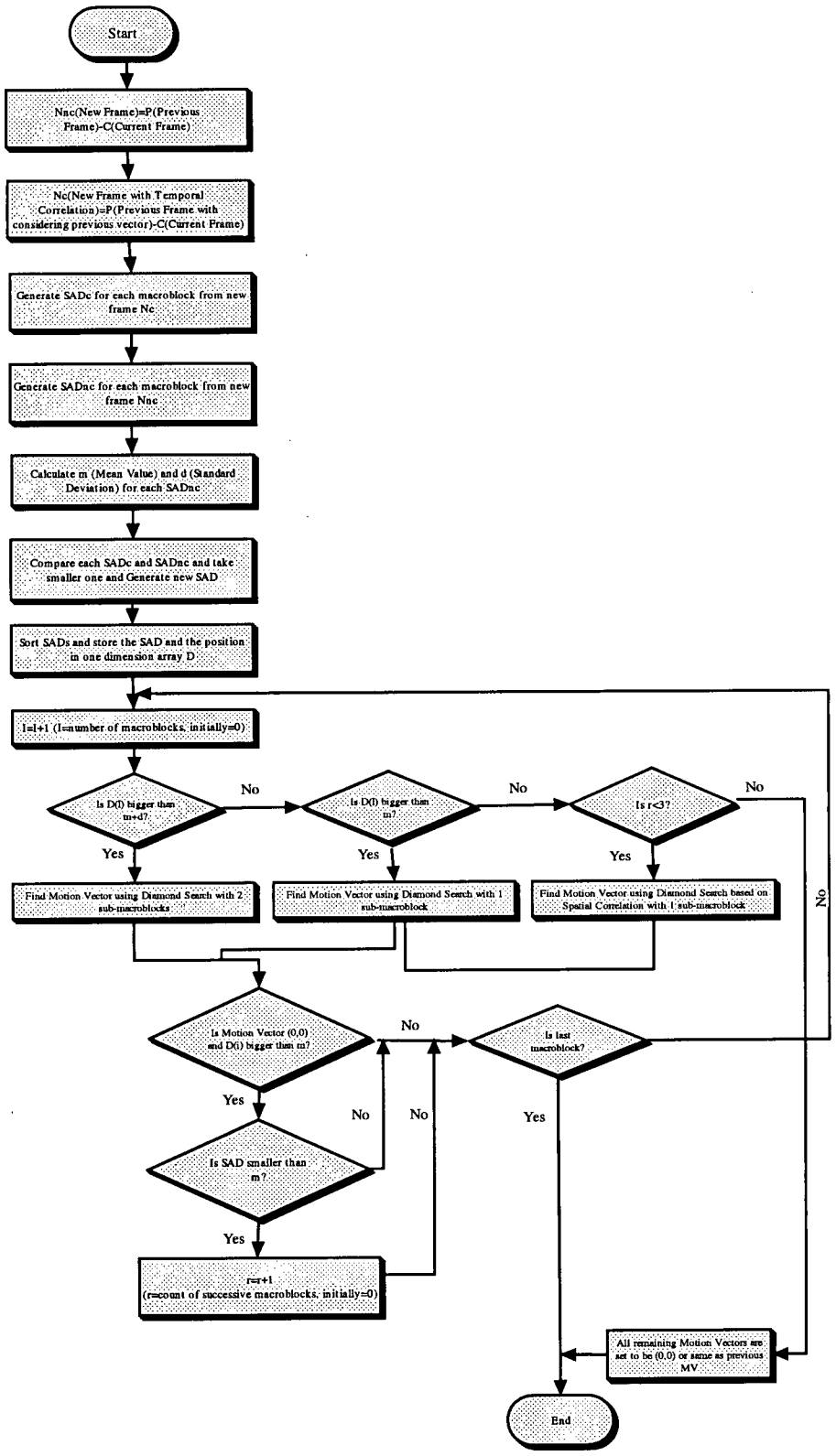


Figure 23 Flowchart for the second proposed motion estimation algorithm

Step 2: Frame N_{nc} and N_c is divided into 16x16 macroblocks. Each macroblock can be indexed by $M_{nc}(n,m)$ and $M_c(n,m)$ and the sum of absolute difference of the pixels of each macrobloack can be represented as $SAD_{nc}(n,m)$ and $SAD_c(n,m)$. Let the mean value of $SAD_{nc}(n,m)$ to be $\overline{m}$ and the standard deviation of $SAD_{nc}(n,m)$ be σ.

If there are both $SAD_{nc}(n,m)$ and $SAD_c(n,m)$, the new $SAD(n,m)$ can be generated as follow,

If $SAD_{nc}(n,m) >= SAD_c(n,m)$ then

$SAD(n,m) = SAD_c(n,m)$

Else

$SAD(n,m) = SAD_{nc}(n,m)$

End if

Step 3: Sort $SAD(n,m)$ in descending order and store the value and the order in an 1-dimensional structure array D_i . The array D_i contains the values of $SAD(n,m)$ and the positions of the macroblock (m, n.) in addition to the information whether it is from $SAD_{nc}(n,m)$ or $SAD_c(n,m)$.

Step 4: Let the motion vector of previous frame be $MV'(n,m)$ and the current frame's motion vector of this frame to be $MV(n,m)$. Initialize the number of repetitions of successive motion vectors to zero (r=0).

$r = 0$

While D_i has SAD values

If $D_i \geq \overline{m} + \sigma$ then

If D_i is from SAD_{nc} then

Find the exact motion vector using two of new subsets (Figure 18 and 19) of a macroblock using diamond search (DS) algorithm

Else

Find the exact motion vector using two of new subsets (Figure 18 and 19) of a macroblock using diamond search (DS) algorithm with initial search point of MV'

End if

Else if $D_i < \overline{m} + \sigma$ and $D_i \geq \overline{m}$ then

If D_i is from SAD_{nc} then

Find the exact motion vector using only one of new subset of a macroblock using diamond search (DS) algorithm

Else

Find the exact motion vector using only one of new subset of a macroblock using diamond search (DS) algorithm with the initial search point moved to MV'

End if

If (MV =(0,0) or $MV = MV'$) and $r > 3$ then

Set rest of MV to (0,0) or MV' depending on whether D_i is from SAD_{nc} or SAD_n

Break while loop

Else if (MV =(0,0) or $MV = MV'$) and $r < 3$

$r = r + 1$

Else

$r = 0$

End if

Else if $D_i < \overline{m}$ then

Find exact motion vector using one of new subset of a macroblock using a combination of DS and prediction method described in section 3.3.

If (MV =(0,0) or $MV = MV'$) and $r > 3$ then

Set rest of MV to (0,0) or MV' depending on whether D_i is from SAD_{nc} or SAD_n

Break while loop

Else if (MV =(0,0) or $MV = MV'$) and $r < 3$

$r = r + 1$

Else

$r = 0$

End if

End if

End While

3.3 Determining Initial Search Position for Our Algorithm

Figure 24 illustrates the real motion vectors of two consecutive frames of the news (qcif) stream using full search method. The shaded areas are showing the non-zero motion vectors and their corresponding temporal motion vectors in the previous frame. We observe that the correlation between these vectors is very high. Experimental results from a large set of video streams yielded similar results. Based on this observation, we developed a simple algorithm for predicting motion vectors and therefore starting search point using information of neighbouring vectors and the previous frame vector.

Because our method sorts the macroblocks according to their SAD values, we need to design a process for predictiong the initial value of the motion vector for each macroblock. In case that the SAD value of the specific macroblock is derived from spatial correlation only (i.e., direct subtraction of two frames), then the following two steps are executed.

If all the surrounding motion vectors have not been calculated (e.g., first macroblock used in a frame), then the initial value is set to (0, 0).

If one or more of the surrounding motion vectors is known, then the mean value of the known motion vectors is used as the initial MV values.

In case that the SAD value of the specific macroblock results from temporal correlation, then the following two steps are used.

If all the surrounding motion vectors have not been calculated (e.g., first macroblock used in a frame), then the initial value is set to be equal to the motion vector value of the previous frame.

If one or more of the surrounding motion vectors is known, then the procedure can be described as follows:

Let the previous motion vector be $MV_p(x_p, y_p)$ and the neighbouring motion vectors be $MV_n(x_n, y_n)$. In general, if there are n known motion vectors, we can simply obtain the predicted motion vector $MV(x, y)$ of a macroblock as follows:

$$MV_{SPATIAL}(x_{SPATIAL}, y_{SPATIAL}) = \frac{1}{n}\sum_{k=1}^{n} MV_k(x_k, y_k) \tag{4.1}$$

(Case 1) $MV_p(x_p, y_p) = MV_{SPATIAL}(x_{SPATIAL}, y_{SPATIAL})$

$$MV(x, y) = MV_p(x_p, y_p) \tag{4.2}$$

(Case 2) $MV_P(x_P) = MV_{SPATIAL}(x_{SPATIAL})$ and $MV_P(y_P) \neq MV_{SPATIAL}(y_{SPATIAL})$

$$MV(x) = MV_p(x_p) \tag{4.3}$$

$$MV(y) = \frac{MV_p(y_p) + MV_{SPATIAL}(y_{SPATIAL})}{2} \tag{4.4}$$

(Case 3) $MV_P(x_P) \neq MV_{SPATIAL}(x_{SPATIAL})$ and $MV_P(y_P) = MV_{SPATIAL}(y_{SPATIAL})$

$$MV(x) = \frac{MV_p(x_p) + MV_{SPATIAL}(x_{SPATIAL})}{2} \tag{4.5}$$

$$MV(y) = MV_p(y_p) \tag{4.6}$$

(Case 4) $MV_p(x_p, y_p) \neq MV_{SPATIAL}(x_{SPATIAL}, y_{SPATIAL})$

$$MV(x, y) = \frac{MV_p(x_p, y_p) + MV_{SPATIAL}(x_{SPATIAL}, y_{SPATIAL})}{2} \tag{4.7}$$

0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0
0 0	0 0	0 0	0 0	0 0	0 0	1 0	0 0	0 0
0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	1 1
1 1	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0
0 0	1 2	0 1	0 0	0 0	0 0	0 0	0 0	0 0
0 0	0 0	0 0	1 0	0 0	1 0	0 0	0 0	0 0
0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0
0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0
0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0
0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0
0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0

(a)

0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0
0 0	0 0	0 0	0 0	0 0	0 0	1 0	0 0	0 0
0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	1 1
0 0	0 0	0 0	0 -7	0 0	0 0	0 0	0 0	0 0
0 0	1 1	0 1	0 0	0 0	0 0	0 0	0 0	0 0
0 0	0 0	0 0	1 0	0 0	1 0	0 0	0 0	0 0
0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0
0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0
0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0
0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0
0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0

(b)

Figure 24 Motion vector of news.qcif stream: (a) Motion vector of 71st frame, (b) Motion vector of 72nd frame

4 Implementation of Our ME Algorithm

A variety of conventional MPEG encoders is used for different multimedia applications. H.263 is widely used for low-bit rate multimedia delivery in wireless environments. The H.323 communication standard supports both H.261 and H.263 video codecs and it includes several other protocols, such as the multimedia control protocol H.245, signaling protocols H.225.0.

In this study, we implemented our new motion estimation algorithm on the H.263 standard to show how our approach improves the computational speed of a real video compression scheme. H.263 is presently the codec of choice for the mobile devices. H.264 is the emerging video standard, but at present its complexity is a challenge for low power consumption applications such as cell phones and PDAs. Implementing our algorithm on the H.264 standard is the next logical research step. Some H.264's advanced features such as different macroblock sizes and multi-frame predictions must be taken into consideration.

We implemented our algorithm on H.263 that runs on a PC platform as well as video enabled PDA. The details of the two implementations and specifications of the devices are described in the following section.

4.1 Specification of Hardware and Software used for Implementing Our ME Algorithm

This section describes the specifications of the hardware and software used to design and implement our new algorithm. Even though the new motion estimation algorithm is focused on mobile devices, such as PDAs and cell phone, we first developed and tested our algorithm on a PC platform and then we imported to a video enabled PDA.

4.1.1 Hardware

For the design and implementation of our method, we used an HP Pavilion ze4430us which is powered by AMD Athlon XP-M 2400+. Table 5 shows the specifications of the PC. All the coding was completed on this laptop and the complied code was uploaded to an HP iPAQ h4150 handheld device through USB cradle. This HP iPAQ h4150 was to test our algorithm. This uses a 400MHz Intel® XScale™ processor and includes 64MB internal RAM and 32MB integrated ROM. Its 3.5" Transflective TFT LCD display uses 64,000 colors and has maximum resolution of 240x320 pixels.

Table 5 Specification of HP Pavillion ze4430us

Processor	1.80-GHz Athlon XP-M 2400+
Operating system	Microsoft® Windows® 2003 Standard Edition
Memory	512 MB RAM(SDRAM)
HDD	40GB Hard Drive
Display	15.0" TFT (1024-by-768)
Video Card	ATI MOBILITY RADEON 4x AGP graphics with 64MB of shared memory
Expansion	PC Card Slot, FireWire Port, Parallel Port, Serial Port, S-Video Out Port
Modem	Integrated 56k v.92 Data/Fax Modem
Network	10-/100-Mbps Ethernet
Integrated wireless	Integrated WLAN 802.11g

The 400MHz Intel® XScale processor PXA255 outperforms Intel's StrongARM processor by a factor of 2 and is currently used by iPAQ, Zaurus and Dell. PXA255 is a highly integrated, 32-bit RISC processor utilizing advanced Intel 0.18μ process for high core speeds at

low power that combines the efficiency of Intel design with the ARM v.5TE instruction set architecture. This process is able to support games, movies and richer music as well as many of the latest applications being developed for the home and workplace. It also complements the Intel StrongARM SA-1110 applications processors, the leading platform for Pocket PC devices today, and sets the stage for a new class of high-performance, low-power wireless communications devices.

Table 6 summarizes the specifications of the HP iPAQ h4150 PDA.

Table 6 Specification of HP iPAQ h4150

Processor	400MHz Intel® XScale™ processor
Operating system	Microsoft® Windows® Mobile 2003 Premium for Pocket PC
Memory	64MB SDRAM (55MB user accessible), 32MB Flash ROM
Display	3.5" Transflective TFT display with 64,000 colors
Expansion	Integrated SD slot - supports SD/MMC type standard, SDIO ready
Integrated wireless	Integrated WLAN 802.11b, Bluetooth®, IrDA

4.1.2 Software

Microsoft Visual C++ 6.0 (with Service Pack 6) which is widely used in academic and industrial fields was used to implement our PC based H.263 Encoder on Microsoft Windows Server 2003 Standard Edition. This tool makes use of a wide range of Microsoft Window applications and is ideal for designing and implementation for the most proficient and efficient applications.

Microsoft eMbedded Visual C++ 4.0 (with Service Pack 3) is appropriate for developing the H.263 Encoder for Pocket PC 2003 platform application such as HP iPAQ h4150. The

Software Development Kit for Windows Mobile 2003-based Pocket PCs is also required. The Microsoft eMbedded Visual C++ 4.0, which is used to implement applications running on Windows CE.NET 4.0, 4.1, 4.2 and PocketPC 2003, is a powerful development tool that provides significant benefits to companies which are building "native" executables for the next generation of Windows CE-based devices.

5 Experimental Results

The Telenor H.263 encoder 1.7 with variable bitrate, quantization parameter of 10, 0 frame skip, reference frame rate of 30 was used for our tests. The macroblock size was fixed at 16x16 pixels and the maximum value of motion vector displacement was set to ± 7 pixels in both rows and columns. Variable bit rates were used, which means that the network usage changes in order to ensure that the image quality is within a certain range.

5.1 Performance Evaluation of Proposed Motion Estimation Algorithm

In our tests we used thirteen QCIF file video sequences, which are known to be very good content representation for mobile applications. These sequences are "Akiyo", "Carphone", "Children", "Claire", "Container", "Foreman", "Grandma", "Miss America", "News", "Salesman", "Silent", "Suzie" and "Trevor".

At first we compare our method against four conventional, widely used algorithms, the Full Search (FS), Diamond Search (DS), Four Step Search (4SS) and New Three Step Search (NTSS), in terms of SNR, average bit rates and computational complexity.

Table 7 and Table 8 show the SNR and the average bit rates for all the tested methods. Figures B-1 to B-13 in Appendix B show the MSE values of all the encoded frames of the video sequences, indicating that the picture quality of each video stream remains constant. For subjective comparisons, Figures 25 a, b, c, d, e, f, g show the 72^{nd} frame of the "salesman" video sequence, obtained by each of the tested motion estimation methods. We observe that the visual quality of all these frames is the same. From above observations (subjective quality, MSE and SNR values and the bit rates) we can conclude that the picture quality is kept the same for all our

tests. Table 9 shows the average number of processed points per macroblock and Table 10 shows the total number of processed points used by every method to determine the motion vectors for 100 frames. The last two columns show the results obtained by our method when only spatial correlation is used and when spatial and temporal correlation are combined (last column). It is clear that the performance of our method improves significantly when spatial and temporal correlations are used.

When compared with all the other methods, we observe that our approach drastically improves the motion estimation search speed. In particular, our algorithm has on average only 0.5% of the computational complexity of the Full Search (FS) method and is almost 19 times faster than 4SS, the best among the conventional methods.

Table 7 Average SNR of Y component of 100 Frames

Seq.	FS (dB)	NTSS (dB)	4SS (dB)	DS (dB)	Proposed (Spatial only) (dB)	Proposed (dB)
akiyo	34.34	34.26	34.28	34.28	34.19	34.22
carphone	33.42	33.07	33.12	33.15	32.97	33.08
children	30.54	30.55	30.54	30.55	30.47	30.42
claire	36.17	36.07	36.06	36.06	35.94	35.94
container	32.51	32.51	32.51	32.51	32.48	32.48
foreman	31.92	31.29	31.39	31.43	31.20	31.36
grandma	33.27	33.25	33.25	33.25	33.17	33.18
miss_am	37.14	36.76	36.87	36.90	36.79	36.83
news	32.46	32.43	32.43	32.43	32.37	32.38
salesman	31.71	31.71	31.71	31.71	31.67	31.67
silent	32.25	32.27	32.27	32.25	32.19	32.19
suzie	34.25	33.98	34.09	34.07	33.90	33.94
trevor	32.62	32.56	32.61	32.59	32.46	32.35

Table 8 Average Bit Rate

Seq.	FS (kbps)	NTSS (kbps)	4SS (kbps)	DS (kbps)	Proposed (Spatial only) (kbps)	Proposed (kbps)
akiyo	27.48	28.50	28.05	27.78	28.86	28.55
carphone	82.72	100.94	94.99	93.13	97.98	95.15
children	230.29	247.53	250.57	248.23	251.67	243.81
claire	26.91	27.70	27.16	26.99	27.25	27.38
container	43.51	43.81	43.75	43.82	44.53	43.98
foreman	111.84	207.14	193.33	189.75	208.40	192.53
grandma	25.04	25.42	25.20	25.13	26.29	26.22
miss_am	27.56	34.08	31.37	30.54	31.40	31.15
news	62.91	65.03	64.55	63.88	66.22	65.89
salesman	39.98	40.83	40.12	40.02	40.63	40.79
silent	57.03	60.18	58.68	58.51	61.20	60.63
suzie	64.98	85.56	79.97	79.43	82.23	85.44
trevor	96.72	109.94	106.25	105.48	112.63	129.19

Table 9 Average number of processed points per macroblock

Seq.	FS	NTSS	4SS	DS	Proposed (Spatial only)	Proposed
akiyo	183.5	19.7	11.1	11.4	1.25	1.11
carphone	183.5	20.3	11.4	12.5	2.05	1.91
children	183.5	20.5	11.3	12.2	1.52	1.08
claire	183.5	19.5	11.1	11.5	1.17	1.16
container	183.5	19.7	11.7	11.4	1.22	1.04
foreman	183.5	20.8	11.6	13.3	2.36	1.71
grandma	183.5	19.7	11.1	11.5	1.17	1.15
miss_am	183.5	19.9	11.2	11.8	1.53	1.31
news	183.5	19.7	11.1	11.5	1.01	0.98
salesman	183.5	19.7	11.1	11.5	0.99	0.90
silent	183.5	20.1	11.3	11.9	1.05	0.93
suzie	183.5	20.5	11.6	12.9	2.35	1.71
trevor	183.5	20.0	11.3	12.1	1.58	1.25

Table 10 Total number of processed points

Seq.	FS	NTSS	4SS	DS	Proposed (Spatial only)	Proposed
akiyo	1817200(100%)	194612(10.7%)	109526(6.0%)	113267(6.2%)	12460(0.7%)	11019 (0.6%)
carphone	1817200(100%)	201333(11.1%)	113160(6.2%)	124147(6.8%)	20336(1.1%)	18882(1.0%)
children	1817200(100%)	202956(11.2%)	111918(6.2%)	121033(6.7%)	15077(0.8%)	10658(0.6%)
claire	1817200(100%)	192929(10.6%)	109814(6.0%)	113719(6.3%)	11574(0.6%)	11503(0.6%)
container	1817200(100%)	194812(10.7%)	109575(6.0%)	113228(6.2%)	12053(0.7%)	10287(0.6%)
foreman	1817200(100%)	205713(11.3%)	114971(6.3%)	131938(7.3%)	23394(1.3%)	16923(0.9%)
grandma	1817200(100%)	194844(10.7%)	109712(6.0%)	113712(6.3%)	11632(0.6%)	11402(0.6%)
miss_am	1817200(100%)	197052(10.8%)	110565(6.1%)	116481(6.4%)	15142(0.8%)	13011(0.7%)
news	1817200(100%)	195368(10.8%)	109787(6.0%)	114192(6.3%)	9993(0.5%)	9743(0.5%)
salesman	1817200(100%)	195054(10.7%)	109748(6.0%)	113763(6.3%)	9773(0.5%)	8944(0.5%)
silent	1817200(100%)	199035(11.0%)	111388(6.1%)	118102(6.5%)	10410(0.6%)	9211(0.5%)
suzie	1817200(100%)	203142(11.2%)	114693(6.3%)	127771(7.0%)	23277(1.3%)	16908(0.9%)
trevor	1817200(100%)	197547(10.9%)	111495(6.1%)	119339(6.6%)	15663(0.9%)	12374(0.7%)

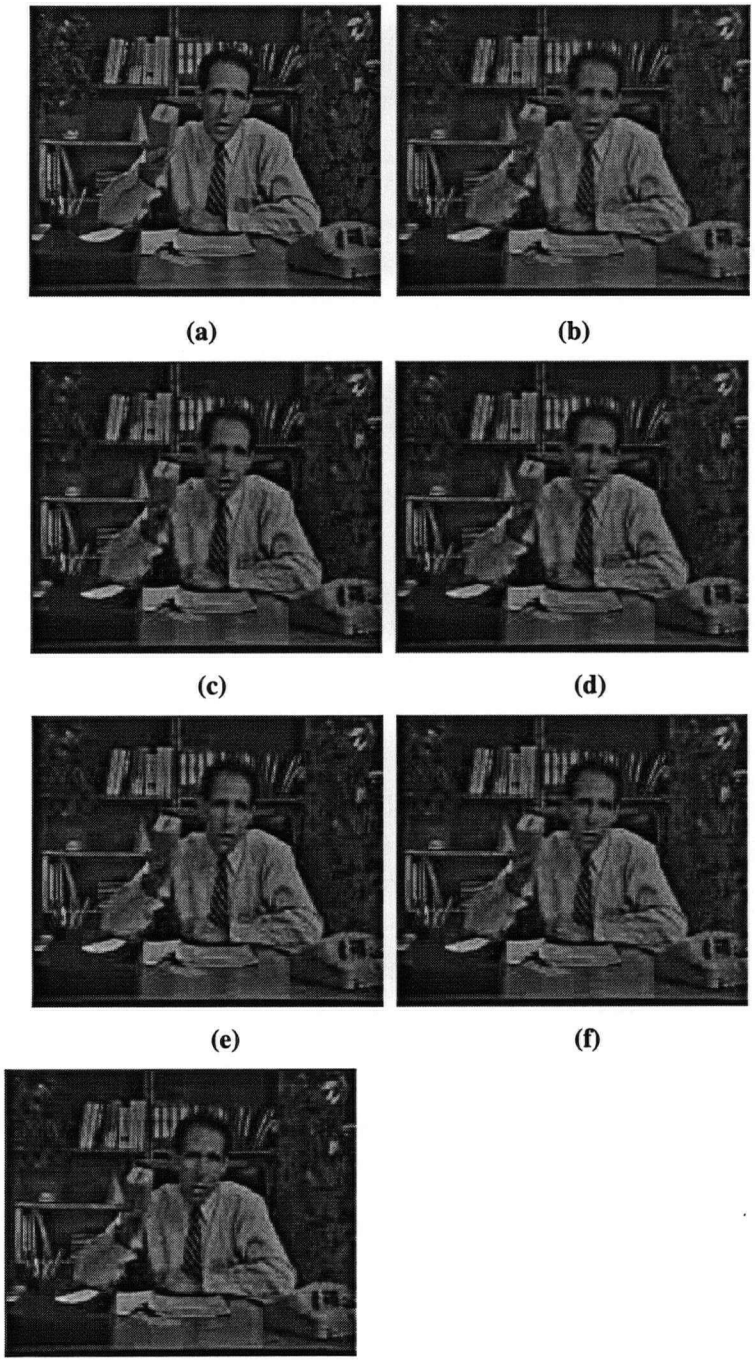


Figure 25 Decoded 72nd frame for the "salesman" sequence: (a) Original, (b) FS, (c) DS, (d) NTSS, (e) 4SS, (f) Proposed (Spatial only) Algorithm, (g) Proposed Algorithm

We also compared our method with a non conventional motion estimation approach known as Adaptive Motion Estimation (AME) method, which recent studies have shown that outperforms all the existing motion estimation methods [26].

Table 11 shows the MSE values for two conventional methods (TSS and 4SS), the non conventional AME method and our method obtained for 3 different video streams. Table 12 shows the average number of search points per macroblock for the above motion estimation methods. We observe that our spatial-temporal based method outperforms AME, resulting in search speed that is 7.5 times faster than that of AME.

Table 11 Average MSE per Frame

	TSS	4SS	AME	Proposed (Spatial only)	Proposed
Akiyo	36.91	36.41	36.34	30.52	30.28
Foreman	84.413	81.78	81.03	80.45	78.11
Stefan	160.16	158.27	153.37	150.28	152.19

Table 12 Average Number of Search Points per Macroblock

	TSS	4SS	AME	Proposed (Spatial only)	Proposed
Akiyo	25	33	11.38	1.31	1.11
Foreman	25	33	13.99	2.49	1.71
Stefan	25	33	12.99	3.28	1.73

6 Conclusion

The focus of this thesis is on the development of novel motion estimation algorithms for video compression, which are computationally very efficient without sacrificing the quality of the video, as required for low bit rate video coding in battery powered mobile devices such as personal digital assistants and cellular telephones. The proposed methods are mainly based upon the spatial correlation among neighbouring macroblocks, or spatial-temporal correlation among neighbouring and previous frame's macroblocks.

The main features of the proposed motion estimation algorithms are: (1) methods to classify macroblocks according to the amount of motion as indicated by the SAD of each macroblock compared the SAD statistics among all the macroblocks; (2) different efficient search methods for each class of macroblocks; and (3) searching over a subset of the pixels in each macroblock for further speed-up. The proposed algorithms have been implemented using the diamond search pattern that is widely used and studied in conventional search algorithms. However, the proposed algorithms are independent of the search pattern and other search patterns can easily be adopted. To further reduce computations, all the remaining motion vectors in a frame are set to (0,0) if the motion vectors of several successive macroblocks are found to be (0,0).

We have evaluated the performance of the proposed algorithms and compared with several existing methods. We have presented the results of the performance evaluations, which clearly show that the proposed algorithms significantly reduce the number of computations for motion estimation without degrading picture quality. For the same picture quality, the proposed algorithms are about 200 times faster than the full search method, and almost 19 times faster than 4SS, the best presently available conventional method. When compared to the most recently

published non-conventional motion estimation method, which outperforms all the existing methods, our approach improves the computational speed of motion estimation by 7.5 times.

In this thesis, we only explore the possibility of speed-up of H.263 standard. It is, however, imperative to look for a way to implement the proposed algorithm on the emerging H.264 multimedia communication standard. In addition, since H.264 uses several new operations that are computationally intensive, such as intra frame estimation, macroblock partitioning, multi reference in motion estimation and quarter-pel motion estimation, we might need to find new solutions to improve the efficiency of video compression and reduce power consumption in mobile devices employing this new standard.

7 Bibliography

[1] S.-H. Han, S.-W. Kwon, T.-Y. Lee and M.-K. Lee, "Low power motion estimation algorithm based on temporal correlation and its architecture," *International Symposium on Signal Processing and its Applications*, vol. 2, pp. 647-650, Aug. 2001.

[2] J. Biemond, L. Looijenga, D. Boekee and R. Plompen, "A pel-recursive Wiener-based displacement estimation algorithm," *Signal Processing*, vol. 13, pp 399-412, 1987.

[3] H.-M. Hang, A, Puri and D. Scilling, "Motion-compensated transform coding based on block motion tracking algorithm," *IEEE International Conference on Communicatons*, vol. 1, pp. 136-140, 1987.

[4] Y.T. Tse and R.L. Baker, "Global zoom/pan estimation and compensation for video compression," *Proc. International Conference on Acoustics, Speech, and Signal Processing*, pp. 2725-2728, 1991.

[5] T. Koga, K. Iinuma, A. Hirano, Y. Iijima and T. Ishiguro, "Motion compensated interframe image coding for video conference," *Proc. NTC*, pp. C9.6.1-9.6.5, 1981.

[6] R. Li, B. Zeng and M.L. Liou, "A new Three-Step Search Algorithm for Block Motion Estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 4, pp. 438-442, Aug. 1994.

[7] T.-S. Choi and J.-N. Kim, "Real-time video coding," *IEEE Transactions on Consumer Electronics*, vol. 45, pp. 417-426, May 1999.

[8] L.M. Po and W.C. Ma, "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, June 1996.

[9] R.S. Richmond II and D. S. Ha, "A low-power motion estimation block for low bit-rate wireless video," *International Symposium on Low Power Electronics and Design*, pp. 60-63, Aug. 2001.

[10] L.K. Liu and E. Feig, "A block-based gradient descent search algorithm for block motion estimation in video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, pp. 419-422, Aug. 1996.

[11] A. Averbuch and Y. Keller, "Fast motion estimation using bidirectional gradient methods," *Proc. Acoustics, Speech, and Signal Processing*, vol. 4, pp. 3616-3619, May 2002.

[12] Y. Keller and A. Averbuch, "Fast gradient methods based on global motion estimation for video compression," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, pp. 300 -309, Apr. 2003.

[13] J. Y. Tham, S. Ranganath, M. Ranganath and A. A. Kassim, "A Novel Unrestricted Center-Based Diamond Search Algorithm for Block Motion Estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, pp. 369-377, Aug. 1998.

[14] S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast block matching motion estimation," *Proc. International Conference on Information, Communications and Signal Processing*, vol. 1, pp. 292 -296, Sep. 1997.

[15] C. Zhu, X. Lin, L.-P. Chau, H.-A. Ang and C.-Y. Ong, "An optimized diamond search algorithm for block motion estimation," *IEEE International Symposium on Circuits and Systems*, vol. 2, pp. 488-491, May 2002.

[16] T. Sappasitwong, S. Aramvith, S. Jitapunkul, A. Tamtrakarn, P. Kitti-Punyangam and H. Kortrakulkij, "Adaptive asymmetric diamond search algorithm for block-based motion

estimation," *International Symposium on Video/Image Processing and Multimedia Communications*, pp. 283-288, Jun. 2002.

[17] W. Zheng, I. Ahmad and M.L. Liou, "Real-time software based MPEG-4 video encoder," *Proc. of Workshop and Exhibition on MPEG-4*, pp. 71-74, Jun. 2001.

[18] P.H.N. de With, "A simple recursive motion estimation technique for compression of HDTV signals," *IEEE International Conference on Image Processing And its Application*, pp. 417-420, 1992.

[19] G. de Haan, P. W.A.C. Biezen, H. Huijgen and O. A. Ojo, "True-Motion Estimation with 3-D Recursive Search Block Matching," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 3, pp. 368-379, Oct. 1993.

[20] L. Luo, C. Zou, X. Gao and Z. He, "A new prediction search algorithm for block motion estimation in video coding," *IEEE Transactions on Consumer Electronics*, vol. 43, pp. 56-61, Feb. 1997.

[21] K.-L. Chung and L.-C. Chang, "A new predictive search area approach for fast block motion estimation," *IEEE Transactions on Image Processing*, vol. 12, pp. 648-652, Jun. 2003.

[22] J. Jain and A. Jain, "Displacement Measurement and Its Application in Interframe Image Coding," *IEEE Transactions on Communications*, vol. 29, pp.1799-1808, Dec. 1981.

[23] R. Srinivasan and K. Rao, "Predictive Coding Based on Efficient Motion Estimation," *IEEE Transactions on Communications*, vol. 33, pp. 888-896, Aug. 1985.

[24] M. Ghanbari, "The cross-search algorithm for motion estimation [image coding]," *IEEE Transactions on Communications*, vol. 38, pp. 950-953, Jul. 1990.

[25] S.-H. Han, S.-W. Kwon, T.-Y. Lee and M.-K. Lee, "Low power motion estimation algorithm based on temporal correlation and its architecture," *International Symposium on Signal Processing and its Applications*, vol. 2, pp. 647-650, Aug. 2001.

[26] J.-H. Lim and H.-W. Choi, "Adaptive motion estimation algorithm using spatial and temporal correlation," *IEEE Pacific Rim Conference on Communications, Computers and signal Processing*, vol. 2, pp. 473-476, Aug. 2001.

[27] J. Feng, T.Y. Liu, K.T. Lo and X.D. Zhang, "Adaptive motion tracking for fast block motion estimation," *IEEE International Symposium on Circuits and Systems*, vol. 5, pp. 219-222, 6-9 May 2001.

[28] C. de Vleeschouwer and T. Nilsson, "Motion estimation for low power video devices," *Proc. International Conference on Image Processing*, vol. 2, pp. 953-956, Oct. 2001.

[29] C. de Vleeschouwer, T. Nilsson, K. Denolf and J. Bormans, "Algorithmic and architectural co-design of a motion-estimation engine for low-power video devices," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, pp. 1093-1105, Dec. 2002.

[30] S. Mietens, P.H.N. de With and C. Hentsschel, "Flexible frame-reordering and multi-temporal motion estimation for scalable MPEG encoding in mobile consumer equipment," *International Conference on Consumer Electronics*, pp. 342-343, Jun. 2002.

[31] S. Mietens, G. Hekstra, P.H.N. de With and C. Hentsschel, "New flexible motion estimation technique for scalable MPEG encoding using display frame order and multi-temporal references," *Proc. International Conference on Image Processing*, vol. 1, pp. 701-704, Sep. 2002.

[32] F.G.B. de Natale, F. Granelli and G. Vernazza, "Low-complexity motion estimation for VLBR video coders," *Proc. International Conference on Image Processing*, vol. 1, pp. 685-688, Sep. 2002.

[33] D. Alfonso, F. Rovati, D. Pau and L. Celetto, "An innovative, programmable architecture for ultra-low power motion estimation in reduced memory MPEG-4 encoder," *IEEE Transactions on Consumer Electronics*, vol. 48, pp. 702-708, Aug. 2002.

[34] J.-B. Xu, L.-M. Po and C.-K. Cheung, "A new prediction model search algorithm for fast block motion estimation," *Proc. International Conference on Image Processing*, vol. 3, pp. 610-613, Oct. 1997.

[35] J.-Y. Nam, J.-S. Seo, J.-S. Kwak, M.-H. Lee and Y. H. Ha, "The adaptive predicted direction search algorithm (APDSA) New fast-search algorithm for block matching motion estimation using temporal and spatial correlation of motion vector," *IEEE Transactions on Consumer Electronics*, vol. 46, pp. 934-942, Nov. 2000.

[36] M. Brunig and W. Niehsen, "Fast full-search block matching," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, pp. 241-247, Feb. 2001.

[37] J.-N. Kim, S.-C. Byun and B.-H. Ahn, "Fast full search motion estimation algorithm using various matching scans in video coding," *IEEE Transactions on Systems, Man and Cybernetic*, vol. 31, pp. 540-548, Nov. 2001.

[38] Y. Naito, T. Miyazaki and I. Kuroda, "A fast full-search motion estimation method for programmable processors with a multiply-accumulator," *Proc. IEEE International Conference on Acoustics*, Speech, and Signal Processing, vol. 6, pp. 3221-3224, May 1996.

[39] Y.-C. Lin and S.-C. Tai, "Fast full-search block-matching algorithm for motion-compensated video compression," *IEEE Transactions on Communications*, vol. 45, pp. 527-531, May 1997.

[40] V.L. Do and K.Y. Yun, "A low-power VLSI architecture for full-search block-matching motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, pp. 393-398, Aug. 1998.

[41] C.-H. Cheung and L.-M. Po, "A novel cross-diamond search algorithm for fast block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, pp. 1168-1177, Dec. 2002.

[42] C.-H. Cheung and L.-M. Po, "A novel small-cross-diamond search algorithm for fast video coding and videoconferencing applications," *Proc. International Conference on Image Processing*, vol. 1, pp. 681-684, Sep. 2002.

[43] Y. Nie and K.-K. Ma, "Adaptive rood pattern search for fast block-matching motion estimation," *IEEE Transactions on Image Processing*, vol. 11, pp. 1442-1449, Dec. 2002.

[44] K.-K. Ma and G. Qiu, "An improved adaptive rood pattern search for fast block-matching motion estimation in JVT/H.26L," *Proc. International Symposium on Circuits and Systems*, vol. 2, pp. 708-711, May 2003.

[45] C. Zhu, X. Lin, L.-P. Chau, K.-P. Lim, H.-A. Ang and C.-Y. Ong, "A novel hexagon-based search algorithm for fast block motion estimation," *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 3, pp. 1593-1596, May 2001.

[46] S.-B. Hong, H.-S. Lee, G.-Y. Chun, H. Baik and M. Par, "FCBHS: a fast center-biased hybrid search algorithm for fast block motion estimation," *Proc. International Conference on Information Technology: Coding and Computing*, pp. 254-259, Apr. 2002.

[47] A. Ahmed, S.K. Nandy and P. Sathya, "Content adaptive motion estimation for mobile video encoders," *IEEE International Symposium on Circuits and Systems*, vol. 2, pp. 237-240, May 2001.

[48] P.I. Hosur, "Motion adaptive search for fast motion estimation," *IEEE International Conference on Consumer Electronics*, pp. 172-173, Jun. 2003.

[49] P. L. Tai, C. T. Liu and J. S. Wang, "Complexity-adaptive search algorithm for block motion estimation," *Proc. International Conference on Image Processing*, vol. 2, pp. 969-972, Oct. 2001.

[50] J. Xin, M.-T. Sun and V. Hsu, "Diversity-based fast block motion estimation," *Proc. International Conference on Multimedia and Expo*, vol. 3, pp. 525-528, Jul. 2003.

[51] M.E. Al-Mualla, C.N. Canagarajah and D.R. Bull, "Simplex minimization for single- and multiple-reference motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, pp. 1209-1220, Dec. 2001.

[52] S.-M. Jung, S.-C. Shin, H. Baik and M.-S. Park, "Efficient multilevel successive elimination algorithms for block matching motion estimation," *Proc. Vision, Image and Signal Processing*, vol. 149, pp. 73 -84, Apr. 2002.

[53] C.-H. Cheung and L.-M. Po, "A novel block motion estimation algorithm with controllable quality and searching speed," *IEEE Transactions on Circuits and Systems*, vol. 2, pp. 496-499, May 2002.

[54] C.-H. Cheung and L.-M. Po, "Adjustable partial distortion search algorithm for fast block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, pp. 100-110, Jan. 2003.

Appendix A

Table A- 1 Relation between SAD and MV for the 26th frame in news.qcif. Average SAD and standard deviation are 1065 and 724, respectively

x	y	SAD	MV
6	2	4608	1 , 0
6	3	3788	1 , 1
4	3	2757	-1 , 0
6	4	2744	1 , 0
6	1	2644	1 , 0
7	2	2109	0 , 0
8	5	1881	0 , 0
2	6	1842	0 , 0
2	5	1831	0 , 0
8	3	1828	0 , 0
3	4	1813	0 , 0
8	4	1786	0 , 0
2	4	1780	0 , 0
2	7	1742	0 , 0
7	4	1716	0 , 0
4	4	1713	2 , 0
9	8	1602	0 , 0
7	1	1548	0 , 0
1	7	1479	0 , 0
4	2	1467	0 , 0
3	3	1407	0 , 0
8	6	1406	0 , 0
7	8	1343	0 , 0
9	5	1319	0 , 0

2	8	1304	0 , 0
10	2	1302	0 , 0
7	5	1283	0 , 0
7	3	1270	0 , -1
6	6	1244	0 , 0
9	6	1238	0 , 0
9	4	1217	0 , 0
9	7	1213	0 , 0
9	2	1211	0 , 0
0	7	1207	0 , 0
3	7	1188	0 , 0
10	8	1182	0 , 0
10	6	1160	0 , 0
10	7	1151	0 , 0
2	3	1127	0 , 0
3	5	1123	0 , 0
0	6	1087	0 , 0
5	6	1079	0 , 0
5	4	1074	0 , 0
3	1	1061	0 , 0
5	1	1053	0 , 0
4	1	1012	0 , 0
8	7	1004	0 , 0
3	2	961	0 , 0
8	8	944	0 , 0
5	3	942	-1 , 0
7	6	938	0 , 0
0	5	936	0 , 0
10	5	900	0 , 0
7	7	897	0 , 0

9	3	888	0 , 0
6	5	880	0 , 0
3	6	852	0 , 0
3	8	832	0 , 0
10	4	830	0 , 0
1	8	815	0 , 0
1	5	780	0 , 0
6	8	776	0 , 0
10	3	760	0 , 0
0	4	748	0 , 0
5	2	741	0 , 0
1	4	737	0 , 0
5	7	733	0 , 0
6	7	696	0 , 0
10	0	665	0 , 0
1	6	664	0 , 0
4	6	662	0 , 0
0	8	596	0 , 0
5	8	572	0 , 0
5	0	556	0 , 0
4	5	554	0 , 0
4	0	551	0 , 0
0	0	532	0 , 0
5	5	509	0 , 0
2	1	505	0 , 0
9	0	503	0 , 0
6	0	492	0 , 0
2	2	491	0 , 0
8	0	482	0 , 0
2	0	474	0 , 0

3	0	462	0 , 0
1	3	409	0 , 0
0	3	408	0 , 0
1	0	407	0 , 0
7	0	394	0 , 0
8	1	381	0 , 0
4	7	354	0 , 0
4	8	337	0 , 0
8	2	317	0 , 0
10	1	239	0 , 0
0	2	116	0 , 0
1	1	92	0 , 0
1	2	79	0 , 0
9	1	62	0 , 0
0	1	52	0 , 0

Table A- 2 Relation between SAD and MV for the 52th frame in silent.qcif. Average SAD and standard deviation are 1384 and 1024, respectively

x	y	SAD	MV
3	4	6695	2 , -7
4	3	5724	2 , 1
3	3	5503	-1 , -3
4	4	4778	-1 , -7
2	4	3691	1 , -2
4	5	2748	-1 , -2
3	5	2482	0 , -2
5	5	2082	0 , 0
6	2	1985	0 , 0
3	6	1896	0 , 0
2	2	1852	0 , 0
1	6	1841	0 , 0
2	3	1806	0 , 0
6	1	1759	0 , 0
5	4	1742	0 , 0
6	3	1706	0 , 0
2	5	1674	0 , 0
1	4	1610	0 , 0
2	6	1604	0 , 0
5	1	1562	0 , 0
5	2	1541	0 , 0
8	2	1493	0 , 0
0	5	1472	0 , 0
6	4	1467	0 , 0
9	3	1444	0 , 0
1	3	1431	0 , 0

1	2	1430	0 , 0
0	3	1400	0 , 0
3	2	1384	0 , 0
0	2	1365	0 , 0
9	6	1358	0 , 0
10	6	1358	0 , 0
0	4	1345	0 , 0
2	1	1343	0 , 0
4	1	1340	0 , 0
7	5	1340	0 , 0
1	1	1330	0 , 0
3	1	1322	0 , 0
5	3	1319	0 , 0
10	4	1310	0 , 0
6	5	1293	0 , 0
10	3	1260	0 , 0
10	2	1259	0 , 0
7	2	1248	0 , 0
9	4	1244	0 , 0
9	5	1223	0 , 0
2	8	1214	-3 , 0
5	0	1207	0 , 0
7	1	1193	0 , 0
10	5	1193	0 , 0
7	8	1193	0 , 0
3	8	1175	0 , 0
3	7	1166	0 , 0
7	4	1141	0 , 0
4	2	1140	0 , 0
8	3	1131	0 , 0

10	1	1130	0 , 0
4	0	1123	0 , 0
0	6	1110	0 , 0
8	4	1107	0 , 0
10	7	1105	0 , 0
0	1	1095	0 , 0
8	6	1080	0 , 0
8	1	1070	0 , 0
2	7	1054	0 , 0
9	1	1039	0 , 0
9	2	1030	0 , 0
7	7	1024	0 , 0
7	3	1007	0 , 0
8	8	1006	0 , 0
8	7	994	0 , 0
6	8	978	0 , 0
0	7	894	0 , 0
9	7	889	0 , 0
6	6	871	0 , 0
7	6	868	0 , 0
8	5	825	0 , 0
10	0	806	0 , 0
5	8	800	0 , 0
10	8	797	0 , 0
1	7	794	0 , 0
6	0	792	0 , 0
4	8	789	0 , 0
6	7	780	0 , 0
1	0	775	0 , 0
2	0	679	0 , 0

1	5	669	0 , 0
8	0	631	0 , 0
9	0	622	0 , 0
7	0	593	0 , 0
0	0	584	0 , 0
0	8	578	0 , 0
1	8	553	0 , 0
5	6	523	0 , 0
4	7	472	0 , 0
9	8	445	0 , 0
5	7	436	0 , 0
3	0	433	0 , 0
4	6	400	0 , 0

Appendix B

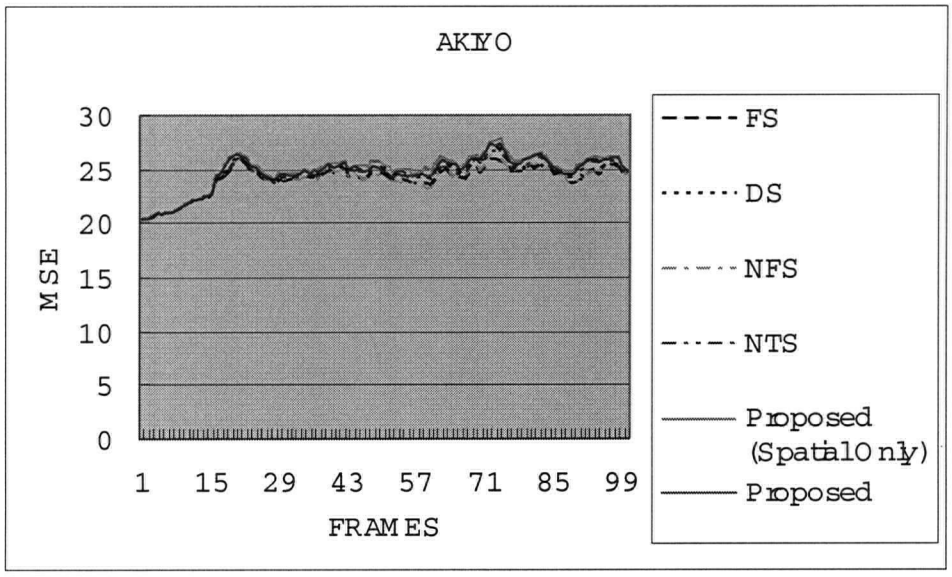


Figure B-1 MSE values of encoded frame of akiyo.qcif

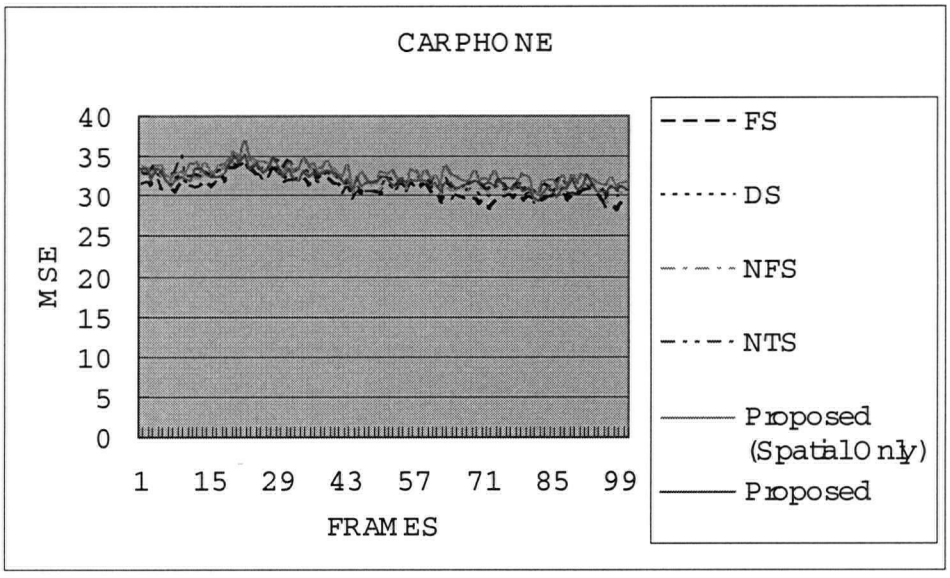


Figure B-2 MSE values of encoded frame of carphone.qcif

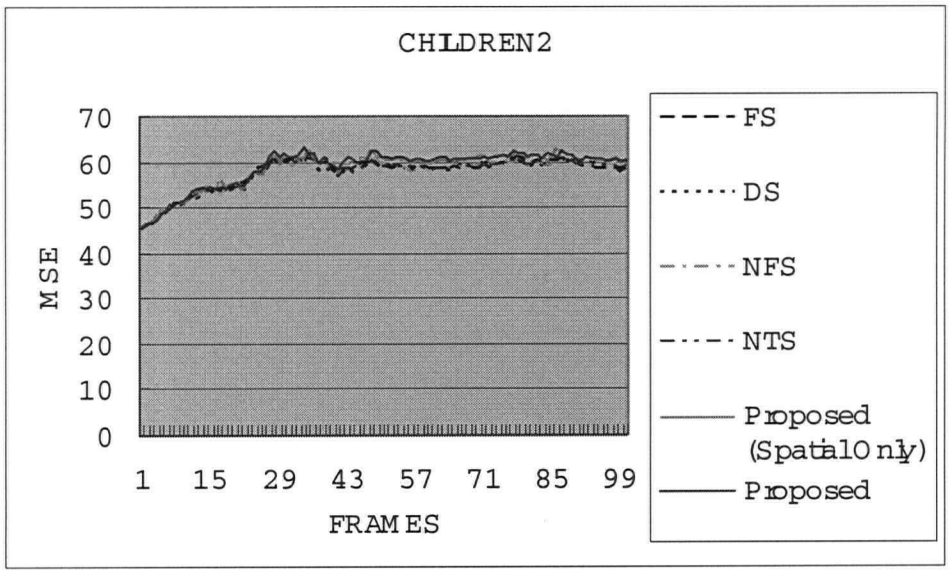


Figure B-3 MSE values of encoded frame of children.qcif

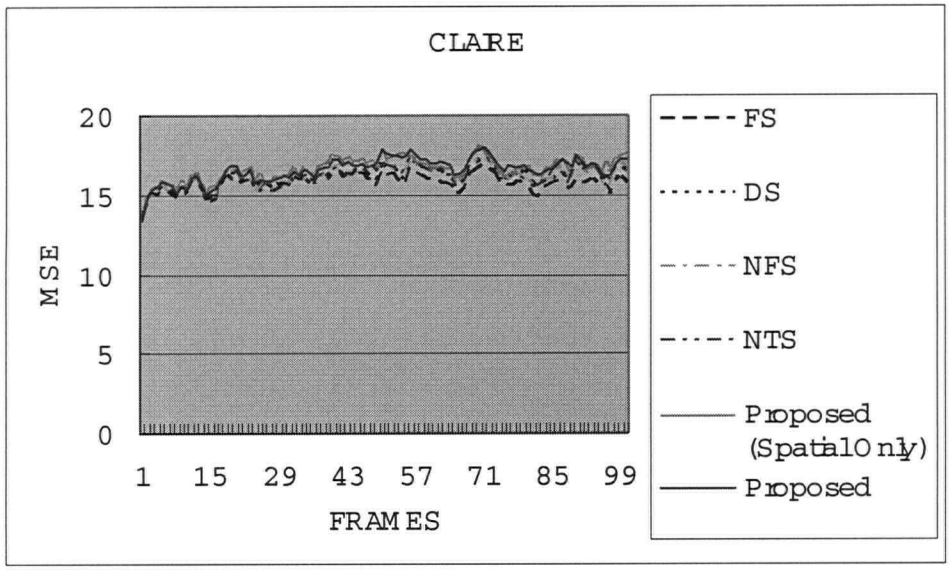


Figure B-4 MSE values of encoded frame of claire.qcif

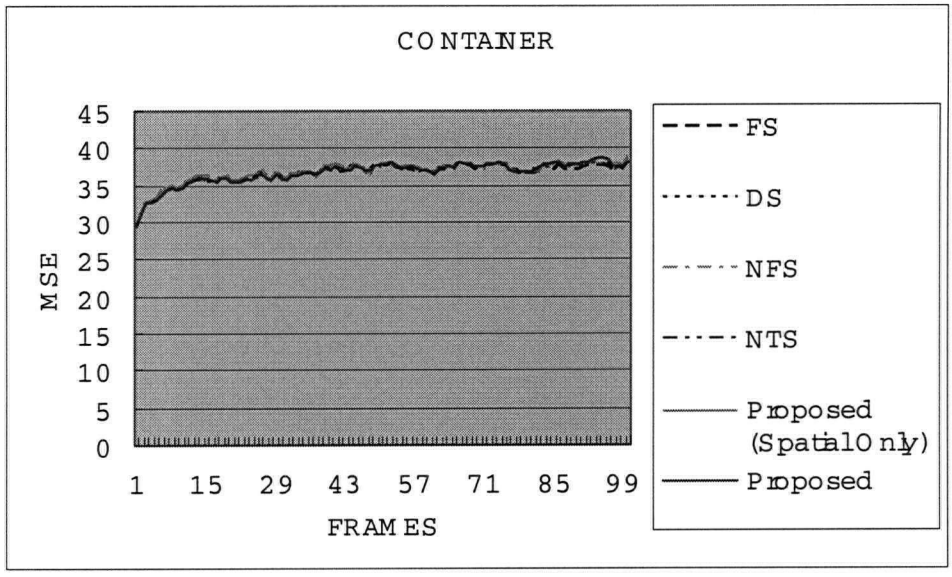


Figure B-5 MSE values of encoded frame of container.qcif

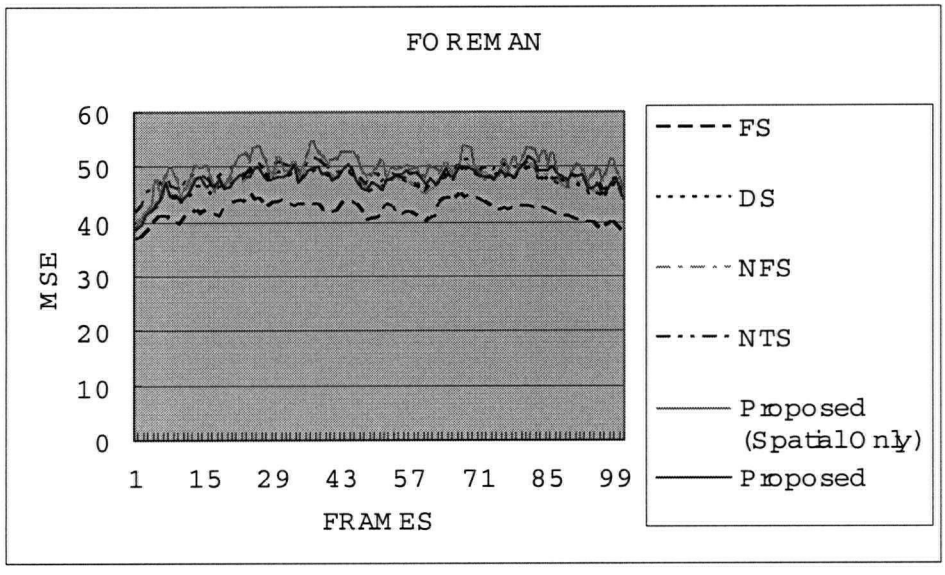


Figure B-6 MSE values of encoded frame of foreman.qcif

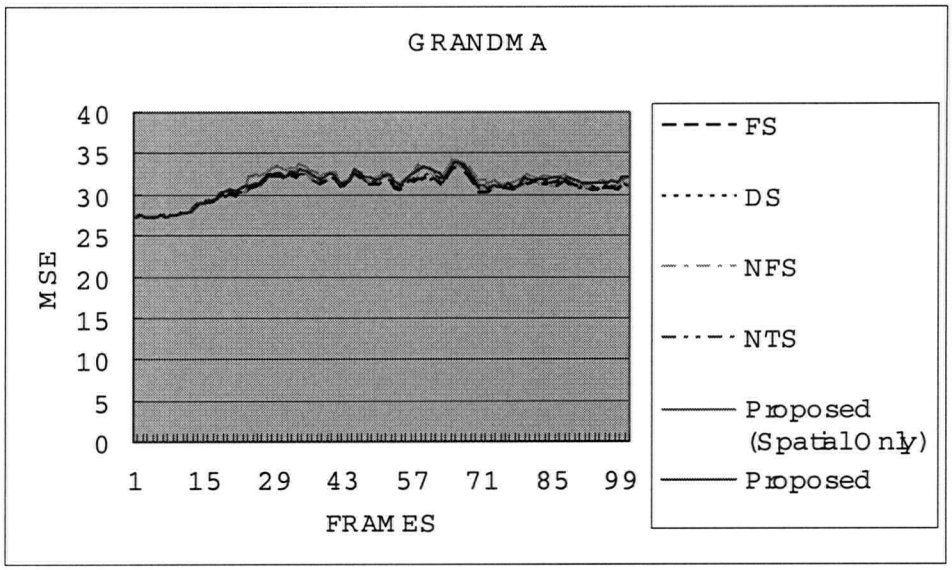


Figure B-7 MSE values of encoded frame of grandma.qcif

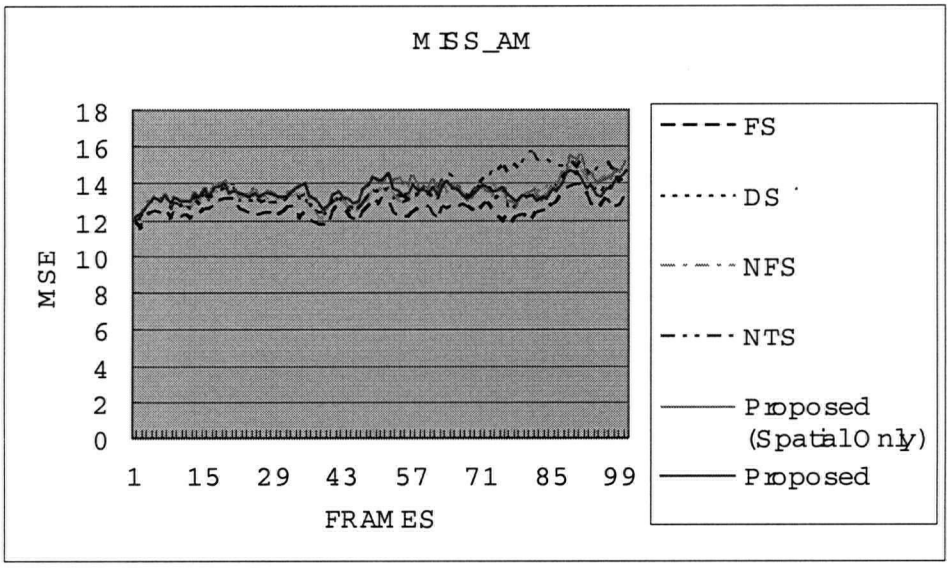


Figure B-8 MSE values of encoded frame of miss_america.qcif

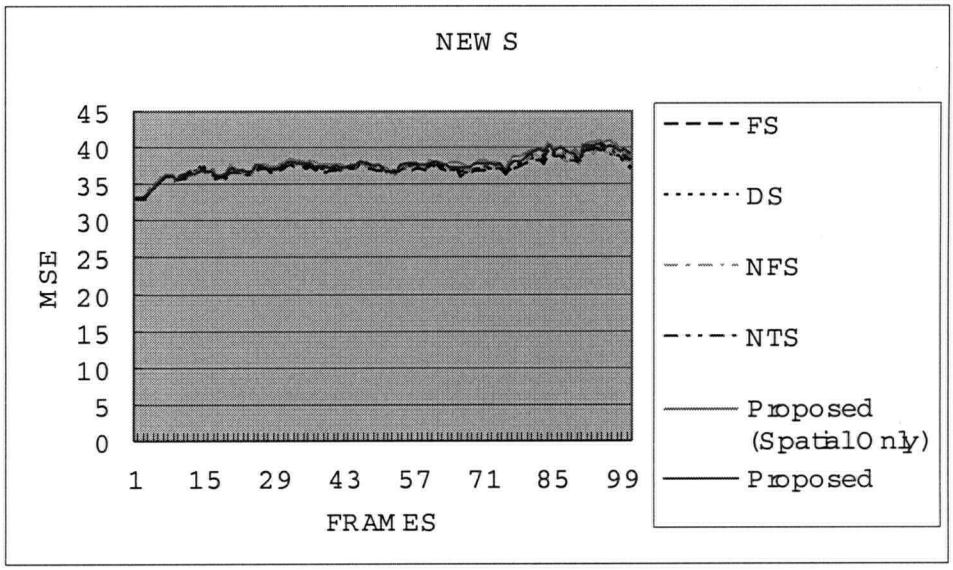


Figure B-9 MSE values of encoded frame of news.qcif

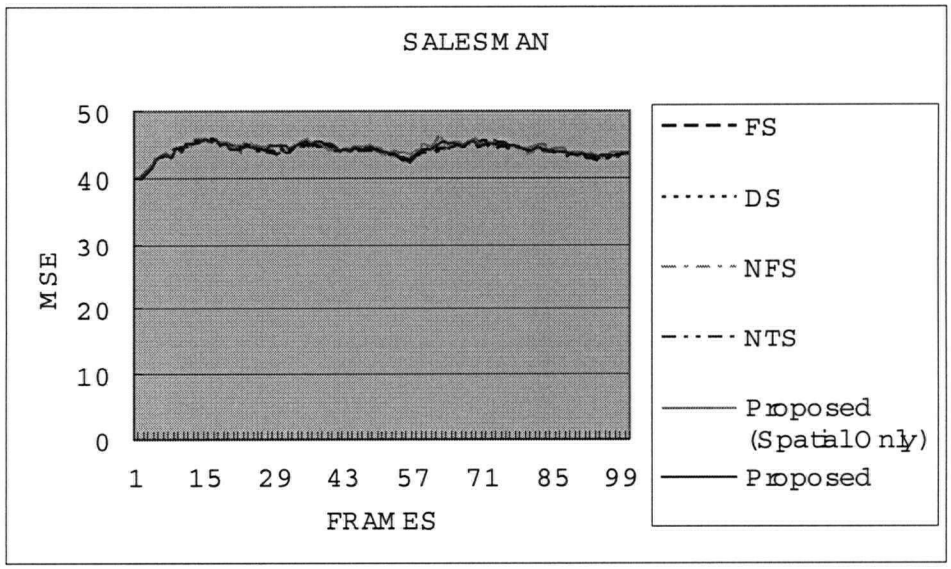


Figure B-10 MSE values of encoded frame of salesman.qcif

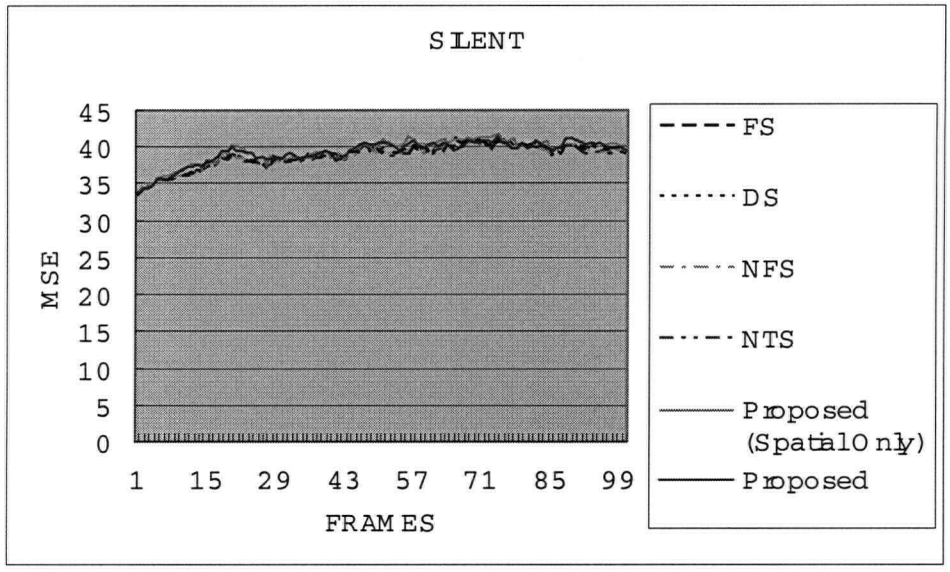


Figure B-11 values of encoded frame of silent.qcif

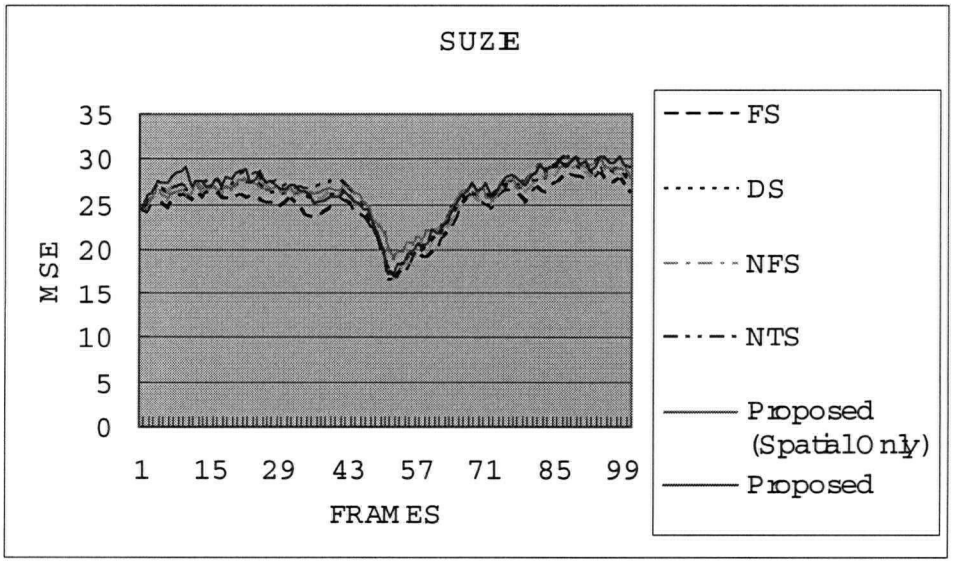


Figure B-12 MSE values of encoded frame of suzie.qcif

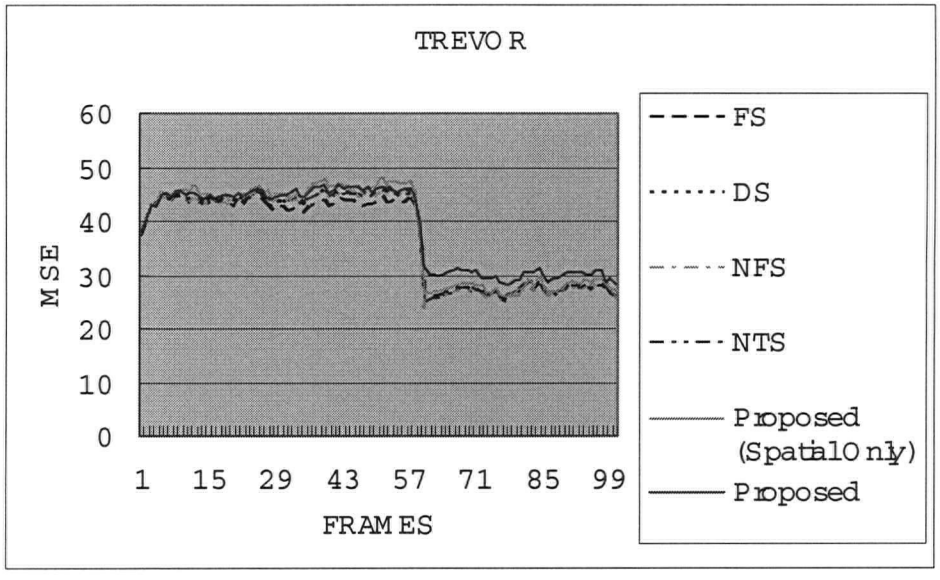


Figure B-13 MSE values of encoded frame of trevor.qcif