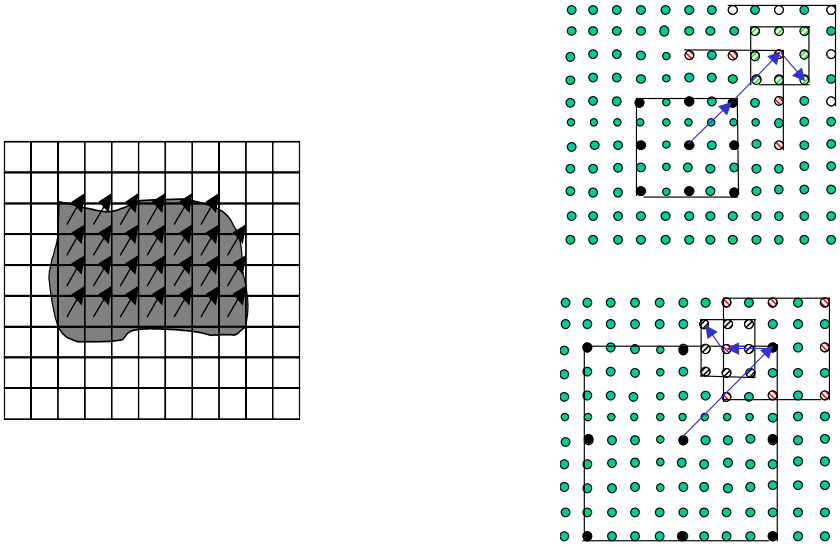
Carnegie Mellon

Electrical & Computer ENGINEERING

Estimation and Mode Decision for Spatially Correlated Motion Sequences

Deepak S. Turaga and Tsuhan Chen
Advanced Multimedia Processing Lab



Technical Report AMP 01-01
February 2001

Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

There are several parts of the video encoding process that can be optimized for high quality, high compression ratio and fast encoding speed. In this paper we focus on some of these encoder optimization issues. We first study motion estimation, a computationally intensive part of encoding, and propose efficient algorithms based on exploiting spatially correlated motion, that perform well in terms of speed, quality and the bit rate for different kinds of motion sequences as well as picture formats. We extend our motion estimation algorithms to find motion vectors for sub-parts of a block, as allowed for in the H.263 and MPEG-4 standards. We study the one-four motion vector decision and propose an efficient two-tier decision strategy. We then study the error measurement or the measure of similarity that is an important parameter for motion estimation as well as the mode decision. We introduce a new error measurement that can improve the mode decision significantly. Furthermore for certain motion estimation strategies, this new measurement can also improve the bit rate resulting from a better motion estimation. Another issue that we address in this paper is the estimation of delta motion vectors in the H.263 and the MPEG-4 standards. We have implemented these optimizations in a H.263 framework and the results are encouraging.

I. Introduction

Standard compliant video coding involves converting the raw video data into a standard specified bitstream. There are many requirements of a good video coder. It should achieve a good compression ratio, leading to an efficient bitstream and should do so without sacrificing video quality. It should also be fast and easy to implement. Video coding standards specify the syntax of the bitstream, but allow for optimizing the encoding process to achieve one or more of the above goals, specifically targeted towards different applications. In this paper we focus on optimizations of parts of the video coding process. In particular, we focus on block-based coding schemes. In these schemes, the pictures are sub-divided into smaller units called blocks that are processed one by one. In this paper we use the term block to represent a unit of 16 pixels by 16 pixels. These are also referred to as macroblocks in video coding standards. Standards such as the H.26x series and MPEG series use block-based strategies. Some more information about these standards may be obtained from [1,2]. In these schemes each frame of the video sequence is

Corresponding Author: Tsuhan Chen, Electrical and Computer Engineering, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213. Email: tsuhan@cmu.edu.

Accepted for publication in *IEEE Transactions on Circuits and Systems for Video Technology*, 2001.

decomposed into non-overlapping blocks and each of these is treated as a unit for coding purposes.

Motion estimation is part of the process to remove temporal redundancies and is a computation intensive operation in the encoding process [3]. Block based schemes assume that each block of the current frame is obtained from the translation of some corresponding region in a reference frame. Motion estimation tries to identify this best matching region in the reference frame for every block in the current frame. An example of motion estimation is shown in Figure 1.

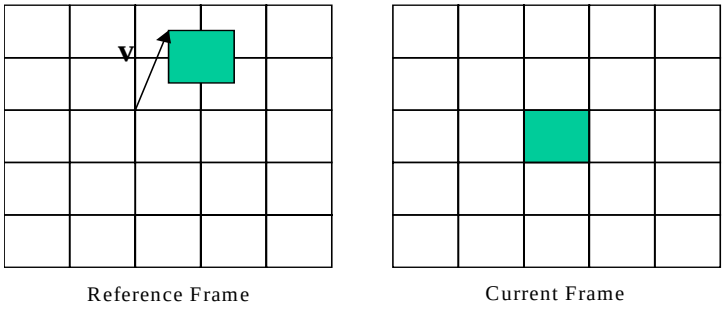


Figure 1. Illustration of Motion Estimation

In the figure, the gray block on the right corresponds to the current block being coded and the gray area on the left represents the best match found for the current block, in the previous frame. The displacement is called the motion vector. Typically, motion vectors are restricted to within a range $\mathbf{v} = (v_x, v_y)$. The search range specified by the baseline H.263 standard allows motion vectors to range between –15 pixels and 16 pixels in either dimension. This is also equivalent to saying that we have a search window of size 32×32 about the search center. The residue between the best match and the current block and the motion vector are then coded and transmitted. A good motion estimation algorithm has to perform well in terms of three parameters. These form what we call the "speed-quality-bit rate" tradeoff, which we introduced in [4]. It should be as fast as possible and should use as little computation as possible. It should provide a good quality of the match, i.e. the best match should be as similar to the current block as possible. Finally it should also try to reduce the number of bits needed for coding. A large part of the bit rate comes from coding the residue, however coding the motion vectors also needs bits and most of the work in literature does not consider motion vector bits explicitly. Some previous work does explicitly consider the tradeoff between the bits needed to code the motion vectors and the distortion. Notably, the work by Chen and Willson [5] involves finding the best motion vector using a cost function consisting of the distortion and the rate to code the motion vectors. They generate a

trellis of candidate motion vectors using predictors derived from previously coded blocks and use the Viterbi algorithm to identify the best motion vectors. The work by Chen and Kung [6] also considers the rate explicitly during motion estimation. They use a neighborhood relaxation scheme to obtain a piecewise continuous motion field. Both of these papers try to obtain a set of motion vectors optimal in the rate-distortion sense using a cost function that is a combination of the rate and the distortion and using different search strategies that exploit the spatially correlated motion information. They are, however not very efficient in terms of computational complexity and their performance suffers when motion is large. The objective of the work in this paper is to develop an efficient motion estimation algorithm that can provide good rate and distortion performance. We do not explicitly consider the effect of rate during motion estimation, however by using the spatially correlated motion information we can identify a homogeneous motion field that leads to the better rate performance.

The motion estimation algorithms we propose in this paper combine features from existing motion estimation schemes to perform well in terms of all three parameters. There are different kinds of motion estimation algorithms that exist in literature. In this paper we use search strategies based on gradient-descent methods. Among these strategies some algorithms are center-biased i.e. they are more efficient when the best match is close to the center of the search and some are non-center-biased, i.e. they are more efficient when the best match is far from the center. We use spatially correlated motion information to adaptively choose between a center-biased and a non-center-biased search algorithm to combine their respective advantages. We then extend our work by using the correlated motion information to also choose a starting point for these search strategies. This extension is useful in exploiting the correlated motion information more effectively when there is a large amount of correlated motion in the scene. Previous work on using spatially correlated motion information to enhance motion estimation has also been done by Gallant et al [7]. They use information from the motion vectors of the neighbors to predict a starting point for their search strategy. Similar work has also been done by Xu et al [8], only they use a different set of neighbors to choose the search center. In addition to using spatially correlated motion information to predict the starting point of the search, our work also uses this information to choose between different search strategies. This choice of search strategies is useful when the prediction of the search center is erroneous, as described later in this paper.

Some other work has been done in [9-12]. Weng et al [9] model 2-D motion correlation using two 1-D models and the prediction for the motion vector of the current block is obtained through Kalman filters and linearly combining the two 1-D estimates. References [10-12] contain hierarchical motion estimation algorithms. Lim and Ra [10] obtain the motion vector for a large

block size and use that as an initial estimate for the motion vector for the smaller blocks contained in the large block while Lin and Wu [11] use a multiresolution approach to reduce the search space. Chalidabhongse and Kuo [12] combine the multiresolution scheme with spatio-temporal correlations and use candidates for motion vectors from temporally and spatially neighboring blocks as well as coarser level blocks. Our work differs from these strategies in that spatial correlation is used explicitly to choose the search strategy, besides the starting point of the search.

We then look at one of the advanced optional coding modes in both the H.263 and MPEG series standards that allows for the use of four motion vectors for a block. The use of four motion vectors can lead to a better quality of matches, thereby reducing the bit rate needed for the residue. However it adds computation complexity besides adding an overhead in terms of sending four motion vectors as against one. We extend our spatial correlation based algorithms to estimate these motion vectors. In addition, a decision needs to be made whether the improvement in the quality due to the four motion vectors warrants this extra computation and the overhead of sending these extra motion vectors. Some enhancements in this decision process are also investigated.

One of the requirements of motion estimation is block matching which needs a measure of similarity between two blocks. Traditionally the Mean Absolute Difference (MAD) has been used for this purpose. The MAD between two $N \times N$ blocks is the mean of the absolute pixel difference between the two blocks and is defined as $MAD = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left|x_{ij} - y_{ij}\right|$. This measure however does not always relate monotonically with the number of bits required to code the residue. This means that even if motion estimation finds the best match in terms of the MAD, it is not guaranteed that the residue from this best match requires the fewest bits for coding. As we want to finally reduce the number of bits required for coding, we need a better measure of similarity. We introduce a new measure called the mean-removed mean absolute difference (mrMAD) and implement some motion estimation strategies, using genetic algorithms, that show that using the mrMAD as a measure of similarity does indeed lead to greater savings in bit rate. This new measure also leads to a better mode decision between Intra and Inter coding. Intra coding involves coding the block itself while Inter coding involves coding the residue between the block and the best match and the motion vector. The mode decision is useful when good matches for the block cannot be found in the previous frame, for instance in cases with out of plane motion, or scene changes. In such cases it costs more bits to code the residue as against the block itself. A good mode decision tries to use

the best form of coding (Intra vs. Inter) for every block to minimize the bits needed for coding. We will show that the mrMAD, introduced here, can be used to obtain a good mode decision.

We also examine PB frames, another advanced optional mode in the H.263 standard. We propose the use of a center-biased algorithm for the estimation of Delta motion vectors and show the improvements in speed over using the full search strategy. Delta motion vectors are also used similarly in the bi-directional prediction mode for B frames as specified in the MPEG-4 standard. Our work on the estimation of Delta motion vectors may also be used to estimate these.

This paper is organized as follows. Section II describes our proposed motion estimation algorithms and includes a discussion of their performance. Section III talks about the one-four motion vector decision strategy. Section IV describes the mrMAD as a new measure of similarity and how it leads to good motion estimation and improved mode decision. Section V details the estimation of delta motion vectors. Concluding remarks are in Section VI.

II. Motion Estimation Algorithms

As we have mentioned before, motion estimation involves finding the best matching region for every block in the current frame, from a reference frame. It is a very important part of the encoding process and affects its performance greatly. Many search strategies have been developed to find the best match.

The first of these strategies is the full search. This strategy examines every region in a specified search area in the reference frame. Clearly, it is optimal in terms of finding the best match, but it is computationally very expensive. As a result, several sub-optimal search algorithms that are not as computationally expensive have been developed. These assume that the best match is likely to be found in the vicinity of reasonably good matches and proceed to search in direction of locally better matches. Alternately this may be viewed as moving in the direction of decreasing local error to reach the global minimum on the error surface. Such algorithms are likely to get trapped in local minima and hence their performance depends on the choice of the starting point as well as the actual error surface. Among these algorithms are the three step search (TSS) [13] and the four step search (FSS) [14].

The TSS is a coarse to fine search. The starting step-size for the search is large, typically half the search range, and at every stage the center of the search is moved in the direction of the best match at that stage and the step size is reduced by half. In contrast, the FSS starts with a fine step size, typically 2, and the center of the search is moved in the direction of the best match without changing the step-size, until the best match at that stage is at the center. The step-size is then

halved to 1 to find the best match. Some sample convergence paths for the TSS and the FSS are shown in the following figures.

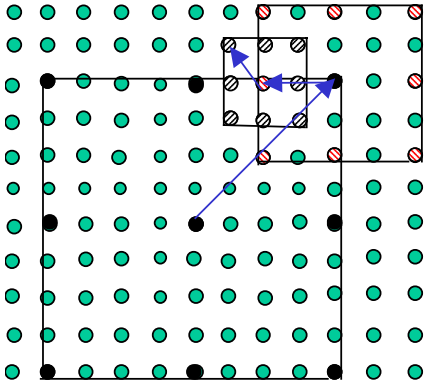


Figure 2. Sample convergence path for the TSS

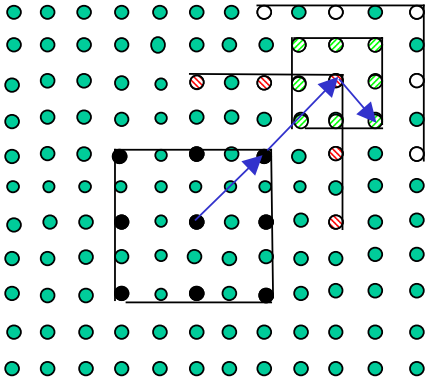


Figure 3. Sample convergence path for the FSS

From the two preceding figures we can see that the FSS is more center-biased, i.e., it converges more rapidly when the best match is close to the center of the search. On the other hand the TSS performs better when the best match is located far from the center of the search.

Most video sequences have spatially homogeneous frames with a lot of spatially correlated information in different parts. This information can be used by motion estimation algorithms to perform well in terms of the “speed-quality-bit rate” tradeoff. For instance many objects in the scene are composed of more than one block. So when the object moves a set of blocks moves together. An example of this is shown in Figure 4.

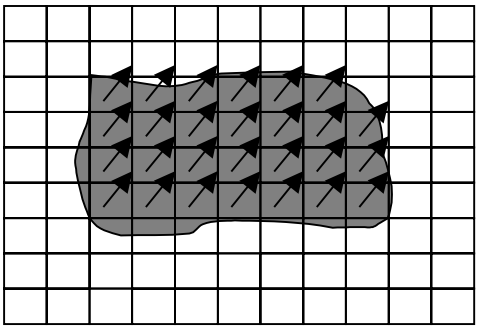


Figure 4. Example of correlated motion

As can be seen from Figure 4 most blocks are either completely in the background or part of the moving object and in most cases (except boundary blocks) neighboring blocks move together.

Given that a majority of the neighbors of a block move in a certain direction, it is reasonable to assume that the block also moves in the same direction. Similarly, the magnitude of the motion can also be predicted from neighbors. Both algorithms introduced in this paper use this spatial correlation information. The use of this information helps in reducing the search space, thereby increasing the speed of convergence. It can also help in obtaining a better match when coupled with a sub-optimal search strategy by providing a better starting point for the search. Bit rate is coupled with the quality of the match, the better the match, the smaller the bits needed for coding the residue. Another component of the bit rate is the bits needed for coding motion vectors. Motion vectors are differentially coded to reduce the bits needed for coding them. In the MPEG-1 and 2 these are differentially coded from the previous motion vector, while in H.263 and MPEG-4 these are differentially coded from the median of three neighboring blocks. Hence a homogeneous set of motion vectors is more likely to require fewer bits to code than randomly oriented motion vectors. Use of spatial correlation information to direct the search is likely to produce a homogeneous motion field. Hence, this leads to a reduction of bit rate in two ways.

The algorithms introduced in this section combine the advantages of the TSS and the FSS by adding spatial correlation information. As the neighbors for considering the spatial correlation information, we chose blocks to the left and above the current block as shown in Figure 5.

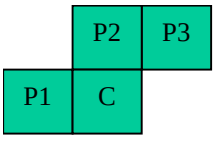


Figure 5. Neighbors for correlation information

The neighboring blocks P1, P2 and P3 are also called predictors for the current block C. We use the term block to refer to each unit in motion estimation and compensation. This is sometimes called a macroblock in the standards. This choice of neighbors for spatial correlation information also reduces the bit rate for the motion vectors as the same set of neighbors is used, in H.263 and MPEG 4, for their predictive coding. When the block in consideration is near the picture boundary or near a Group of Blocks boundary, not all predictors are available and in such cases, we use only the available predictors, which is consistent with the H.263/MPEG-4 standards.

A. Majority Voting Algorithm (MVA)

Each of the predictor blocks votes for either the TSS or the FSS based on whether its motion vector is small or large and the algorithm with the majority of votes is chosen for the search. Predictors with small motion vectors vote for the FSS and those with large motion vectors vote for the TSS. The assumption is that if the majority of the predictors have small motion vectors, it is likely that the current block also has a small motion vector, in which case a more center biased algorithm like the FSS is a better choice. A similar argument holds when the majority of motion vectors are large. Hence in order to choose between the two different kinds of algorithms we need to classify motion vectors as large and small. In order to do this we performed a best case analysis for the FSS and the TSS, similar to the work done in [15]. This analysis involves computing the smallest number of comparisons each algorithm would take to get to a particular position. It assumes that we have an error surface that is smooth, unimodal and isotropic. Under these assumptions these gradient descent strategies will require the smallest number of comparisons to reach any desired point. We have performed this analysis and we show an example assuming that the search window size is fixed at 16×16. These numbers are shown in Figure 6 with the center square corresponding to the search center.

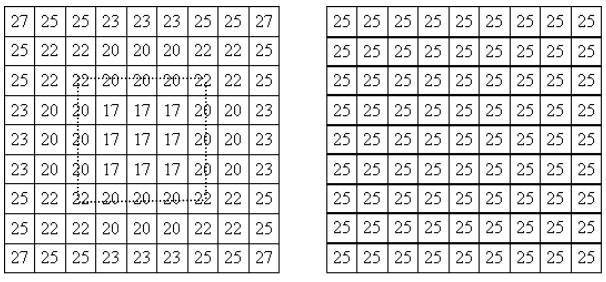


Figure 6. Best case analysis for the FSS and the TSS

The figure shows the comparison only for motion vectors within the range $|v_x| \, and \, |v_y| \le 4$. The best case number of comparisons for the FSS always goes up with the distance from the center and this is independent of the search window size. Also, we can see that if we look along the edges of a square centered around the search center (e.g. the dotted square shown in the figure for the FSS), the points at the corners of the square require the largest number of comparisons, in the best case. For these corner points the number of comparisons in the best case may be written as follows. $Best\ case\ number\ of\ comparisons = \left\lfloor \frac{a \times 5}{2} \right\rfloor + 17;\ |v_x| = |v_y| = a$.

In contrast, the number of comparisons for the TSS depends only on the search window size and remains the same irrespective of the distance from the center, for a given search window size. This number is $(\log_2 N) \times 8 + 1$ where N is one side of the search window. This number of comparisons for the TSS changes only when the search range changes. Using these facts, we can determine when we should use the FSS and when the TSS. As an example, for a 16×16 search window the TSS requires 25 comparisons for any point, while the FSS requires less than this, in the best case, for all points inside the square with a = 3. This means that we should use the FSS when both v_x and v_y are smaller than 4 in magnitude and the TSS otherwise, if the FSS always requires the best-case number of comparisons. Similarly for a 32×32 search window, which is what we use, the TSS requires 33 comparisons for every point, while the FSS requires more than this only when $a \ge 8$. So in the best case we should use the FSS when $|v_x| \, and \, |v_y| < 8$ and the TSS otherwise. However during experiments we realize that in general the FSS requires more than these best case number of comparisons to converge. This is because the best case analysis assumes that the error surface is unimodal and isotropic. This is not the case in practice and the FSS usually requires a larger number of comparisons to reach any point. Hence the threshold can be reduced and we determine empirically that we should use the FSS when $|v_x| \, and \, |v_y| \le 4$ and the TSS outside this range. This definition is used throughout this paper to distinguish between small and large motion vectors.

B. Extended Majority Voting Algorithm (EMV)

The previous algorithm uses spatial correlation information to estimate the magnitude of the motion of the current block and hence choose between the FSS and the TSS appropriately. Spatial correlation information can also be used to predict the direction of the motion of the current block. The use of this information effectively reduces the search space as well as helps in

choosing a better starting point for the search. The choice of a starting point is critical to both the TSS and the FSS as they can get trapped in local minima. The median of the motion vectors of the predictors is computed and the center of the search is moved to this value. The median operation is used to suppress the effect of significantly different motion vectors. Following this moving of the center of the search, the predictors also vote for either the TSS or the FSS as for the majority voting scheme. It can be argued that once the center of the search is moved to the predicted value, the best match is likely to be close to this new center and hence it is better to always use the center-biased FSS. However, this assumption is invalid when the current block lies across an object boundary and its motion is different from the predictors. In such cases the search strategy starts from a center far away from the best match where the TSS is better. So the choice between the FSS and the TSS allows for correcting any invalid predictions. The performance of the MVA and the EMV as compared with the TSS and the FSS is included in the following section.

C. Simulation Results

The motion estimation algorithms were tested in terms of the speed (as measured in CPU time), MAD as a measure of the quality of motion compensation and the bits needed for coding. The conversion to bits was done in compliance with the H.263/MPEG-4 standard. The tests were done on four different sequences. Each sequence is QCIF with 300 frames at 30Hz. The four sequences chosen for the tests are Akiyo, Hall, Coastguard and Foreman. Some sample frames from these sequences are shown in the following figures.

Figure 7. Akiyo (left) and Hall (right)

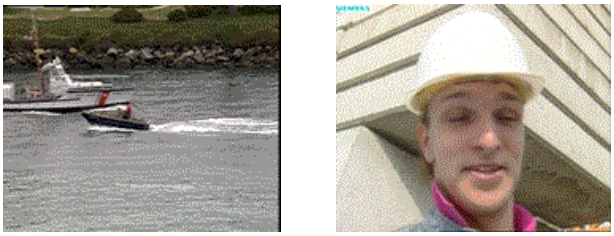


Figure 8. Coastguard (left) and Foreman (right)

Results for these sequences are included in the following table.

Table 1. Motion Estimation Results for QCIF sequences

Sequence	Algorithm	Quality (MAD)	Speed (CPU Time)	Bit Rate		Comments
				Residue bits	MVD bits	
Akiyo	FSS	0.469	7.75	4.08M	59925	Small motion sequence. All algorithms perform well.
	TSS	0.471	13.07	4.08M	59975	
	MVA	0.469	7.73	4.08M	59925	
	EMV	0.469	7.75	4.08M	59893	
Hall	FSS	1.74	8.02	4.48M	66920	Moderate and correlated motion. MVA and EMV perform better than TSS and FSS. EMV performs best.
	TSS	1.75	13.07	4.48M	67308	
	MVA	1.74	8.03	4.43M	66832	
	EMV	1.71	8.00	4.38M	66775	
Coastguard	FSS	3.62	8.35	5.82M	78664	Large and correlated motion. MVA and EMV perform better than FSS and TSS, performance worse than TSS in terms of bits for residue.
	TSS	3.55	13.52	5.73M	82753	
	MVA	3.57	8.17	5.79M	80426	
	EMV	3.55	8.02	5.75M	78541	
Foreman	FSS	3.85	8.40	5.90M	138388	Large and uncorrelated motion. MVA performs well, but EMV performance worse due to the uncorrelated motion.
	TSS	3.84	13.55	5.87M	152344	
	MVA	3.81	8.02	5.86M	145090	
	EMV	3.91	8.17	5.97M	157582	

We repeat these experiments with the CIF versions of the same sequences in order to illustrate the performance of our algorithms across different formats as well as different types of motion sequences. These results are included in Table 2.

Table 2. Motion Estimation Results for CIF sequences

Sequence	Algorithm	Quality (MAD)	Speed (CPU Time)	Bit Rate		Comments
				Residue bits	MVD bits	
Akiyo	FSS	0.5908	33.89	16.25M	247261	Small motion sequence. EMV and MVA perform better than both TSS and FSS.
	TSS	0.5930	52.16	16.26M	247614	
	MVA	0.5907	32.19	16.22M	246771	
	EMV	0.5907	32.36	16.22M	246782	
Hall	FSS	2.52	33.99	16.71M	367312	Moderate and correlated motion. MVA and EMV perform better than both TSS and FSS in speed and bits for MVD.
	TSS	2.52	50.37	16.72M	387273	
	MVA	2.52	33.21	16.71M	367429	
	EMV	2.52	32.91	16.71M	367379	
Coastguard	FSS	4.705	41.14	25.31M	347121	Large and correlated motion. MVA and EMV perform better than FSS and TSS.
	TSS	4.683	57.45	25.17M	394600	
	MVA	4.652	40.65	24.96M	341546	
	EMV	4.601	38.89	24.89M	340322	
Foreman	FSS	4.079	40.21	24.06M	740202	Large and uncorrelated motion. MVA performs well, but EMV performance worse due to the uncorrelated motion.
	TSS	3.942	61.62	23.86M	829883	
	MVA	3.861	40.56	23.56M	713510	
	EMV	3.947	41.23	23.86M	749309	

From the tables we can see that the FSS converges faster than the TSS for both the QCIF as well as the CIF sequences. However the relative difference in the speed (in terms of percentage) is smaller for the CIF sequences than for the QCIF sequences as the CIF sequences typically have larger motion (due to the scaling up). For low motion sequences, the performance of the FSS is also better in terms of the quality as well as the bits needed for coding the residue. However, for the sequences with large motion, i.e. Foreman and Coastguard, we can see that the FSS does not converge to very good matches, thereby leading to worse performance than the TSS in terms of the quality as well as the bits needed for coding the residue. Our proposed algorithms try to combine the best features of these two strategies so that they may be used across different motion sequences as well as picture formats.

The Akiyo sequence has very little motion and hence all the algorithms find the best match of around the same quality. The results for the QCIF sequence show that the MVA and the EMV perform better than both the FSS and the TSS in terms of the speed as well as the MVD bits. From the results for the CIF sequence, we can see that the MVA and the EMV perform better than both the TSS and the FSS in terms of bits for residue, MVD bits and speed. The performance of the EMV is better than that of the MVA as it exploits the spatially correlated motion information to a greater extent. The Hall sequence has moderate and correlated motion. Due to

this correlated motion, the MVA and the EMV perform better than both the TSS and the FSS in terms of the quality, the speed as well as the bits for residue and MVD bits. As before, the EMV performs better than the MVA for both the QCIF as well as the CIF sequence. The Coastguard sequence has large, but correlated motion. Again, due to the presence of the correlated motion, the MVA and the EMV outperform both the TSS and the FSS in terms of speed, quality as well as MVD bits across both the QCIF as well as the CIF version, with the EMV performing the best of the four sequences. The Foreman sequence has large and uncorrelated motion. For this sequence, the performance of our strategies goes down, with the EMV suffering a larger degradation, as it is more dependent on spatially correlated motion. However, we can see that the performance of our algorithms is still better than the worse of the FSS and the TSS. In fact for the CIF sequence, the MVA still outperforms the TSS and the FSS.

From all these results we can see that our algorithms combine the center-biased nature and speed of the FSS with the ability of the TSS to converge to good matches for large motion sequences. For sequences with correlated motion, our proposed algorithms perform better than both the TSS and the FSS. For sequences with uncorrelated motion, the performance of our algorithms degrades, but is still always better than the worse of the TSS and the FSS and is comparable to the performance of the better of the two. Clearly, our algorithms can be used across different types of motion sequences as well as different picture formats to provide good performance in terms of the "speed-quality-bit rate" tradeoff. This is not true of either the FSS or the TSS as their performance depends on the kind of sequence under test.

Comparing our approach to that done by Gallant et al [7] the difference is as follows. Gallant et al. use spatial correlation information from neighbors to predict the location of the best match. This is used as a starting point for a center biased search. They do not use a choice between a center-biased and a non-center biased algorithm. As we have pointed out before, the option to use a non-center biased algorithm (TSS) is useful in cases when our prediction is erroneous and helps in enhancing the performance of the search strategies. Based on our approach, we choose to use the TSS around 8~9% of the time, on average.

III. One/Four Motion estimation and Decision

An advanced option in H.263 and the MPEG-4 standards involves using four motion vectors per block. Each block is further subdivided into four parts and the best match for each of these parts is computed. This is shown in the following figure.

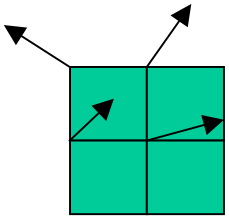


Figure 9. Motion vectors for sub-parts of blocks

Allowing motion vectors for sub-parts of a block provides a better match in terms of quality and should also help reduce the bits needed to code the residue as compared to when only one motion vector is allowed per block. In order to compute these four motion vectors for every block we have extended our search strategies MVA and EMV appropriately. Now for each sub-part we choose a set of predictors, based on the same set defined in the H.263 standard for differential coding of motion vectors of these sub-parts as shown in Figure 10.

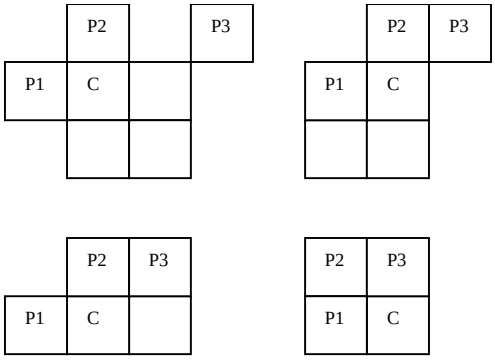


Figure 10. Redefinition of neighbors for sub-parts of block

As before, the predictors vote for either the TSS or the FSS based on their own motion vector and the center of the search for each sub-part can also moved based on the median of the predictor motion vectors. Also, we use only available predictors in accordance with what is specified in the standards. This leads to efficient motion estimation for all the four motion vectors and as our algorithms bias a smooth motion field this also helps in reducing the bit rate further when this mode is used in conjunction with the overlapped block motion compensation (OBMC) [16]. OBMC involves using motion vectors of neighboring sub-parts of blocks to reconstruct a sub-part, thereby leading to an overall smoothing of the image and removal of blocking artifacts. Each pixel in the prediction for the sub-part is obtained as a weighted average of three pixels that depend on the motion vectors of the sub-part and two of its neighbors. The averaging helps in obtaining a better prediction and thus helps in reducing the bits needed for coding the residue.

The MPEG-4 verification model (VM14) [17] recommends a search strategy for these four motion vectors that is efficient and provides motion vectors that lead to reduction of bit rate in

conjunction with OBMC. The strategy recommended uses a full search to determine the one motion vector for the block and using a 2×2 search window about this to determine the four motion vectors for the sub-parts of the block. When we compare our search strategy with this recommended strategy, we find that our strategy is 3~4 times faster and the bit rate using both the strategies is very close to (within 0.5%) each other for all sequences.

Although we have extended the efficient search strategies to compute these motion vectors, the number of times motion estimation needs to be done has increased four-fold, leading to a corresponding increase in computation. Besides this additional computational cost, there is also an overhead associated with the sending of four instead of one motion vector. Many times when the entire block lies completely inside the background or a moving object, all four sub-parts of the block move together which is equivalent to having one motion vector for the entire block. Clearly in such cases sending four motion vectors instead of one motion vector is inefficient. In order to avoid this overhead the standard allows for the encoder to decide whether it should send one motion vector or four motion vectors. Work on selecting the optimal combination of modes in order to minimize the distortion subject to a rate constraint has been done by Weigand et al [18]. They propose a generic algorithm for optimizing the encoder operation in a rate-constrained framework, using a Lagrangian formulation. They illustrate their approach using the one-four motion vector mode decision and propose a solution, using dynamic programming that accounts for dependencies between adjacent blocks and selects the mode that minimizes a cost function consisting of rate and distortion. This strategy is general enough for different mode decisions, however it is complex and requires a large amount of computation to account for the dependencies as well as the rate constraints.

In contrast, the decision strategy recommended by the test model of the H.263 standard does not take into account any dependency between neighboring blocks and also does not consider a rate-constrained problem. It is a simple decision strategy and is as follows

$$\begin{cases} \textit{Choose four motion vectors if } MAD_1 - MAD_4 > T_2 \\ \textit{Choose one motion vector otherwise} \end{cases}$$

where T_2 is an empirical threshold, MAD_1 is the MAD for the best match for the block using one motion vector and MAD_4 is the average of the MADs from the best match for each of the four sub-parts. In order to increase the efficiency of the decision, we propose a two-tier decision strategy.

Step 1. Find the MAD_1.

If $MAD_1 < T_3$

choose one motion vector for the block and stop

else

find MAD_4

Step 2. Compare MAD_1 and MAD_4. Use the previously defined decision strategy i.e. if $MAD_1 - MAD_4 > T_2$ then choose four motion vectors otherwise choose one motion vector.

If the MAD_1 is small it is quite likely that the best match for the block is a sufficiently good match for each of the sub-parts of the block. Hence, there is no need to find the best match for each of the individual sub-parts of the block. The threshold T_3 is also found empirically.

When we use the new two-tier decision strategy to decide between using one and four motion vectors per block, the increase in speed is 15~20% and there is no difference in quality (average MAD) or size of the bitstream. The largest savings in speed were obtained for the Akiyo sequence as it really smooth and has low motion, while the smallest savings were achieved for the Foreman sequence, as there are cases when the best match with one motion vector does not fall below the threshold defined. To achieve these gains the threshold T_3 was choused as 4, which means that an MAD of 4 is a reasonable small MAD below which there is no perceptible gain by using four motion vectors instead of one.

The MPEG-4 VM14 also recommends a decision strategy similar to the one proposed by the H.263 test model, the only difference being that there is a bias towards favoring a zero motion vector, during the one motion vector estimation. This is done by subtracting a constant from the MAD for the zero motion vector case. This bias is thus, also carried over to the one-four motion vector decision. We also include a similar bias in our mode decision implicitly, due to our motion estimation algorithms. During motion estimation for the one motion vector, if we find that the MAD_1 at the search center is smaller than a certain threshold, we stop the search. We also decide at this stage not to do the four motion vector search and to use the one motion vector. A bias may also be explicitly added into the mode decision by changing thresholds T_2 and T_3 appropriately. For instance when we have a zero motion vector in the one motion vector case we can increase both the thresholds to bias the decision towards using this zero motion vector. The thresholds may also be changed heuristically based on the type of video sequence. For instance, for a low motion sequence with a small number of moving blocks, we should increase the bias towards using one motion vector and so can increase the thresholds, while for a high motion sequence with a large number of moving blocks it is likely that using four motion vectors gives better performance in

terms of quality as well as bit rate, so in we need to reduce the bias towards using one motion vector and so can reduce the thresholds appropriately.

IV. Mean Removed Mean Absolute Difference

The MAD is often used to measure the similarity between blocks and the smaller it is between two blocks, the more similar they are. So a lot of motion estimation algorithms use this as a block matching criterion. Video coding not only requires a good quality of the match, but also requires minimizing the number of bits for coding the residue and motion vectors. The relationship between the MAD and the number of bits needed to code the residue is not monotonic. This is illustrated by the following figure.

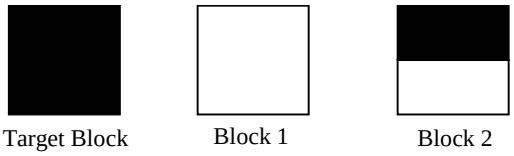


Figure 11. Example of bad block matching

As can be seen from Figure 11 Block 2 has a smaller MAD with respect to the target block than Block 1 and using the MAD to find the best match chooses Block 2. However, the residue between Block 1 and the target block is just a constant. Coding such a residue requires very few bits and so actually Block 1 should be preferred. If before computing the MAD we remove the means from each of the blocks, it can be seen that Block 1 is chosen as the better match, as desired. This is shown in Figure 12.

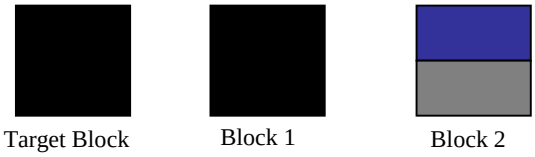


Figure 12. Mean reduced blocks

It can be easily verified that finding the best match through mean removal is effectively using a new measure of similarity between blocks. We call this new measure the mean removed MAD (mrMAD). It is computed as follows.

$$z_{i,j} = x_{i,j} - y_{i,j}$$

$$m_z = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} z_{i,j} : mean\ of\ block\ z$$

$$mrMAD = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left|z_{i,j} - m_z\right|$$

In order to evaluate the relationship between mrMAD and bits we perform the following test. We collect 10000 blocks from various sequences and assign motion vectors to each of them. Using these motion vectors and the original blocks we compute the bits needed to code the residue and the mrMAD between the block and its predictor. Simultaneously we also compute the MAD between the block and its predictor. We then try to predict the number of bits to code residue using linear prediction with mrMAD as a predictor. We also do this using the MAD as a predictor. We derived the linear minimum mean squared error (MMSE) prediction for the bits from the MAD and the mrMAD and then computed the prediction error in each case. We found that the root mean squared error in predicting the bits using the MAD was 59.74 bits, while the error using mrMAD was 56.7 bits (around 5% better) showing that the mrMAD is indeed a better linear predictor.

In practice, during motion estimation, a bias is added towards the zero motion vector, hence the error measurement does not play as significant a role as for the non-zero motion vectors. We further analyze the data corresponding to the non-zero motion vectors and see that the mrMAD is indeed a better predictor for bits than the MAD. For this analysis we include 10000 blocks with non-zero motion vectors from different sequences and look at the MAD, the mrMAD and the bits needed to code the residue. We generate scatter plots of the bits versus the MAD and the bits versus the mrMAD and these plots are shown in Figure 13.

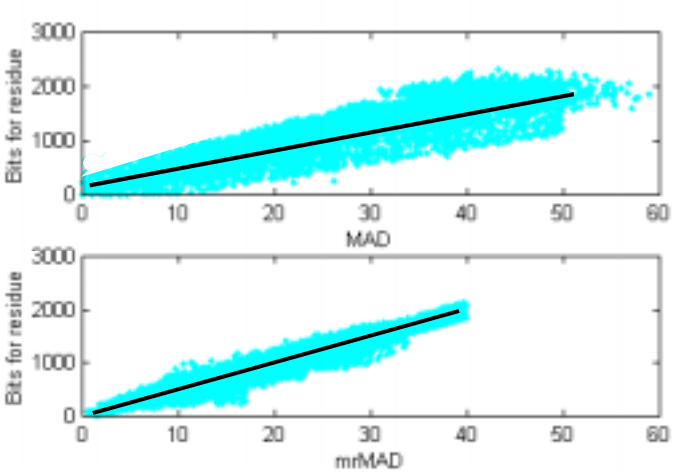


Figure 13. Scatter plots for bits versus MAD and mrMAD for non-zero motion blocks

It can be seen from the plots that the scatter plot for the bits versus the MAD has a much larger spread around the linear prediction, shown as the dark superposed line, as compared to the plot for the bits versus the mrMAD. For these non-zero motion blocks, we repeat the linear MMSE prediction for bits using the mrMAD and the MAD as predictors. We find that the root mean squared error in predicting bits for the mrMAD is 10.3% smaller than that for the MAD. This means that the mrMAD is a better predictor for the number of bits than the MAD, especially for blocks with non-zero motion vectors. Hence the mrMAD is a better feature to use than the MAD, as a measure of the number of bits and this is useful in motion estimation and mode decisions.

A. mrMAD for Better Motion Estimation

A lower error in terms of predicting the bit rate from the mrMAD should help in getting better matches in terms of bit rate if this is used a block matching measure. We have implemented the full search algorithm with the mrMAD as a matching criterion and the results are included in the following table.

Table 2. Bit rate with different block matching criteria for full search

Sequence	Error bits with MAD as matching criterion	Error bits with mrMAD as matching criterion
Akiyo	3.93M	3.92M
Hall	4.32M	4.23M
Coastguard	5.20M	5.10M
Foreman	4.78M	4.59M

As can be seen, by using the mrMAD we get savings in bit rate of around 1~4% over using the MAD. This comes at the expense of a reasonable increase in computation time due to the computation of the mean of the difference block. We also tried to implement the mrMAD with the sub-optimal search strategies. The results showed that the additional cost in terms of computation time went down as compared to the MAD, but no clear conclusion could be drawn whether the use of the mrMAD actually leads to a smaller bit rate. The size of the bitstream was larger for some sequences, while it was smaller for other sequences. This is probably due to the implicit assumption made by most sub-optimal search strategies. They assume that the best match lies in the vicinity of reasonably good matches and hence moving in the direction of better matches (in terms of the block-matching criterion) leads towards the best match. Therefore these strategies all take a gradient-descent approach. We assert that the error surface with the MAD is more likely to be smooth and hence suited for the gradient descent searches as opposed to the mrMAD or the bit error surfaces. We provide an illustration to highlight this assertion. We randomly choose a block from the Foreman sequence and for every point in it's search space we

find the MAD, the mrMAD and the number of bits needed to code the residue between the original block and the block at that position. We call these the error surfaces corresponding to the MAD, the mrMAD and the bits. These error surfaces are shown in the following three figures.

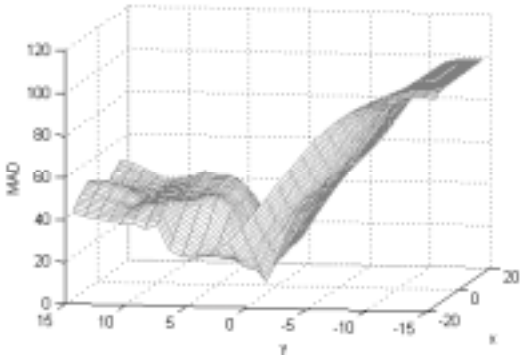


Figure 14. MAD vs. displacement

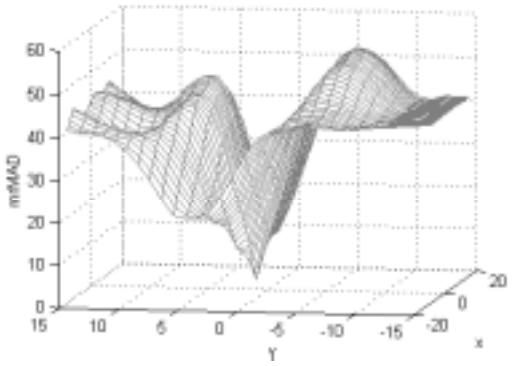


Figure 15. mrMAD vs. displacement

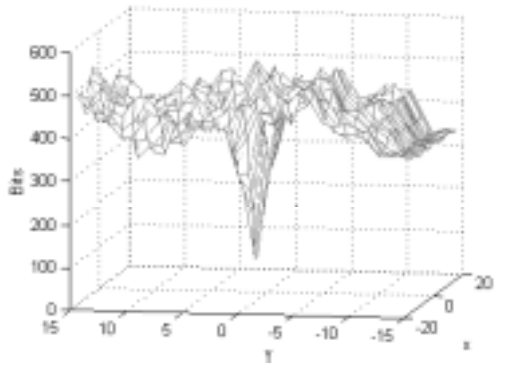


Figure 16. Bits vs. displacement

The three figures are plots of the MAD, mrMAD and bits across the search space. The horizontal axes represent the x and y displacement from the search center while the vertical axis shows the MAD, mrMAD or bits as the case may be. As can be seen from the plots, the MAD surface is

well suited to a gradient-descent based algorithm due to the fact that it has relatively fewer crests and troughs. This means that the chance that a gradient descent based search will get trapped in a local minima is smaller for this surface. On the other hand, if we look at the surface corresponding to the bits needed to code the residue, we see that there are many crests and troughs and so a gradient descent based search is more likely to get trapped in a local minima. As the mrMAD surface follows the bit error surface more closely than the MAD surface and shows the multiple crests and troughs, it is ill-suited for the use of a gradient-descent based search strategy. However, the mrMAD can be still effective with other non-gradient descent strategies, for instance stochastic search strategies like simulated annealing or genetic algorithms etc. As a preliminary test of this hypothesis we implement a simple genetic algorithm. More information about genetic algorithms can be found in [19]. For every block we start with a population of 10 random test vectors. We then rank these in terms of the error measurement and use proportional crossover and mutation to generate the next population. We repeat this ten times and the motion vector with the smallest error measurement among the remaining population is chosen as the best match. We then compute the residue and the number of bits needed to code the sequence. We find that for the same starting set of test vectors, using mrMAD as a measure instead of the MAD leads to a reduction in the bitstream size between 2.5 and 5.3%. This is an indication that mrMAD is also good for non-gradient descent based search strategies, especially stochastic search strategies, besides the full search.

B. mrMAD for Better Mode Decision

Sometimes, especially during scene changes, it is not possible to find a good match for a block. It can occur that coding the residue requires more bits than coding the block by itself. So the encoder has to decide between Intra and Inter coding. This decision is called the mode decision. To do a mode decision the encoder needs to know how many bits intra coding would take and how many bits inter coding would take. As converting to bits is computationally expensive, encoders instead use estimates for the bits. Typically the energy (with DC value removed) in the block is used as an estimate of the bits needed for intra coding, while the MAD is used as an estimate for the bits needed for inter coding. For example, the mode decision as recommended by the test model of the H.263 standard [20] is

$$\begin{cases} Intra\ Coding\ if\ MAD - E_x > T \\ Inter\ Coding\ \ otherwise \end{cases}$$

where $E_x = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left|x_{i,j} - m_x\right|$ and T is an empirically found threshold.

As we have mentioned before, the mrMAD provides a better estimate of the number of bits as compared to the MAD. Hence we can replace the MAD with the mrMAD to obtain a new mode decision as follows.

$$\begin{cases} Intra\,Coding \;\; if \;\; mrMAD - E_x > T \\ Inter\,Coding \;\; otherwise \end{cases}$$

To test whether the mrMAD provides a better mode decision the following was done. For every block for each of the four sequences (Akiyo, Hall, Coastguard and Foreman), E_x, MAD for the best match and mrMAD for the best match were computed. In addition bits needed for intra coding, bits needed for coding residue of best match with MAD and bits for coding residue of best match with mrMAD were also computed.

Each block corresponds to a point in the MAD-E_x space. With each of these points we associate a height that is just the difference between the bits needed for intra coding the block and bits needed for inter coding residue of best match with MAD. Blocks that should be inter coded have positive heights and blocks that should be intra coded have a negative height. Figure 17 illustrates this.

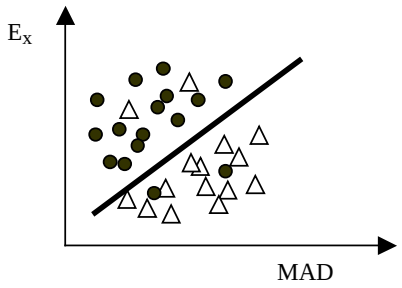


Figure 17. Example of MAD-Energy space

In the preceding figure the circles represent blocks with positive heights and triangles represent blocks with negative heights. The mode decision tries to partition this space into the positive half and the negative half. The mode decision equation is linear to avoid any excess computation (also shown in Figure 17). Hence we wish to find a line that partitions this space into points with positive heights on one side and those with negative heights on the other side. In general this is not perfectly possible (because of the linear boundary constraint). So we try to find the line that partitions this space such that sum of heights is as positive as possible on one side and as negative

as possible on the other side. To find the best line we adjust the intercept (threshold *T*) and keep the slope at unity. We found that we did not get much benefit by allowing for a different slope.

Similarly, we collect all the blocks in the mrMAD-E_x space and try to find the best linear boundary there. The thresholds (intercepts of the lines) for both cases were empirically determined to maximize the difference in either space. The threshold T determines the intercept of the linear boundary in the feature space. The threshold may have different values for the best boundary for different kinds of sequences. For instance, low motion sequences have a large number of blocks that can be well predicted leading to the Inter decision being more likely and so we can reduce the threshold to allow for this. In contrast, high motion sequences are more likely to have Intra coded blocks and this can be allowed for by raising the threshold. For the purposes of this paper we determine one threshold obtained using training data from different kinds of motion sequences. We conducted experiments on the same four sequences (Akiyo, Hall, Coastguard and Foreman) and found that maximum difference in sum of heights between the positive and negative halves in the mrMAD-E_x space is larger than that in the MAD-E_x space by around 6~6.5% for all the four sequences. This difference in heights corresponds to the savings in error bits by using mrMAD based mode decision. We implemented this in the H.263 framework and observed that the corresponding savings in total bit rate (including residue bits, motion vector bits and overhead bits) were around 4.5~4.8% for all the four sequences.

V. Delta motion vectors

The PB frame mode is a particular enhanced option that is part of the H.263 coding standard. MPEG-1, 2 and 4 have a similar mode that uses B frames. This is different from the PB frame mode of H.263 as the B frames in MPEG are independent entities, unlike in H.263 where the B blocks are always sent together with the P blocks. The PB frame mode is very useful for increased efficiency in coding and helps reducing the bitstream size for sequences with smooth motion. The following paragraph details the use of delta motion vectors for the PB frame mode in the H.263 standard.

A PB frame is a unit that is made of two frames, one of which is called the P picture and the other is called the B picture. Both these pictures are decoded together and are treated as a single entity. This mode supports three different pictures: the I picture, the P picture and the B picture. An I picture is Intra coded and is not predicted from any other picture. A P picture is predicted from the previous P picture or from a preceding I picture using motion estimation and compensation. A B picture is called so, because parts of it may be bi-directionally predicted, forward predicted

from the previous frame or backward predicted from the future P picture. The motion vectors for motion compensated prediction of the B picture are derived from the motion vectors of the P picture and a set of correction motion vectors called delta motion vectors. Without delta motion vectors, motion is assumed to be smooth between the previous I or P picture and the current P picture. This means that the motion vectors for the intermediate B picture are linearly interpolated. For example half the motion may be assumed to have taken place between the previous I or P picture and the B picture and the remaining half between the B picture and the current P picture. This may not be true as the motion may be accelerated or not linear at all. To correct for such cases delta motion vectors are provided. These are shown graphically in Figure 18.

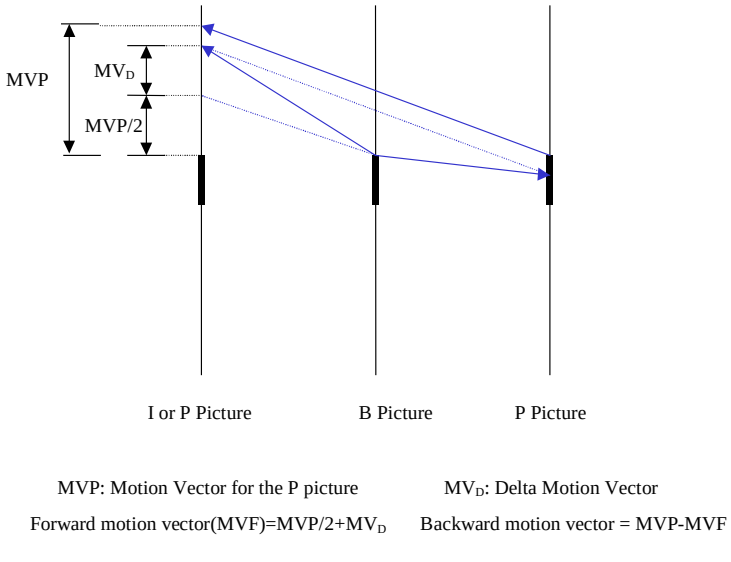


Figure 18. Example Delta Motion Vectors for 1-D case

Without delta motion vectors MVP/2 motion is assumed to have taken place between the I or P picture and the B picture and the remaining MVP/2 motion is assumed to have taken place between the B picture and the current P picture. With the delta motion vectors MV_D the forward and backward motion vectors are appropriately derived. MPEG-4 has separate P and B frames as opposed to the PB frame being treated as one unit, but the concept of delta motion vectors carries over.

Delta motion vectors are usually small as motion between frames is generally smooth and the need for correction is small. In tests that we conducted on the same four video sequences as before, using our H.263 codec, we found that less than 1% of delta motion vectors have values that may be considered large ($|v_x|$ or $|v_y| > 4$). Hence, motion estimation algorithms for delta

motion vectors need to be center biased and so we choose the FSS for this purpose. The use of the FSS does not affect the quality of the motion compensation adversely, but it reduces significantly the time for the delta motion vector estimation. The FSS was implemented and it leads to a speed up by a factor of 5~8 over the use of the full search, while the bitstream size increases by less than 1%. A speed up factor of over 7 was achieved for three sequences Akiyo, Hall and Coastguard, all of which have correlated and smooth motion, while a factor of 5.3 was achieved for the Foreman sequence, which has some degree of uncorrelated and large motion.

All of the previous discussion was for the estimation of delta motion vectors for the PB frame mode in H.263. MPEG-4 has a direct coding bi-directional mode, which extends the H.263 approach of using the P block motion vectors and scaling them to derive forward and backward motion vectors for the B frames. Similar to the PB frame mode in H.263, this mode also allows for Delta motion vectors to account for non-linear motion between frames. Due the similarity in the nature of the delta motion vectors as used in this mode and those used in the PB frame mode, our work may be easily extended to estimating these motion vectors. Hence all of the previous discussion holds for the Delta motion vectors for the direct bi-directional coding mode for the B frames in MPEG-4.

VI. Conclusion

The work presented in this paper deals with the optimization of some of the parts of the encoding process. The motion estimation algorithms we presented use spatial correlation information to choose a starting point for the search as well as the search strategy from this starting point. The correlation information is also used to decide between a center-biased search (FSS) and a non-center-biased search (TSS). This improves speed, quality and bit rate and provides a good search strategy across different kinds of motion sequences as well as different picture formats. The algorithm can be modified to replace the FSS with other center-biased algorithm like the Diamond search [15] and the TSS with other strategies such as the simple and efficient search [21]. The search strategy can also be extended to estimate motion vectors for sub-parts of a block, as allowed for in optional modes in H.263 and MPEG-4. Besides making the motion estimation more efficient, we also introduce an efficient two-tier decision strategy to choose between one and four motion vectors. This leads to an increase in speed of over 15~20% over the strategy recommended in the test model for H.263. The mrMAD that we have introduced in this paper is better related to bit rate than the more traditional MAD and it can lead to a better mode decision. It can also improve the bit rate for coding residue when used with the full search and non-gradient

descent algorithms for motion estimation. Savings of around 4.5~4.8% are obtained in total bit rate when we use the mrMAD instead of the MAD. We also propose the use of a center-biased algorithm (FSS) for the estimation of Delta motion vectors in the PB frame mode and this can lead to a speed up over the full search by a factor of 5~8. This work may also be used for estimating the Delta motion vectors for the direct bi-directional coding mode for B frames in MPEG-4. These optimizations have been fully tested and included in the H.263 video codec developed at Carnegie Mellon University [22].

VII. Acknowledgements

The work on spatial correlation based motion estimation algorithms was initiated in collaboration with Mohamed Alkanhal. We are very grateful to Prof. Vijaya Kumar for his valuable suggestions with the study of the mode decision. We are also grateful to the anonymous reviewers whose insightful comments helped improve the paper.

References

[1] *Video Coding for Low Bit rate Communication*, ITU-T Recommendation H.263 Version 2, Jan. 1998.

[2] Motion Pictures Experts Group, "Overview of the MPEG-4 Standard", ISO/IEC JTC1/SC29/WG11 N2459, 1998.

[3] J. Jain and A. Jain, “Displacement Measurement and its application in interframe image coding”, *IEEE Trans. on Communications*, vol. COM-29, pp. 1799-1808, Dec. 1981.

[4] Mohamed Alkanhal, Deepak Turaga and Tsuhan Chen, “Correlation based search algorithms for motion estimation,” *Picture Coding Symposium*, Apr. 1999, pp. 99-102.

[5] Michael Chen and Alan Willson, Jr., “Rate-Distortion optimal motion estimation algorithm for video coding,” *Proc. International Conference on Acoustics Speech and Signal Processing*, 1996, pp. 2096-99.

[6] Yen-Kuang Chen and S. Y. Kung, “Rate optimization by true motion estimation,” *Proc. IEEE Workshop on multimedia Signal Processing*, June 1997, pp.187-194.

[7] Michael Gallant, Guy Cote and Faouzi Kossentini, “An efficient computation-constrained block-based motion estimation algorithm for low bit rate video coding,” *32 Asilomar Conference on Signals, systems and computers*, 1-4 Nov. 1998.

[8] Jie-Bin Xu, Lai-Man Po and Chok-Kwan Cheung, "A new prediction model search algorithm for fast block motion estimation," *Proc. International Conference on Image Processing*, vol. (lii+951+892+748), pp. 610-3, Oct. 1997.

[9] Shiuh-Ku Weng, Chung-Ming Kuo and Chaur-Heh Hsieh, "Motion estimation algorithm for image sequence coding," *IEE Optical Engineering*, vol. 36, no. 12, pp. 3272-80, Dec. 1997.

[10] Kyoung Won Lim and Jong Beom Ra, "Improved hierarchical search block matching algorithm by using multiple motion vector candidates," *Electronics Letters*, vol. 33, no. 21, pp. 1771-2, Oct. 1997.

[11] J. Li, X. Lin and Y. Wu, "Multiresolution tree architecture with its application in video sequence coding: A new result," *Proc. SPIE Visual Communication and Image processing*, vol. 2094, pp. 730-41, 1993.

[12] J. Chalidabhongse and C. –C. J. Kuo, "Fast motion vector estimation using multiresolution spatio-temporal correlations," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 7, no. 3, pp. 477-88, June 1997.

[13] R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation", *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 4, no. 4, pp. 438-442, Aug. 1994.

[14] L. Po and W. Ma, " A Novel Four-Step Search Algorithm for Fast Block Motion Estimation", *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 6, no. 3, pp. 313-317, June 1996.

[15] J.Y Tham et al., "A novel unrestricted center-biased diamond search algorithm for block motion estimation," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 8, no. 4, pp. 369-77, Aug. 1998.

[16] Orchard, M. T., and Sullivan, G. J., "Overlapped block motion compensation - an estimation-theoretic approach," *IEEE Transactions on Image Processing*, v3 (5), pp. 693-699, Sept 1994.

[17] MPEG-4 Video Group, "MPEG-4 Verification Model v14.0," ISO/IEC N2932, Oct. 1999

[18] Thomas Wiegand, et al, "Rate-Distortion optimized mode selection for very low bit rate video coding and the emerging H.263 standard," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 6, no. 2, pp. 182-190.

[19] Mitsuo Gen and Runwei Cheng, *Genetic Algorithms and Engineering Optimization*, Wiley-Interscience, 1994.

[20] Video Codec Test Model, Near-Term, Version 10 (TMN10) Draft 1, Apr. 1998.

[21] Jianhua Lu and Ming L. Liou, “A simple and efficient search algorithm for block-matching motion estimation,” *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 7, no. 2, Apr. 1997.

[22] CMU H.263 codec. For more information, see http://amp.ece.cmu.edu