Probabilistic Partial Distance Fast Matching Algorithms for Motion Estimation

Krisda Lengwehasatit and Antonio Ortega

Integrated Media Systems Center

and

Signal and Image Processing Institute

Department of Electrical Engineering-Systems

University of Southern California,

Los Angeles, California 90089-2564

CORRESPONDING AUTHOR:

Antonio Ortega

Phone: (213) 740-2320

Fax: (213) 740-4651

E-mail: ortega@sipi.usc.edu

This work was published in part in [1], [2]

This work was supported in part by the National Science Foundation under grant MIP-9804959 and the Integrated Media Systems Center, a National Science Foundation Engineering Research Center, with additional support from the Annenberg Center for Communication at the University of Southern California and the California Trade and Commerce Agency.

Abstract

Motion search is by far the most complex operation to be performed in a video encoder. This complexity stems from the need to compute a matching metric for a potentially large number of candidate motion vectors, with the objective being to find the candidate with the smallest metric. Most work on fast motion estimation algorithms has focused on reducing the number of candidate motion vectors that have to be tested. Instead, this paper proposes a novel fast matching algorithm to help speed-up the computation of the *matching* (distance) metric used in the search, e.g. the sum of absolute difference (SAD). Based on a *partial distance* technique, our algorithm reduces complexity by terminating the SAD calculation early (i.e. when the SAD has only been partially computed) once it becomes clear that, given the partial SAD, it is *likely* that the total SAD will exceed that of the best candidate encountered so far in the search. The key idea is to introduce models to describe the probability distribution of the total distance given a measured partial distance. These models enable us to evaluate the risk involved in "trusting" a distance estimate obtained from a partial distance. By varying the amount of risk we are willing to take, we can increase the speed, but we may also eliminate some good candidates too early and thus increase the distortion of the decoded sequence. Because our approach requires knowledge of the statistical characteristics of the input, we also propose two approaches that allow these models to be obtained online. Our experimental results (based on an actual software implementation of an MPEG encoder) demonstrate that significant gains can be achieved with this approach. For example, reductions in the motion estimation computation time as compared with the original partial distance search (where computation stops if the partial SAD is already larger than the SAD of the best candidate so far) can be as high as 45% for 2-D Log search and 65% for exhaustive full search with a small penalty of 0.1dB degradation in PSNR of the reconstructed sequences.

Keywords

Motion Estimation, Complexity, Fast Matching, Fast Search, Partial Distance, Hypothesis Testing.

I. Introduction

Motion estimation (ME) is an essential part of well-known video compression standards, such as MPEG1-2 [3] and H.261/263 [4]. It is an efficient tool for video compression that exploits the temporal correlation between adjacent frames in a video sequence. However, the coding gain comes at the price of increased encoder complexity for the motion vector (MV) search. Typically ME is performed on macroblocks (i.e., blocks of 16×16 pixels) and its goal is to find a vector pointing to a region in the previously reconstructed frame (reference frame) that best matches the current macroblock (refer to Fig. 1). The most frequently used criterion to determine the best match between blocks is the sum of absolute differences (SAD).

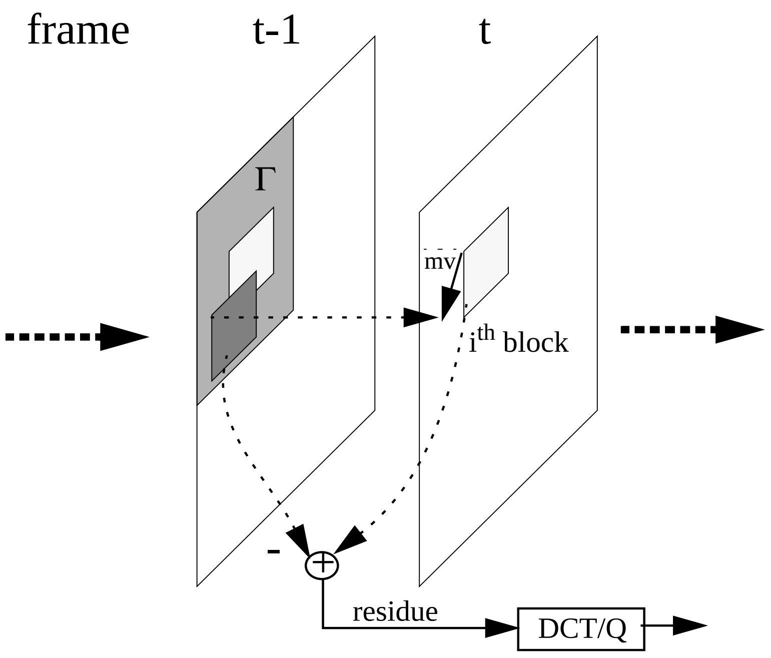


Fig. 1. Motion estimation of i-th block of frame t predicted from the best block in the search region Γ in frame t-1.

A. Example: Conventional Exhaustive Search

We start by introducing the notation that will be used throughout the paper (refer to Table I). Let us consider the i-th macroblock in frame t. For a given macroblock and candidate motion vector $\vec{mv}$, let the sum of absolute difference matching metric be denoted as $SAD(\vec{mv}, \beta)$, where[1]

$$SAD(\vec{mv}, \beta) = \sum_{(n_x,n_y)\in\beta} |I_t(n_x, n_y) - I_{t-1}(n_x + mv_x, n_y + mv_y)|, \tag{1}$$

and where β is a subset of the pixels in the macroblock. This notation will allow us to represent the standard SAD metric based on the set B of all pixels in a macroblock, as

[1]Note that, since our approach will be the same for all macroblocks in all motion-compensated frames, we will not consider explicitly the macroblock and frame indices (i and t) unless necessary.

TABLE I

Notation Table

$I_t(n_x, n_y)$	intensity level of (n_x, n_y) pixel relative to the upper-left-corner pixel of the macroblock.
B	the set of pixels constituting a macroblock
β	a subset of B
$\vec{mv} = (mv_x, mv_y)$	a candidate motion vector
Γ	the set of allowable $\vec{mv}$ in a pre-defined search region e.g., $\Gamma = \{(mv_x, mv_y) : mv_x, mv_y \in \{-16, -15.5, -15, ..., 15, 15.5\}\}$.
γ	a set of $\vec{mv}$ ($\gamma \subset \Gamma$) actually tested for a given search scheme

well as partial SAD metrics based on pixel subsets β. A ME algorithm will return as an output the best vector for the given search region and metric, $MV^*(\gamma, \beta)$, i.e. the vector out of those in the search region $\gamma \subseteq \Gamma$ that minimizes the SAD computed with $\beta \subseteq B$ pixels,

$$MV^*(\gamma, \beta) = \arg \min_{\vec{mv} \in \gamma} SAD(\vec{mv}, \beta).$$

In the literature, a search scheme is said to provide an optimal solution if it produces $MV^*(\Gamma, B)$, i.e., the result is the same as searching over all possible $\vec{mv}$ in the search region (Γ) and using a metric based on all pixels in the macroblock (B), where $MV^*(\Gamma, B)$ can typically be found using an exhaustive full search (ES). In this paper, we will term "exhaustive" any search such that $\gamma = \Gamma$ regardless of the particular β chosen.

In general, motion search is performed by computing the SAD of all the vectors in the search region sequentially (following a certain order, such as a raster scan or an outward spiral), one vector at a time. For each vector, its SAD is compared with the SAD of the "best found-so-far" vector. Without loss of generality, let us assume that we are considering the i-th candidate vector in the sequence, $\vec{mv}_i$ for $i = 1, ..., |\Gamma|$, and we use B for the SAD computation. Thus we define the "best found-so-far" SAD as

$$SAD_{bsf}(\gamma_i, B) = \min_{\vec{mv} \in \gamma_i} SAD(\vec{mv}, B)$$

where $\gamma_i = \bigcup_{j=1}^{i} \{\vec{mv}_j\} \subseteq \Gamma$ is the set of vectors that have already been tested up to $\vec{mv}_i$

and the associated "best found-so-far" vector is denoted by $MV_{bsf}(\gamma_i, B)$. Note that when all vectors in the search region have been tested, $MV^*(\Gamma, B)$ is equal to $MV_{bsf}(\Gamma, B)$.

To complete the encoding process, MV^* is transmitted. The residue block, which is the difference between the motion estimated block and the current block, is transformed, quantized, entropy coded and then sent to the decoder, where the process is reversed to obtain the reconstructed images.

B. Fast Motion Estimation Techniques

We now provide a quick overview of fast ME techniques. Our goal is to provide a rough classification of the various strategies that have been used to reduce complexity while introducing the novel features in our proposed algorithm.

B.1 Fast Search vs. Fast Matching

The total complexity of the ME process depends on (i) the number of candidate vectors in the search region, Γ, and (ii) the cost of the metric computation to be performed for each of the candidates (e.g. computing a SAD based on the set B.) Thus, fast ME techniques are based on reducing the number of candidates to be searched (fast search) and/or the cost of the matching metric computation (fast matching).

B.1.a Fast search (FS). In order to improve the efficiency of the search, fast ME algorithms can restrict the search to a subset of vectors $\gamma \subset \Gamma$. This subset of vectors can be pre-determined and fixed as in [5] or it can vary as dictated by the specific search strategy and the characteristics of the macroblock. Examples of the latter case are 2-D log search [6], conjugate directions and one-at-a-time search [7], new three step search [8], gradient descent search [9], center-biased diamond search [10] which exploit in various ways the assumption that the matching difference is monotonically increasing as a particular vector moves further away from the desired global minimum. A good initial point can also be used to reduce the risk of being trapped in local minima. Approaches to find a good initial point include hierarchical and multiresolution techniques [11], [12], [13], [14], [15]. Another successful class of techniques seeks to exploit the correlations in the motion field, e.g., MVs of spatially and temporally neighboring blocks can be used to initialize the search as in [16] and [17].

B.1.b Fast matching (FM). Another approach for fast ME, which can also be combined with a FS technique, consists of devising matching criteria that require less computation than the conventional sum of absolute difference (SAD) or mean square error (MSE). One example of this approach consists of computing a partial metric, e.g., the SAD based on $\beta \subset B$ [5]. Of particular relevance to our work are the partial distance search techniques, which have also been proposed in the context of VQ [18], [19]. In a partial distance approach the matching metric is computed on successively larger subsets of B but the computation is stopped if the partial metric thus computed is found to be greater than the total metric of "best found-so-far" vector. For example if $SAD(\vec{mv}_i, \beta \subset B) > SAD_{bsf}(\gamma_{i-1}, B)$ there is no need to complete the metric computation and calculate $SAD(\vec{mv}_i, B)$. Many implementations of FS algorithms include this partial distance technique to speed up their metric computation. Other early termination criteria have been proposed in [20]. Alternatively, matching metrics other than SAD or MSE can also be used. For example, in [21], adaptive pixel truncation is used to reduce the power consumed. In [22], the original (8-bit) pixels are bandpass filtered and edges are extracted, with the final result being a binary bit-map that is used for matching. Other approaches include hierarchical feature matching [23], normalized minimum correlation techniques [24], and minimax matching criterion [25].

In this paper, we will focus on FM approaches based on the partial distance technique. The novelty of this work is the probabilistic stopping criterion of the partial distance computation. It should be emphasized that the FM techniques we propose can be applied along with any FS strategy. We also note that, while our experimental results are provided for a software implementation, focusing on FM approaches may also be attractive in a hardware environment. For example, from a hardware architecture point of view, some FS designs have the drawback of possessing a non-regular data structure, given that the blocks that have to be searched in the previous frame depend on the selection of initial point, and thus vary from macroblock to macroblock. Conversely, ES algorithms have the advantage of operating based on a fixed search pattern (this could also facilitate parallelizing the algorithm). In general, FS algorithms such as that in [16] will have to be modified for hardware implementation, with one of the main goals being to minimize the overhead due to the non-regularity of the algorithm. As an alternative, if the goal is an efficient hardware design one may choose to design an efficient FM approach (e.g., [21], [26], [27], [28], [29]) and combine it with a simple search technique, such as ES.

B.2 Fixed vs. Variable Complexity

We can also classify ME techniques into fixed complexity algorithms (FCAs) and variable complexity algorithms (VCAs). The complexity in FCAs is input-independent and remains constant (e.g. a ME technique with fixed β and γ), while in this work we will consider VCAs, where complexity is input dependent (e.g. β and γ are different for each macroblock and/or frame.) The goal when designing a VCA is then to achieve low complexity in the average case. Thus, we expect the "worst case" complexity of the VCA to be higher than that of a comparable FCA, but hope that on the average, a VCA will have lower complexity. In practice, this is done by making reasonable, though typically qualitative, assumptions about the characteristics of typical sequences. For example, consider the algorithm of [16], which, as indicated earlier, exploits the correlations in the motion field. For this algorithm, performing ME in a scene with smooth motion (e.g. a scene with panning) tends to require less complexity (and to be closer to the optimal ES result) than finding the motion field for a scene with less correlated motion (e.g. a scene with several independent moving objects). Thus, such an algorithm provides a good average case performance under the assumption that typical video sequences have predominantly smooth motion. For similar reasons, algorithms in [6], [7], [8], [10] perform well for sequences with low motions.

A second example of a VCA algorithm can be found in the partial distance approach discussed earlier. The underlying assumption here is that the distribution of SADs for typical blocks has large variance, with few vectors having SAD close to the minimum (i.e. the SAD of the optimal vector). Thus, on average one can expect to eliminate many bad candidate vectors early (those having large metric) and thus to achieve a reduction in overall complexity. Once again this is an implicit assumption about the statistical characteristics of these matching metrics for typical blocks. In this paper we argue that substantial gains can be achieved by making these assumptions *explicit*, and therefore our novel probabilistic stopping criterion for the metric computation will be based on explicit statistical models of the distribution of SAD and partial SAD values.

B.3 Computationally Scalable Algorithms

Finally, we consider the computational scalability property, which is being considered as a desirable feature in many applications (e.g. to operate the same algorithm in different platforms, or to run at various speeds in the same platform). Computational scalability allows to trade-off speed with performance (e.g. the energy of the prediction residue in the case of ME). There has been some recent interest in computation scalability in the context of video coding in general (e.g., our work on DCT/IDCT [30], [31]) and ME in particular. For example, [32] addresses computationally constrained motion estimation where the number of vectors to be searched (the size of γ) is determined by a complexity constraint based on a Lagrangian approach. This technique adopts an idea similar to that in [33] but using complexity rather than rate as a constraint.

Our proposed probabilistic stopping criterion approach is the first that provides a computational scalability for the FM approaches. In our approach we still compute partial distances and use those to determine when a candidate vector should be eliminated. The basic idea is to stop the computation of the matching metric when the partial metric indicates that the total metric is *likely* to exceed that of the best candidate so far. when "bad" candidate vectors are being tested, since these can be discarded before the B-pixel metric has been computed. To achieve this goal we formalize a hypothesis testing framework where the decision to stop the metric computation is done based on probabilistic models of the distribution of the actual metric based on the calculated partial metric. We select a given probability of error (i.e. missing a "good" vector) based on these models and by varying this probability we can achieve a computationally scalable calculation. Our results show significant complexity reduction, e.g., 35% compared to the original partial distance search with insignificant degradation in distortion, e.g., loss of $0.01dB$ in PSNR. The complexity can be made variable by adjusting a threshold which determines the probability of eliminating good MVs. When this threshold is lowered we can speed-up the matching (e.g. by 50%) at the cost of a further, but still slight, increase in distortion (e.g. less than $0.1dB$ loss in PSNR).

Following the above categorizations, this paper contributes to fast motion estimation an algorithm which is FM, VCA and computationally scalable. The paper is organized as follows. In Section II, the original partial distance algorithm is reformalized and a new macroblock partitioning is introduced. In Section III, we introduce the proposed hypothesis testing framework that forms the basis of our algorithm. There, we assume all necessary statistics of a particular sequence are known to the encoder. In Section IV, we propose efficient techniques to obtain those statistics for a particular sequence adaptively during the encoding. In Section V, we show the results of our proposed algorithm with adaptive parameter estimation, as compared to the original partial distance search using ES, 2-D Log search [6] and ST1 search [16]. Finally, concluding remarks are given in Section VI.

II. A Review on Partial Distance Fast Matching

In this paper, we use SAD as the matching criterion[2]. In all our experiments SAD calculations are based on at most 128 pixels out of the 256 pixels of a macroblock. As in [5], this subset $\beta_q \subset B$ is obtained by subsampling using a quincunx grid (see Fig. 2.) As shown by Fig. 3, this particular subsampling tends to provide sufficient accuracy for the SAD calculation (see [5], [16]), i.e., the overall MSE does not increase significantly if we use β_q instead of B.

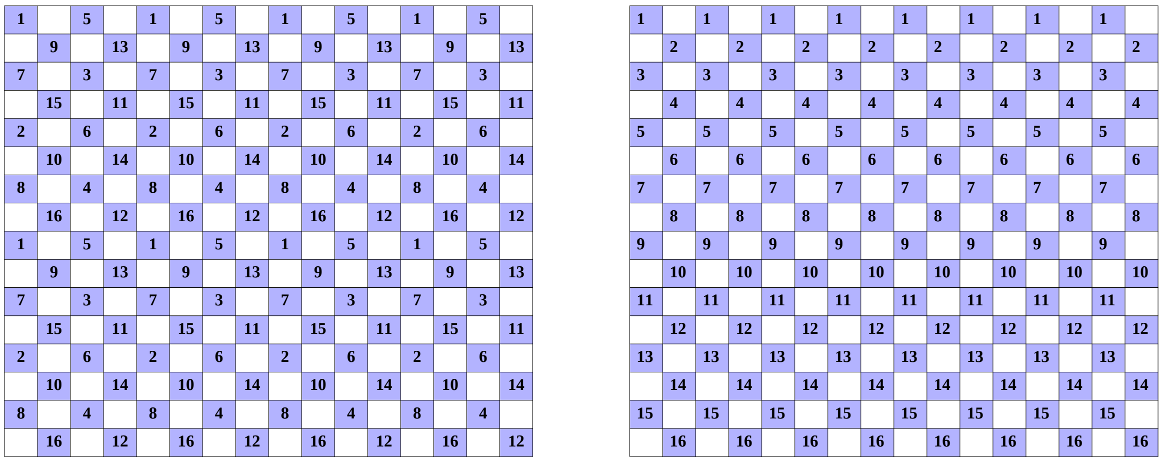
Our work is based on splitting each set β_q into b subsets of pixels, y_i, for $i \in \{1, 2, ..., b\}$, such that $\bigcup_{i=1}^{b} y_i = \beta_q$ and $y_i \cap y_j = \phi$ for $i \neq j$ (see for example Fig. 2). Let us define $x_i = \bigcup_{j=1}^{i} y_j$. During the SAD calculation of $\vec{mv}_j$, we compare the partial calculation of the SAD, $SAD(\vec{mv}_j, x_i)$ with the best SAD obtained out of all previously tested candidates, $SAD_{bsf}(\gamma_{j-1}, \beta_q)$. If the partial SAD is greater than SAD_{bsf}, we can terminate the computation and continue to the next vector. Otherwise, we compute $SAD(\vec{mv}_j, x_{i+1})$ for the next stage and perform the test again. If no early termination occurs, the process repeats until the final stage, b, is reached.

There are many possible ways to partition β_q into subsets y_i. In this paper, we propose two methods, which both have $|y_i| = 8$, $\forall i$, and thus result in 16 stages of testing: these are the uniform partition (UNI)[3] and the row-by-row partition (ROW) shown in Fig.2(a)

[2]Note, however, that our approach could be easy generalized to other additive distance metrics such as MSE (see for example our work in applying this approach to searching of a Vector Quantizer codebook [2]).

[3]During the review process of this paper, we became aware of the work by Cheng and Sun [34] which independently proposed using a uniform partition (dithering pattern pixel decimation). It is important to note, however,

and 2(b), respectively. In Fig.2(a), the partition is designed such that the correlation between $SAD(\vec{mv}, \beta_q)$ and $SAD(\vec{mv}, x_i)$ is maximum, given that pixels are uniformly spaced. This results in fewer pixel comparisons on the average, since early termination is more likely. However, for a hardware implementation, UNI may not be desirable as it results in more irregular memory access patterns. Conversely, ROW (see Fig.2 (b)) provides a more regular memory access that could simplify the implementation, although we can expect ROW to be worse than UNI in terms of producing a reliable estimate of the total SAD.



(a) (b)

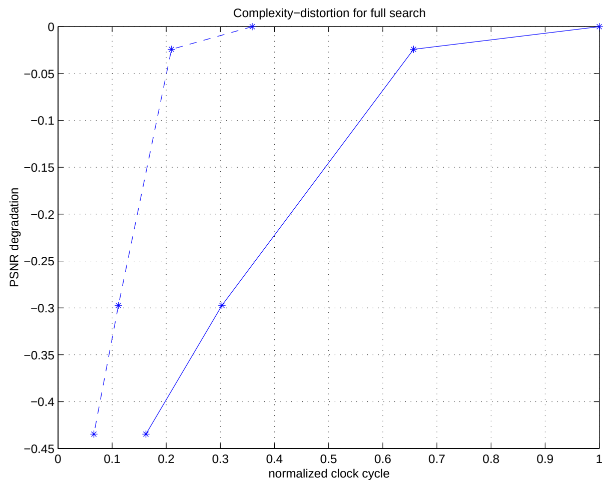
Fig. 2. Subset partitionings for 128 pixels subsampled using a quincunx grid into 16 subsets for partial SAD computation. Only highlighted pixels are used to compute SAD. Two types of subsets are used (a) uniform subsampling (UNI) and (b) row-by-row subsampling (ROW). Partial distance tests at the i-th stage are performed after the metric has been computed on the pixels labeled with i (corresponding to y_i).

To simplify the notation, when there is no ambiguity we omit to write the terms $\vec{mv}_j$, γ_{j-1} and B. Also we use $PSAD_i$ to represent the partial SAD at stage i, i.e., $SAD(\vec{mv}, x_i)$ for $i = 0, ..., b-1$. Note that the partial SAD can be computed recursively as

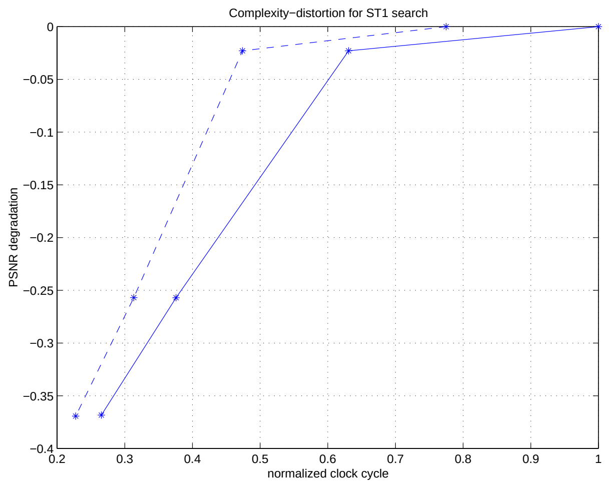
$$PSAD_{i+1} = PSAD_i + SAD(\vec{mv}, y_i) \tag{2}$$

where $PSAD_0 = 0$ and $PSAD_b = SAD$.

that neither this or other FM works use a variable number of pixels for matching.



(a)



(b)

Fig. 3. Complexity-Distortion of reduced set SAD computation with ROW DTFM ('dotted') and without DTFM ('solid') using (a) ES and (b) ST1 search, averaged over 5 test sequences. Points in each curve from right to left correspond to $|\beta| = 256$, 128, 64 and 32, respectively. Note that there is a minimal difference between computing the SAD based on 256 and 128 pixels. For this reason in all the remaining experiments in this paper we use at most 128 pixels for the SAD computation.

It is clear from (2) that $PSAD_i \leq SAD$, for $\forall i$. Therefore, if $PSAD_i$ is greater than the SAD_{bsf}, there is no need to complete the SAD computation and we can move on to evaluate the next vector. Otherwise, we compute $PSAD_{i+1}$ and perform the test again. As a result the MV^* obtained by the partial distance method is obviously the same as that obtained by computing directly the full metric. Thus we call this technique a *deterministic testing fast matching (DTFM)*, as it deterministically provides the optimal solution. Note that in this paper the "optimal" motion vector is based on SAD computed from β_q (i.e., 128 pixels), and therefore of a solution based on $x_i \subset \beta_q$ will tend to be "sub-optimal" since we cannot guarantee that it will produce the same motion vector selection obtained using β_q. The DTFM approach can be summarized as follows

Algorithm 1 (DTFM)

Step 1: *At the beginning of motion search for a particular block, compute the SAD of the first candidate MV, assign it to* SAD_{bsf} *and set* MV_{bsf} *accordingly.*

Step 2: *Every time a new* $\vec{mv}$ *is considered, as dictated by the FS strategy, set* SAD *to zero. Set* $i = 0$. *If there is no next unevaluated vector,* $MV^* = MV_{bsf}$.

Step 3: *Compute* $PSAD_i$

Step 4: *If* $i < b$*, go to step 5, else let* $SAD = PSAD_b$ *and go to step 6.*

Step 5: *If* $PSAD_i \geq SAD_{bsf}$*, we eliminate the current candidate and go to step 2. Otherwise, let* $i = i + 1$ *and repeat step 3.*

Step 6: *If* $SAD < SAD_{bsf}$*,* $SAD_{bsf} = SAD$ *and* $MV_{bsf} = \vec{mv}$*. Go to step 2.*

The partial distance technique, we have just described is well-known and is implemented in many actual software implementations, where ROW subsampling is typically used (e.g. [35],[36]). The complexity savings of this technique come from the possibility of early termination in Step 5. The amount of complexity reduction varies depending on the nature of the sequence. Also, since we can use DTFM with any FS algorithm, the efficiency of the FS algorithm will affect the savings stemming from DTFM. For example, for efficient FS algorithms the tested MVs are likely to have small SAD and their SAD values will tend to be fairly similar. Therefore there is less chance to terminate the matching computation at an early stage, and the benefits of DTFM will be reduced. In general, the complexity reduction contributed by DTFM can be significant, e.g., about 3-5 times speedup in ES case. In Fig.3, complexity-distortion (C-D) results with and without DTFM are compared. The C-D curves are obtained by changing the set β. One can observe that the relative gain in using DTFM is lower when a fast search algorithm is used.

TABLE II

Profile of MPEG2 encoder on the "mobile&calendar" sequence comparing ES and ST1.

component	ES	ES-DTFM	ST1	ST1-DTFM
Motion estimation	86.6%	69.3%	22.9%	20.2%
Quant. + Inv.Quant	3.7%	8.3%	20.0%	21.7%
DCT + IDCT	1.9%	5.9%	13.0%	12.6%
Others	7.8%	16.5%	44.1%	45.5%
Relative total time	1	0.44	0.2142	0.2106

An example of the CPU time profiles of the major components in an MPEG2 encoder using a FS algorithm (the ST1 algorithm in [16]) and ES are shown with DTFM and without DTFM in Table II. The last row shows relative total complexity of each algorithm.

We can see that DTFM reduces the ME complexity significantly in the ES case while the reduction is modest when used with a FS.

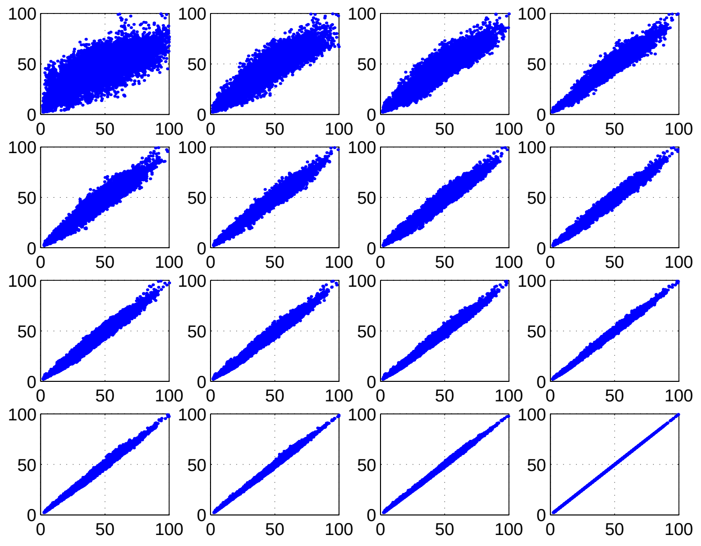
III. Hypothesis Testing Fast Matching

In this section, we propose a new algorithm, *hypothesis testing fast matching* (HTFM), that enables additional complexity savings as compared to DTFM by allowing an early termination of the SAD calculation based on the *likelihood* that the SAD will be greater than SAD_{bsf}, given the current $PSAD_i$. This complexity reduction over DTFM comes with the cost of potentially not finding the best motion vector for some blocks, which leads to an increase in the energy of the motion compensated predicted frame.

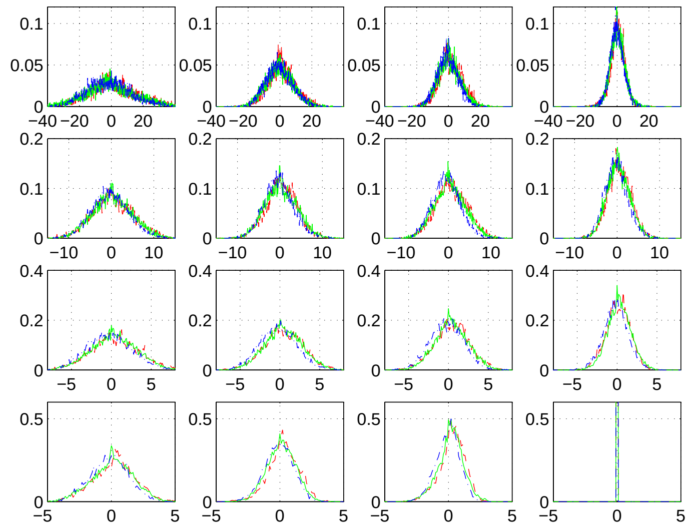
In our formulation, we will use the mean absolute difference (MAD) defined as $MAD = SAD/|B|$, where $|B|$ is the number of pixels in set B. Similarly, we write the "best-found-so-far" MAD as $MAD_{bsf} = SAD_{bsf}/|B|$ and the partial MAD as $PMAD_i = PSAD_i/|x_i|$. It is worth noting that *SAD is always greater than or equal to $PSAD_i$ but MAD can be either greater or smaller than $PMAD_i$.*

Our proposed approach comes from the observation that the PMAD becomes increasingly correlated with the MAD as the partial metric computation proceeds to more stages, i.e., more and more pixels are used. For example, consider Fig.4(a) where scatter plots of MAD vs. $PMAD_i$ are shown. It can be seen that there is a high correlation and $PMAD_i$ is an increasingly good estimate of MAD as i grows. The histograms of the difference between MAD and the PMADs are also shown in figure 4(b). From these figures we can conclude that the following are good approximations: (i) the partial MAD is a good estimate of the final MAD, i.e., $E\{MAD|PMAD_i\} \approx PMAD_i$, and (ii) there exists a reliable model for the error, and this model is about the same for any values of PMAD, i.e., $p_{MAD|PMAD_i}(x) \approx p_{MAD-PMAD_i}(x - PMAD_i)$.

In DTFM, we stopped the metric computation as soon as $PSAD_i$ is greater than SAD_{bsf}. In HTFM, given the $PMAD_i$ at the i-th stage we want to decide whether the MAD is *likely* to be larger than MAD_{bsf}. If that is the case, we can terminate the matching distance computation early, with the risk that the actual MAD may turn out to be smaller than MAD_{bsf}. We refer to this risk as the probability of false alarm, P_F. More formally, our goal is



(a)



(b)

Fig. 4. (a) Scatter plot between MAD (y-axis) and $PMAD_i$ (x-axis) and (b) corresponding histograms of $MAD - PMAD_i$. These are plotted for 16 stages of UNI subsampling, with number of pixels ranging from 8 (top left) to 128 (bottom right). We use UNI subsampling and ES on the "mobile &calendar" sequence. Similar results can be obtained with other sequences, search methods and subsampling grids.

Formulation 1 (HTFM) *Given $PMAD_i$ at the i-th stage ($i = \{1, 2, ..., b\}$) and MAD_{bsf}, we want to decide between the following two hypotheses:*

$$H_0 \; : \; MAD < MAD_{bsf}$$
$$H_1 \; : \; MAD \geq MAD_{bsf}$$

such that the constraint

$$P_F \equiv Pr(H_0|decide \;\; H_1) < P_f \text{ is satisfied.}$$

where P_f is the targeted P_F.

If H_1 is chosen, we stop the SAD computation. Otherwise, we compute $PMAD_{i+1}$ and perform another hypothesis testing at the $i+1$-th stage. Note that we could formulate the problem using the Neyman-Pearson criterion in which $Pr(\text{decide} \;\; H_1|H_0)Pr(H_0)$ (probability of false alarm) is constrained and $Pr(\text{decide} \;\; H_0|H_1)Pr(H_1)$ (probability of miss detect) is minimized. However, the resulting decision rule turns out to be the same. Also, we treat MAD_{bsf} and $PMAD_i$ as constants rather than assuming some prior distribution for them. Thus, we only need to consider and model the conditional probability of MAD given $PMAD_i$. P_F can then be rewritten as (see also Fig. 5),

$$P_F = \int_{-\infty}^{MAD_{bsf}} p_{MAD|PMAD_i}(x)dx = \int_{-\infty}^{MAD_{bsf}-PMAD_i} p_{MAD-PMAD_i}(x)dx$$

Given this probability, we can find a threshold parameter, Th_i, such that P_F is less than some threshold P_f,

$$\int_{-\infty}^{-Th_i} p_{MAD-PMAD_i}(x)dx = P_f \tag{3}$$

so that the decision rule becomes (see Fig.5)

$$PMAD_i - MAD_{bsf} \underset{H_0}{\overset{H_1}{\gtrless}} Th_i \tag{4}$$

Now we can replace the PSAD testing at step 5 of Algorithm 1 (DTFM) by (4). As illustrated in figure 5, $Pr(H_0|\text{decide} \;\; H_1)$ is the area under the $p_{(MAD|PMAD_i)}(x)$ function to the left of MAD_{bsf}. In general, we could select different P_f thresholds for each stage in the hypothesis testing. However, for simplicity, we fix P_f to a constant at all stages in our experiments.

From the histogram in Fig.4(b) we can model the difference between MAD and $PMAD_i$ as a random variable with Laplacian distribution, i.e., $p_{(MAD-PMAD_i)}(x) = \frac{\lambda_i}{2}e^{-\lambda_i|x|}$ where

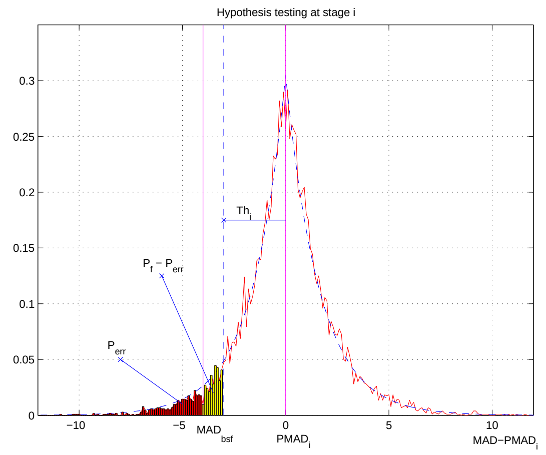


Fig. 5. Empirical pdf of $MAD - PMAD_i$ (estimation error) obtained from histogram of training data (solid line) and the corresponding parametric model (dashed line). HTFM terminates the matching metric computation at stage i if $PMAD_i - MAD_{bsf} > Th_i$.

λ_i is the Laplacian parameter for stage i. We found that this model is accurate for many test sequences and FS methods. With a Laplacian model, the threshold (3) can be written as a function of λ_i and P_f as follows

$$Th_i = \begin{cases} -\frac{1}{\lambda_i} ln(2P_f) & P_f \leq 0.5 \\ -\frac{1}{\lambda_i} ln2(1-P_f) & P_f > 0.5 \end{cases} \qquad (5)$$

Note that the Th_i of each stage is different because the model (and therefore λ_i) is different even if the same P_f is used for each stage.

So far, we have formalized a hypothesis testing framework based on likelihood testing of PMAD. However, there is a situation where it is useful to combine HTFM and DTFM. In some cases $PSAD_i$ is already larger than SAD_{bsf} but our probabilistic estimate indicates that $PMAD_i$ is not large enough to be eliminated (for example if P_f is very small), i.e., $PSAD_i|X|/|x_i| - SAD_{bsf} < Th_i|X|$ but $PSAD_i > SAD_{bsf}$. This would result in computing successive refinements of PMAD. This situation happens more in the last few stages and when SAD_{bsf} has a small value, i.e., when we are close to finding the best point in the search region . Therefore, in order to take full advantage of all information available at a certain stage, our HTFM also incorporates the partial SAD test in conjunction with the likelihood test at each stage. The HTFM, thus, can be summarized as

Algorithm 2 (HTFM)

Same as Algorithm 1 except that **Step 5** *is replaced by:*

Step 5 *If $PSAD_i \geq SAD_{bsf}$ or $PMAD_i > MAD_{bsf} + Th_i$, we eliminate the current candidate and go to step 2. Otherwise, let $i = i + 1$ and repeat step 3.*

The proposed HTFM technique reduces matching metric computation cost, but introduces an overhead due to the hypothesis test at each stage (one more comparison and two additions). While this additional complexity is outweighed in general by the gain from early termination, it is also possible to optimize the testing. Thus some tests could be pruned with the goal of minimal overall complexity for a specific data with known statistics, i.e., the probability mass distribution of being terminated at certain stage as in [1]. For simplicity, in this paper we perform the test in every stage without any optimizations.

IV. Online Parameter Estimation

In section III, we assumed that the conditional p.d.f. of MAD given $PMAD_i$ at stage i is known, so that the appropriate thresholds can be derived from this distribution. In practice, however, these statistics are not known a priori for a particular sequence. A possible solution would be to select in advance these probability distributions, through training over a set of video sequences. However, we have observed that the statistics of different sequences can differ significantly, depending on the frequency content of each frame, motion of objects in a frame and the FS techniques used. For example, a frame consisting of only low spatial frequency components tends to have less MAD estimation error than a frame with high frequency components. A frame with many moving objects causing uncorrelated motion vector also gives higher MAD estimation error. Moreover, with initialization-refinement approaches to FS (e.g. [16]), the MAD estimation error is smaller than for ES because the statistics based on a set of candidate vectors that are already expected to be good (i.e., their SAD will be close to the optimal one). For these reasons, we focus on approaches that estimate the probability models online, with updates taking place every few frames in a video sequence, under the assumption that the statistics do not change rapidly over a small group of frames.

We model the conditional density to be a Laplacian distribution with parameter λ_i. Thus we will only need to estimate the λ_i's in order to update the HTFM thresholds. Furthermore, λ_i is related to the variances, $\sigma_i^2 = E\{(MAD - PMAD_i)^2\}$, and the first order absolute moments, $\mu_i = E\{|MAD - PMAD_i|\}$ by

$$\lambda_i = \sqrt{2/\sigma_i^2} = 1/\mu_i \tag{6}$$

Therefore, obtaining any one of these three parameters is equivalent. Obviously, obtaining exact statistics would require that we compute the full MAD for each block, so that no fast matching speed up would be possible for the training frame. We now propose two training techniques for fast approximation of the threshold parameters for both ROW and UNI subsampling. In both techniques, our goal is to maintain the speed up due to DTFM while estimating the statistics reliably.

A. Model Estimation for ROW

Assume that when using DTFM for one block, the SAD calculation is terminated at stage t. In this case we have no information about $PMAD_i$ for $i > t$, but, given that we terminated early, the corresponding $PMAD_t$ can be thought to be a good approximation of the overall true MAD. Thus, our estimate $\tilde{\sigma}_i^2$ for σ_i^2 can be computed by assuming that for each block $MAD \simeq PMAD_t$, so that our estimated error variance is

$$\begin{aligned}\tilde{\sigma}_i^2 &= E_{t|i}\{(PMAD_t - PMAD_i)^2\} && \text{for } t > i \\ &\approx 2E_{t|i}\{|PMAD_t - PMAD_i|\}^2 && \text{with the Laplacian assumption}\end{aligned} \tag{7}$$

The update algorithm then becomes

Algorithm 3 (Variance estimation for ROW)

Step 1:*For a selected training frame, for every tested MV, perform DTFM in SAD calculation but also save $|PMAD_t - PMAD_i|$ for $i < t$ where t is the stage of DTFM termination.*

Step 2:*Compute $\tilde{\sigma}_i^2$ from $2E_{t|i}\{|PMAD_t - PMAD_i|\}^2$.*

Step 3:*Compute $\lambda_i = \sqrt{2/\tilde{\sigma}_i^2}$ and update thresholds for HTFM using this new estimate set of variances and (5).*

Thus for the training frames, we collect PMAD data only before the DTFM termination. The result of variances obtained this way is shown in Fig.6 averaged over 150 frames for each of the test sequences (ES is used in the experiment).

Two main observations can be made from Fig.6. First, it can be seen that $\tilde{\sigma}_i^2$ is obviously lower than the σ_i^2, thus resulting in smaller value of Th_i for a given targeted P_f which means that $Pr(SAD < SAD_{bsf}|\text{decide } \; H_1)$ is larger. Therefore, in order to obtain the same probability of error, P_f must be set to a smaller value. Second, we can further reduce the complexity of the data collection/manipulation by using linear interpolation to find $\tilde{\sigma}_i^2$ for $i \neq \{1, 2, 4, 8\}$ and only computing $\tilde{\sigma}_1^2, \tilde{\sigma}_2^2, \tilde{\sigma}_4^2$, and $\tilde{\sigma}_8^2$.

B. Model Estimation for UNI

In general, we can still use Algorithm 3 to approximate the model parameters for UNI subsampling. However, a better estimate $\hat{\sigma_i^2}$ of the true variance σ_i^2 can be obtained in this case. Let us consider the pixel at position $\vec{k} = (k_x, k_y)$ and denote $d(\vec{k})$ the pixel difference at that position,

$$d(\vec{k}) = |I_t(\vec{k}) - I_{t-1}(\vec{k} + \vec{mv})|$$

We can write $PMAD_i$ as

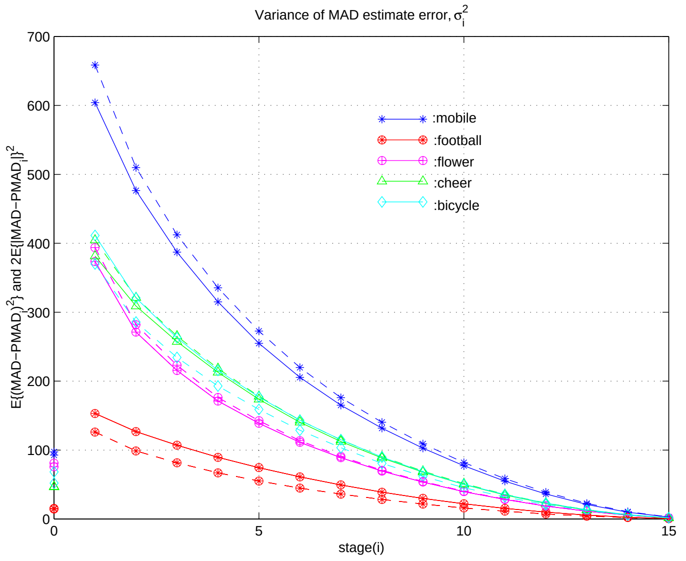
$$PMAD_i = \sum_{\vec{k} \in x_i} d(\vec{k})/|x_i| \tag{8}$$

Consider the first two stages, $PMAD_1$ and $PMAD_2$. Because the pixels are uniformly spaced we can assume that the pixel difference, $d(\vec{k})$, is an i.i.d. random sequence with average MAD and variance σ_d^2. Hence, $PMAD_1$ and $PMAD_2$ can be viewed as time averages. Therefore, we have

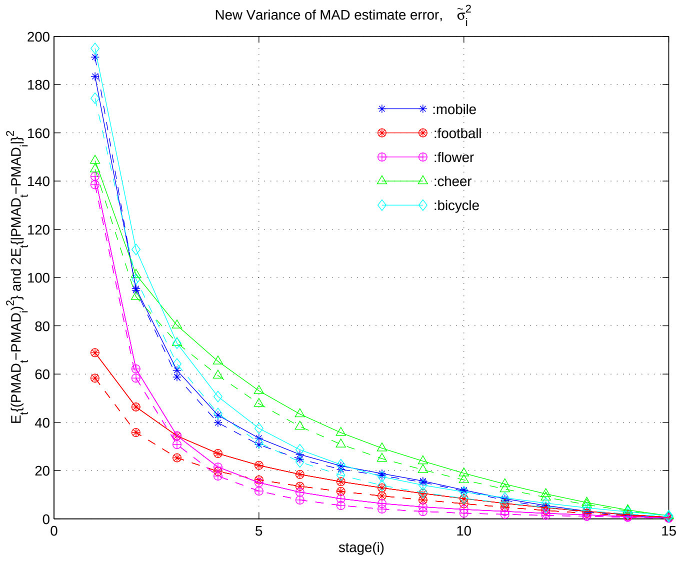
$$\begin{aligned} E\{(PMAD_1 - PMAD_2)^2\} &= \sigma_d^2(|x_2| - |x_1|)/|x_2||x_1| \\ &= \sigma_1^2(|x_2| - |x_1|)/|x_2| \\ &= \sigma_2^2(|x_2| - |x_1|)/|x_1| \end{aligned} \tag{9}$$

where σ_i^2 for $i = 1, 2$ are as defined previously. Therefore, using (9) for the first two test stages ($|x_2| = 16$ and $|x_1| = 8$), we can approximate σ_1^2 as $2E\{(PMAD_1 - PMAD_2)^2\}$. Besides, $PMAD_1 - PMAD_2$ can also be modeled to have Laplacian distribution. Hence its variance can be obtained without a square operation from the first order moment (expected absolute value), i.e., we can approximate σ_1^2 by

$$\sigma_1^2 \simeq 4E\{|PMAD_1 - PMAD_2|\}^2 = \hat{\sigma_1^2}.$$



(a)



(b)

Fig. 6. (a)σ_i^2 and (b) $\tilde{\sigma}_i^2$ of ROW computed from the definition (mean square) ('solid') and computed from the first order moment ('dashed'). The left-most points in (a) are $E\{(PMAD_1 - PMAD_2)^2\}$ and $2E\{|PMAD_1 - PMAD_2|\}^2$.

Our experiments (see Fig. 7(a)) show that this is a fairly accurate approximation. To approximate the variances at other stages, $\hat{\sigma_i^2}$, we observe that (see Fig.7(b)) the ratios between the first stage variance σ_1^2 and other σ_i^2 are almost constant regardless of the sequence and FS scheme used (see Fig. 7(b)). This can be justified based on the i.i.d. assumption of the pixel residue. From (8), it can be derived that if $d(\vec{k})$ is i.i.d. then the ratio of variances of $PMAD_i$ and $PMAD_j$ does not depend on σ_d^2 but is a function of only i and j. Since the i.i.d. assumption is good enough under UNI, it makes sense to see consistent ratios among all test sequences. As a result, in our approach we only estimate $\hat{\sigma_1^2}$ and obtain the remaining $\hat{\sigma_i^2}$ by applying the scaling factors shown in Fig. 7(b). We also note (see Fig. 7(a)) that the variances can be estimated accurately from the first order moment, μ_i^2. This can be justified from the Laplacian modeling shown in Fig. 5.

Therefore, the algorithm to estimate model parameter for UNI can be summarized as

Algorithm 4 (Variance estimation for UNI)

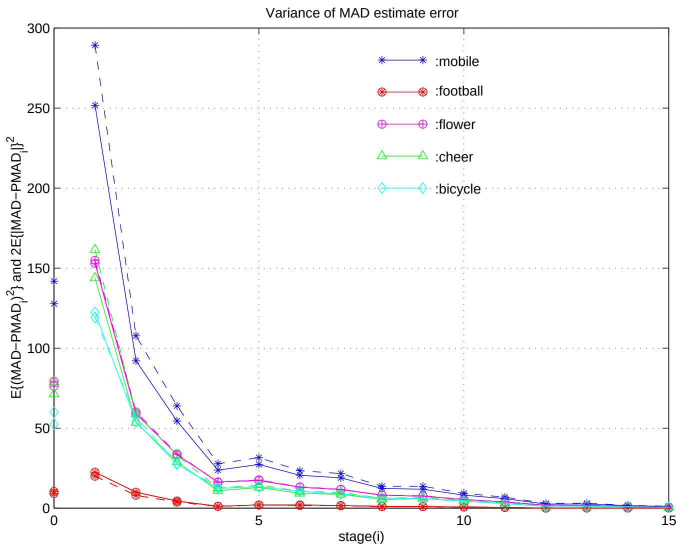
Step 1: *For a selected training frame, for every tested MV, always compute $PMAD_1$ and $PMAD_2$ and save $|PMAD_1 - PMAD_2|$. (DTFM or HTFM tests can be used for the following stages.)*

Step 2: *Compute $\hat{\sigma}_1^2 = 4E\{|PMAD_1 - PMAD_2|\}^2$. Compute other $\hat{\sigma}_i^2$ by dividing $2\hat{\sigma}_1^2$ with the ratios in Fig. 7(b).*

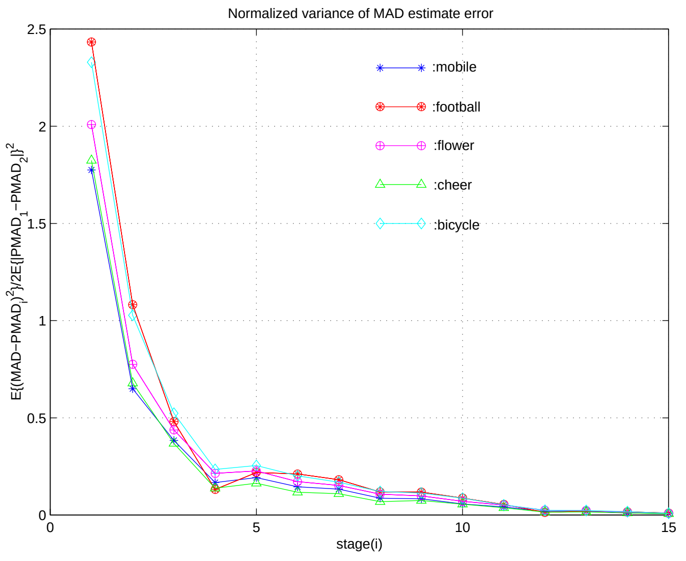
Step 3: *Compute $\lambda_i = \sqrt{2/\hat{\sigma}_i^2}$ and update thresholds for HTFM using this new estimate set of variances and (5).*

This variance approximation technique is fast because the only data we have to collect is $PMAD_1 - PMAD_2$ and we can apply DTFM test (when no previous statistics are available) or HTFM test (using previous HTFM test parameters) at stage 2, 3 and so on, i.e., we still gain some computation saving while performing the training. Once again, the limitation of Algorithm 4 is that the i.i.d. assumption is only reasonable when using UNI. For ROW, it is obvious that we cannot apply the UNI parameter estimation technique.

To demonstrate the effectiveness of our online training scheme we show in Fig. 8 a comparison between the σ obtained by online training and the actual error variances for the corresponding frames. Our results show that these methods provide a good approximation without a significant impact in the complexity of the training frames. In our experimental



(a)



(b)

Fig. 7. (a) σ_i^2 ('solid') and $2\mu_i^2$ ('dashed') of MAD estimate error at 15 stages using ES and UNI, respectively. The left-most points shows $E\{(PMAD_1 - PMAD_2)^2\}$ and $2E\{|PMAD_1 - PMAD_2|\}^2$ for each sequence. (b) Ratio of $\sigma_i^2/\hat{\sigma}_1^2$ for each sequence. Note that this ratio is nearly the same for all sequences considered.

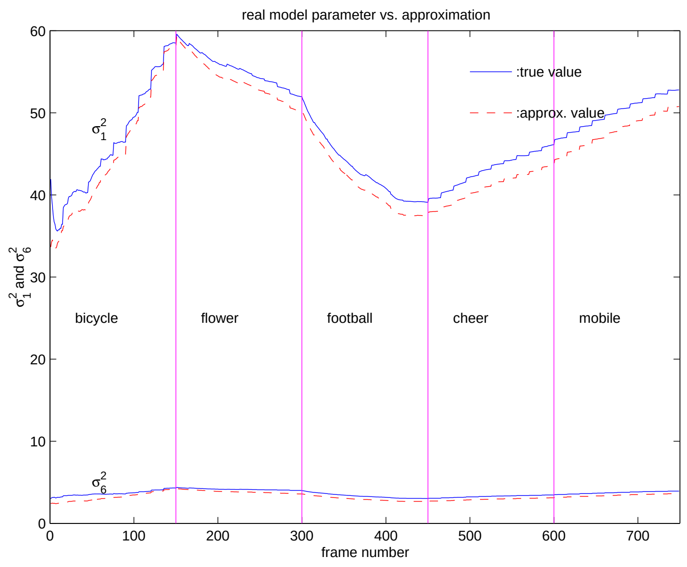


Fig. 8. Example of tracking of statistics σ_i under UNI subsampling. Note that the approximated values track well the actual ones, even though the parameters do change over time. We use several different sequences to provide the comparison. This serves as motivation for using online training, rather than relying on precomputed statistics.

results these training techniques will be used as appropriate and the corresponding training overhead will be included in the measured computation cost.

V. Experimental Results

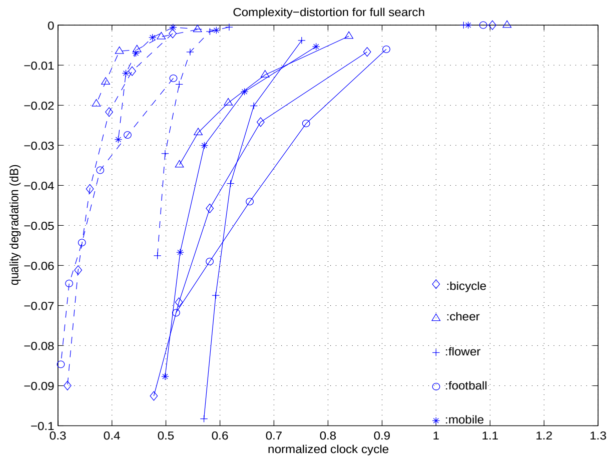
We discuss the conditions of our experiments first. We use 5 test sequences, namely, “mobile&calendar”, “football”, “flower”, “cheer” and “bicycle”. All of them consist of 150 frames of size 360 by 240. We encode them with an MPEG2 coder based on the “mpeg2encode” source code of [36]. We use frame prediction mode only (in MPEG2 there are other modes of motion compensation such as field and dual-prime prediction). The range of motion search is -15 to 15. We set the target rate at 5 Mbps. All the results are generated on a Pentium II 300 MHz processor.

In addition to ES for the best motion vector, we also use 2-D log search [6] and ST1 algorithm [16] as fast search techniques. We compare results between these three techniques with and without our HTFM using either UNI or ROW. This allows us to assess how our FM algorithm performs when a more efficient FS is also used. We note again that our FM algorithm could be combined with other FS approaches.

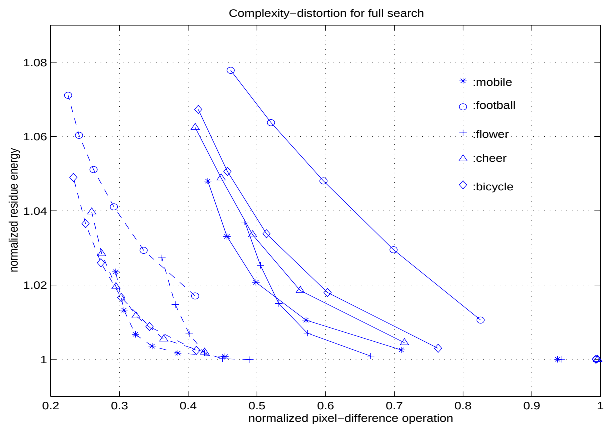
2-D log search starts from a small set of vectors uniformly distributed across the search region and moves on to the next set more densely clustered around the best vector from the previous step (if there is a change in direction, otherwise, the next set would be the same farther apart). The ST1 algorithm employs the spatial and temporal correlation of motion vectors of adjacent macroblocks. It starts with the best candidate motion vectors from a set of neighboring macroblock both spatially and temporally, if available. Then it performs local refinement on a small 3x3 window search until it reaches the minimum point. In general, ST1 algorithm achieves a higher speed-up than 2-D log search, with also lower overall residue energy.

We use each one of these three FS algorithms at integer pel motion accuracy. In order to obtain half-pel motion accuracy, we perform a one step search over a small 3x3 window (on half-pel grid) around the best integer motion vector from the FS algorithms.

A. UNI versus ROW



(a)



(b)

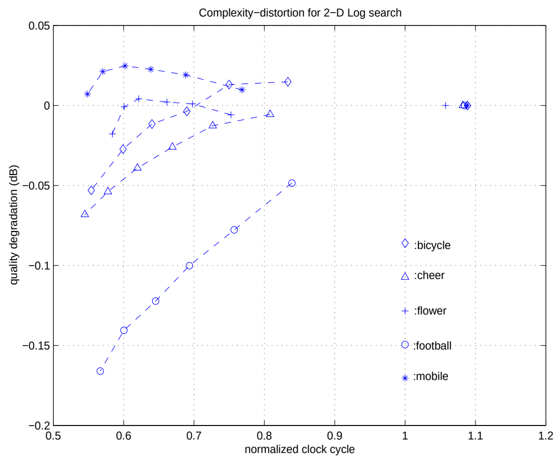
Fig. 9. Complexity-distortion of HTFM with ES and variance estimation on-the-fly, ROW ('solid') and UNI ('dashed'), (a) PSNR degradation vs. clock cycle and (b) residue energy per pixel vs. number of pixel-difference operations. Both clock cycle and number of pixel-difference operations are normalized by the result of ES with ROW DTFM. It can be seen that UNI HTFM performs better than ROW HTFM. The transform coding mitigates the effect of the increase of residue energy in the reconstructed frames. The testing overhead reduces the complexity reduction by about 5%. The complexity reduction is up to 65% at 0.05 dB degradation.

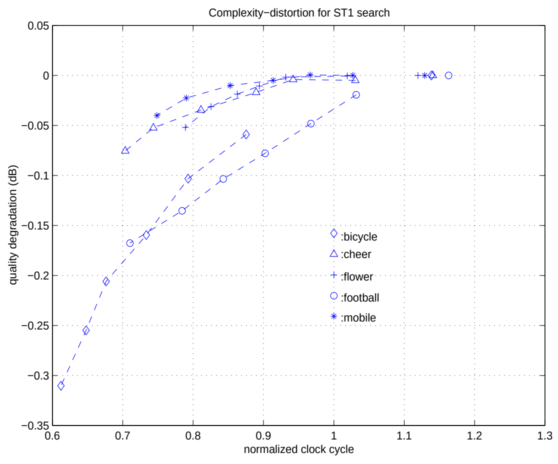
In Figure 9 (a), we show the complexity-distortion of 5 test sequences using ES motion estimation with the HTFM. The unit of complexity is the actual CPU clock cycles spent in motion estimation normalized by the same quantity using ROW DTFM. The distortion is given in terms of PSNR degradation from the DTFM (both ROW and UNI result in the same quality). If UNI is applied to DTFM, the resulting complexity is higher, as can be seen by isolated points on 0 dB line. However, with HTFM, the UNI complexity is less than ROW by about 15% as the pixel difference saving overcomes the data access complexity and the likelihood of making wrong decision is smaller due to the reduced estimation error variance.

Figure 9 (b) presents the same results as 9 (a) but with the number of pixel-difference operations is shown instead of the CPU clock cycle, and the average energy of residue pixel instead of the reconstructed PSNR degradation. It can be seen that the savings measured in terms of number of pixel-difference are always greater by about 5% because this measure does not take the complexity of search strategy, the test (DTFM or HTFM test), and parameter estimation into account. Therefore, it can be seen that the DTFM with UNI yields lower number of pixel-difference operations than ROW, but the actual CPU clock cycle is higher. It can also be seen that the decrease in reconstructed frame quality is less than the increase in residue energy. This happens because the resulting smoother motion field requires fewer bits for coding and therefore allows more bits to be used to code the residue, increasing the overall quality. Since, as mentioned earlier UNI performs better than ROW, in Fig 10(a) and 10(b), we only show complexity-distortion using UNI HTFM for 2-D Log search and ST1 search, respectively.

B. Scalability

Figs.9 and 10 illustrate the computational scalability of the HTFM. In order to obtain a complexity-distortion curve we plot the complexity and distortion pair at different P_f values (ranging from 0.05 to 0.30 for UNI, and 0.01 to 0.20 for ROW.) For ROW we have to select lower P_f due to the underestimation of σ_i^2 by $\tilde{\sigma}_i^2$, and thus the real probability of error is larger than the targeted P_f. The statistics are updated every GOP of size 15. Both complexity and distortion are normalized by the complexity and distortion values of ROW DTFM.





(a) (b)

Fig. 10. Complexity-distortion of UNI HTFM with variance estimation on-the-fly of (a) 2-D Log search and (b) ST1 search. The axes are clock cycle and PSNR degradation normalized/compared to the 2-D Log search (a) or ST1 search (b) with ROW DTFM. The complexity reduction is up to 45% and 25% at 0.05 dB degradation for 2-D Log and ST1 search, respectively.

We can see that even though the gain is not as large as in the ES case, the complexity reduction can still be achieved with FS algorithms. For example, we achieve a 45% complexity reduction with around 0.05 dB loss for 2-D Log search and a 25% complexity reduction for ST1 search. In terms of subjective quality, we observe no perceptual difference between DTFM and the HTFM at this level of degradation. In all experiments, one can see that the complexity-distortion performance of HTFM on the "football" sequence is the worst because the high motion content results in high MAD estimation error variance. Therefore, ideally the HTFM works the best for sequence with low MAD estimation error variance. In other words, the residue has to be relatively smooth, which implies smooth moving object with a smooth background content.

C. Temporal Variation

Fig. 11, 12 and 13 show frame by frame speedup in clock cycle average of "mobile" and "football" using ES, 2-D Log search and ST1 search, respectively. Speedup is computed by comparing with the original FS algorithm (2-D Log or ST1 search) *without FM*. The P_f for HTFM is set to 0.1 or 10% for ES, 20% and 30% for 2-D Log and ST1 search, respectively. One can see that with fast model parameter estimation which takes place in the first P-frame of a GOP (size 15), we still perform almost as well as DTFM. By comparing Figs. 11 and 13, we can see that with an efficient FS algorithm, the extra speedup from DTFM is smaller, and thus speedup from HTFM algorithm is more difficult to get. Otherwise, the P_f can be set to larger value for greater speedup but that comes with the price of relatively larger distortion increase. It can also be seen from Fig. 13 that in some frames, the DTFM results in slower motion estimation (speedup less than 1). This is because the candidates being evaluated by ST1 are all good, thus resulting in almost no early terminations. This case is not observed by HTFM because the probabilistic testing still terminates candidates. In such case, the P_f can be set to as high as 40-50% without much PSNR degradation since any of the candidates evaluated by the FS are equally good.

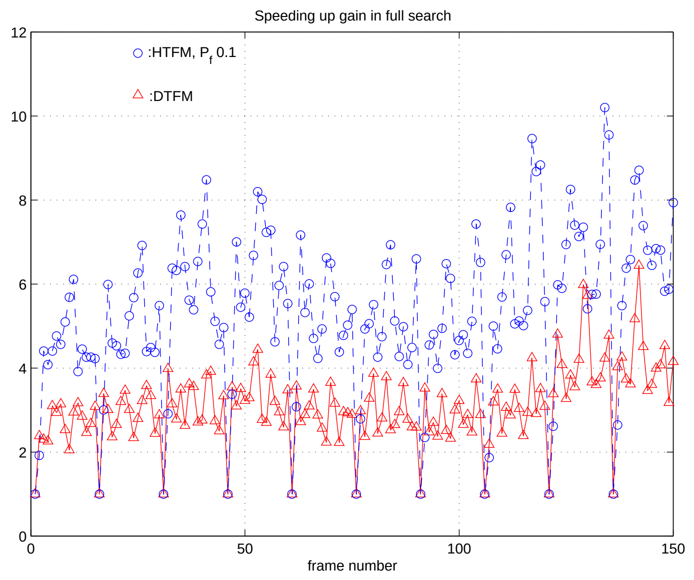


Fig. 11. Frame-by-frame speedup factor for ES using ROW and 'Δ': DTFM, and 'o': ROW HTFM with $P_f = 0.1$ and 0.01 dB degradation.

D. *Overall Performance*

Table III and IV show the results in terms of *total encoding time* (seconds) needed to encode 150 frames of the 5 test sequences using i) $|B| = 128$ pixels without FM, ii) $|B| = 128$ pixels with ROW DTFM iii) $|B| = 64$ pixels with ROW DTFM, iv) $|B| = 32$ pixels with ROW DTFM, v) UNI HTFM $P_f = 20\%$, vi) UNI HTFM $P_f = 30\%$, and vii) UNI HTFM $P_f = 40\%$ (all HTFM results are based on 128 pixels). Note that for DTFM with

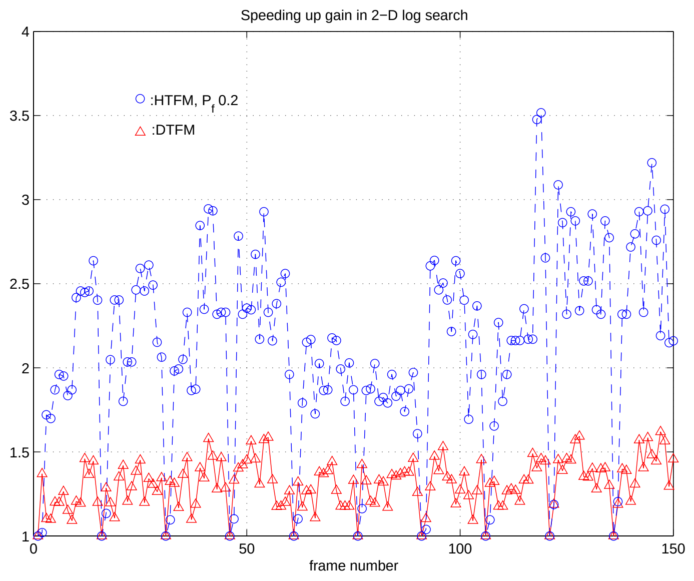


Fig. 12. Frame-by-frame speedup factor for 2-D Log search using ROW and '*': no FM, 'Δ': DTFM, and 'o': ROW HTFM with $P_f = 0.2$ and 0.04 dB degradation.

$|B| = 64$ (and 32), the number of pixels in each stage is 8 (4) and the number of stages reduces to 8 (8) while in HTFM the number of pixels used in each stage and number of stages are fixed to 8. Once again, we can see that the relative speedup due to FM is much more significant when using ES rather than a FS. Furthermore, we can see that in general our HTFM with $P_f = 30\%$ provides speedup as good as or better than using 64 pixels with ROW DTFM but with less distortion.

VI. Conclusion

In this paper, we have proposed a fast ME by using FM in a variable complexity framework, in which the complexity is input-dependent and is adjustable by the degree of probability to make wrong decision. We formalize the problem with the assumption of the knowledge of the distribution of the MAD estimation error. Then hypothesis testing is applied to minimize the probability of error. We call this novel algorithm HTFM. We presented fast "on the fly" model parameters estimation to cope with statistical change within a single sequence and can be universally applied to any sequences and any FS. For a very efficient FS, we achieve even further complexity reduction beyond the already fast result.

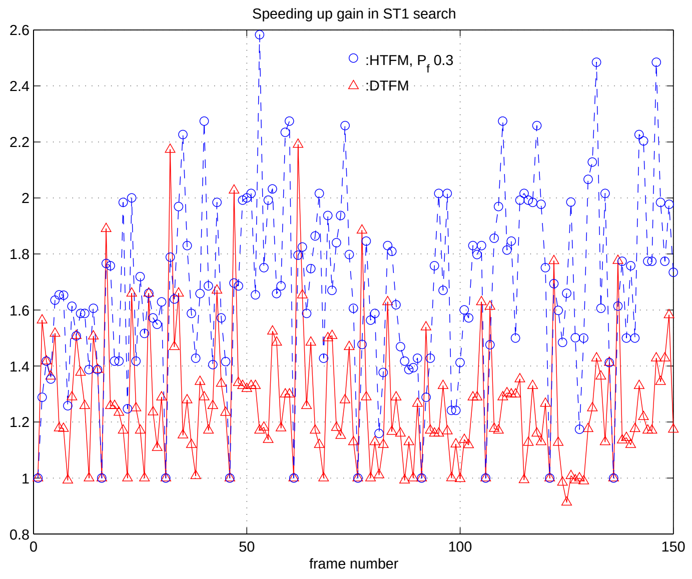


Fig. 13. Frame-by-frame speedup factor for ST1 algorithm using ROW and '*': no FM, 'Δ': DTFM, and 'o': ROW HTFM with $P_f = 0.3$ and 0.12 dB degradation.

Acknowledgments

The authors would like to acknowledge the support of Intel Corporation, for the donation of a Pentium II PC that was used in all the experiments. The authors would also like to thank Andrea Basso and Amy Reibman from AT&T Labs, New Jersey, for their collaboration in an initial version of this work [1] and their valuable comments.

References

[1] K. Lengwehasatit, A. Ortega, A. Basso, and A. Reibman, "A novel computationally scalable algorithm for motion estimation," in *Proc. of VCIP'98*, San Jose,CA, January 1998, vol. SPIE-3309, pp. 68–79.

[2] K. Lengwehasatit and A. Ortega, "Complexity-distortion tradeoffs in vector matching based on probabilistic partial distance techniques," in *Proc. of Data Compression Conference, DCC'99*, Snowbird, UT, Mar. 1999.

[3] J. Mitchell, W. Pennebaker, C. E. Fogg, and D. J. LeGall, *MPEG Video Compression Standard*, Chapman and Hall, New York, 1997.

[4] Study Group 16, "H.263 video coding for low bit rate communication," Tech. Rep., ITU-T, September 1997.

[5] B. Liu and A. Zaccarin, "New fast algorithms for the estimation of block motion vectors," *IEEE Trans.Cir.Sys. for Video Tech.*, vol. 3, pp. 148–157, April 1993.

[6] J.R. Jain and A.K. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. on Comm.*, vol. COM-29, pp. 1799–1808, December 1981.

[7] R. Srinivasan and K.R. Rao, "Predictive coding based on efficient motion estimation," *IEEE Trans.Comm.*, vol. COMM-33, pp. 888–895, August 1985.

[8] R. Li, B. Zeng, and M.L. Liou, "A ,new three-step search algorithm for block motion estimation," *IEEE Trans.Circ.Sys. for Video Tech.*, vol. 4, pp. 438–442, August 1994.

[9] L.-K. Liu and E. Feig, "A block-based gradient descent search algorithm for block motion esitmation in video coding," *IEEE Trans.Circ.Sys. for Video Tech.*, vol. 6, pp. 419–422, August 1996.

[10] J. Y. Tham, S. Ranganath, and M. Ranganath, "A novel unrestricted center-biased diamond search algorithm for block motion estimation," *IEEE Trans. on Circ. and Sys. for Video Tech.*, vol. 8, no. 4, pp. 369–377, August 1998.

[11] M. Bierling, "Displacement estimation by hierarchical block matching," in *Proc. of VCIP'88*, 1988, vol. 1001, pp. 942–951.

[12] F. Dufaux and M. Kunt, "Multigrid block matching motion estimation with an adaptive local mesh refinement," in *Proc. of VCIP'92*, 1992, vol. 1818, pp. 97–109.

[13] S. Zafar, Y. Q. Zhang, and B. Jabbari, "Multiscale video representation using multiresolution motion compensation and wavelet decomposition," *IEEE J. on Sel. Areas in Comm.*, vol. 11, pp. 24–35, January 1993.

[14] Y. Q. Zhang and S. Zafar, "Motion-compensated wavelet transform coding for color video compression," *IEEE Trans. on Circ. and Sys. for Video Tech.*, vol. 2, pp. 285–296, September 1992.

[15] K. M. Uz, M. Vetterli, and D. J. LeGall, "Interpolative multiresolution coding of advanced television with compatible subchannels," *vid*, vol. 1, no. 1, pp. 86–99, March 1991.

[16] J. Chalidabhongse and C.-C. Kuo, "Fast motion vector estimation using multiresolution-spatio-temporal correlations," *IEEE Trans. on Circ. and Sys. for Video Tech.*, vol. 7, no. 3, pp. 477–488, June 1997.

[17] J.-B. Xu, L.-M. Po, and C.-K. Cheung, "Adaptive motion tracking block matching algorithm for video coding," *IEEE Trans. on Circ. and Sys. for Video Tech.*, vol. 9, no. 7, pp. 1025–1029, October 1999.

[18] C.-D. Bei and R. M. Gray, "An improvement of the minimum distortion encoding algorithm for vector quantization," *IEEE Trans. on Comm.*, vol. COM-33, no. 10, pp. 1132–1133, October 1985.

[19] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Pub., 1992.

[20] Y.-C. Lin and S.-C. Tai, "Fast full-search block-matching algorithm or motion-compensated video compression," *IEEE Trans. on Comm.*, vol. 45, no. 5, pp. 527–531, May 1997.

[21] Z.-L. He, K.-K. Chan, C.-Y. Tsui, and M. L. Liou, "Low power motion estimation design using adaptive pixel truncation," in *In Proc. of Int'l Symp. on Low Power Electronics and Design 1997*, pp. 167–172.

[22] V. Bhaskaran B. Natarajan and K. Konstantinides, "Low-complexity block-based motion estimation via one-bit transforms," *IEEE Trans.Circ.Sys. for Video Tech.*, vol. 7, pp. 702–706, August 1997.

[23] X. Lee and Y.-Q. Zhang, "A fast hierarchical motion-compensation scheme for video coding using block feature matching," *IEEE Trans.Circ.Sys. for Video Tech.*, vol. 6, pp. 627–634, December 1996.

[24] M. Khosravi and R.W. Schafer, "Low complexity matching criteria for image/video applications," *ICIP'94*, pp. 776–780, November 1994.

[25] M.-J. Chen, L.-G. Chen, T.-D. Chiueh, and Y.-P. Lee, "A new block-matching criterion for moion estimation and its implementation," *vid*, vol. 5, no. 3, pp. 231–236, June 1995.

[26] V. L. Do and K. Y. Yun, "A low-power vlsi architecture for full-search block-matching motion estimation," *IEEE Trans. on Circ. and Sys. for Video Tech.*, vol. 8, no. 4, pp. 393–398, August 1998.

[27] V. Bhaskaran and K. Konstantinides, *Image and Video Compression Standards: Algorithms and Architectures 2nd Edition*, Kluwer Academic Publishers, Boston, 1997.

[28] P. Pirsch, N. Demassieux, and W. Gehrke, "VLSI architecture for video compression - A survey," *Proceedings of IEEE*, vol. 83, no. 2, pp. 220–246, February 1995.

[29] H. Yeo and Y. H. Hu, “A novel modular systolic array architecture for full-search block matching motion estimation,” *vid*, vol. 5, no. 5, pp. 407–416, October 1995.

[30] K. Lengwehasatit and A. Ortega, “Distortion/decoding time trade-offs in software dct-based image coding,” in *Proc. of ICASSP'97*, Munich, Germany, Apr 1997.

[31] K. Lengwehasatit and A. Ortega, “Dct computation based on variable complexity fast approximations,” in *Proc. of ICIP'98*, Chicago, IL, Oct. 1998.

[32] F. Kossentini and Y.-W. Lee, “Computation-constrained fast mpeg-2 encoding,” *IEEE Signal Proc. Letters*, vol. 4, pp. 224–226, August 1997.

[33] F. Chen, J.D. Villasenor, and D.S. Park, “A low-complexity rate-distortion model for motion estimation in h.263,” *Proc.ICIP'96*, vol. 2, pp. 517–520, 1996.

[34] F.-H. Cheng and S.-N. Sun, “New fast and efficient two-step search algorithm for block motion estimation,” *IEEE Trans. on Circ. and Sys. for Video Tech.*, vol. 9, no. 7, pp. 977–983, October 1999.

[35] “University of British Columbia, Canada, TMN H.263+ encoder version 3.0,” .

[36] MPEG Software Simulation Group, “MPEG2 video codec version 1.2,” .

TABLE III

Total time for encoding 150 frames and PSNR.

sequence	FM	ST1		Log		ES	
		sec.	PSNR	sec.	PSNR	sec.	PSNR
mobile	128 pels	37.78	32.46	43.79	32.15	472.75	32.37
	DTFM (128 pels)	35.42	32.46	39.33	32.15	152.37	32.37
	DTFM (64 pels)	33.63	32.12	35.54	31.37	98.25	31.95
	DTFM (32 pels)	32.58	31.93	33.59	30.97	71.05	31.69
	HTFM 20%	34.44	32.45	35.14	32.16	84.59	32.36
	HTFM 30%	33.78	32.42	34.50	32.14	80.39	32.34
	HTFM 40%	33.52	32.35	34.11	32.08	77.88	32.27
football	128 pels	39.05	40.43	44.72	40.05	449.64	40.52
	DTFM (128 pels)	37.75	40.43	42.19	40.05	231.80	40.52
	DTFM (64 pels)	34.40	40.24	36.34	39.77	129.63	40.32
	DTFM (32 pels)	32.69	40.16	33.62	39.68	88.31	40.23
	HTFM 20%	36.39	40.33	36.70	39.89	102.43	40.46
	HTFM 30%	35.16	40.26	35.64	39.85	93.38	40.43
	HTFM 40%	34.25	40.19	34.92	39.79	87.94	40.39
flower	128 pels	37.75	35.30	44.67	34.62	466.02	35.31
	DTFM (128 pels)	35.75	35.30	40.69	34.62	148.64	35.31
	DTFM (64 pels)	33.19	35.04	35.71	33.77	94.88	35.02
	DTFM (32 pels)	32.37	34.93	33.41	33.29	68.80	34.91
	HTFM 20%	34.67	35.28	35.81	34.63	92.80	35.29
	HTFM 30%	34.00	35.25	35.24	34.61	87.78	35.25
	HTFM 40%	33.50	35.19	34.66	34.58	82.23	35.17

TABLE IV

cont. Total time for encoding 150 frames and PSNR.

sequence	FM	ST1		Log		ES	
		sec.	PSNR	sec.	PSNR	sec.	PSNR
cheer	128 pels	42.61	35.04	47.30	35.00	464.34	35.01
	DTFM (128 pels)	40.67	35.04	43.34	35.00	175.66	35.01
	DTFM (64 pels)	37.96	34.94	39.26	34.88	107.44	34.91
	DTFM (32 pels)	36.77	34.91	37.37	34.85	77.58	34.86
	HTFM 20%	39.21	35.01	39.30	34.96	92.25	35.00
	HTFM 30%	38.31	34.97	38.60	34.93	86.29	34.99
	HTFM 40%	37.66	34.92	37.85	34.90	83.32	34.97
bicycle	128 pels	42.28	35.17	47.58	34.44	461.28	35.19
	DTFM (128 pels)	40.47	35.17	45.04	34.44	224.13	35.19
	DTFM (64 pels)	36.78	34.89	38.99	34.07	128.43	34.84
	DTFM (32 pels)	35.06	34.75	36.16	33.96	87.87	34.68
	HTFM 20%	37.66	34.96	39.56	34.39	102.76	35.16
	HTFM 30%	36.89	34.86	38.42	34.35	93.73	35.11
	HTFM 40%	36.18	34.75	37.53	34.29	87.95	35.04

List of Figures

List of Tables