Complexity-Distortion Tradeoffs in Vector Matching based on Probabilistic Partial Distance Techniques*

Krisda Lengwehasatit and Antonio Ortega
Integrated Media Systems Center
Signal and Image Processing Institute
University of Southern California
Los Angeles, CA 90089-2564

Abstract

In this paper we consider the problem of searching for the best match for an input among a set of vectors, according to some predetermined metric. Examples of this problem include the search for the best match for an input in a VQ encoder and the search for a motion vector in motion estimation based video coding. We propose an approach that computes a partial distance metric and uses prior probabilistic knowledge of the reliability of the estimate to decide on whether to stop the distance computation. This is achieved with a simple hypothesis testing and the result, an extension of the partial distance technique of Bei and Gray provides additional computation savings at the cost of a (controllable) loss in matching performance.

1 Introduction

Many lossy compression applications require that the encoder perform vector matching of some kind. A classical example is *vector quantization* (VQ) where the objective is to find for each input vector the best match among all the vectors in the codebook. The complexity of the search then depends on the dimension of the vectors as well as the size of the codebook. A second example of vector matching can be found in *motion estimation* (ME) in video coding. In this case the goal is to find for each block in the current video frame the best match among blocks within a predetermined search region in the previous frame. The information about the best match is then sent to the decoder in the form of a motion vector.

*This work has been funded in part by the 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, by the National Science Foundation under grant MIP-9804959, and by an equipment grant from the Intel Corporation.

In both VQ and ME, vector matching for a given input is performed by computing, for each vector, a metric to determine how good the match is. Then the matching proceeds through all the elements of the codebook or the blocks in the search region until the best (i.e., lowest metric) match is found.

In the VQ case, the matching metric normally used is the *Euclidean distance* between the input and the codeword, defined as $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x}-\mathbf{y}\|^2 = \sum_{m=1}^{k}(x_m-y_m)^2$, where $\mathbf{x}$ is the input vector of dimension k and $\mathbf{y}$ is a codeword in the codebook $C = \{\mathbf{y_1}, \mathbf{y_2}, ..., \mathbf{y_n}\}$ of size n. Therefore, the *quantization rule* is $Q(\mathbf{x}) = \mathbf{y_i}$ if $\|\mathbf{x} - \mathbf{y}_i\|^2 < \|\mathbf{x} - \mathbf{y}_j\|^2$, $\forall j \neq i$.

In the ME case, the *sum of absolute difference* (SAD) is a more popular choice as a matching metric because of its lower computation requirements. The SAD of a motion vector $\vec{mv}$ is defined as $S(\vec{mv}) = \sum_{(n_x,n_y)\in B} |I_t(n_x, n_y) - I_{t-1}(n_x + mv_x, n_y + mv_y)|$ where $I_t(n_x, n_y)$ is the intensity level of pixel (n_x, n_y) at frame t, B is normally the set of all pixels in the block being coded and $\vec{mv} = (mv_x, mv_y) \in M$, where M is the allowable range of MV and thus also defines the size of the search region. For example a typical value for M is [-16,15.5]x[-16,15.5]. The SAD normalized by $|B|$, the size of B, is the *mean absolute difference* (MAD). We will use both SAD and MAD throughout this paper.

The metric computation dominates the overall complexity of the system. Basically, the system has to compute the metric between an input and all codewords (in VQ) or all possible MVs (in ME) in order to find the best match codeword or MV with the lowest metric. Obviously the sources of complexity are two, namely, (i) comparing the input with all the vectors in the codebook or search region and (ii) computing a metric for each of the vectors, where the complexity of the metric computation depends on the dimension of the vectors.

Thus, fast vector matching techniques, both for VQ and ME, can be achieved by attacking each of these complexity factors, i.e., by reducing the number of searched codewords or MVs, and by reducing the number of operation required for metric computation. We refer to the first class of techniques as *fast search* (FS) and to the second class of techniques as *fast matching* (FM). Note that fast techniques (FS or FM) can be designed to provide the same result as an exhaustive search, or to be suboptimal, given that the faster matching comes at the expense of a slightly degraded performance. Throughout this paper we will call "optimal" matching a matching technique that computes the exact metric values for all the candidates.

FS techniques for ME rely on searching fewer points within the search region by imposing a structure to the search. Furthermore, the search can be initialized with a "good" candidate so that the search can be restricted to a relatively small region close to the candidate input. The search can then be stopped whenever a "good enough" match has been found. Most of these ME FS techniques are suboptimal, i.e., they do not find the best match within the search region, although the degradation as compared to exhaustive search tends to be small. Examples of fast search for ME include [1], [2], [3], [4] and [5]. In the VQ case, there are many FS algorithms that achieve the optimal solution ([6], [7], [8], [9], [10]). In these approaches codewords that are known to be too far from the input are eliminated first, then the exhaustive search is performed on the rest of the codewords. An alternative approach is to build

the codebook with a structure that allows a faster search. The best known approach of this type is the tree-structured VQ (TSVQ) algorithm [11].

While the gains of FS approaches are well known and documented, less attention has been given to FM techniques. In the VQ case, to the best of our knowlegde, only the *partial distance* technique of Bei and Gray [12] has been proposed. This technique consists of incrementally (i.e., one dimension at a time) computing the metric and stopping the computation as soon as the partial distance exceeds the best distance so far. This approach can provide significant gains (e.g. a factor of 3-4 speed up in some cases) without any suboptimality. As for the ME case, most research on FM ME has focused on finding a computationally cheaper distance metric that preserves the ability to discriminate between good and bad matches. However, the partial distance method can also be applied to speed up the SAD computation. In fact this approach is succesfully implemented in many software video encoders (e.g. [13],[14]). Other approaches for fast SAD computation with suboptimal solution can be found in [15], where the number of pixels used to calculate the SAD is reduced by subsampling pixels in a macroblock. Other alternative distance metrics used in ME can be found in [16], [17], [18], etc.

In this paper, we focus on the FM approach by presenting a novel *hypothesis testing fast matching* (HTFM) approach for vector matching which extends the partial distance techniques by allowing a probabilistic stopping criterion. We start with a review of the partial distance technique in Section 2. Given that the result of the vector search using partial distance is optimal, we refer to these techniques as *deterministic testing fast matching* (DTFM). In Section 3, we present a general framework for HTFM. The basic idea is to use the partial distance estimate to predict the total distance. This prediction will be subject to an error which can be modeled. The HTFM approach will stop the metric computation when it has *sufficient confidence*, given the probabilistic model, that the total distance will exceed the best match so far. Note that this approach was originally proposed for ME in [19], although here we present an improved version that includes adaptive "on the fly" parameter estimation (see also [20]). In this paper we generalize the idea to include VQ. In Section 3.1 and 3.2 we apply HTFM to ME and VQ, respectively, and provide results of complexity saving versus degradation in matching performance. Conclusions are provided in Section 4.

2 Deterministic Testing Fast Matching

We present the partial distance matching method in the context of VQ as described in [12]. The basic idea is to test an incomplete calculation of the distance with the "best-found-so-far" distance while the computing process is not finished yet. Based on the monotonicity of the distance (i.e., total distance increases with dimension), we can terminate the computation if the partial distance is already greater than the "best-found-so-far" distance. Then the current codeword being compared cannot be optimal and we move on to evaluate next codeword. We formalize the partial distance for both VQ and ME as follows.

We define the partial distance between input $\mathbf{x}$ and codeword $\mathbf{y}$ as $d_{k'}(\mathbf{x},\mathbf{y}) = \sum_{m=1}^{k'}(x_m - y_m)^2$, i.e., the sum of square difference up to dimension k' between input vector $\mathbf{x}$ and codeword $\mathbf{y}$. It is obvious that $d_{k'}(\mathbf{x},\mathbf{y}) \leq d(\mathbf{x},\mathbf{y})$ for $\forall k'$. Next we define the "best-found-so-far" codeword, $\mathbf{y}^*$ for input vector $\mathbf{x}$ with their corresponding distance, d^*, as $\mathbf{y}^* = \arg\min_{\mathbf{y_i}\in C_s} d(\mathbf{x},\mathbf{y}_i)$. C_s are subset of codewords in the codebook C whose distances to $\mathbf{x}$ have been tested.

Similarly, in the ME case the partial SAD computation can be defined by dividing the set B in k subsets $B_1, ..., B_k$ where $B_i \subset B_j$ for $i < j$ and $B_k = B$. Therefore, the partial SAD computed at stage k' for block B with MV $\vec{mv}$ is $S_{k'}(\vec{mv}) = \sum_{(n_x,n_y)\in B_{k'}} |I_t(n_x,n_y) - I_{t-1}(n_x + mv_x, n_y + mv_y)|$, and $S_{k'}(\vec{mv}) \leq S(\vec{mv})$ for $\forall k'$. The "best-found-so-far" MV, $\vec{mv}^*$ for block B, $\vec{mv}^* = \arg\min_{\vec{mv}\in M_s} S(\vec{mv})$, and its corresponding SAD, S^*, are defined similarly. M_s is analogous to C_s, i.e., it includes the blocks that have been tested thus far. We now show the partial distance algorithm for ME; its VQ counterpart is similar.

Algorithm 1 (Partial Distance Search (DTFM))

Step 1: *At the beginning of motion search for a particular block, initialize S^* to a very large number and set $\vec{mv}^* = (0,0)$.*

Step 2: *The next $\vec{mv}$ in the search region (or the one dictated by any applicable FS strategy) is considered. Set $k' = 0$. If there are no more vectors to evaluate, return $\vec{mv}^*$.*

Step 3: *Compute $S_{k'}(\vec{mv})$.*

Step 4: *If $k' < k$, go to step 5, else go to step 6.*

Step 5: *If $S_{k'}(\vec{mv}) \geq S^*$, go to step 2. Otherwise, $k' = k' + 1$ and go to step 3.*

Step 6: *If $S(\vec{mv}) < S^*$, $S^* = S(\vec{mv})$ and $\vec{mv}^* = \vec{mv}$. Go to step 2.*

The complexity savings of this technique come from the possibility of early termination in Step 5. This reduction is data dependent, i.e., it varies according to the nature of the input. Obviously if a FS technique is also used (and determines the order in which the vectors are tested in Step 2) then the gain achievable will depend on the FS strategy. In general, the more efficient FS is, the less computational savings can be achieved with DTFM: since we are already considering only candidate vectors that are more likely to provide "good" matches early termination will not be as likely.

3 Hypothesis Testing Fast Matching

One drawback of the DTFM approach is that it does not provide any *computation scalability*, i.e., DTFM achieves the same solution as optimal matching but does not allow us to obtain a faster solution at the cost of some quality reduction. In this section, we investigate algorithms which possess computational scalability with gradual reductions in complexity coming at the cost of corresponding reductions in the matching quality. These algorithms are based on the Hypothesis Testing Fast Matching (HTFM) approach, which uses hypothesis testing to decide when to terminate the search at Step 5 in Algorithm 1.

The basic idea is to first estimate the distance metric from the partial distance at stage k'. The difference between the actual and estimated distance can be modeled as a random variable with a certain probability density function (pdf). In addition to the partial distance testing, we can add a decision rule to determine whether our estimate based on the partial distance allows us to decide that the final total distance will be greater than the "best-found-so-far" distance. We make this decision by taking into account the reliability of the estimate as determined by our probability model. If we have sufficient confidence we can terminate the search, otherwise we continue computing the metric at the next stage (i.e., including more vector dimensions.) Thus HTFM combines partial distance test and hypothesis test. We present two examples of applications of HTFM to ME and VQ.

3.1 Motion Estimation

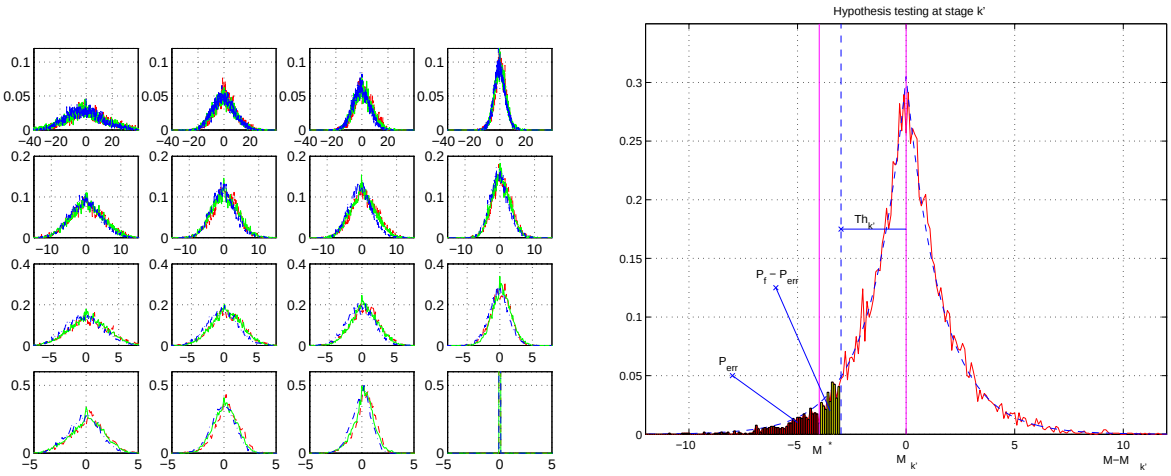
Let us consider the MAD as our distance metric. Thus, the "best-found-so-far" MAD and the partial MAD at stage k' can be directly defined as $M^* = S^*/|B|$ and $M_{k'} = S_{k'}/|B_{k'}|$, respectively where $|\cdot|$ denotes size of the set. We start by formulating the estimate of MAD from partial MAD. By considering the distance and partial distances as random processes, the best estimation in mean square sense is the expected value of distance given partial distance i.e. $E\{M|M_{k'}\}$. From our observation $E\{M|M_{k'}\}$ can be approximated by $M_{k'}$, with the estimation error, $E\{(M - M_{k'})^2\}$ getting smaller as k' grows as shown in Figure 1(a). This figure shows histograms from a typical sequence of the estimation error $M - M_{k'}$ at various values of k' where we partition B into sixteen subsets $B_1 \subset B_2 \subset ... \subset B_{16} = B$. In this paper, B_i is the set of pixels in the first i row of a macroblock[1].

Intuitively it should be obvious that, as confirmed by our experiments, the more pixels we use, the more accuracy we will achieve, on the average, for our SAD estimation, and therefore the better MV we will obtain if only the partial SAD is used. Moreover, the resulting histograms can be seen as estimates of the pdf of the estimation error. For ME, in most cases the pdf can be well approximated by a Laplacian distribution. Our next procedure is to design a decision rule for hypothesis testing. The decision is *given $M_{k'}$ and M^*, we want to decide between two hypotheses: (i) there is a high likelihood that the final M_k will be larger than the current M^* and thus we stop computing, or (ii) we do not have a reliable enough estimate and thus we proceed to compute $M_{k'+1}$, i.e., compute the $k'+1$-th stage for the metric, and test again.* Thus our hypothesis are, given $M_{k'}$

$$\begin{aligned} H_0 &: M_k \geq M^* \\ H_1 &: M_k < M^* \end{aligned}$$

The one parameter that determines the performance of the decision rule is the probability of false alarm, $P_{err} = Pr(H_1|\ H_0$ was chosen). Therefore, our problem is to find the optimal decision rule such that $P_{err} < P_f$, where P_f is a given target probability of false alarm.

[1]The choice of partition method also plays an important role in the system. However, due to the space limitation, we refer the interested readers to [20] for details.



(a) (b)

Figure 1: (a) histograms of estimation error using $M_{k'}$ as an estimate for M at 16 stages stepping from left to right and top to bottom. (b) estimation error obtained from 'solid line': histogram of training data and 'dashed line': parametric modelling. Hypothesis testing to determine whether to terminate the search given M^* and $M_{k'}$ at stage k' with certain the probability of false alarm, P_f, that determines the threshold, $Th_{k'}$.

Thus the condition $P_{err} < P_f$ can be written as follows,

$$P_{err} = \Pr(H_0|M_{k'}) \cdot \Pr(M_k < M^*|M_{k'}, M^*) \leq P_f,$$

and the optimal decision rule is

$$Pr(M_k < M^*|M_{k'}, M^*) \underset{H_1}{\overset{H_0}{\lessgtr}} P_f$$

Since $p_{(M|M_{k'})}(y) = p_{(M-M_{k'})}(y-M_{k'})$, the left-hand side can be written as $\int_{-\infty}^{M^*} p_{(M-M_{k'})}(y-M_{k'})dy$. For the right-hand side, we find Th_i such that $P_f = \int_{-\infty}^{-Th_i+M_{k'}} p_{(M-M_{k'})}(y-M_{k'})dy$ then the decision rule becomes

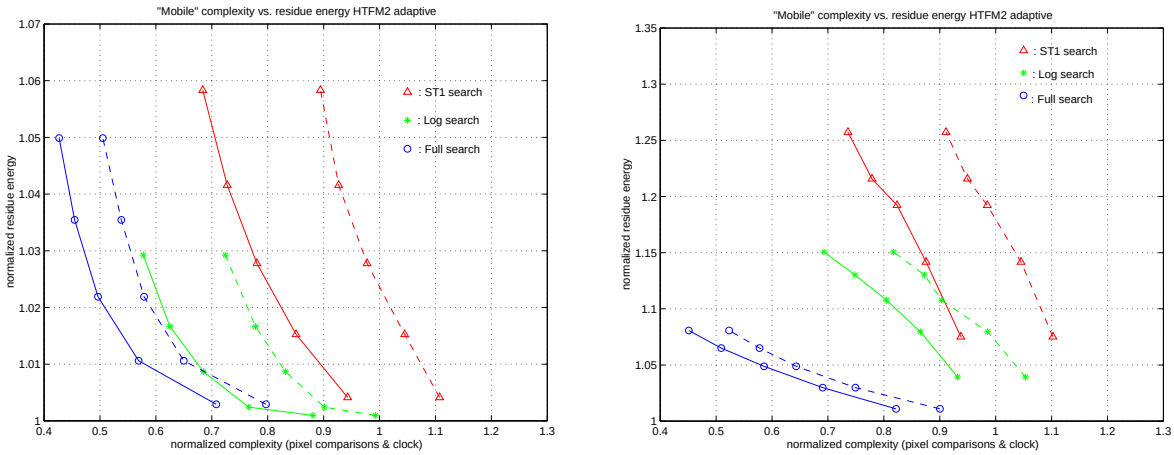
$$M_{k'} - M^* \underset{H_1}{\overset{H_0}{\gtrless}} Th_i \tag{1}$$

which can be visualized in Figure 1(b). Now we can add (1) to the partial distance test in section 2. With Laplacian distribution model, $p_{(M-M_{k'})}(y) = \frac{\lambda_{k'}}{2} e^{-\lambda_{k'}|y|}$ where $\lambda_{k'}$ is Laplacian parameter for stage k', the threshold can be written as

$$Th_{k'} = \begin{cases} -\frac{1}{\lambda_{k'}} ln(2P_f) & P_f \leq 0.5 \\ -\frac{1}{\lambda_{k'}} ln2(1-P_f) & P_f > 0.5 \end{cases} \tag{2}$$

In general, we can design hypothesis testing for each stage differently, i.e., having a different P_f at each stage. However, for simplicity, we set P_f to a constant at all stages in our experiment. However, even with the same P_f, $Th_{k'}$ varies depending on $\lambda_{k'}$ of each stage. In [20], we address the issue of obtaining a fast estimation of the parameter $\lambda_{k'}$ for any sequence while performing motion search. Here we use a fast parameter approximation derived in [20] that adaptively estimates the $\lambda_{k'}$ at every GOP (size 15 frames).

The result from applying HTFM to ME is shown in Fig.2 where the distortion is the energy of the residue blocks, and the complexity is measured in terms of number of pixel comparison as well as the CPU clock cycles spent in our software simulation on the PentiumII 300 MHz. Both distortion and complexity units are normalized by the results of DTFM. We traverse along complexity-distortion curve by increasing P_f from 0.01 to 0.2. In Figure 2, we show our HTFM applied to exhaustive search, 2-D Log search [1] and ST1 search [5].



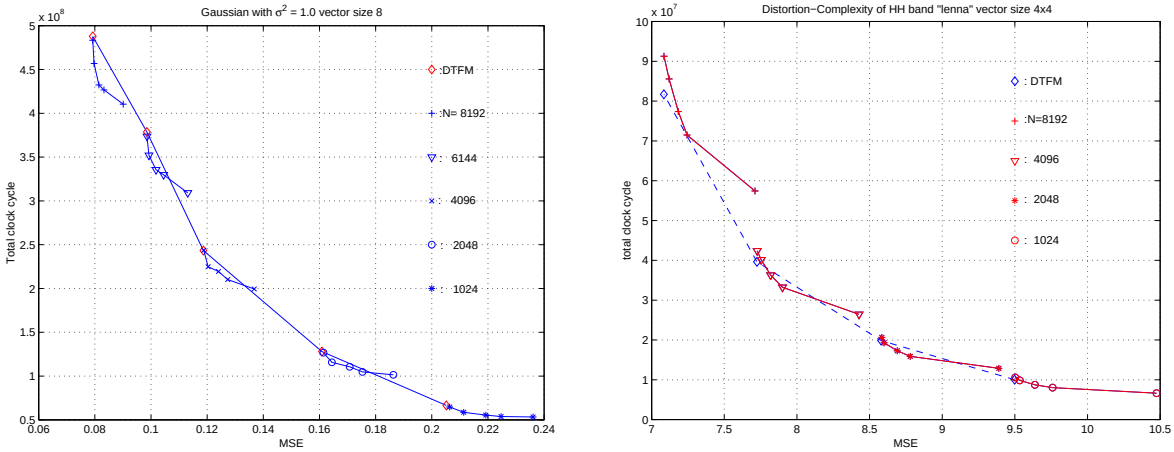
(a) (b)

Figure 2: Complexity-distortion of 150 frames of (a) "mobile& calendar" (b) "football" sequence where 'dashed': CPU clock cycle, 'solid': no. pixel comparisons.

3.2 Vector Quantization

As in the ME case, we first change the unit of the distance to be a per-dimension distance. In this case we define $\tilde{d}(\mathbf{x},\mathbf{y}) = d(\mathbf{x},\mathbf{y})/k$ and $\tilde{d}_{k'}(\mathbf{x},\mathbf{y}) = d_{k'}(\mathbf{x},\mathbf{y})/k'$. Our first goal is to estimate $\tilde{d}(\mathbf{x},\mathbf{y})$ from the $\tilde{d}_{k'}(\mathbf{x},\mathbf{y})$. Then we find the estimation error pdf and model it such that we can design a decision rule based on hypothesis testing to meet the targeted probability of false alarm. As in the ME case we found that $E\{\tilde{d}|\tilde{d}_{k'}\}$ can be well approximated by $\tilde{d}_{k'}$ For simplicity, we can also approximate the estimated error pdf using a Laplacian distribution, as in ME case, and design the decision rule based on this assumption. The Laplacian parameter in this VQ case is obtained from the training vectors. The complexity-distortion result using HTFM is shown in Figure 3, which shows the result of DTFM at different codesizes, as well

as HTFM curves corresponding to one codebook size with P_f ranging from 0.05 to 0.55. Figure 3(a) is for an i.i.d. Gaussian source with unit variance and Figure 3(b) is the high-high band from subband decomposition of "lenna" image. In both cases, the codebook is designed using the LBG [21] algorithm from training vectors which are i.i.d. Gaussian and typical images, respectively.



(a) (b)

Figure 3: Complexity-distortion of HTFM VQ with vector size (a) 8 for i.i.d. source and (b) 16 (4x4) for high-high band of "lenna" image.

We can see that unlike the ME case, in which the equivalent codebook size is fixed by the search region, in this VQ case the size of the codebook can be chosen to meet a complexity-distortion requirement. Note, however, that in order to operate in a computation scalable mode, in the DTFM case the codebook size has to be modified, while scalability is achieved with a constant codebook size for the HTFM case. In Figure 3, complexity-distortion performance achieved by HTFM is approximately the same as that achieved with DTFM using different codebook size within a certain range. This is due to several factors. First, the speedup from using DTFM alone is already large, i.e., about 3 to 4 times faster than original MSE computation. More than 90% of the distance computations are terminated early by DTFM, and most of the terminations occur at early stages. Second, the HTFM introduces more overhead cost for testing while providing more early termination at first few stages. However, the number of additional early termination is relatively small compared to the overall gain achieved by DTFM. And finally, the vector dimension in VQ case is still far less than in the ME case (16x16 macroblock). Thus, a few extra early termination at a little bit earlier stages is outweighted by the overhead cost for extra testing. Therefore, in order to get the maximum speedup performance, in our experiment for subband data VQ, Figure 3(b), we apply the HTFM test to the first one quarter of dimensions and simply use the DTFM test for the rest. As a conclusion, the HTFM for VQ, even though it does not provide a significant speedup over DTFM, provides computational scalability for a given fixed codebook, while scalability can only be achieved with different codebook sizes with DTFM.

4 Conclusion

We categorized the fast algorithms for vector matching into 2 classes, FS and FM. We focused on the partial distance method for FM and gave a brief overview of the algorithm. As an extension of the partial distance approach we present a general framework for HTFM which consists of first distance estimation from partial distance, then modeling the error pdf and finally finding a decision rule to meet a desired target probability of false alarm. We apply our HTFM to two vector matching applications, ME and VQ. Our results show computational scalibility by trading off the quality of best matching with complexity budget. The only caveat is that we have to carefully apply HTFM in the case of VQ since the overhead of more test may outweigh the early termination gain. For future work, a more practical tree-structured VQ (TSVQ) will be considered to explore all the possibility of speeding up gain for low dimension vector.

References

[1] J. Jain and A. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. on Comm.*, 1981.

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

[3] R. Li, B. Zeng, and M. 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.

[4] 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.

[5] J. Chalidabhongse and C.-C. Kuo, "Fast motion vector estimation using multiresolution-spatio-temporal correlations," *IEEE Trans.Cir.Sys. for Video Tech.*, April 1997.

[6] D. Y. Cheng, A. Gersho, B. Ramamurthi, and Y. Shoham, "Fast search algorithms for vector quantization and pattern matching," in *Proc. of ICASSP'84*, (San Diego), pp. 9.11.1–9.11.4, March 1984.

[7] D. Y. Cheng and A. Gersho, "A fast codebook search algorithm for nearest-neighbor pattern matching," in *Prof. of ICASSP'86*, vol. 1, pp. 265–268, April 1986.

[8] M. Orchard, "A fast nearest neighbor search algorithm," in *Proc. of ICASSP'91*, (Toronto, Canada), pp. 2297–2300, May 1991.

[9] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Comm. ACM*, vol. 18, pp. 509–517, September 1975.

[10] I. Katsavounidis, C.-C. J. Kuo, and Z. Zhang, "Fast tree-structured nearest neighbor encoding for vector quantization," *IEEE Trans. on Image Processing*, 1996.

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

[12] 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, pp. 1132–1133, October 1985.

[13] University of British Columbia, Canada, "Tmn h.263+ encoder version 3.0."

[14] Mpeg Software Simulation Group, "Mpeg2 video codec version 1.2."

[15] 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.

[16] V. B. 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.

[17] 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.

[18] E. Chan, A. Rodriguez, R. Gandhi, and S. Panchanathan, "Experiments on block-matching techniques for video coding," *Multimedia Systems 2(5)*, pp. 228–241, 1994.

[19] K. Lengwehasatit, A. Ortega, A. Basso, and A. Reibman, "A novel computationally scalable algorithm for motion estimation," in *Proc. of VCIP'98*.

[20] K. Lengwehasatit and A. Ortega, "Hypothesis testing based fast matching for motion estimation," *IEEE Trans. on Circ. and Sys. for Video Tech.*, 1999. To be submitted.

[21] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. on Comm.*, 1980.