COMPUTATIONALLY SCALABLE PARTIAL DISTANCE BASED FAST SEARCH MOTION ESTIMATION

Krisda Lengwehasatit[*]

PacketVideo Corp.
Core Engineering Division
10350 Science Center Drive
San Diego, CA 92121
krisda@packetvideo.com

Antonio Ortega [†]

Integrated Media Systems Center
Department of Electrical Engineering-Systems
University of Southern California
Los Angeles, California 90089-2564
ortega@sipi.usc.edu

ABSTRACT

In this paper, we present a class of algorithms that use a partial distance metric to speedup the motion estimation process. The partial distance metric is used within the motion search to eliminate unlikely candidates through a thresholding process that enables computation scalability. We also propose a multiresolution variant which further reduces the complexity. Our results are comparable to state of the art approaches present a regular structure and are computationally scalable.

1. INTRODUCTION

The task of motion estimation is to find the best match for the block currently coded from a region in the previously reconstructed frame. The matching metric typically used is the sum of absolute difference (SAD) defined as

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

where $I_t(n_x,n_y)$ is the intensity level of the (n_x,n_y)-th pixel at frame t, B is the set of all pixel in the macroblock (size 16x16) of interest, and $\vec{mv}$ is a candidate motion vector in a search region Γ. The "best" motion vector, i.e., the one having the minimum matching metric denoted by $MV^*(\Gamma, B)$, can be found through exhaustive search.

$$MV^*(\Gamma, B) = \arg\min_{\vec{mv}\in\Gamma} SAD(\vec{mv}, B). \quad (2)$$

[*]This work has been done in part while this author was at University of Southern California.

[†]This work has been supported in part by Integrated Media Systems Center, a National Science Foundation Engineering Research Center, cooperative Agreement No. EEC-9529152 and by the National Science Foundation under grant MIP-9804959.

It can be seen that the major source of complexity in motion estimation comes from the metric computation that must be performed for every candidate. Thus, there are basically two approaches to speedup the motion estimation; either reducing the number of candidate vectors for which the metric will be computed, i.e., *fast search (FS)*, or reducing the complexity of the metric calculation itself, i.e., *fast matching (FM)*. In the FS approach, only a subset of promising candidates in the search region, $\gamma \subset \Gamma$ in (2), is considered. Examples of FS approaches include the 2-D log search [1], the new three step search [2] and the diamond center-based search [3]. These algorithms are based on the monotonicity assumption of the matching metric as a candidate vector moves further away from the global minimum. A good initial point can also be used to reduce the risk of being trapped in local minima. Hierarchical [4], multiresolution [5] and spatio-temporal [6] techniques have been used to find a good initial point by exploiting the correlation of neighboring candidate vectors and motion vector from neighboring blocks for better initial candidate. In the FM approach[1], the basic idea is to reduce the metric computation complexity by using a computationally cheaper lower bound (e.g.,[7]) or trying to use a smaller set of pixels in a macroblock, i.e., using $\beta \subset B$ in (1) (e.g., the well known *partial distance search (PDS)* by Bei and Gray [8]), while maintaining the accuracy of the SAD. In the PDS, the metric calculation stops once it is clear that the metric of a tested vector will exceed that of the best found-so-far one. In our previous work [9], we proposed a probabilistic version of the PDS which gives a suboptimal solution but allows the computation to be scalable.

Unlike our previous work [9] in which only the FM approach is considered, in this paper we propose algorithms that combine both FM and FS techniques. Thus, we use the partial SADs (PSADs) to limit the number of candidates

[1]We consider only algorithms that are based on the conventional SAD, although our work could be extended to other additive cost functions.

which are computed. Our algorithms use thresholds as a way to trade off the computational savings and the motion estimation accuracy which, in turns, affects the coding efficiency.

2. PDS-BASED CANDIDATE ELIMINATION

We first describe the PDS algorithm. Let us denote the partial SAD of candidate $\vec{mv}$ by $PSAD_i = SAD(\vec{mv}, x_i), \forall i$ where the x_i are embedded subsets of pixels, i.e., $x_i \subset B$ and $x_i \subset x_j$ for $i < j$. Therefore, PSAD is the SAD metric computed on a subset of pixels. The PDS algorithm can be summarized as follows. Candidates are evaluated sequentially. When a given candidate is evaluated, we know the best-so-far candidate with its corresponding best-so-far SAD, SAD_{bsf}. We can compare $PSAD_i$ of the current candidate with the SAD_{bsf}. If $PSAD_i \geq SAD_{bsf}$, it is clear that the SAD will be larger than SAD_{bsf}, and therefore the current candidate can be eliminated. If $PSAD_i < SAD_{bsf}$, we compute the next stage $PSAD_{i+1}$ and compare to SAD_{bsf} again until the termination occurs or the complete SAD is computed. Then the process is repeated for the next candidate.

The choice of the sequence of subsets x_i can be made such that the terminations occur earlier. We have found that by using a set of pixels uniformly distributed around the macroblock, the PDS computation is better than, say, having x_i as one row of pixels. Therefore, in our implementation we use 16 stages, each of which is a 16-pixel uniformly distributed set for x_i's (see also [9]). Note that if the candidate with the smallest total SAD is coincidentally computed first, it is certain that the computation of other candidates will be terminated at their earliest stage. We will refer to this case as an ideal case for minimal complexity. However, in practice, the best candidate is obviously not known beforehand. Only a good initial candidate can be guessed. However, if only the pixel difference operator is considered, the ideal complexity can be achieved without knowing the best candidate beforehand by using Algorithm 1.

Algorithm 1 (Ideal Candidate Elimination (ICE-PDS))

Step 1: *Compute the $PSAD_1$ of every candidate in the search region. Set the current stage of every candidate to 1.*

Step 2: *Find candidate whose current stage is less than 16 (total number of stages) such that its PSAD is smallest.*

Step 3: *If there exists such candidate, update this candidate's PSAD to the next stage and repeat Step 2. Otherwise, return the candidate with smallest SAD (current stage is 16).*

This algorithm can seen to be analogous to the traditional greedy bit allocation in which one bit at a time is iteratively allocated to the transform coefficient showing the maximal distortion improvement. Here we take the most promising candidate at any given stage (given its PSAD) and spend on it the additional computation to update it. However, this approach is complex since it requires finding the minimum PSAD at each step and each candidate is accessed several times. We can, however, reduce the complexity overhead by limiting the number of times each candidate is considered, i.e., reducing the number of PSAD calculation stages. In this paper, we propose a two-step Candidate Elimination PDS (see Alg. 2). In the first step we compute the partial SAD of all candidates up to stage m. Then the candidate with minimal $PSAD_m$ is considered first by continuing SAD calculation from stage m. Then other candidates' SADs are computed *using the PDS*, i.e., at every stage after the m-th stage its PSAD is compared with the best-so-far SAD. In fact, the PDS can also be used in the first step when $PSAD_m$'s of every candidate are computed. In this case, the stage number where termination occurs (before m), $n(j)$, must be kept as the starting point for the second step SAD calculation. The algorithm is summarized as follows.

Algorithm 2 (2-Step Candidate Elimination (2-CE-PDS))

Step 1: *With a preselected m value, find $MV^*(\Gamma, x_m)$ using PDS and keep the partial SAD, $SAD(\vec{mv}_j, x_{n(j)})$ where $n(j) \leq m$ is defined above for the j-th candidate.*

Step 2: *Update $SAD(MV^*(\Gamma, x_m), B)$ first, set it to SAD_{bsf} and use PDS to compute $SAD(\vec{mv}_j, B)$ $\forall j, \vec{mv}_j \neq MV^*(\Gamma, x_m)$, starting from $PSAD_{n(j)}$.*

No sorting is needed as in ICE-PDS. Only finding minimum in the first step is needed. This method does not employ an initialization technique like other FS algorithms. The second step initial candidate is based solely on the first step PSAD. With this algorithm, the memory corresponding to the search area has to be accessed only twice. From our experiments, we found that using the value $m = 1$ provides good complexity result since the PSAD computation in the first step is small. However, with the spirally outward order of the full search ([10],[11]) the complexity savings from 2-CE-PDS or even ideal ordering PDS are not so significant.

3. SUBOPTIMAL CE-PDS

To further reduce the complexity, we can limit the number of candidates considered in the next stage by comparing the PSADs with a threshold. Consider the 2-step algorithm with a first stop at step m. In the second step, after updating the best PSAD to obtain SAD_{bsf}, we can surmise that the actual minimal SAD would have the value around (strictly equal or less) the current SAD_{bsf}. We can then set up a threshold with the value proportional to the SAD_{bsf} scaled down to the number of stages in the first step. Specifically, the threshold, T, is defined as

$$T = t \cdot SAD(MV^*(\Gamma, x_m), B) \cdot m/16$$

where t is a control parameter to adjust the tradeoff between the complexity and quality, e.g., if t is small, fewer candidates (those with smaller $PSAD_m$) are considered in the second step thus reducing complexity. However acuracy may be lost since the best candidate could have been eliminated from the motion search. As can be seen, the threshold is proportional to the SAD scaled to the stage m. The computationally scalable algorithm can be summarized as follows.

Algorithm 3 (2-Step Threshold Fast CE-PDS (2T-FCE))

Step 1: *Find $MV^*(\Gamma, x_m)$ using PDS and keep the partial SAD, $SAD(\vec{mv}_j, x_t(j))$ $\forall \vec{mv}_j \in \Gamma$ where $t(j) \leq m$ is the termination stage for the j-th candidate.*

Step 2: *Update $SAD(MV^*(\Gamma, x_m), B)$ first, set it to SAD_{bsf} and find $MV^*(\gamma, B)$ using PDS by updating $SAD(\vec{mv}_j, B)$ for candidates $\vec{mv}_j \in \gamma = \{\vec{mv}_j | SAD(\vec{mv}_j, x_t(j)) < T\}$ from $PSAD_{t(j)}$.*

We can further reduce the number of candidates considered by exploiting the spatial correlation of SADs among neighboring candidates, as in multiresolution motion estimation approaches ([4, 5]). We assume that an initial candidate with full SAD is computed, SAD_{init}. The set of thresholds at each step/resolution, $T(r)$ for $r = 1, ..., R$, are computed based on SAD_{init} scaled to the r-th resolution (i.e., $T(r) = t \cdot SAD_{init} \cdot 1/4^{(R-r)}$). At each resolution, a subset of candidates corresponding to the resolution grid of the motion field which coincides with the subset of pixels used for the particular PSAD stage is considered. The first resolution consists of x_1, the second resolution corresponds to x_1,x_2,x_3,x_4 and the third corresponds to the whole set B. Let γ_r be such a set of candidates on the subsampled grid at the r-th resolution. Let $\phi_r(\vec{mv})$ be the set of motion vectors in the neighborhood of $\vec{mv}$ in the r-th resolution.

The partial SAD of candidates in a particular resolution determine which candidates (and their surrounding neighbors) will be considered in the finer resolution. Note that there can be many surviving candidates as a result of using the PSAD thresholding instead of selecting only one survivor at each resolution as in the conventional MR. We propose two variants, *breadth-first (MR1-FCE)* in which the current resolution determines the stage of PSAD calculation, whereas in the second variant, *depth-first (MR2-FCE)*, the SAD calculation can be completed even if we are not in the finest resolution. The details of these multiresolution algorithms are summarized as follows.

Algorithm 4 (Breadth-First Multiresolution (MR1-FCE))

Step 1:*Find the SAD of the initial motion vector, set it to SAD_{bsf}. Compute $T(r)$ for $r = 1, ..., R$ as stated above. Set $r = 1$.*

Step 2:*Compute $SAD(\vec{mv}, x_1)$ using PDS, $\forall$ $\vec{mv} \in \gamma_1$. $r \leftarrow r + 1$.*

Step 3:*If $SAD(\vec{mv}, x_{n(r-1)}) < T(r-1)$, i) use PDS to update to $SAD(\vec{mv}, x_{n(r)})$ and ii) compute the neighboring $SAD(\vec{v}, x_{n(r)})$ using PDS $\forall \vec{v} \in \phi_r(\vec{mv})$.*

Step 4:*$r \leftarrow r + 1$. If $r < R$, repeat step 3. Otherwise, go to step 5.*

Step 5:*If $SAD(\vec{mv}, x_{n(R-1)}) < T(R-1)$, i) use PDS to update to $SAD(\vec{mv}, x_{n(R)})$ and ii) compute $SAD(\vec{v}, x_{n(R)})$ using PDS $\forall$ $\vec{v} \in \phi_R(\vec{mv})$.*

Step 6:*Return MV_{bsf} as the best motion search result.*

Algorithm 5 (Depth-First Multiresolution (MR2-FCE))

Step 1:*The same as Step 1 in MR1-FCE.*

Step 2:*The same as Step 2 in MR1-FCE.*

Step 3:*If $SAD(\vec{mv}, x_{n(r-1)}) < T(r-1)$, i) use PDS to update to $SAD(\vec{mv}, x_{n(R)})$ and remove $\vec{mv}$ from the list of candidates to be processed, and ii) compute the neighboring $SAD(\vec{v}, x_{n(r)})$ using PDS $\forall$ $\vec{v} \in \phi_r(\vec{mv})$.*

Step 4:*$r \leftarrow r + 1$. If $r < R$, repeat step 3. Otherwise, go to step 5.*

Step 5:*The same as Step 5 in MR1-FCE.*

4. EXPERIMENTAL RESULTS

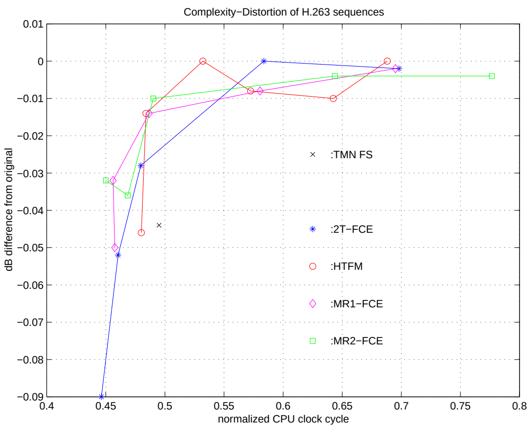

Fig. 1. Complexity-Distortion of motion estimation part of TMN's H.263 encoder averaged over several test sequences. The complexity unit is the number of clock cycles normalized by the original PDS. t is varied for computational scalability of the proposed algorithms (2T-FCE, MR1-FCE and MR2-FCE). 'TMN FS' is TMN's fast search and 'HTFM' is the algorithm in [9]. The result of conventional MR algorithm is 0.347 normalized clock with -0.436 dB.

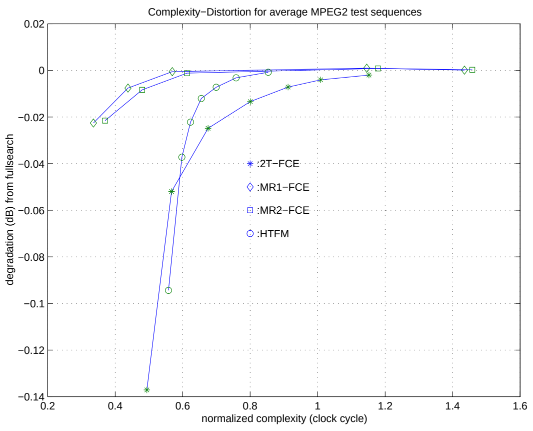

Fig. 2. Complexity-Distortion of motion estimation part of MSSG's "mpeg2encode" encoder averaged over several test sequences. The complexity unit is the clock cycles normalized by the original PDS. t is varied for computational scalability of the proposed algorithms (2T-FCE, MR1-FCE and MR2-FCE). 'HTFM' is the algorithm in [9]. The result of conventional MR algorithm is 0.217 normalized clock with -0.915 dB.

Figures 1 and 2 show the result of our proposed algorithms compared with others using H.263 encoder [11] and MPEG2 encoder [10], respectively. The complexity-distortion tradeoffs are obtained from varying the parameter t for threshold. Each of the complexity-distortion operating pointd are averaged among test sequences which are "Miss america", "Salesman", "Foreman", "Mother & daughter" and "Grandmother" for H.263, and "Mobile & calendar", "Football", "Cheer", "Bicycle", and "Flower" for MPEG2.

It can be seen that our algorithms are better than TMN's fast search and the two MR algorithms give better C-D results than those of HTFM in [9]. It can be seen that for mpeg2encode the MR approach gives much better results since the initial candidate is the zero MV as compared to the H.263 encoder's result in which the initial candidate is the median of MVs of neighboring block. As a result, there is not much difference between the HTFM and the proposed algorithms using TMN's H.263 coder. Thus it can be concluded that the proposed algorithms work well for the case where no initial good candidate is provided among a large set of candidates with high magnitude of redundancy (correlations) since they use the only information available, i.e., the partial SAD. Therefore, they are suitable for a particular applications such as the first P-frame after an I-frame, scenes with random motions, or high accuracy motion estimation with large search region.

5. CONCLUSIONS

We have proposed algorithms that exploit the partial metric information in speeding up the motion search. We addressed the importance of initial candidate and then we proposed a method that use the PSAD to obtain the good candidate. Furthermore, we proposed an algorithm that eliminates a number of candidates considered while computing the distance metric. We also proposed multiresolution based algorithms that also take advantage of spatial correlation among neighboring candidates. The experimental results on H.263 and MPEG2 encoders show better complexity-distortion performance than our previous work in [9] in the case of fullsearch.

6. REFERENCES

[1] 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.

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

[3] 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.

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

[5] 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.

[6] J. Chalidabhongse and C.-C. Kuo, "Fast motion vector estimation using multiresolution-spatio-temporal correlations," *IEEE Trans. on Circ. and Sys. for Video Tech.*, pp. 477–488, June 1997.

[7] 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.

[8] 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.

[9] K. Lengwehasatit and A. Ortega, "Probabilistic partial distance fast matching algorithms for motion estimation," Accepted for publication *in IEEE Trans on Circ. and Sys. for Video Tech.*, 1999.

[10] Mpeg Software Simulation Group, "MPEG2 video codec version 1.2," *http://www.creative.net /∼tristan/MPEG/mssg/mpeg2vidcodec_v12.tar.gz.*

[11] "University of British Columbia, Canada, TMN H.263+ encoder version 3.2,".