MOTION ESTIMATION FOR LOW POWER VIDEO DEVICES

Christophe De Vleeschouwer and Tord Nilsson

IMEC - DESICS, Kapeldreef 75, 3001 Leuven, BELGIUM

Ericsson Radio Systems AB, SWEDEN

ABSTRACT

We propose a block motion estimation (ME) algorithm that meets high quality requirements and allows for cost efficient VLSI realizations. It relies on a set of rules common to all recent fast BMA algorithms and has been designed in order to allow for easy and prolific data reuse.

1. INTRODUCTION

Companies seek to combat the obvious drawbacks of running power-hungry and computationally intensive multimedia applications on battery powered portable devices. The emergence of markets for such applications has created the need for low-power VLSI video compression processors. The compression technology has been defined by H.263 and MPEG-4 standards. A common and major feature of these standards is block motion estimation/compensation techniques that reduce temporal redundancy. For each block of the current frame, the block-matching algorithm (BMA) considers a search window in a previously reconstructed frame. In this window, it searches for the block that is closest to the current block and defines a motion vector (MV) which points to it. Various measures, as the mean squared error (MSE), the mean absolute error (MAE) or sum of absolute difference (SAD), can be used in the matching criterion. These measures make the BMA a computationally intensive task and a data-dominated system [1], which means that the overall system performance, including its power consumption, is dominated by data transfer and storage operations. Many fast BMA algorithms have been proposed. They reduce the number of positions to check in the search window and, consequently, reduce the amount of manipulated data. An overview of these schemes is provided in Section 2. It highlights a small set of common design rules. Considering hardware, the variable distances between candidate locations and the often unpredictable data requirement complicate the control scheme, lower the efficiency of the computation kernel and make it difficult to reuse data for reducing the number of external memory accesses. In contrast, as in [2], our algorithm, presented in Section 3, has been designed with both the set of common design rules and the very large scale integration (VLSI) implementation constraints in mind. Our major concern has been to enable a maximal data reuse while still minimizing the amount of computation.

2. FAST MOTION ESTIMATION ALGORITHMS

A number of conventional algorithms have been proposed as an alternative to the computationally intensive full search BMA. They include the 2-D logarithm search, the three step search (TSS), the conjugate directional search, the cross search, etc. These algorithms track the direction of a minimal distortion measure, i.e. a best matching. For all of them, a logarithmic step search is performed. The distortion is first computed in a number of positions surrounding the center of the search and fixed by a pre-defined step size. The step size is then halved and the search is continued around the position of minimal distortion. The procedure finishes when the step size becomes one. The variable distances between candidate locations break the regularity of data flow. Even careful designs result in architectures that require a large number of input ports and/or poorly enable reuse of data [3]. Extensive studies on the statistical behavior of real-world sequences motion vectors (MVs) [4] have shown that their distribution is centrally biased and that there exist a high spatial and temporal correlation among vectors of adjacent blocks. These observations have been utilized to improve conventional algorithms or to guide the development of a new generation of adaptive schemes. These schemes are based either on the adaptive growing of a search area [5], or on the propagation of a search pattern [6],[7]. In both cases, the checking point pattern is not uniformly and a priori allocated. It depends on the sequence features and allows for computation savings when possible.

From this large set of contributions, one can draw some **general rules to guide the design of efficient motion estimation algorithms**:

- The algorithm should exploit the **spatial correlation of motion vectors** within a frame. Often this is done through the prediction of the search starting point.
- To exploit the MVs **center-biased distribution**, the checking points should be chosen on a pattern that is more compact around the initial and central position.
- For faster **convergence towards a local optimum**, the checking points should be chosen adaptively, in the direction of an improvement of the matching criterion (Gradient descent algorithms).
- The search should **stop as soon as possible**, i.e. once the matching criterion is “good enough” (see 3.3.2).

3. PROPOSED ALGORITHM DESIGN

3.1 Center-biased and spatially correlated motion vector distribution

In order to exploit the center-biased motion vector distribution, as in [5], the search locations have been split into square patterns around a central location (Figure 1).

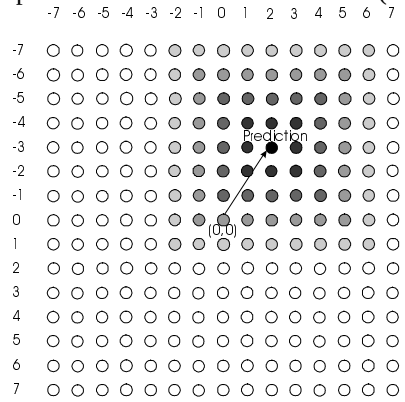

Figure 1: Definition of search zones. Darkest zones are searched first.

The central position is predicted from the neighboring (macro-) blocks motion vectors, according to the MVs differential encoding scheme of MPEG-4 [9]. Figure 2 compares the optimal motion vectors distribution around the origin and around the predicted MV. As expected, the distribution is more compact around the predicted value. The optimal MV will be found faster if the search is started from the predicted value.

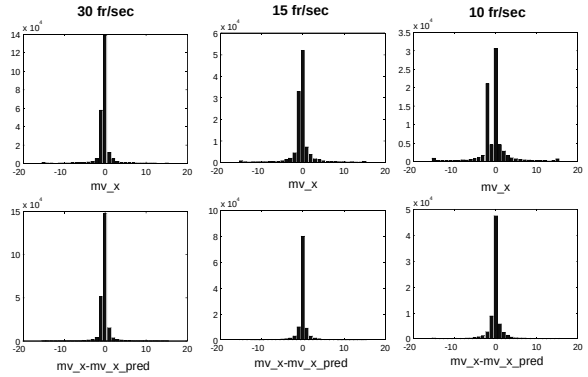

Figure 2: Statistical distribution of the horizontal component of motion vectors (full search algorithm, search range = 16). Top: MVs distribution around the origin of the search window. Bottom: MVs distribution around the predicted position. Graphs result from the aggregation of statistics gathered on three sequences (Foreman, Mother&Daughter, and Mobile&Calendar) encoded in CIF format, with two constant quantization parameter (QP = 5 and 20).

In our algorithm, the search center is determined. In contrast to [5], it is not chosen among a set of candidates that are "randomly" distributed in the search window. Such "random" accesses make the data reuse difficult to implement and increase the I/O bandwidth.

3.2 Gradient descent or directional search

For faster convergence towards a local optimum, the checking points have to be chosen adaptively, in the direction of an improvement of the matching criterion. To check this assertion, we have performed the search depicted on Figure 1 on a large set of sequences. When the matching criterion improved on a square, that position is used to predict the optimal position in the next square. Actually, the predicted position is the one pointed by the vector originated in the center of the search and passing through the last optimal position. Figure 3 draws, for a given square, histograms of the distance measured between the predicted and the actual optimal position, if there is an improvement on the considered square. It appears that the improvement is most likely located around the predicted position. It leads to the search strategy depicted on Figure 4. Only the neighbors of the predicted position are investigated before going to the next square. The search is stopped as soon as there is no further improvement. On Figure 4, a 5-points neighborhood is considered.

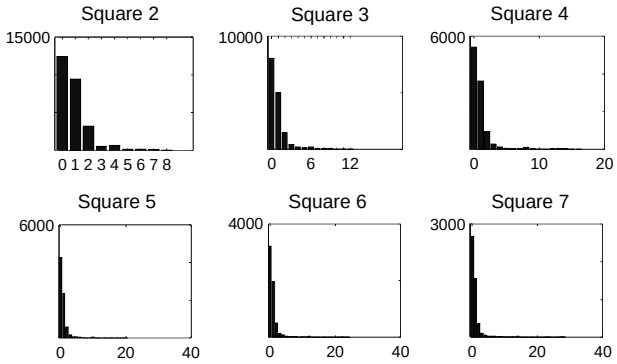

Figure 3: Distance to prediction for CIF format. A set of sequences (M&D, Foreman, Mobile) has been considered at 15 fps, for QP = 5 and 20.

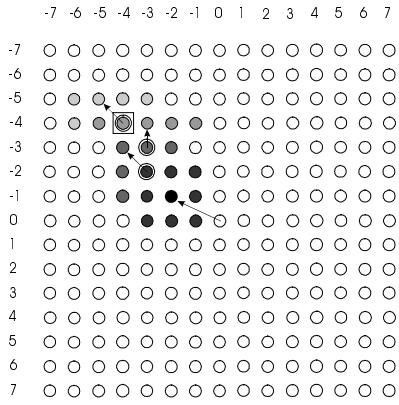

Figure 4: Directional search (DS).

The drawback of propagation or gradient descent methods is that, for sequences with large, random and non-correlated motion vectors, they are easily trapped in a local minimum. A way to avoid that drawback for our directional search (DS) is to modify the search according to Figure 5. DS is completed with a global search (DS+G). All positions on the square on which no improvement has been detected are considered. If an improvement is detected, the search goes on locally, until no further local improvement is obtained. If not, the search immediately stops. This method is based on the observation that, when no improvement is detected on a complete square there is little probability of further improvement. Performing the global search on a square means that we go horizontally or vertically through adjacent positions. It is suited to a cost efficient implementation as it enables easy data reuse [2].

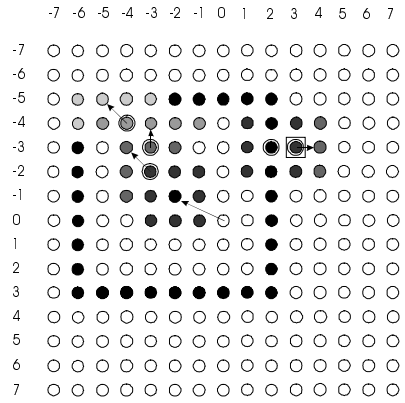

Figure 5: Local directional search terminated with a global search (DS+G).

Now that the search strategy has been defined, one can estimate its performance. The bit-rate increase due to the DS+G has been measured for a set of sequences encoded at constant quality. M&D, Foreman, Mobile&Calendar, have been encoded in CIF and QCIF formats, with 30, 15, 10 or 7.5 fps and with QP = 5 or 20. For this test-bench, the bit-rate increase remains within 4% of the size of the bitstream obtained when using a full search ME (15x15 search window). The worst results appear for sequences encoded with low quality (high QP) and low frame rate.

3.3 Stopping criteria

3.3.1 Search range vs. frame rate

An obvious way to limit the number of searched positions is to limit the size of the search window or the search range (SR). Of course, without *a priori* knowledge about the resolution and the movements in the scene, it is not possible to fix a threshold valid for any kind of sequence. Nevertheless, for a given scene, the SR should increase with the time elapsed between two frames. Indeed, for a given motion, a longer period between frames considered by the BMA results in larger displacements. In a coding context, even if the scene is captured at a constant frame rate (e.g. 30 fps), not all the frames are actually encoded. The encoder configuration or the rate control algorithm [9] may drop frames to avoid channel overload. The actual time elapsed between two successively encoded frames is thus likely to vary. REFOn graphics such as the ones from Figure 2, larger time elapsed between frames, i.e. lower frame rate, result in less compact distribution of the MVs. From such graphics, drawn for a number of sequences and frame rates, it has been possible to estimate the required search range and its dependency with the frame rate. It increases as the square root of the time elapsed between encoded frames. In practice, it is defined by:

$$Search\ Range = SR \cdot \sqrt{DROP + 1}$$

- *DROP* is the number of frames dropped between successive frames. Reference frame-rate is 30 fps.
- *SR* is a parameter depending on the frame resolution and on the movement in the scene. Typically, SR=2 (3) for QCIF (CIF).

The cost of SR limitation in terms of coding performance can be measured by the bitrate increase to encode a set of sequences at constant quality. Fig. 6 and 7 illustrate the performance/complexity trade off. The DS+G (Figure 5) is performed with different *SR* parameter. Reducing the *SR* parameter to 2 only slightly decreases the complexity for a significant loss in performances. *SR* is set to 3.

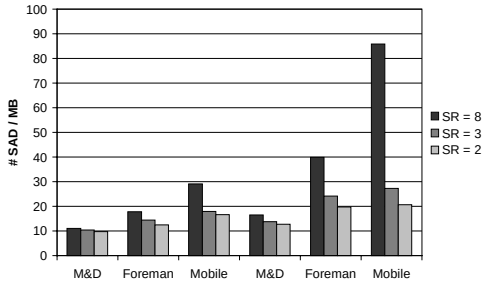

Figure 6: DS+G Computational complexity (# SAD /MB) decrease with SR parameter (SR = 2 and 3). CIF format sequences have been encoded with constant quality (QP=20). On the left (right), frame rate is 30 fps (7.5 fps).

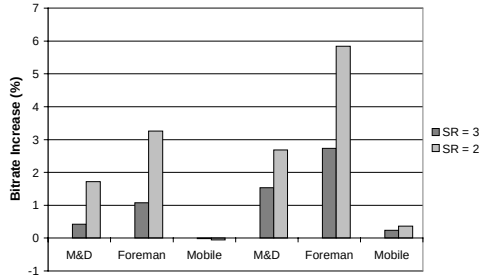

Figure 7: Bitrate increase with SR = 2 and 3. Sequences have been encoded with constant quality (QP=20). Reference is DS+G with *SR=8*. Frame rate is 30 fps (left) or 7.5 fps (right).

3.3.2 "Good enough" matching criterion

Once the motion prediction is good enough, there is no reason to continue the optimization process. Indeed, as the quantization error in the reference frame propagates to the current frame, there is little improvement to expect once the prediction error is in the range of that error. Moreover, after prediction, the error blocks are transformed and quantized for compression. The higher the quantization step and the smaller the signal energy, the higher the probability that DCT coefficients are quantized to zeros. Based on a theoretical model for the DCT coefficients[8], Pao and al. were able to link the SAD value on 16x16 blocks (SAD_{16x16}) to the probability that the block is only composed of zero quantized 8x8 subblocks (if $SAD_{16x16} < 50.QP$ it is true with 99 % probability). Similarly, the MPEG-4 community has derived a conservative threshold on the SAD computed on 8x8 blocks (if $SAD_{8x8} < 20.QP$ all coefficients are zero quantized) to reduce the encoder complexity [9]. In both works, these thresholds are used to skip DCT, quantization, and corresponding inverse operations for zero blocks. In the ME context, we do not have to be conservative. We have empirically defined a threshold $T=256+70.QP$ on the macro-block SAD (if $SAD_{16x16} < T$, the search stops). Experiments have demonstrated that this choice is not critical. As in [8] and [9], T depends on QP.

4. CONCLUSION

The DS+G has been defined (Figure 5) and search range and *SAD* thresholds have been fixed (Section 3.3). For a given quality, the performance cost is small: the bitstream increase remains within a few percent (~5% worst case) of the total size of the optimal bitstream, i.e. the one obtained with the full search. Figure 8 completes this result. It presents the average loss in dB due to our algorithm. It remains below 0.5 dB.

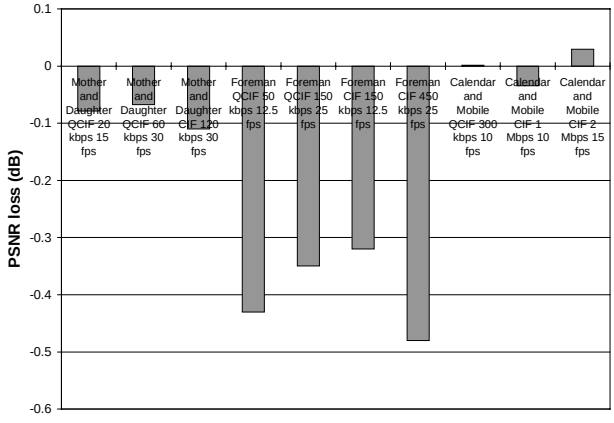

Figure 8: Coding efficiency degradation due to the proposed algorithm. Reference is the full search (search range = 16).

Additionally, during the search, checking points are adjacent, which increases the locality and allows for data reuse. This results in a decrease of the data transfers and storage and, together with the achieved complexity reduction (Figure 9), enables cost efficient VLSI realizations.

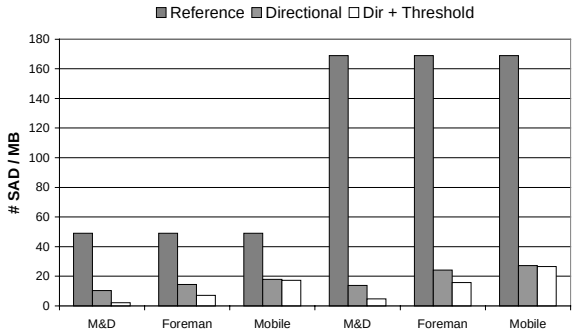

Figure 9: Number of SAD per macroblock. Sequences have been encoded with constant quality (QP=20). Left: 30 fps, right: 7.5 fps. Reference is full search with SR = 3.

5. REFERENCES

[1] P. Khun, "Fast MPEG-4 motion estimation: processor based and flexible VLSI implementations", Journal of VLSI Signal Proessing. Syst. For Signal, Image And Video Technology, vol. 23, No.1, pp. 67-92, Oct. 99.

[2] Z. He and M. L. Liou, "Design of Fast ME Algorithm Based on Hardware Consideration", IEEE Trans. CSVT, vol. 7, No. 5, pp.819-823, Oct. 97.

[3] H. Yeo and Y. H. Hu, "A Modular High-Throughput Architecture for Log. Search Block ME", IEEE Trans. on CSVT, vol. 8, No. 3, pp.299-315, June 98.

[4] B. Zeng, R. Li, and M. L. Liou, "Optimization of Fast Block Motion Estimation Algorithms", IEEE Trans. on CSVT, vol. 7, No. 6, pp.833-844, December 1997.

[5] A. M. Tourapis, O. C. Au, M.L. Liou, and G. Shen, "An Advanced Zonal Block Based Algorithm for Motion Estimation", ICIP'99, 26PO3.1, Kobe, Japan.

[6] L.-K. Liu and E. Feig, "A Block-Based Gradient Descent Search Algorithm for Block ME", IEEE Trans. on CSVT, vol. 6, No. 4, pp.419-422, Aug. 96.

[7] S. Zhu and K.-K. Ma, "A New Diamond Search Algorithm for Fast Block ME", IEEE Trans. on Image Proc., vol. 9, No. 2, pp.287-290, Feb. 2000.

[8] I-M. Pao and M.-T. Sun, "Modeling DCT coefficients for fast video encoding", IEEE Trans. on CSVT, vol. 9, No. 4, pp.608-616, June 1999.

[9] "MPEG-4 Video Verification Model version 16.0. ISO/IEC JTC1/SC29/WG11 Doc. N3312, March 2000, Noordwijkerhout.