MOTION ESTIMATION IN MPEG-2 VIDEO ENCODING USING A PARALLEL BLOCK MATCHING ALGORITHM

Daniel Grosu, Honorius Gâlmeanu
Multimedia Group - Department of Electronics and Computers
Transilvania University of Braşov
Politehnicii 1-3, 2200 Braşov, ROMÂNIA
e-mail: grosu, galmeah3@vega.unitbv.ro

Abstract

Motion estimation is the most computing-intensive component of the MPEG-2 encoding process. It refers to the action of searching for the closest prediction of a certain image block. The more accurate the prediction, the better achieved compression and quality are. The optimal solution is guaranteed by performing an exhaustive search for all possible predictions for a certain block. Such an approach this component is characterized by a polynomial time, and it is responsible for about 90% of the encoding time. The reduction of the prediction search time is performed by parallelizing the block-matching search algorithm.

In brief, a video sequence with a throughput of 50.688Mbps is MPEG-2 encoded, preserving very good image quality, at a 500Kbps bitrate. The compression ratio achieved this way is 100 to 1. The compression time was reduced from 115 sec/frame to 29 sec/frame, about 4 times faster, using 6 Pentium 166MHz Ethernet connected machines.

Keywords: MPEG, motion estimation, motion vector, block matching, macroblock, PVM.

1 Problem statement

Motion estimation refers to the way of constructing prediction for a certain block in an image, referred with the term of *macroblock*. A video sequence is formed from succesive static frames whose encoding becomes a target MPEG bitstream [6] to be sent over a computer network at a specified bitrate [7]. Each frame is formed from a dense grid of adjacent pixel macroblocks. Depending on coding pattern, each frame can be:

- intra coded (I) frame, with no references to other frames;
- predicted (P) or bidirectionally predicted (B) frame, whose macroblocks contain references to previous or both previous and next frame.

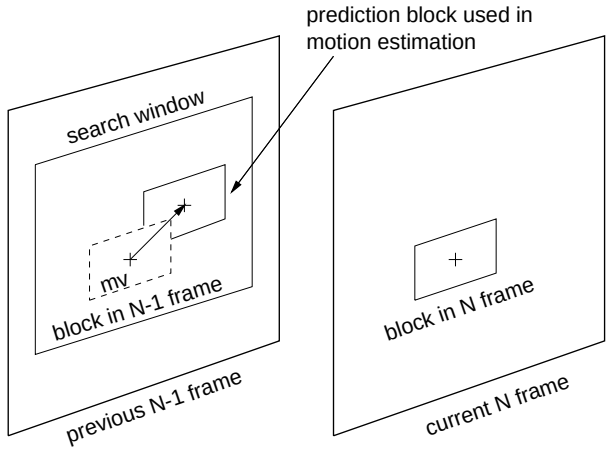


Figure 1: Motion of a macroblock through consecutive frames is estimated by the corresponding motion vector.

The article is structured the following way: first, the concept of motion estimation is described in brief. Then, there are presented existing solutions of the described problem, with their disadvantages from the MPEG coding point of view. Next, it is shown the parallel encoder model, then the experimental results. Finally, the conclusions present once more the goods and the bads of the parallel approach.

The main goal of motion estimation algorithms is the accurate determination of a prediction macroblock for which the absolute difference from the current macroblock is minimum. Such a macroblock roughly approximates the true motion that appears in a video sequence. As seen in Figure 1, the prediction macroblock is completely determined by the image it is part of, the displacement between it and the macroblock for which this prediction is determined and the corresponding prediction error. The displacement is referred with the term of *motion vector*, and it is the variable to be determined by motion estimation algorithms. The possible values for a motion vector determines the characteristic *search window*. A macroblock is coded by storing in the bitstream only the estimated motion vector and attached prediction error, efficiently coded with discrete cosine transform [1, 4]. The MPEG standard does not impose nor recommends any motion estimation algorithms for determining the motion vectors, although the proper choice of these ones greatly impacts both on compression performance and image quality.

2 Motion estimation algorithms

The existing motion in a video sequence is represented using a scalar spatio-temporal variation of image luminance intensity. Motion estimation algorithms attempt to find the best past pixel which is characterized approximately by the same luminance intensity as the one of a given pixel:

$$I(\vec{r}, t) \approx I(\vec{r} - \vec{d}, t - \Delta t) \qquad (1)$$

There are mainly three categories of motion estimation algorithms [3]:

- gradient techniques, which rely on hypothesis that image luminance is invariant along motion trajectories, which are determined by calculating the corresponding gradient of two pixel positions. Because of additional constraints involved by problem solving, motion vectors are not able to accurately describe the motion according to minimum absolute error hypothesis;

- pel-recursive techniques, which recursively minimize the prediction error, on a pixel-to-pixel basis. They rely on the same hypothesis of dense motion field, but attempts to determine the best motion vector using a pixel recursive approach. Because of this, slow motion is not accurately

represented, error resulted from determining motion vectors being propagated along the lines of a frame;

- block matching techniques are based on the matching of blocks of two consecutive images, the desired goal being to minimize the characteristic absolute error. In this case, the motion vector is not asssociated to only one pixel, but it is the same for all pixels within the block. For a block, the corresponding motion vector is the displacement between the block and another block, within the search window, that has the closest content luminance information to the that of the block for which the prediction (i.e. the best matching block) is to be determined:

$$\vec{d} = \min_{\vec{d} \in S} \sum_{\vec{r} \in B} \|I(\vec{r}, t) - I(\vec{r} - \vec{d}, t - \Delta t)\| \quad (2)$$

The absolute minimum error is obtained by performing an exhaustive search of all discrete candidate displacements (motion vectors) within a maximum displacement range, technique called full search block matching.

Although sometimes accurate, gradient-based techniques generate too much overhead information associated with motion vectors to be suited for MPEG-2 coding. The most convenient are block matching ones, because of their possibility to find the optimal macroblock, characterized by a minimum absolute error and only one motion vector.

3 Parallelization of block matching algorithm

The classic block matching algorithm requires a full search to find the best candidate for each macroblock in a frame. The time required for encoding a frame is in the order of $O(N^4)$, where N is frame dimension. In a parallel environment this time may be reduced by evenly distributing macroblocks in contiguous slices among the processors. This requires initial frame to be partitioned in slices, as shown in Figure 2. Each processor (here called *estimator*) performs an exhaustive search among all macroblocks in a defined search window for finding the best prediction of each macroblock in the slice it received. The parallel environment is represented in the chosen implementation by Parallel Virtual Machine, a library of functions that makes a UNIX computer network act as a Multiple Instruction Multiple Data parallel architecture. PVM [2] allows tasks on different machines constituting the parallel environment to communicate to each other, and also synchronizing, starting and shutting down remote processes using a message passing function library. PVM routines which perform these tasks are called by the user processes. The parallel encoder model is depicted in Figure 3.

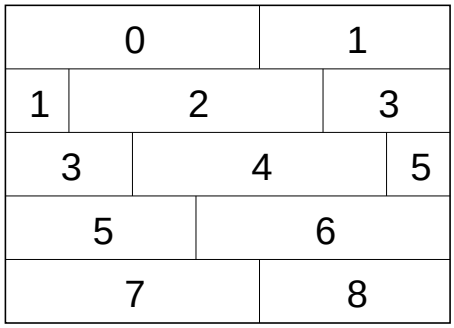


Figure 2: Each processor estimates motion for each macroblock in one slice of the initial frame. Here there are 9 slices, corresponding to 9 processors.

A cycle time t_{cycle} for encoding one frame is contributed by the following:

- the encoder reads current frame in time

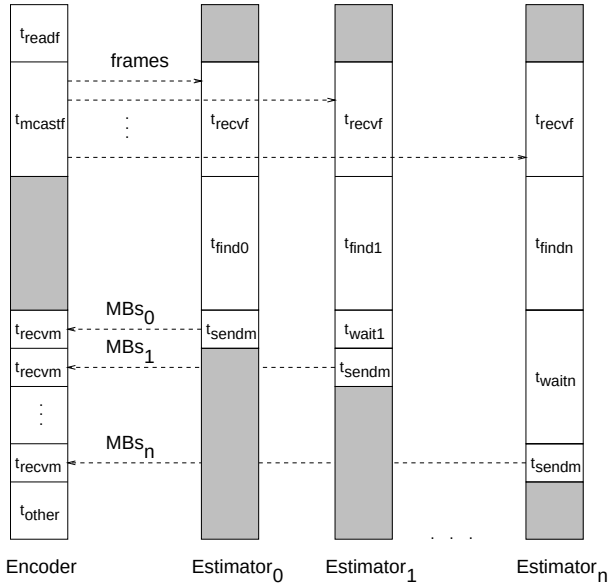


Figure 3: Master-slave model of the encoder.

t_{readf} then sends it to estimators in time $t_{mcastf} = t_{recvf}$;

- each estimator determines the solutions for all macroblocks from the slice of macroblocks it received by the time $t_{find0} \ldots t_{findn}$ since it finishes receiving frames;

- the encoder eventually collects results (i.e. motion vectors and prediction errors) for all macroblocks of frame, from every estimator, in the time $t_{recvm} = t_{sendm}$;

- idle times $t_{wait1} \ldots t_{waitn}$ exist due to the sequential way the master encoder receives results from slave estimators.

According to the master-slave model, there are two components which determines the total encoding cycle time, where $t_{cycle} = t_{calc} + t_{comm}$:

$$t_{calc} = \max(t_{find0} \ldots t_{findn}) + t_{readf} + t_{other} \quad (3)$$

$$t_{comm} = t_{mcastf} + n \cdot t_{recvf} \quad (4)$$

The variation of search window's size S and that of the number n of estimators of the parallel environment has a great impact on t_{calc} and t_{comm}:

- an increase of n determines reduction of t_{find}, in fact t_{calc}, but increases t_{comm};

- enlarging the search window, both t_{calc} and t_{comm} increase in magnitude.

The main goal is to achieve as low t_{cycle} as possible, meaning that for a particular search window must be determined an optimum number n of estimators.

4 Experimental results

The parallel MPEG-2 software encoder was tested on six Pentium 166MHz machines running PVM 3.3 under Linux 2.0.22, connected through an Ethernet 10BaseT hub. A sequence of 34 frames representing a motion video scene at a resolution of 352×240 at 25 frames/sec was encoded. The original video throughput of 50.688Mbps was compressed at a 500Kbps bitrate without a visible degradation of original image quality. Table 1 shows differrent times for encoding this video sequence for different search windows and number of processors.

Figure 4 shows the reduction of encoding time by increasing the number of estimators in the parallel environment.

In order to compare the performances of the parallel implementation for different sizes of the search window with those of the sequential implementation, gained speedup was determined for each case. Speedup determines how many times a parallel implementation runs faster than its sequential correspondent [5]:

$$SpeedUp = \frac{\min(t^1_{Seq} \ldots t^n_{Seq})}{t_{Par}} \quad (5)$$

Srch wind.	Seq. srch	Parallel srch. (no. of estimators)				
		2	3	4	5	6
8	93	69	84	99	120	158
16	206	125	117	124	146	172
32	400	239	231	204	217	246
64	986	536	417	367	359	365
128	2517	1345	1048	802	712	697
256	3925	2025	1552	1233	1038	990

Table 1: Time (in seconds) for encoding the test seqence for different search windows and different configurations of parallel environment.

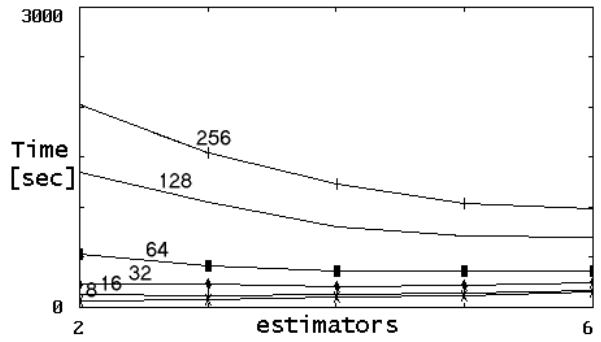


Figure 4: Encoding times vs. number of estimators.

Parallel running time is t_{Par}, and sequential times observed by running the same problem on each of the n machines were $t^1_{Seq} \ldots t^n_{Seq}$. Figure 5 shows that for a certain search window, there is an optimum number of estimators used in parallel encoding that maximizes the speedup, minimizing the encoding time. For example, for a 64×64 search window a number of 5 estimators is optimum.

Efficiency is used to determine the fraction of time per second a processor is used for effective computing:

$$Efficiency = \frac{SpeedUp}{n} \qquad (6)$$

Figure 6 shows that enlarging the search window, the efficiency of processor's using also increases.

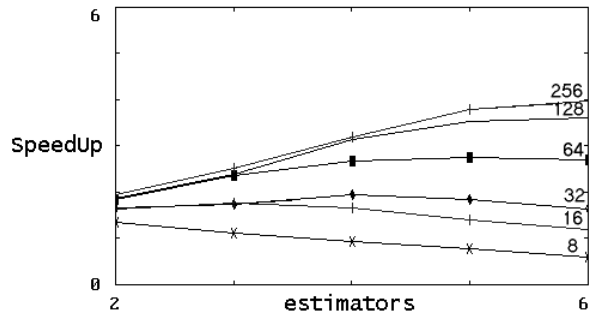


Figure 5: Variation of speedup with the size of the search window and number of estimators.

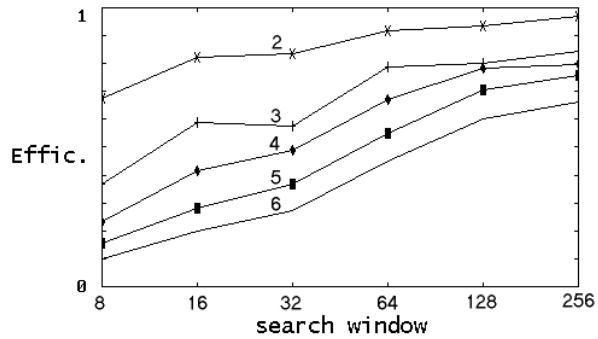
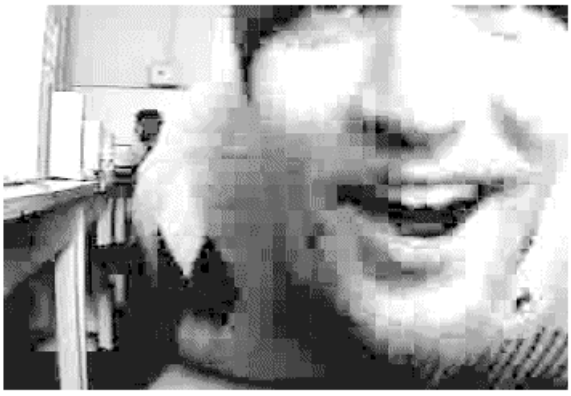


Figure 6: Efficiency increases by enlarging the search window.

The time taken to compress the video sequence for the sequential approach was 115 sec/frame, while the parallel approach, using 6 estimators working independently, required only 29 sec/frame, about four times faster, and preserving the same image quality.

5 Conclusions

Motion estimation is the main factor that determines image quality in MPEG-2 encoding at low bit rates. The more accurate prediction is, the better image quality is achieved. For example, the same frame of the video sequence is presented in Figure 7, at the same bitrate.



8 x 8 search window

256 x 128 search window

Figure 7: Image quality enhances enlarging the search window.

Because of transform coding and cuantization involved, image quality visibly differs by varying the size of the search window.

At low bit rates, image quality is directly influenced by the way motion estimation prediction is formed. The optimum solution is only guaranteed by performing an exhaustive search among all possible ones. The prohibitive time required for finding the best prediction was reduced by the means of a parallel approach. For a given size of the search window (the greater the better), it was determined an optimum number of estimators specific to the parallel environment used, minimizing the search time. This way was achieved the best balance between time used by the parallel environment in communicating results and effective computing time.

References

[1] N. Ahmed, T. Natarajan and K.R. Rao, Discrete Cosine Transform, *IEEE Trans. Comput.*, vol. C-23, pp. 90-93, Jan. 1974

[2] A.L. Beguelin, J.J. Dongarra, G.A. Geist, W.C. Jiang, R.J. Manchek, B.K. Moore, V.S. Sunderam, *PVM: Parallel Virtual Machine. A User's Guide and Tutorial for Networked Parallel Computing*, MIT Press, Cambridge, Massachusetts, 1994

[3] F. Dufaux, Motion Estimation Techniques for Digital TV: A Review and a New Contribution, *IEEE Proc.*, Vol. 83, No. 6, June 1995

[4] R.C. Gonzales, R.E. Woods, *Digital Image Processing*, Addison Wesley, 1992

[5] D. Grosu, Some Performance Metrics for Heterogeneous Distributed Systems, *Proc. of the Intl. Conf. on Parallel and Distrib. Proc. Tech. and Applic.*, Vol. III, pp. 1261-1268, Sunnyvale, California, August 1996

[6] ISO/IEC 13818-2, Information technology - Generic coding of moving pictures and associated audio - Part 2: Video, *Tech. Rep.*, 1995

[7] R. Schäfer and T. Sikora, Digital Video Coding Standards and Their Role in Video Communications, *IEEE Proc.*, Vol. 83, No. 6, June 1995