A monotonicity principle for information theory

Frame-reordered multi-temporal motion estimation for scalable MPEG

Stephan Mietens[1], Peter H.N. de With[2], and Christian Hentschel[3]

[1]Eindhoven University of Technology, EESI, Eindhoven, The Netherlands
[2]CMG Eindhoven / Eindhoven University of Technology, The Netherlands
[3]Philips Research Labs., Eindhoven, The Netherlands

Abstract. The applicability of MPEG video coding can be improved by scaling the resource usage, considering the desired application and the device that is going to be used. For this purpose, we present a new resource-scalable motion estimation technique, based on a flexible three-stage process. This process involves frame processing in display order independent of GOP structures and approximation of MPEG motion-vector fields using multiple references. Quality refinements are optional, giving a flexible framework with a large scalability range (with a factor of 14) of computational effort and memory bandwidth usage, resulting in different picture-quality levels or bitrates. Experiments show that in high-quality operation, the new method slightly outperforms a full-search motion estimation with a search window of 32×32 pixels, although it is based on a recursive block matcher that has only a fraction of the complexity of a full-search block matcher. For e.g. mobile applications with power constraints, the computational effort can be further reduced by at least 43% compared to a single-pass recursive block matcher, at an expense of a slight increase of the bitrate (0.013 bits/pixel).

1 Introduction

Internet-based applications (e.g. video conferencing), portable television applications and mobile consumer terminals impose variable video quality requirements on the system architecture. These requirements can be exploited for scaling the algorithmic complexity of the applications (such as MPEG coding), while accepting a certain quality loss under circumstances as indicated below. Firstly, a part of the available general-purpose computation power of a TV can be saved on-the-fly to be able to perform other tasks in parallel by request. Secondly, when using small displays in e.g. mobile devices, the observer cannot perceive fine details that are contained in the complete video signal. However, for such devices, still a full and thus costly processing of the video is performed.

In this paper, it is our objective to design a scalable MPEG encoding system, featuring scalable video quality and a corresponding scalable resource usage [1]. Such a system enables advanced video encoding applications on a plurality of low-cost or mobile consumer terminals, having limited resources (available memory, computing power, stand-by time, etc.) as compared to high-end computer systems or high-end consumer devices. Note that the advantage of scalable systems is that they are designed once for a whole product family instead of a single product.

A computationally expensive corner stone of an MPEG encoder is motion estimation, which is used to achieve high compression rates by exploiting the temporal redundancy. The search of motion vectors in the motion estimation process requires significant computational effort, because large temporal distances between reference frames lead to large search areas. We have found considerable savings in computation and introduce scaling by performing an initial motion estimation with the video frames at the entrance of the encoder. This estimation is exploited to efficiently derive the desired motion vector fields needed for the final MPEG encoding process. Furthermore, the quality of full-search motion estimation can be obtained with an optional refinement stage.

The paper is organised as follows. Section 2 gives a brief introduction to the motion estimation process and Section 3 adresses the problem statement. Section 4 presents a new technique to perform motion estimation with considerable savings in computational effort and memory bandwidth for resource-constrained applications. Section 5 shows experimental results and Section 6 concludes the paper.

2 Motion estimation for MPEG coders

Successive frames of a video sequence have a temporal correlation that is utilised for video data compression, by sending frame differences instead of complete frames to the decoder, saving bandwidth and/or storage space. Motion in video sequences introduced by camera movements or moving objects result in high spatial frequencies occuring in the frame-difference signal. A higher compression rate is achieved by flattening these spatial frequencies using motion estimation and compensation techniques. Figure 1 shows the basic architecture of an MPEG encoder.

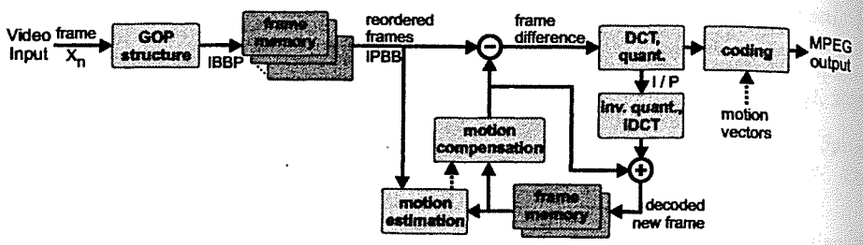

Fig. 1. Basic architecture of an MPEG encoder.

The motion estimation process in MPEG systems divides each frame into rectangular macroblocks (16 × 16 pixels each) and computes motion vectors per block. A motion vector signifies the displacement of the block (in the x-y pixel plane) with respect to a reference image. For each block, a number of candidate motion vectors are examined. For each candidate, the block under test in the current image is compared with the corresponding block fetched from the reference image, displaced by the motion vector. After testing all candidates, the one with the best match is selected. This match is usually done on basis of the sum of absolute differences (SAD) between the current block and displaced block. The collection of motion vectors for a frame forms a motion vector field.

The MPEG coding standard defines three different types of frames, namely I-, P- and B- frames, where I-frames are coded without any temporal reference, thus as completly independent frames. P-frames are based on motion estimation using one temporal reference, namely the previous reference frame. The motion between the reference frame and the P-frame is called forward motion. B-frames are based on motion estimation that can use both backward and forward motion, hence they can also refer to the upcoming reference frame. Reference frames are I- and P-frames. Since B-frames refer to future reference frames, they cannot be (en/de)coded before this reference frame is received by the (en/de)coder. Therefore, the video frames are processed in a reordered way, e.g. "IPBB" (transmit order) instead of "IBBP" (display order).

Notations

For ease of discussion, we form subgroups of pictures (SGOP) that have the form $(I/P)BB...B(I/P)$ within a group of pictures (GOP) according to the MPEG definition and we refer to Figure 2.

The number of pictures within a subgroup k is denoted by M_k, analogous to the prediction depth M of a GOP, and can vary from SGOP to SGOP. The MPEG forward vector-field, which is used in the prediction of the i^{th} frame of the SGOP, is denoted by f_i^k. The MPEG backward vector-field is denoted by b_i^k. Arbitrary vector fields are denoted by $(X_m \rightarrow X_n)$ for the forward case and $(X_m \leftarrow X_n)$ for the backward case, indicating motion between frame X_m and X_n with $n > m$. To indicate the frame type of X, it can be replaced by I, P, or B. In this paper, we discuss motion estimation in terms of entire vector fields, rather than the individual motion vectors.

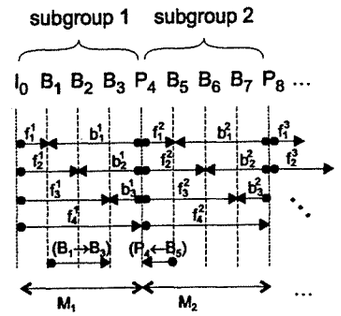

Fig. 2. Example of vector fields used for motion estimation in MPEG encoding after defining a GOP structure. In this example, a GOP with a constant $M = 4$ was chosen.

3 Problem statement

A large number of algorithms has been proposed for reducing the computational effort of a full-search motion estimation. The algorithms make a trade-off between complexity and the quality of the computed vector fields. When compared to full search, popular algorithms like New Three Step Search [2] and Center-Biased Diamond Search [3] provide a good quality of motion vector fields at low cost. However, the accuracy of the motion vectors is limited for fast motion in the video sequence.

Further reduction of the computational effort is achieved by using recursive motion estimation (RME) [4][5] that derives candidate motion-vectors from previously computed motion vectors in both the current motion vector field (so-called "spatial" candidates) or the previous motion vector field (so-called "temporal" candidates). Currently, the usage of RME algorithms is limited to a fixed GOP structure with $M = const$. Only the vector field f_M^k is used once for the computation of the next field f_M^{k+1} and without modification.

A more sophisticated approach for motion estimation is presented in [6], featuring a two-step estimation process and enhanced vector-field prediction. The first step of this approach is a coarse RME to estimate forward vector-fields. The second step uses the vector fields computed in the first step as prediction and performs an additional RME. Vector fields used for prediction are scaled to the required temporal distance.

The problem of the aforementioned algorithms is that a higher value of M increases the prediction depth, which implies a larger frame distance between reference frames, thereby hampering accurate ME. Furthermore, the algorithms do not provide sufficient scalability. For these reasons, we introduce a new three-stage motion estimation that works not only for the typical MPEG GOP structures, but also for more general cases. Furthermore we introduce a new technique called *multi-temporal* ME. These new techniques give more flexibility for a scalable MPEG encoding process.

4 New three-stage motion estimation

Temporal candidate motion vectors play a key role within our new technique for ME. For this reason, we propose the usage of RME (which is based on block matching), since such algorithms make use of temporal candidates and provide a good prediction of the current motion in a video sequence at low cost.

It is obvious that the prediction quality improves with a smaller temporal distance D, where the parameter D denotes the difference between the frame numbers of the considered frames. Therefore, we commence with estimating the motion using the minimum temporal distance $|D| = 1$, which results in an accurate motion estimation that can be performed at low computational effort. Since

a common MPEG GOP structure has $M > 1$ and thus some of the required vector fields have $D > 1$, we consider this as a first stage to derive a prediction of vector fields. In a second stage, these predicted vector fields are used to calculate the required vector fields according to the MPEG standard. In a further stage, the vector fields can be refined by additional - although simple - motion vector refinement. This concept results in a new motion estimation process that is carried out in three stages as follows.

- *Stage 1.* Prior to defining a GOP structure, we perform a simple RME for every received frame X_n and compute the forward motion-vector field $(X_{n-1} \rightarrow X_n)$ and then the backward field $(X_{n-1} \leftarrow X_n)$. For example, in Figure 2 this means computing vectors like f_1^1 and b_3^1, but then for every pair of sequential frames.
- *Stage 2.* After defining a GOP structure, all the vector fields $F \in \{f, b\}$ required for MPEG encoding are approximated by appropriately accessing multiple available vector fields F_A and F_B and combine them using the linear relation

$$F = \alpha * F_A + \beta * F_B, \tag{1}$$

  where the scaling factors α and β depend on the processed fields to obtain the correct temporal distance needed for F. For example, $f_2^1 = f_1^1 + (B_1 \rightarrow B_2)$ (see Figure 2), thus having $\alpha = \beta = 1$. Note that α and β become different if the frame distances change or when complexity scaling is applied (see below).
- *Stage 3.* For final MPEG motion estimation in the encoder, the computed approximated vector fields from the previous stage are used as an input. Beforehand, an optional refinement of the approximations can be performed with a second iteration of RME.

Architecture

Figure 3 shows the architecture of the three-stage motion estimation embedded in an MPEG encoder. Note that the amount of memory needed for the new architecture is the same as used for the architecture shown in Figure 1, except for the additional motion vector memory. The amount is the same because the entrance frame memory for Stage 1 can be integrated with the memory for the reordering process. With respect to the additional vector memory, the following can be said. The memory needed to store a vector field is negligible compared to the memory consumption of a video frame. For comparison, the vector needs only 2 bytes, whereas a macroblock contains 256 bytes (luminance only).

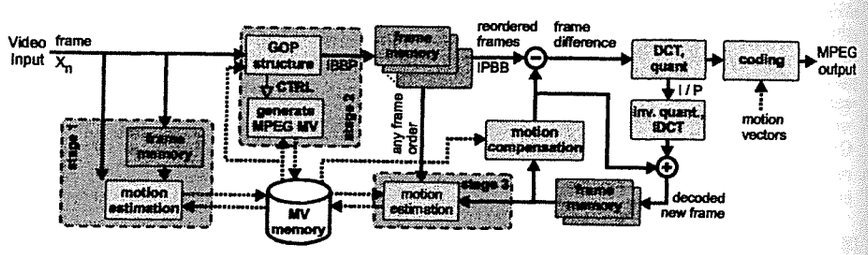

Fig. 3. Architecture of an MPEG encoder with the new scalable three-stage motion estimation.

Let us now discuss several aspects of the architecture.

- The initial motion estimation process in Stage 1 is performed on succeeding frames, which have a temporal distance of 1. Furthermore, Stage 1 uses original frames, without quantisation errors. For these reasons, the RME results in higher quality prediction based on accurate motion vector fields.

- Stage 2 can optimally choose a GOP structure by e.g. analysing the computed motion vector fields. For example, if motion occurs in a sequence, the first frame that is followed by (almost) zero motion can be defined as a reference frame.
- With Equation (1), Stage 2 introduces a new concept called ***multi-temporal*** ME. It means that the computation of a vector field results from combining two other vector fields (see example in the description of Stage 2) and/or the total prediction is based on various vector fields (as follows). We define the computations using various vector fields as a "motion model". Note that such a motion model is independent of the initial stage of motion estimation, e.g. in our case recursive block matching. For example, an affine motion estimation would simply lead to more accurate motion vector fields. A prediction based on various vector fields can be used for high-quality applications to approach different motion models like real velocity, acceleration, zoom, rotation, etc. To give an example of multi-temporal motion estimation for an object with constant motion speed, we predict a motion-vector field $\hat{f}$ by specifing the motion model "most recent velocity" by

$$\hat{f}_i^k = \begin{cases} 2 * f_{i-1}^k - f_{i-2}^k & \text{if } i \geq 3 \\ 2 * f_{i-1}^k & \text{if } i = 2 \\ -b_{M_{k-1}}^{k-1} & \text{if } i = 1,\ M_{k-1} > 1 \\ f_1^{k-1} & \text{if } i = 1,\ M_{k-1} = 1 \end{cases}. \tag{2}$$

  Note that besides our framework, any state-of-the-art motion estimation algorithm can be improved by using multi-temporal vector-field predictions. This implies that a higher number of predictions are generated for the computation of one vector field.
- Stage 3 is intended for high-quality applications to reach the quality of a conventional MPEG motion estimation algorithm by refining the motion vectors that are approximated in Stage 2.
- Figure 3 indicates a central memory for motion vectors that is available for all three stages. The advantage of this central memory is that it can be used in combination with multi-temporal motion estimation to enhance the vector-field predictions. As indicated in the example presented in Equation (2), the selection of vector fields for a further vector-field prediction is not limited to vector fields that are found within the current SGOP, but also to the previous SGOP (e.g. the use of $b_{M_{k-1}}^{k-1}$ in Equation (2)).
- The three stages are decoupled from the actual coding process and are connected with each other via the central motion-vector memory. This concept leads to a flexible usage of the three-stage motion estimation, where within an SGOP, the stages can be processed in sequence or in parallel.

The main advantage of the proposed architecture is that it enables a broad scalability range of resource usage and achievable picture quality in the MPEG encoding process. This is illustrated by the following statements.

- Stage 1 can omit the computation of vector fields (e.g. the backward vector fields) or compute only significant parts of a vector field to reduce the computational effort and memory.
- The effort for computing a vector field can be reduced in all stages if a constant motion velocity is measured in the current video stream.
- The set of motion vectors that are refined in Stage 3 can be limited to significant parts of a vector field to save computations and memory accesses. If this refinement is omitted completely, the new technique can take advantage of further reduced computational effort, because the processing of vector fields in Stage 1 and 2 is much simpler than motion estimation itself when applied to frames with a large temporal distance.

5 Experiments and results

The flexible scalable motion-estimation technique introduced in Section 4 gives a good opportunity for quality and computational scaling. The results of a proof-of-concept experiment using the

"Stefan" sequence (tennis scene) is shown in Figure 4, based on a GOP size of $N = 16$ and $M = 4$ (thus "IBBBP" structure). We use the RME taken from [4] (limited to pixel-search) in Stage 1 and 3 because of its simple design. Note that the RME for this proof-of-concept does not yet take advantage of the reduced temporal distance during the initial stage at the entrance of the encoder. In this experiment, the scaling of complexity is introduced by gradually increasing the vector field computations in Stage 1 using the order f_1, f_2, f_3, f_4, b_3, b_2, b_1, and then the same order was used for the vector field refinements in Stage 3. The area with the white background shows the scalability of the quality range that results from downscaling the amount of computed motion-vector fields. Each vector field requires 14% of the effort compared to a 100% simple RME based on 4 forward vector fields and 3 backward vector fields when going from one to the next reference frame. If all vector fields are computed and the refinement Stage 3 is performed, the computational effort is 200% (not optimised).

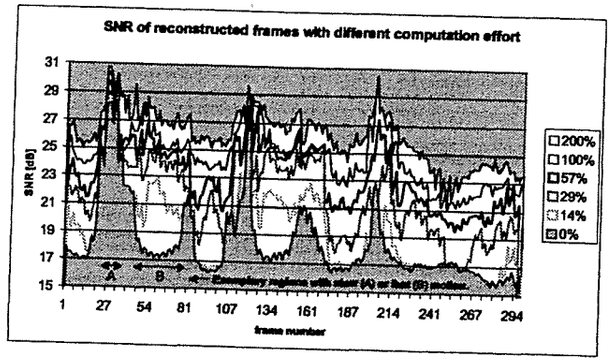

Fig. 4. SNR of reconstructed B-frames of the "Stefan" sequence (tennis scene) with different computational effort, P-frames are not shown for the sake of clearity ($N = 16$, $M = 4$). The percentage shows the different computational effort that results from omitting the computation of vector fields in Stage 1 or performing additional refinement in Stage 3.

The average SNR of the reconstructed P- and B-frames (taken after motion compensation and before computing the differential signal) of this experiment and the resulting bitrate of the encoded MPEG stream is shown in Figure 5. Note that for comparison purpose, no bitrate control is performed during encoding and therefore, the output quality of the MPEG streams for all complexity levels are equal. The quantisation factors $MQUANT$ we use are 12 for I-frames and 8 for P- and B-frames. For a full quality comparison (200%), we consider full-search block-matching with a search window of 32×32 pixels. The new motion estimation technique slightly outperforms this full search by 0.36 dB SNR measured from the reconstructed P- and B-frames of this experiment (25.16 dB instead of 24.80 dB). The bitrate of the complete MPEG sequence is 0.012 bits per pixel (bpp) lower when using the new technique (0.096 bpp instead of 0.108 bpp). When reducing the computational effort to 57% of a single-pass RME, an increase of the bitrate by 0.013 bpp compared to the 32×32 full search is observed.

6 Conclusions

We have presented a new scalable technique for motion estimation in MPEG encoding. The scalability can be exploited to reduce the computational effort over a large range making our system feasible for low-cost mobile MPEG systems. The motion-estimation technique has been split into

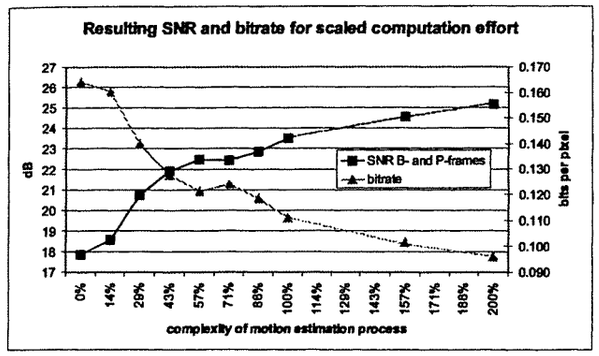

Fig. 5. Average SNR of reconstructed P- and B-frames and resulting bitrate of the encoded "Stefan" stream at different computational efforts. A lower average SNR results in a higher differential signal that must be coded, which leads to a higher bit-rate.

a precomputation stage and an approximation stage. Optionally, a refinement stage can be added to come to the quality of a conventional MPEG encoder (or even outperform it). In the precomputation stage, we used a simple recursive block matcher to find rather good motion estimates because the frames are processed in time-consecutive order. In the approximation stage, vector fields are scaled and added or substracted (thus having multi-temporal references), which is less complex than performing advanced vector searches. The computation of e.g. the backward motion estimation can be omitted to save computational effort and memory bandwidth usage.

First experiments with the new method show that a full processing of our framework slightly outperforms a 32×32 full-search motion estimation when quality is compared. For comparison of the computational effort, we have used a simple recursive block matcher as a reference (note that this is a challenging comparison, because the algorithms are completely different). With a 12% increase of the bitrate, a reduction of 43% in computational effort is obtained, compared to a single-pass recursive motion estimation. The advantage of the proposed system is that it offers a large range in saving computational effort. In our experiments, the computational effort is scaled by a factor of 14, resulting in an average SNR range from 17.84 dB to 25.16 dB of the reconstructed P- and B-frames and a bitrate range from 0.164 bpp to 0.096 bpp.

References

1. C. Hentschel *et al.*: Scalable Algorithms for Media Processing. IEEE Int. Conf. on Image Processing (ICIP 2001) **3** (2001) 342–345
2. T. Koga, K. Iinuma, A. Hirano, Y. Iijima and T. Ishiguro: Motion-compensated interframe coding for video conferencing. Proceedings NTC **81** (1981) C9.6.1–9.6.5
3. J.Y. Tham *et al.*: A Novel Unrestricted Center-Biased Diamond Search Algorithm for Block Motion Estimation. IEEE Trans. Circuits Systems Video Technol. **8** (1998) 369–377
4. P.H.N. de With: A simple recursive motion estimation technique for compression of HDTV signals. IEE Int. Conf. Image Proc. and its Applic. (IPA'92) (1992) 417–420
5. G. de Haan *et al.*: True-Motion Estimation with 3-D Recursive Search Block Matching. IEEE Trans. Circuits Systems Video Technol. **3** (1993) 368–379
6. F.S. Rovati *et al.*: An Innovative, High Quality and Search Window Independent Motion Estimation Algorithm and Architecture for MPEG-2 Encoding. IEEE Trans. Consum. Electron. **46** (2000) 697–705