FAST Rate Allocation through Steepest Descent for JPEG2000 Video Transmission

Francesc Aulí-Llinàs, *Member, IEEE,* Ali Bilgin, *Senior Member, IEEE,* and Michael W. Marcellin, *Fellow, IEEE*

***Abstract*—This work addresses the transmission of pre-encoded JPEG2000 video within a video-on-demand scenario. The primary requirement for the rate allocation algorithm deployed in the server is to match the real-time processing demands of the application. Scalability in terms of complexity must be provided to supply a valid solution by a given instant of time. The FAst rate allocation through STeepest descent (FAST) method introduced in this work selects an initial (and possibly poor) solution, and iteratively improves it until time is exhausted or the algorithm finishes execution. Experimental results suggest that FAST commonly achieves solutions close to the global optimum while employing very few computational resources.**

***Index Terms*—Video transmission, rate allocation, rate-distortion optimization, motion JPEG2000.**

I. Introduction

RATE allocation methods for video transmission are commonly devised to provide the best possible video quality to the end-user. This is achieved by either minimizing the average distortion for all frames, or providing constant (or pseudo-constant) quality for the video sequence [1]. Mean squared error (MSE) is the distortion measure used in most video transmission schemes, so the former criterion is formally stated as the minimization of the MSE (MMSE). It has been shown [2] that the problem of achieving constant quality is equivalent to the one of minimizing the maximum distortion over the sequence, so the latter criterion is more commonly stated as the minimization of the maximum MSE (MMAX).

A fundamental aspect for transmission schemes is the rate allocation problem that emerges when variable bitrate (VBR) video is transmitted under the constraints imposed by the scenario. VBR video refers to the ability of the coding system to transmit a different number of bytes for each frame. Regardless of the coding system, VBR video is the key mechanism through which the optimization of the MMSE or MMAX are accomplished. Hence the rate allocation problem is the root of most video transmission schemes.

This work considers a video-on-demand scenario in which the server transmits pre-encoded JPEG2000 video to clients over a constant bitrate channel. The sequence is rendered by the client at some specified cadence and, to allow VBR video, the client has a limited buffer in which frame data are initially stored before being decoded and displayed. We assume that the transmission rate and buffer size may vary from client to client, and therefore is unknown at the time the video is compressed.

Once the server receives a client request, it allocates a rate to each frame while guaranteeing that the transmission will not cause buffer overflow or underflow at the client. The optimization criterion for choosing among valid solutions can be either MMSE or MMAX. We refer to a rate allocation that satisfies these buffer requirements as "a valid solution." Frame codestreams at the selected rates are then created from the pre-compressed ones using quality scalability properties of JPEG2000.

A primary requirement of the application is that the rate allocation algorithm utilizes very few computational resources (both in terms of CPU time and memory), and that it provides scalability in terms of complexity to allow real-time processing tasks. Timing is essential since the server must respond to clients within a short and predictable interval of time. Thus, the algorithm must reach a valid solution by a given instant of time.

JPEG2000, and certain other image and video coding systems, produce codestreams containing several quality layers. Layers represent convenient truncation points for the compressed representation of the image. When the desired rate falls between those provided by layers, quality layers may, or may not, be truncated depending on the implementation and the requirements of the application [3]. Thus, two versions of the proposed algorithm are presented, corresponding to whether or not layer fragmentation is allowed.

To the best of our knowledge, this problem has not been addressed in the context of JPEG2000. Our previous work [4] is focused on the coding of JPEG2000 video under constrained encoder resources, while [5] addresses the same optimization problem introduced herein, though assuming that the server can only access the rate-distortion characteristics of the current and previous frames. Although there exist rate allocation methods for VBR video that handle some of the raised demands (e.g., [6]–[9]), none of them fulfill the requirements imposed by this application. The application has been implemented employing standard tools and protocols provided in JPEG2000 Part 1 (ISO/IEC 15444-1 - Core coding system), Part 3 (ISO/IEC 15444-3 - Motion JPEG2000), and Part 9 (ISO/IEC 15444-9 - Interactivity tools, APIs and protocols).

This paper is organized as follows: Section II describes rate allocation methods for VBR video with similar requirements to those proposed in this scenario; Section III introduces the proposed method; and Section IV assesses the performance of our approach in terms of coding performance and computational complexity. Section V presents concluding remarks.

II. Rate allocation for VBR video transmission

We start with the optimization for MMSE. Let R^{total} be the total rate (bits/sequence) available to satisfy the client's request for a sequence, and let N be the number of frames of the sequence. The ith frame can be truncated at points $x(i)$ corresponding to increasing bitrates $r_{ix(i)}$ (bits/frame) and decreasing distortion $d_{ix(i)}$, with $1 \leq i \leq N$ and $1 \leq x(i) \leq Q_i$, where Q_i is the number of truncation points available for frame i. The objective is to find

Dr. Francesc Aulí-Llinàs is with the Department of Information and Communications Engineering, Universitat Autònoma de Barcelona, Spain (phone: +34 935813571; fax: +34 935814477; e-mail: fauli@deic.uab.es). Dr. Ali Bilgin is with the Department of Biomedical Engineering and with the Department of Electrical and Computer Engineering, University of Arizona, USA (e-mail: bilgin@email.arizona.edu). Dr. Michael W. Marcellin is with the Department of Electrical and Computer Engineering, University of Arizona, USA (e-mail: marcellin@ece.arizona.edu). This work was completed during a research stage of the first author at the University of Arizona. A preliminary version of this research was presented at the IEEE International Conference on Image Processing 2009. This work has been partially supported by the Spanish Government (MICINN), by FEDER, and by the Catalan Government, under Grants MEC_BPD-2007-1040, TIN2009-14426-C02-01, and 2009-SGR-1224.

the truncation points $\mathbf{x} = \{x(1), x(2), ..., x(N)\}$ that minimize the distortion without exceeding the bit budget, i.e.,

$$\min_{\mathbf{x}} \sum_{i=1}^{N} d_{ix(i)} \quad \text{s.t.} \quad \sum_{i=1}^{N} r_{ix(i)} \leq R^{total} \ . \tag{1}$$

Since well designed coding systems generate truncation points that lie on the convex hull of the operational rate-distortion function, several studies have proven that near-optimal solutions can be found through the use of a generalized Lagrange multiplier for a discrete set of points. This leads to low complexity implementations due to the transformation of the constrained problem of (1) to an unconstrained one. The idea is to use the Lagrange cost $d_{ix'(i)} + \lambda r_{ix'(i)}$ so that the truncation points $\mathbf{x}'$ that minimize

$$\sum_{i=1}^{N} \left(d_{ix'(i)} + \lambda r_{ix'(i)} \right) \tag{2}$$

for the particular λ for which the total rate $R(\lambda) = R^{total}$ represents the optimal solution to (1). Although λ may not be exactly adjusted to attain the total bit budget, close approximations are enough to find near-optimal solutions.

Unfortunately, this approach can not be directly applied when additional constraints are imposed. Let $B(t)$ denote the buffer occupancy in the instant after frame t is rendered according to

$$B(t) \ = \ \frac{R^{total}}{N} \cdot t - \sum_{i=1}^{t} r_{ix(i)} + \frac{R^{total}}{N} \cdot t^{delay} \ . \tag{3}$$

The first term of this equation represents the buffer filling at an assumed constant transmission rate corresponding to R^{total}/N bits per frame time. The second term represents the buffer emptying as frames are rendered. The third term represents buffering delay, expressed as the constant transmission rate multiplied by t^{delay} frame times. Buffering delay is introduced in the above expression to partially fill the buffer prior to rendering the first frame. Taking into account that the maximum buffer size is B^{max}, and that the buffer should not be underflowed or overflowed, the optimization problem is then expressed as

$$\min_{\mathbf{x}} \sum_{i=1}^{N} d_{ix(i)} \tag{4}$$

such that

$$\sum_{i=1}^{N} r_{ix(i)} \leq R^{total} \tag{5}$$

and

$$0 \ \leq \ B(t) \ \leq \ B^{max} \quad \forall \, t, \ 1 \leq t \leq N \ . \tag{6}$$

The second constraint (6) restrains the use of the classical Lagrange approach. It considers the size of the client's buffer, maintaining its occupancy within the imposed limits.

The optimization problem of (4) (5) (6) has been a topic of interest since the mid-90s [10]. Two main approaches have proven effective to tackle the problem [1]: dynamic programming techniques [6], and Lagrange relaxation methods [11]. The first approach employs the Viterbi algorithm to build a trellis structure containing all feasible solutions. It guarantees optimality at the expense of high computational cost. Clustering of frame rates is a complexity reduction technique employed in [6] that does not guarantee optimality, but reduces time and memory requirements while achieving near-optimal performance. Hence, several studies have employed the Viterbi algorithm, or variations of this algorithm, to develop and/or assess the performance of rate allocation methods (e.g., [6], [12], [13]). Accordingly, the method proposed in this work is compared to the Viterbi algorithm with clustering techniques in Section IV.

The second traditional approach to tackle the problem of (4) (5) (6) employs Lagrange optimization by relaxing the constraints, i.e., one Lagrange multiplier is applied to each constraint. For large problems, this is highly computationally demanding, so fast approximations [6]–[8] have been proposed.

Recent work on video streaming applies steepest descent techniques to the rate allocation problem. This technique is used in this work to achieve complexity scalability, and is reviewed in Section III.

With respect to the MMAX criterion, the formulation of the optimization problem is the same as in (4) (5) (6) but the objective function (4) is replaced by

$$\min_{\mathbf{x}} \left(\max_{i=1}^{N} \ d_{ix(i)} \right) \ . \tag{7}$$

Other criteria similar to MMAX are the minimization of the distortion variation [14], or the consideration of an acceptable distortion range [13]. All these criteria pursue the stabilization of video quality, and achieve similar results [15]. Common approaches to the problem of (7) (5) (6) are [13]: dynamic programming techniques [16], lexicographic bit allocation [2], minimum distortion variation [14], or iterative algorithms [15].

Despite the diversity of these approaches, it is stated in [16] that *"any problem which can be solved with the MMSE criterion can also be solved with the MMAX criterion [...] and both criteria can be applied simultaneously within the same optimization framework."* This observation is key in the work reported here. When optimizing MMSE, the objective of our algorithm is to select frame rates that have the lowest Lagrange costs while respecting the rate constraints. By *only* replacing the Lagrange costs by distortion values the algorithm's objective is altered so that it seeks the solution that has the lowest maximum distortion (MMAX).

When codestreams are truncated at points other than at layer boundaries, the aforementioned approaches still hold if the discrete nature of the problem is maintained. For example, codestreams can be truncated at intervals equally spaced in terms of bitrate. More precisely, consider that frame i can be truncated at rates $r_{ik} = k \cdot \Phi$, where $1 \leq k \leq \lfloor r_i^{max}/\Phi \rfloor$, where r_i^{max} corresponds to the length of the codestream i, and Φ denotes the truncation granularity. By reducing Φ, the algorithm achieves more precise solutions. The main difficulty is to determine the distortion and/or Lagrange cost at the truncation points, which is addressed in the next section.

III. Fast rate allocation method

A. JPEG2000 standard

The core coding system of JPEG2000 is wavelet-based with a two tiered coding scheme. A key feature of the system is the coding of small blocks of wavelet coefficients independently. To allow quality scalability, the embedded bitstream generated for each codeblock is partitioned into segments that are placed within successive layers of quality, which eventually form the final codestream.

The bit allocation problem that emerges when forming quality layers is essentially the same as in (1), with coding units represented as codeblocks instead of frames. As mentioned earlier, an efficient solution to this problem is to use a generalized Lagrange multiplier. Commonly, Lagrange optimization is applied in JPEG2000 by computing the operational rate-distortion function of the bitstream generated for each codeblock in order to determine the points that lie on the convex hull. Let R_l and D_l respectively denote the bitrate and distortion of the lth such point of a codeblock bitstream, with $R_l < R_{l+1}$. The distortion-rate slope at this point is then

$$S_l = \frac{D_{l-1} - D_l}{R_l - R_{l-1}} \ . \tag{8}$$

The optimization process may select then from the union of all codeblocks those bitstream segments with highest distortion-rate slope to form layers with specified bitrates or slope thresholds [17].

B. Insights and techniques

The main insight behind the proposed method is to employ a hill climbing technique with steepest ascent/descent. Generally speaking, this technique selects a trivial valid solution to the problem (potentially poor), then iteratively makes small changes to the solution following some heuristic. In each iteration it selects the next valid solution, climbing the path of steepest ascent until it reaches a point in which the solution cannot be improved. This does not guarantee optimality since local optima may stop the progression. However, when applied in a convex space, the heuristic has no local maximums/minimums, so the algorithm's progression is not stopped.

In the context of video transmission, algorithms using the concept of steepest descent were first introduced in [18], and have been used ever since [19]–[25]. The main difference between these methods and the one proposed in this work is that our approach uses alternatively the steepest descent and rate constraints to go backward and forward over the space of solutions in an attempt to reach the optimum. Another point of distinction is the handling of real-time processing tasks through complexity scalability.

When the optimization criterion is MMSE, the heuristic for the steepest descent is the Lagrange cost, which is embodied in the JPEG2000 framework as the distortion-rate slope thresholds of quality layers. To apply the steepest descent in this context, it is mandatory that the operational rate-distortion function of frames lie on the convex hull. This prerequisite is guaranteed when using the generalized Lagrange multiplier based post-compression rate distortion (PCRD) optimization employed in most JPEG2000 implementations. We remark that the extension of the proposed approach to other video transmission schemes must also fulfill this prerequisite. When the optimization criterion is the MMAX, the heuristic is the distortion produced when the layer is decoded.

To make available the distortion-rate slope thresholds and distortions achieved at each quality layer, both slope thresholds and MSE values can be recorded in a comment (COM) marker of the codestream. The popular JPEG2000 implementation Kakadu records distortion-rate slopes and layer lengths by default; our modified version adds distortion values. The overhead associated with this strategy is negligible and allows a rapid recovery of slope thresholds, distortions, and layer lengths from codestream headers. Similar techniques are used in other contexts to accelerate streaming algorithms [26].

This mechanism is of limited use in the scenario when layers are allowed to be truncated. Recording information for enough intra-layer points would significantly increase the size of the file. Techniques to estimate the distortion and slope thresholds of intra-layer fragmentation points become compelling in this case. We adopt the models discussed in [17, Ch. 5.4.4], which suggest that the bitrate varies roughly linearly between layers both with the distortion and the logarithm of the distortion-rate slope.

Figure 1(a) depicts the distortion-rate slope threshold recorded by Kakadu when encoding two frames of the "StEM" sequence (see Section IV for a description of this sequence) at several target bitrates. These frames are chosen with substantially different rate-distortion characteristics. Kakadu records the slope threshold of frame i, layer j as

$$S'_{ij} = \log_2 \frac{\Delta d}{\Delta r} = \log_2 \frac{d_{i(j-1)} - d_{ij}}{r_{ij} - r_{i(j-1)}} \ , \tag{9}$$

with an offset and normalization factor (not included above) to conveniently store values in a 16-bit unsigned integer. Vertical dashed lines in Figure 1(a) depict the allocation rates for representative layers. When frame i is truncated at a rate r that falls between layers j, $j+1$, i.e., $r_{ij} \leq r \leq r_{i(j+1)}$, the slope threshold at r's truncation point is estimated according to the linear form

$$S'_i(r) = S'_{ij} + \frac{(S'_{i(j+1)} - S'_{ij}) \cdot (r - r_{ij})}{(r_{i(j+1)} - r_{ij})} \ . \tag{10}$$

This estimation is depicted in the graph as the straight lines connecting the values at layer boundaries. The figure indicates that linear interpolation between the logarithmic function of layer slope thresholds achieves high accuracy in this context. For frame #2750, for instance, the difference between the actual distortion-rate slopes and the estimated ones is less than 1%, on average.

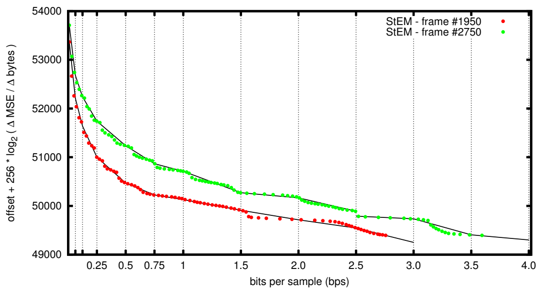
We estimate intra-layer distortion similarly. Figure 1(b) depicts the actual MSE when codestreams containing layers allocated at the rates depicted by the vertical dashed lines are truncated. The straight lines between layers depict the MSE estimated as

$$d_i(r) = d_{ij} + \frac{(d_{i(j+1)} - d_{ij}) \cdot (r - r_{ij})}{(r_{i(j+1)} - r_{ij})} \ , \tag{11}$$

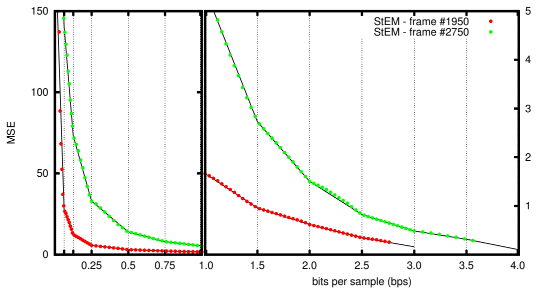
where $d_{ij}, d_{i(j+1)}$ are layer distortion values recorded in the COM marker of the codestream. The accuracy of this technique is notable for most frames. For frame #2750, for instance, the difference between the actual MSE and the estimated one is less than 6% in the bitrate range $(0, 1.0]$ (left part of Figure 1(b)), and less than 1% in the bitrate range $(1, 3.0]$ (right part of the Figure 1(b)). The more layers the codestream has, the higher the accuracy of these estimates.

C. Layer bounded transmission

We first assume that the optimization criterion is MMSE and that layers can not be truncated. The proposed algorithm simultaneously employs the two representations of the problem given by the rate constraints and the Lagrange costs (incarnated as the distortion-rate slope thresholds). The former representation helps maintain the solution within the limits of the feasible area, whereas the latter guides the algorithm to the next solution. To avoid trespassing the rate constraints, every step carried out in the Lagrange space is projected onto the space of rate, so only movements within the feasible region are allowed. With some abuse of notation, the algorithm proceeds as follows:



(a) Distortion-rate slope vs. bitrate



(b) MSE vs. bitrate

Fig. 1: Evaluation of estimation techniques. Coding parameters are: lossy mode, 5 DWT levels, codeblock size of 64×64.

Algorithm 1 Layer bounded allocation

```
 1: initialize $r^* \leftarrow \{r_{1x(1)}, r_{2x(2)}, ..., r_{Nx(N)}\}$ with a solution that
    keeps the buffer at near zero occupancy
 2: repeat
 3:     $r^\star \leftarrow r^*$
 4:     repeat
 5:         seek frame $i' \in r^*$ with lowest $S'_{i'x(i')}$
 6:         $r^{**} \leftarrow \{r_{1x(1)}, ..., r_{i'(x(i')-1)}, ..., r_{Nx(N)}\}$
 7:         if $r^{**}$ respects constraints (5) (6) then
 8:             $r^* \leftarrow r^{**}$
 9:         else
10:             discard frame $i'$ in successive seeks of line 5
11:         end if
12:     until $\nexists\, r^{**}$ that respects constraints (5) (6)
13:     repeat
14:         seek frame $i'' \in r^*$ with highest $S'_{i''(x(i'')+1)}$
15:         $r^{**} \leftarrow \{r_{1x(1)}, ..., r_{i''(x(i'')+1)}, ..., r_{Nx(N)}\}$
16:         if $r^{**}$ respects constraints (5) (6) then
17:             $r^* \leftarrow r^{**}$
18:         else
19:             discard frame $i''$ in successive seeks of line 14
20:         end if
21:     until $\nexists\, r^{**}$ that respects constraints (5) (6)
22:     include all frames in next seeks of lines 5 and 14
23: until $r^\star = r^*$
```

The intention of the first step of the algorithm (line 1) is to find a valid solution. We use the solution given by a pseudo-constant bitrate (CBR) strategy since to deliver the same number of bytes for all frames produces no variations on the buffer occupancy. The purpose of the first inner loop (lines 4 to 12) is to drive the algorithm as far as possible from the constraint given by the maximum channel capacity. During the second inner loop (lines 13 to 21), the path of steepest descent is followed until the maximum channel capacity is reached. This path moves toward the solution given by the generalized Lagrange multiplier until the buffer constraints are reached. These two inner loops are repeated until the algorithm reaches a solution from which it can not improve more.

When optimizing for MMAX, lines 5 (14, respectively) are replaced to seek the frames with the lowest (highest) distortion $d_{i'(x(i')-1)}$ ($d_{i''x(i'')}$). Note that the path of steepest descent for MMAX seeks the frame with the *current* highest distortion since it must minimize the maximum distortion of the current solution, whereas for MMSE it seeks the frame with the *subsequent* layer that has the highest slope threshold.

D. Layer unbounded transmission

In this section we allow layers to be fragmented into equivalently spaced rates given by the truncation granularity Φ. Initially, Φ is set to some maximum, say Φ^{max}, and is then progressively reduced until it reaches Φ^{min}. Following the same notation as in Algorithm 1, the algorithm proceeds as follows:

Algorithm 2 Layer unbounded allocation

```
 1: initialize $r^* \leftarrow \{r_1, r_2, ..., r_N\}$ with $r_i = R^{total}/N \;\forall\, i$
 2: $\Phi \leftarrow \Phi^{max}$
 3: repeat
 4:     repeat
 5:         $r^\star \leftarrow r^*$
 6:         repeat
 7:             seek frame $i' \in r^*$ with lowest $S'_{i'}(r_{i'})$
                (according expression (10))
 8:             $r^{**} \leftarrow \{r_1, ..., r_{i'} - \Phi, ..., r_N\}$
 9:             if $r^{**}$ respects constraints (5) (6) then
10:                 $r^* \leftarrow r^{**}$
11:             else
12:                 discard frame $i'$ in successive seeks of line 7
13:             end if
14:         until $\nexists\, r^{**}$ that respects constraints (5) (6)
15:         repeat
16:             seek frame $i'' \in r^*$ with highest $S'_{i''}(r_{i''} + \Phi)$
                (according expression (10))
17:             $r^{**} \leftarrow \{r_1, ..., r_{i''} + \Phi, ..., r_N\}$
18:             if $r^{**}$ respects constraints (5) (6) then
19:                 $r^* \leftarrow r^{**}$
20:             else
21:                 discard frame $i''$ in successive seeks of line 16
22:             end if
23:         until $\nexists\, r^{**}$ that respects constraints (5) (6)
24:     until $r^\star = r^*$
25:     $\Phi \leftarrow \Phi/2$
26:     include all frames in next seeks of lines 7 and 16
27: until $\Phi < \Phi^{min}$
```

Although this algorithm adds an outer loop with respect to Algorithm 1, our experience indicates that selecting a large truncation granularity Φ^{max} that is progressively reduced is faster than using Φ^{min} invariably, without penalizing performance. When the optimization criterion is MMAX, lines 7 (16, respectively) seek the lowest (highest) distortion $d_{i'}(x(i') - \Phi)$ ($d_{i''}(x(i'')))$ according to expression (11).

After the initialization of $r^\star$ with the CBR strategy, both algorithms can be stopped at any instant taking the most recent $r^\star$ as the final solution. This fulfills the requirements of real-time processing imposed by our application. Experience indicates that the proposed steepest descent algorithm improves every new $r^\star$ (in terms of the chosen heuristic), so the more time the algorithms runs, the better the solution. This is the key-feature that allows scalability in terms of complexity. Experiments in Section IV suggest that both Algorithms 1 and 2 rapidly converge to a solution near the global optimum.

E. Implementation considerations

We call the described method FAst rate allocation through STeepest descent (FAST). The most critical operations of FAST are the seeking of the frame with the lowest/highest distortion-rate slope (or distortion), and the verification of the buffer constraint (6). The computational complexity of the former operation can be reduced using two lists of frames: one ordered by the slope thresholds corresponding to the highest layer currently included in each frame; the other ordered by the thresholds of the next available layer from each frame. The list ordered by currently included layers is sorted in ascending order, so the frame with the lowest slope threshold is always on top. When a layer is removed from this frame, the frame is relocated in the list, which can be accomplished via a bisection search. Similarly, the list ordered by available layers is sorted in descending order, so that the frame to receive the next layer is at the top of the list. Note that the relocation of one frame in one list requires the relocation of the same frame in the other list. Lookup tables holding the position of every frame can be used to avoid the need to search out a frame from the other list. For MMAX, the same optimization process holds. When intra-layer fragmentation is considered, lists are sorted considering truncation points rather than layers.

Verification of the buffer constraints can be efficiently implemented by means of monitoring only the instants in which the trend of the buffer changes. The key-idea is that the instant in which the buffer achieves its maximum fullness happens during an instant in which filling of the buffer changes to emptying, or the other way around for the minimum fullness. Buffer overflow or underflow can be ascertained by examining *only* the fullness of the buffer in these instants. Naturally, these instants have to be updated when layers are added or removed.

Another implementation consideration is that comparing solutions r^* and $r^\star$ may not require computations based on the full solution. A faster technique is to compute the average MSE of the current solution, which can be easily updated every time a layer is added or removed, and to compare average MSE of r^* and $r^\star$ to determine if solutions differ. In the same vein, the termination process can be speeded up by stopping when the difference in average MSE between the current and the previous solution is low[1].

F. Computational complexity

The computational complexity of FAST is the sum of: 1) the initial sorting, 2) the seeking of the highest/lowest slope threshold, 3) the relocation of frames within the sorted lists, 4) the updating of the instants in which the buffer changes trend, and 5) the verification of expression (6). Let M be the sum of the number of layers that are subtracted and added to the solution. In the worst case, the complexity of the algorithm is $O(N \log N + M + M \log N + M + M \cdot N) \cong O(M \cdot N)$. This considers that the buffer reverses trend at every opportunity (i.e., at every frame), so that verifying expression (6) requires N comparisons every time it is carried out. This constant change in tendency is very unlikely to occur in practice. In the experiments of the following section, the buffer changes trend less than 10 times for a sequence of 3,000 frames, and similar results hold for other sequences. Therefore, a more realistic complexity is $O(M \log N)$, which corresponds to the bisection search carried out every time a frame is relocated in the sorted lists. In terms of memory requirements, FAST only needs to maintain in memory the sorted lists of frames, thus memory requirements are linear with the number of frames $O(N)$.

For comparison purposes, the computational complexity of the Viterbi algorithm applied with clustering techniques is $O(N \cdot \frac{B^{max} - B^{min}}{\zeta} \cdot Q)$, where Q denotes the maximum number of layers that the frames contain, and ζ denotes the cluster size. When layers can be truncated, the last term Q is replaced by $\frac{r^{max}}{\zeta}$, with r^{max} denoting the maximum length of the frames' codestreams. In terms of memory requirements, Viterbi has a complexity of $O(N \cdot \frac{B^{max} - B^{min}}{\zeta})$ that, in practice, is much higher than that of FAST as illustrated in the next section.

The Lagrange method is implemented using similar optimization procedures as those of FAST, employing a sorted list of frames. The computational complexity of Lagrange accounts for the initial sorting, and the number of layers included in the final solution, say M', multiplied by the bisection search used to relocate frames in the ordered list, i.e., $O(N \log N + M' \log N) \cong O(M' \log N)$. We note that $M' < M$ since M accounts for all layers that are removed/added by FAST within loops of Algorithms 1 and 2. Lagrange's memory requirements are linear with the number of frames, i.e. $O(N)$.

IV. Experimental results

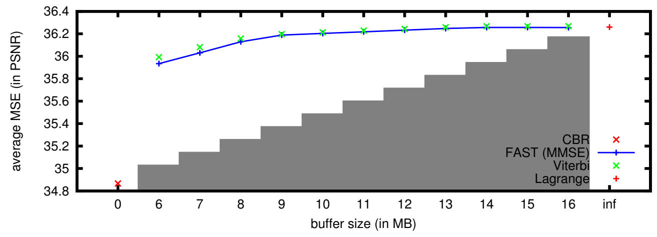
A. Coding performance evaluation

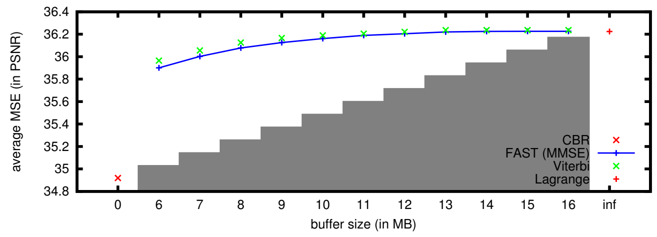
The performance of FAST is evaluated on the "Standard Evaluation Material" (StEM) mini movie, provided by the Digital Cinema Initiatives Consortium. A subsequence of 3,000 frames is selected and, for each frame, codestreams containing 24 quality layers are constructed[2]. The layer allocation strategy distributes quality layers targeting, for each layer, a threshold of distortion-rate slope. The transmission of motion JPEG2000 over JPIP has been implemented in Kakadu v5.2.6. We assume the initial buffer occupancy at half capacity with appropriate buffering delay at the beginning of the transmission.

First, we evaluate the performance of FAST in terms of average MSE for the MMSE optimization criterion. The capacity of the channel is chosen as 3.52 Megabits per second (Mbps), and video is rendered at 10 frames per second (fps), which gives a total bit budget of 132 MB. If the sequence were transmitted using the traditional generalized Lagrange multiplier method –without explicit buffer constraints–, the client would need a buffer of at least 17 MB to absorb the irregularities in rate of the transmitted frames. FAST is applied considering buffer sizes ranging from 6 MB to 16 MB and, for comparison purposes, all graphs depict the performance achieved with the CBR strategy and with the Lagrange method. Graphs also report the performance achieved by Viterbi with clustering techniques under the same constraints. The cluster size in Viterbi is chosen to achieve essentially optimal performance. The Viterbi's cluster size is 1000 bytes. Parameters Φ^{max}, Φ^{min} are respectively set to 1000, 10 for the MMSE criterion, and 4000, 10 for the MMAX criterion. These parameters are selected to minimize running time in the experiments; variations do not change objective performance significantly.

[1]The implementation employed in this work can be found at http://www.deic.uab.es/$\sim$francesc

[2]Frames #1150 through #4149 are selected. The frame size is 2048×857. For simplicity, 8-bit gray-scale versions are used. Coding parameters are: JPEG2000 lossy mode, 5 DWT levels, codeblock size of 64×64.


(a) Average MSE - layer bounded transmission


(b) Average MSE - layer unbounded transmission

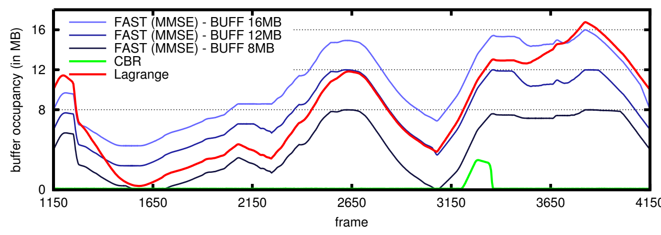
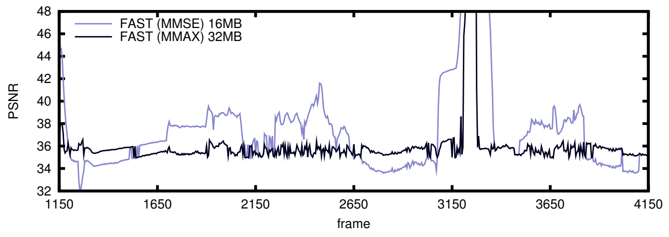

(c) Buffer occupancy - layer bounded transmission

Fig. 2: MSE performance evaluation of FAST, Viterbi, CBR, and Lagrange when the optimization criterion is MMSE.


(a) Layer bounded transmission

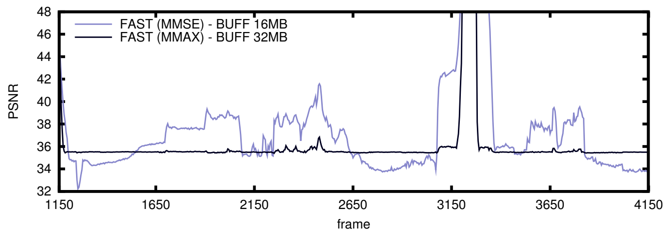
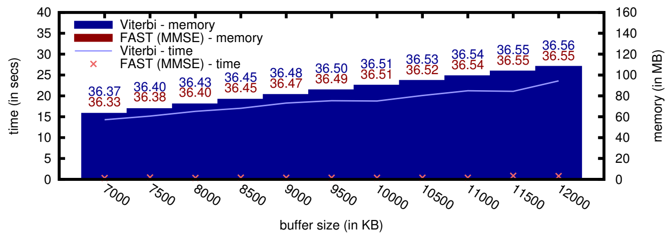

(b) Layer unbounded transmission

Fig. 3: Evaluation of the PSNR of decoded frames.


(a) Performance for different buffer sizes

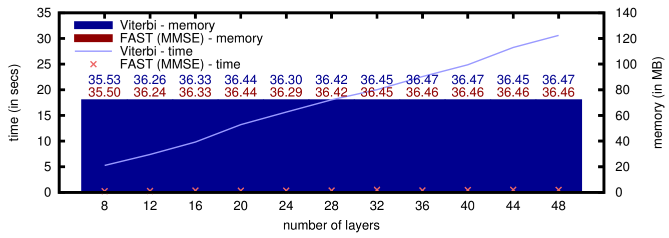

(b) Performance for different number of layers

Fig. 4: Computational performance evaluation of FAST and Viterbi when the optimization criterion is MMSE and transmission is layer bounded.

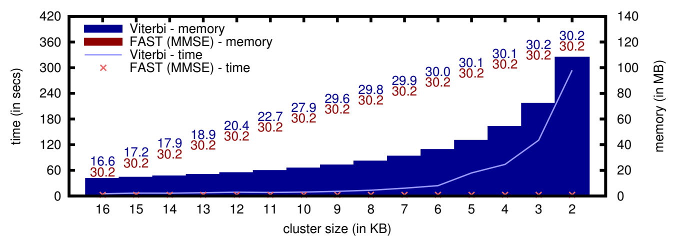
Figures 2(a) and 2(b) depict the average MSE, in terms of Peak Signal to Noise Ratio (PSNR), respectively for the layer bounded and unbounded transmission. In all cases both FAST and Viterbi achieve similar performance, approaching the Lagrange solution as the buffer size grows. Similar results hold for the MMAX criterion. Only when the buffer size is *very* small, is the performance of the Viterbi algorithm slightly better than that of FAST. To explain this behavior we first discuss Figure 2(c), which depicts the buffer occupancy for selected buffer sizes in the experiments above (similar results hold for layer unbounded transmission). The CBR strategy should not require any delay and should maintain the buffer occupancy at 0 for the whole transmission. However, for frames #3200 to #3350 it is not able to do so. This is due to the length of codestreams generated for these frames being significantly less than R^{total}/N. These frames belong to a scene transition and are almost black, yielding lossless compression at unusually small codestreams. Even when the full codestream of these frames is delivered, their transmission fills the client buffer. For small enough buffer size, this would cause buffer overflow at the client. Codestream padding could be used to avoid such overflow. Similarly, the performance of FAST is slightly degraded by the presence of these nearly black frames at very low buffer sizes. Codestream padding actually degrades performance more due to the wasted "overhead", and it is not used here.

Figures 3(a) and 3(b) compare the solutions achieved for both criteria when large buffer sizes are used. Note that the PSNR achieved for the layer unbounded transmission is almost constant even though the intra-layer fragmentation is based on estimates of distortion. Only when frames have very special characteristics, for example, when they are almost constant intensity, the accuracy of this technique is degraded. The PSNR peak from frame #3200 through #3350 is caused by frames that are almost black, so the transmission of very few bytes already achieves very low MSE values. This issue is not perceptually noticeable since the frames are mostly black anyway.

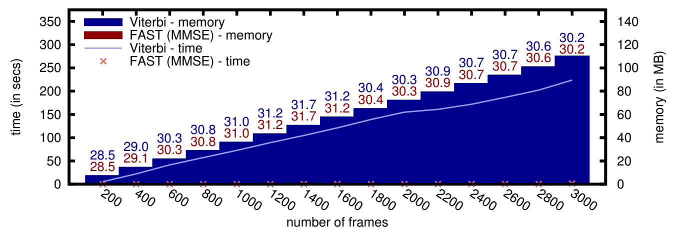
B. Computational costs

Computational performance tests are performed on an Intel Core 2 CPU at 2.4 GHz. All methods are implemented in Java and executed on a Java Virtual Machine v1.6 using GNU/Linux v2.6. Time results report algorithm CPU time, whereas memory results report the size of the data structures employed by the method. Computational performance for both the MMSE and MMAX criteria are similar, thus only graphs for MMSE are provided.

Figures 4(a) and 4(b) assess the computational complexity of the layer bounded transmission compared to Viterbi for different configurations of buffer size, and number of layers. Filled boxes



(a) Performance for different cluster sizes



(b) Performance for different number of frames

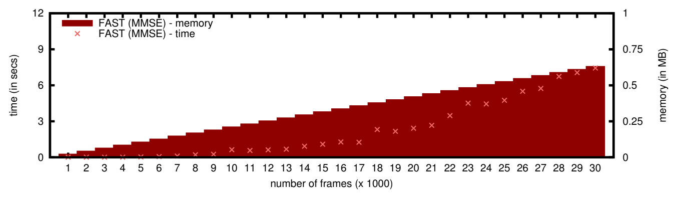
Fig. 5: Computational performance evaluation of FAST and Viterbi when the optimization criterion is MMSE and transmission is layer unbounded.

report memory usage, with the reference axis on the right side of the graph, whereas lines and crosses report CPU time, with the reference axis on the left side of the graph. In addition, graphs report the average MSE (expressed in PSNR) on the top of each column, in blue for Viterbi, and in red for FAST. To fairly compare Viterbi with FAST, the cluster size in Viterbi is chosen as large as possible whilst achieving the same average MSE as FAST. These experiments suggest that the computational load and memory requirements of FAST are extremely low compared to Viterbi. Figures 5(a) and 5(b) assess the computational complexity of the layer unbounded transmission compared to Viterbi for different cluster sizes (of the Viterbi algorithm), and number of frames. Computational complexity of FAST is still very low compared to that of Viterbi. We note that the memory usage of FAST is not discernable in these figures since FAST uses less than 0.5 MB of memory in the reported tests.

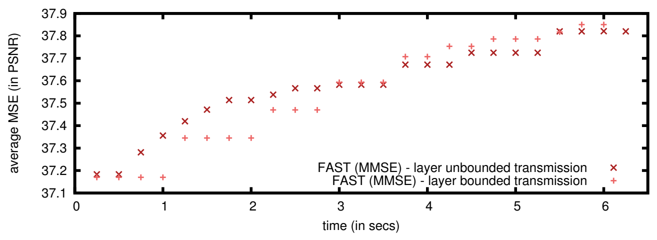
To illustrate the computational performance of FAST for long sequences, 30,000 frames of the movie "Batman Begins" are selected, and codestreams containing 16 quality layers targeted upon slope thresholds are constructed[3]. FAST is applied to select frame rates when the movie is transmitted through a channel with a capacity of 1.2 Mbps, and video is rendered at 15 fps. This gives a total bit budget of 300 MB, and the client buffer size is selected as 8 MB. Figure 6(a) reports computational performance for the layer bounded transmission (similar results hold for layer unbounded transmission). Note that our Java implementation spends less than 7 seconds to optimize 30,000 frames, and that the computational load grows roughly linearly with the number of frames. Parameters Φ^{max}, Φ^{min} are respectively set to 2000 and 250 in these experiments.

To assess the complexity scalability of the algorithm in a real-time environment, Figure 6(b) depicts the average MSE achieved when terminating the algorithm at intervals of time equally spaced every 250 ms, for both the layer bounded and unbounded transmission. Both versions of the algorithm rapidly converge to near optimal solutions. In these graphs, the initial solution corresponds to the CBR strategy, and the periods of time in which the average MSE does not improve correspond to the loops of the algorithm when it goes backwards and forwards over the feasible space of solutions.

[3]The frame size is 590×325, 8-bit gray-scale versions are used. Coding parameters are: JPEG2000 lossy mode, 5 DWT levels, codeblock size of 64×64.



(a) Performance for different number of frames - layer bounded transmission



(b) Real-time processing performance

Fig. 6: Evaluation of the computational performance of FAST for the "Batman Begins" sequence.

V. Conclusions

A fast, complexity scalable rate allocation method is a primary requirement for our video-on-demand application. It is essential that the computational resources of the server are *not* dedicated to the rate allocation method, and that clients receive a response within a minimal –and predictable– interval of time. The lack of computational resources, and real-time processing demands raise a challenging problem. The main contribution of this work is a rate allocation algorithm that provides complexity scalability by means of a steepest descent technique that iteratively enhances an initially poor solution until it reaches near optimal performance. Furthermore, techniques to estimate distortion and distortion-rate slope at intra-layer fragmentation points are developed to allow the use of the proposed method in implementations that truncate quality layers.

The proposed method is evaluated for both the MMSE and MMAX criteria. Experimental results suggest that our method achieves competitive performance with very low computational complexity, allowing the handling of real-time processing tasks efficiently. The scenario has been implemented using standard tools provided by the JPEG2000 standard, demonstrating the suitability of this method to real-world applications.

Acknowledgment

The authors thank D. Taubman for his help with Kakadu, and for hosting the first author during his stay in Australia, and to the associate editor and the anonymous reviewers for their remarks.

References

[1] A. Ortega and K. Ramchandran, "Rate-distortion methods for image and video compression," *IEEE Signal Process. Mag.*, vol. 15, no. 6, pp. 23–50, Nov. 1998.

[2] D. T. Hoang, "Fast and efficient algorithms for text and video compression," Ph.D. dissertation, Brown University, Providence, Rhode Island, May 1997.

[3] F. Auli-Llinas and J. Serra-Sagrista, "JPEG2000 quality scalability without quality layers," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 7, pp. 923–936, Jul. 2008.

[4] J. C. Dagher, A. Bilgin, and M. W. Marcellin, "Resource-constrained rate control for motion JPEG2000," *IEEE Trans. Image Process.*, vol. 12, no. 12, pp. 1522–1529, Dec. 2003.

[5] F. Auli-Llinas and D. Taubman, "Optimal delivery of motion JPEG2000 over JPIP with block-wise truncation of quality layers," in *Proc. IEEE International Conference on Image Processing*, Oct. 2008, pp. 2856–2859.

[6] A. Ortega, K. Ramchandran, and M. Vetterli, "Optimal trellis-based buffered compression and fast approximations," *IEEE Trans. Image Process.*, vol. 3, no. 1, pp. 26–40, Jan. 1994.

[7] A. Ortega, "Optimal bit allocation under multiple rate constraints," in *Proc. IEEE Data Compression Conference*, Mar. 1996, pp. 349–357.

[8] J.-J. Chen and D. W. Lin, "Optimal bit allocation for coding of video signals over ATM networks," *IEEE J. Sel. Areas Commun.*, vol. 15, no. 6, pp. 1002–1015, Aug. 1997.

[9] J. Cabrera, A. Ortega, and J. I. Ronda, "Stochastic rate-control of video coders for wireless channels," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 6, pp. 496–510, Jun. 2002.

[10] A. R. Reibman and B. G. Haskell, "Constraints on variable bit-rate video for ATM networks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 2, no. 4, pp. 361–372, Dec. 1992.

[11] S.-W. Wu and A. Gersho, "Rate-constrained optimal block-adaptive coding for digital tape recording of HDTV," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 1, no. 1, pp. 100–112, Mar. 1991.

[12] C.-Y. Hsu, A. Ortega, and M. Khansari, "Rate control for robust video transmission over burst-error wireless channels," *IEEE J. Sel. Areas Commun.*, vol. 17, no. 5, pp. 756–773, May 1999.

[13] K.-L. Huang and H.-M. Hang, "Consistent picture quality control strategy for dependent video coding," *IEEE Trans. Image Process.*, vol. 18, no. 5, pp. 1004–1014, May 2009.

[14] B. Xie and W. Zeng, "A sequence-based rate control framework for consistent quality real-time video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 1, pp. 56–71, Jan. 2006.

[15] N. Cherniavsky, G. Shavit, M. F. Ringenburg, R. E. Ladner, and E. A. Riskin, "Multistage: A MINMAX bit allocation algorithm for video coders," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 1, pp. 59–67, Jan. 2007.

[16] G. M. Schuster, G. Melnikov, and A. K. Katsaggelos, "A review of the minimum maximum criterion for optimal bit allocation among dependent quantizers," *IEEE Trans. Multimedia*, vol. 1, no. 1, pp. 3–17, Mar. 1999.

[17] D. S. Taubman and M. W. Marcellin, *JPEG2000 Image compression fundamentals, standards and practice*. Norwell, Massachusetts 02061 USA: Kluwer Academic Publishers, 2002.

[18] Z. Miao and A. Ortega, "Optimal scheduling for streaming of scalable media," in *Proc. IEEE Asilomar Conference on Signals, Systems, and Computers*, vol. 2, Nov. 2000, pp. 1357–1362.

[19] ——, "Scalable proxy caching of video under storage constraints," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 7, pp. 1315–1327, Sep. 2002.

[20] Y. Sermadevi and S. S. Hemami, "Efficient bit allocation for dependent video coding," in *Proc. IEEE Data Compression Conference*, Mar. 2004, pp. 232–241.

[21] J. Chakareski, S. Han, and B. Girod, "Layered coding vs. multiple descriptions for video streaming over multiple paths," *SPRINGER Multimedia Systems*, vol. 10, no. 4, pp. 275–285, Apr. 2005.

[22] Y. J. Liang and B. Girod, "Network-adaptive low-latency video communication over best-effort networks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 1, pp. 72–81, Jan. 2006.

[23] P. A. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," *IEEE Trans. Multimedia*, vol. 8, no. 2, pp. 390–404, Apr. 2006.

[24] X. Zhu, P. Agrawal, J. P. Singh, T. Alpcan, and B. Girod, "Distributed rate allocation policies for multihomed video streaming over heterogeneous access networks," *IEEE Trans. Multimedia*, vol. 11, no. 4, pp. 752–764, Jun. 2009.

[25] K. L. Ferguson and N. M. Allinson, "Modified steepest-descent for bit allocation in strongly dependent video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 7, pp. 1057–1062, Jul. 2009.

[26] J. Chakareski, J. G. Apostolopoulos, S. Wee, W. tian Tan, and B. Girod, "Rate-distortion hint tracks for adaptive video streaming," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 10, pp. 1257–1269, Oct. 2005.