# An Algorithm for Lossless Smoothing of MPEG Video*

Simon S. Lam, Simon Chow, and David K. Y. Yau
Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712

## Abstract

Interframe compression techniques, such as those used in MPEG video, give rise to a coded bit stream where picture sizes differ by a factor of 10 or more. As a result, buffering is needed to reduce (smooth) rate fluctuations of encoder output from one picture to the next; without smoothing, the performance of networks that carry such video traffic would be adversely affected. Various techniques have been suggested for controlling the output rate of a VBR encoder to alleviate network congestion or prevent buffer overflow. Most of these techniques, however, are *lossy*, and should be used only as a last resort. In this paper, we design and specify an algorithm for *lossless* smoothing. The algorithm is characterized by three parameters: $D$ (delay bound), $K$ (number of pictures with known sizes), and $H$ (lookahead interval). We present a theorem which guarantees that, if $K \geq 1$, the algorithm finds a solution that satisfies the delay bound. (Although the algorithm and theorem were motivated by MPEG video, they are applicable to the smoothing of compressed video in general.) To study performance characteristics of the algorithm, we conducted a large number of experiments using statistics from four MPEG video sequences.

## 1 Introduction

Recent developments in digital video technology have made possible the storage and communication of full-motion video as a type of computer data, which can be integrated with text, graphics, and other data types. As part of our research project on the design of transport and network protocols for multimedia applications, we have been studying the characteristics of compressed digital video encoded in accordance with MPEG, which is a recently published standard of the International Standards Organization (ISO). The standard is known by the name of the working group, Moving Pictures Expert Group, that developed it [7].

MPEG has been developed for storing video (and associated audio) on digital storage media, which include CD-ROM, digital audio tapes, magnetic disks and writable optical disks, as well as delivering video through local area networks and other telecommunications channels. At rates of several Mbps, MPEG video is suitable for a large number of multimedia applications including video mail, video conferencing, electronic publishing, distance learning, and games.[1]

At this time, there are few alternative industry-wide standards. JPEG, another ISO standard and a precursor of MPEG, was designed for the compression of still images; it does not take into consideration the extensive frame to frame redundancy present in all video sequences. For teleconferencing and videotelephone applications, the CCITT H.261 standard specifies compression techniques at rates of $p \times 64$ kilobits/second, where $p$ ranges from 1 to about 30. Compared to H.261, the MPEG standard was designed for a higher range of rates and a much better visual quality. However, MPEG video is not intended to be broadcast television quality; other standards are being developed to address the compression of television broadcast signals at 10–45 Mbps.[2]

Full-motion video is a set of pictures displayed sequentially. In uncompressed form, each picture is a two dimensional array of pixels, each of which is represented by three values (24 bits) specifying both luminance and color information.[3]

From such uncompressed video data, an MPEG encoder produces a coded bit stream representing a sequence of encoded pictures (as well as some control information for the decoder). There are three types of encoded pictures: I (intracoded), P (predicted), and B (bidirectional). The sequence of encoded pictures is specified by two parameters: $M$, the distance between I or P pictures, and $N$, the dis-

*Research supported in part by National Science Foundation grant no. NCR-9004464, and in part by an unrestricted grant from Lockheed. In *Proceedings of ACM SIGCOMM '94*, London, August 1994.
0

[1] The target rate is 1.5 Mbps for a relatively low spatial resolution, e.g., $350 \times 250$ pixels.

[2] The quality of MPEG video has been compared to that of VHS recording [1].

[3] We use the term *picture* as in [3]. In this paper and the literature, the terms *frame*, *image*, and *picture* are often used interchangeably.

tance between I pictures. Thus, if $M$ is 3 and $N$ is 9, then the sequence of encoded pictures is

I B B P B B P B B I B B P B B ...

where the pattern IBBPBBPBB repeats indefinitely. If $M$ is 1 and $N$ is 5, then the sequence is

I P P P P I P P P P I P P P ...

where the pattern IPPPP repeats indefinitely.

An interframe coding technique called motion compensation is used such that "pieces" of a P picture are obtained from the preceding I or P picture in the sequence, and pieces of a B picture are obtained from the preceding I or P picture and the subsequent I or P picture in the sequence. An I picture is intracoded; that is, it is encoded, and decoded, without using information from another picture. In general, an I picture is much larger than a P picture (in number of bits), which is much larger than a B picture. For typical natural scenes, the size of an I picture is larger than the size of a B picture by an order of magnitude.

An MPEG encoder that compresses a video signal at a constant picture rate (e.g., 30 pictures/s) outputs a coded bit stream with a highly variable instantaneous bit rate. Such a coded bit stream is called variable bit rate (VBR) video. Packet-switching networks—such as ATM networks where transmission capacity is allocated on demand by statistical multiplexing—can in principle carry VBR video traffic without a significant loss in bandwidth utilization. However, it is obvious, and has been demonstrated [10, 11], that the statistical multiplexing gain of finite-buffer packet switches can improve substantially by reducing the variance of input traffic rates.[4] This is one of the objectives of the lossless smoothing algorithm to be presented in this paper.

Changes in the output rate of an MPEG encoder should be viewed on three different time scales: (1) from the encoding of one block to the next within a picture, (2) from one picture to the next within the video sequence being encoded, and (3) from one scene to the next within the video sequence. We will ignore rate fluctuations during the encoding of a picture, since these fluctuations can be smoothed out with a small amount of buffering at the encoder.

The rate fluctuations from one picture to the next are the most troublesome. Consider an I picture, which is 200,000 bits long, followed by a B picture, which is 20,000 bits long. (These are realistic numbers from some of the video sequences we have encoded at a spatial resolution of $640 \times 480$ pixels; see Figure 3 in Section 5.) Suppose the video application specifies a picture rate of 30 pictures/second. Transmitting the I picture in 1/30 second over a network would require a transmission capacity of 6 Mbps to be allocated. Then during the next 1/30 second, the transmission capacity required for the B picture drops precipitously to 0.6 Mbps. These very large fluctuations are a consequence of the use of interframe coding techniques in MPEG.

The encoder output rate also changes, on the average, as the scene in the video sequence being encoded changes. Pictures of more complex scenes require more bits to encode. Pictures also require more bits to encode when there is a lot of motion in a scene (P and B pictures in particular). We observed that the (smoothed) output rates from one scene to the next differ by about a factor of 3 in the worst case, and thus are not as troublesome as rate fluctuations between I and B pictures.

The output rate of an MPEG encoder depends upon the spatial resolution of pictures (number of pixels) and the temporal resolution (picture rate), which are parameters typically specified by a multimedia application. The picture rate, as well as some other MPEG encoder parameters, can be adaptively controlled to modify the encoder output rate (see Section 3). Some researchers have described adaptive techniques for controlling the output rate of VBR encoders [2, 4, 9]. Since these VBR encoders are considered input sources of packet-switching networks, the techniques are sometimes referred to as source rate control or congestion control techniques. Most of these techniques are *lossy*.

In this paper, we present an algorithm for smoothing picture-to-picture rate fluctuations in a video sequence. The algorithm can be implemented in transport protocols for compressed video in general. The algorithm's performance, however, is improved by a lookahead strategy that makes use of the repeating pattern of I, P, and B pictures in an MPEG video sequence. The objective of the algorithm is to transmit each picture in the same pattern at approximately the same rate, while ensuring that the buffering delay introduced by the algorithm is bounded by $D$ for every picture; the delay bound $D$ is a parameter which is to be specified by the multimedia application. The algorithm is lossless because smoothing is accomplished by buffering, not by discarding some information. We believe that an algorithm such as ours should always be used in transmitting MPEG video over a network, while lossy techniques for rate control should be used only as a last resort to alleviate congestion.

Solution to the problem of lossless smoothing is relatively straightforward if picture sizes are known a priori for all pictures in the video sequence.[5] Our main contributions to be presented in this paper are: (1) the design of an algorithm with no knowledge of the sizes of pictures that have not yet been encoded, and (2) an experimental demonstration, using a set of MPEG video sequences, that our algorithm is effective—namely, the delay bound is satisfied for individual pictures and fluctuations in the encoder output rate are reduced to minimum levels (i.e., to those fluctuations caused by motion and scene changes in the video sequence).

The balance of this paper is organized as follows. In Section 2, we provide an introduction to MPEG video—in particular, the structure of an MPEG video bit stream—from the perspective of designers of transport and network protocols. In Section 3, we describe techniques for adaptively controlling the output rate of an MPEG encoder, and explain why lossless smoothing should be used, and lossy techniques only as a last resort. In Section 4, the theoretical basis for algorithm design is stated in a theorem and a corollary. The algorithm is then designed and specified. In Section 5, we first describe the MPEG video sequences used in our experiments. Experimental results are shown to illustrate the performance and demonstrate the effectiveness of our algorithm. Section 6 has some concluding remarks.

[4] For a specified bound on loss probability.

[5] One such solution is given by Ott et al. [8].

## 2 MPEG Video

We describe in this section the structure of an MPEG video bit stream from the perspective of designers of transport and network protocols. For a conventional treatment, see Le Gall [3]; for details, consult the ISO standard [7]. Our observations of the effects of errors, introduced by manually changing some bits in the coded bit stream, can be found in an extended version of this paper [6].

Full-motion video is a set of pictures displayed sequentially. Each picture is represented as a two dimensional array of pixels, each of which is specified by a set of three values giving the red, green, and blue levels of the pixel. This is called the RGB representation. In MPEG encoding, each RGB triplet is first transformed into a YCrCb triplet, where the Y value indicates luminance level and the Cr and Cb values represent chrominance (color information).

As an illustration, a picture with a spatial resolution of $640 \times 480$ pixels and 24 bits per pixel requires about 921 kilobytes to represent when uncompressed. For a video sequence to be displayed at a picture rate of 30 pictures/s, the transmission capacity required is about 221 Mbps.

For compression, MPEG uses *intraframe* techniques that exploit the spatial redundancy within a picture, as well as *interframe* techniques that exploit the temporal redundancy present in a video sequence. These are briefly described below.

The structure of an MPEG video bit stream can be specified as follows in BNF notation:

```
<sequence> ::= <sequence header>
               <group of pictures>
               { [ <sequence header> ]
                 <group of pictures> }
               <sequence end code>

<group of pictures> ::= <group header> <picture>
                        { <picture> }

<picture> ::= <picture header> <slice>
              { <slice> }

<slice> ::= <slice header> <macroblock>
            { <macroblock> }
```

where the curly brackets { } delimit an expression that is repeated zero or more times.

The sequence header contains control information (e.g., spatial resolution, picture rate) needed to decode the MPEG video bit stream. Pictures in an MPEG video sequence are organized into groups to facilitate random access; in particular, a time code specified in hours, minutes, and seconds is included in each group header. Repeating the sequence header at the beginning of every group of pictures makes it possible to begin decoding at intermediate points in the video sequence (facilitating random access). However, only the very first sequence header is required; the others are optional.

The header of a picture contains control information about the picture (e.g., picture type, temporal reference). The header of a slice contains control information about the slice (e.g., position in picture, quantizer scale). Each header (sequence, group, picture, or slice) begins with a 32-bit start code that is unique in the coded bit stream—specifically, the start codes are made unique by zero bit and zero byte stuffing.

Each macroblock in a slice represents an area of $16 \times 16$ pixels in a picture. For example, consider a picture of $640 \times 480$ pixels. There are $40 \times 30$ macroblocks in the picture. The macroblocks are placed in the coded bit stream sequentially in raster-scan order (left to right, top to bottom). It is natural to specify each row of macroblocks in the picture to be a slice. The picture, for the above example, would then be represented by a sequence of 30 slices, one for each row. However, the MPEG standard does not require that a slice contain exactly a row of macroblocks. By definition, a slice contains a series of one or more macroblocks; the minimum is one macroblock, and the maximum can be all the macroblocks in the picture. Also slices in the same picture can have different numbers of macroblocks.

Each macroblock begins with a header containing information on the macroblock address, macroblock type, and an optional quantizer scale.[6] However, the beginning of a macroblock is not marked by a unique start code, and thus cannot be identified in a coded bit stream. Furthermore, macroblocks are of variable length. Hence, a slice is the smallest unit available to a decoder for resynchronization. In particular, whenever errors are detected, the decoder can skip ahead to the next slice start code—or picture start code—and resume decoding from there. One or more slices would be missing from the picture being decoded.

Macroblocks are the basic units for applying interframe coding techniques to reduce temporal redundancy. In an I picture, every macroblock is intracoded. In a P or B picture, a macroblock may be intracoded, or predicted using various interframe motion compensation techniques.

Before describing motion compensation, we first consider intracoded macroblocks and briefly describe the techniques for reducing spatial redundancy. To encode the luminance levels of the $16 \times 16$ pixels in a macroblock, the pixels are subdivided into four blocks of $8 \times 8$ pixels each. MPEG makes use of the fact that the human eye is less sensitive to chrominance than luminance. Therefore, the Cr and Cb planes are subsampled, i.e., for each macroblock, only $8 \times 8$ Cr (Cb) values are sampled, resulting in only one Cr block and one Cb block. Thus following the header of each intracoded macroblock, there are six blocks, each of which is coded as follows.

Applying the discrete cosine transform (DCT) to the 64 values of a block produces 64 coefficients that have a frequency domain interpretation. These coefficients are quantized, with low-frequency coefficients (of basis functions representing large spatial extent) quantized more finely than high-frequency coefficients (of basis functions representing small detail). Significant compression is obtained when many coefficients (typically the higher frequency ones) become zero after quantization. The above technique makes use of two facts: (1) the human eye is relatively insensi-

[6] If specified, this would override the quantizer scale in the slice header.

tive to high-frequency information, and (2) high-frequency coefficients are generally small.

Following quantization, the coefficients are then run length coded to remove zeros, and then entropy coded (actually a combination of variable-length and fixed length codes are used). Although both run length and entropy coding are lossless techniques, quantization is lossy (some image information is discarded).[7]

A macroblock in a P picture is predicted from the reference picture (i.e., the preceding I or P picture in the video sequence) as follows. Various algorithms may be used to search the reference picture for a 16 by 16 pixel area that closely matches this macroblock. (The algorithm is implementation dependent and not specified by the MPEG standard.) If prediction is used, two pieces of information are encoded: (1) a motion vector specifying the x and y translation to the matching area in the reference picture, and (2) an error term, specifying differences between the macroblock and the matching area. The motion vector is entropy coded, while both DCT and entropy coding are applied to the error term. Clearly, prediction is not used if it would take as many bits to code these two pieces of information as the macroblock's pixels; in this case, the macroblock can be intracoded as described above.

Each B picture has two reference pictures, one in the past and one in the future. A macroblock in a B picture may be obtained from a matching area in the past reference picture (forward prediction), a matching area in the future reference picture (backward prediction), or an average of two matching areas, one in each of the two reference pictures (interpolation). For such predicted and interpolated macroblocks, motion vectors and error terms are encoded. But if necessary, a macroblock within a B picture can be intracoded.

Since a B picture depends on a reference picture in the future, it cannot be encoded until the reference picture in the video sequence has been captured and digitized. To do so, an encoder must introduce a delay equal to the time to capture and digitize $M$ pictures (less than or equal to $M\tau$, where $1/\tau$ is the picture rate of the encoder). Similarly, a decoder cannot decode a B picture until its reference picture in the future has been received. Thus, the order in which pictures are transmitted should be different from the order in which a video sequence is displayed. Specifically, the reference picture following a group of B pictures in a video sequence should be transmitted ahead of the group. For example, if the video sequence is

I B B P B B P B B I B B P ...

Then the transmission sequence is

I P B B P B B I B B P B B ....

# 3 Rate Control

In the networking literature, studies on peak rate control of VBR video are concerned with alleviating network congestion. In Section 3.1, we first review techniques that can be used for rate control, all of which are lossy. The smoothing problem of interest in this paper has a different objective, and is unique to VBR video encoded using interframe techniques, such as MPEG video, which has different types of pictures with a wide range of sizes. It has been demonstrated [10, 11] that the statistical multiplexing gain of a finite-buffer packet switch (such as an ATM switch) can be increased by reducing the variance of its input traffic.[8] For the specific objective of reducing picture-to-picture rate fluctuations that are a consequence of interframe coding, lossless smoothing is a more appropriate solution than the lossy techniques. The problem of smoothing is introduced in Section 3.2. Design and specification of our smoothing algorithm are presented in Section 4.

## 3.1 Lossy techniques

An MPEG encoder can control its output rate by setting the quantizer scale in the slice header, and also setting the optional quantizer scale in the header of each macroblock within a slice. A coarser setting would result in a lower bit rate at the expense of poorer visual quality. Additionally, the encoder can also lower its output rate by discarding some of the high-frequency DCT coefficients (under the assumption that the human eye is relatively insensitive to such high-frequency information).

These rate control techniques are described in the MPEG standard as methods for ensuring that the input buffer of the "model decoder" neither overflows nor underflows. As techniques to reduce the output rate of an encoder, both are lossy in that some information is discarded, and may result in visible artifacts in the decoded video. Each technique has been suggested as the basis of congestion control schemes for packet networks that carry VBR video traffic. Specifically, the encoder would control its output rate in response to feedback information from an entry point to a packet network or a point of congestion in the packet network [2, 4, 9].

These lossy techniques for rate control are inappropriate for reducing fluctuations in the bit rate for transmitting I, B, and P pictures in MPEG video for the following reason. I pictures in MPEG video are about an order of magnitude larger than B pictures (see Figure 3 in Section 5).[9] We experimented with changing the quantizer scale of an I picture from 4 to 30. The size of the picture is reduced from 282,976 bits to 75,960 bits. But the picture at the coarser quantizer scale (30) is grainy, fuzzy, and has visible blocking effects. Our observations are in agreement with the following statement from [3]:

> "Intracoded blocks contain energy in all frequencies and are very likely to produce 'blocking effects' if too coarsely quantized; on the other hand, prediction error-type blocks contain predominantly high frequencies and can be subjected to much coarser quantization."

[7] The techniques are essentially the same as those of JPEG. Unlike MPEG, only intracoded pictures are specified by JPEG.

[8] For a specified bound on loss probability.

[9] This observation is consistent with the suggestion in [7] that, for typical natural scenes, P pictures be allocated 2–5 times as many bits as B pictures, and I pictures be allocated up to 3 times as many bits as P pictures.

According to the above statement, I pictures should be quantized less coarsely than P and B pictures, not the other way around.

Another lossy technique that has been suggested for network congestion control is to reduce the picture rate by dropping some B pictures from the video sequence being transmitted [2]. Although dropping B pictures would reduce the average rate of the video sequence, it does not address the problem of picture-to-picture rate fluctuations of interest here.

In summary, both spatial and temporal redundancy are greatly reduced in the coded bit stream of MPEG video. Any lossy technique to reduce the peak rate of the bit stream would degrade the visual quality of I pictures, the largest pictures by far in the video sequence. They are also the most important, since pieces of B and P pictures are obtained from the I pictures.

### 3.2 Lossless smoothing

Consider an MPEG video sequence with picture sizes, $S_1, S_2, S_3, \ldots$ . The size sequence has large fluctuations because I pictures are much larger than B pictures. However, in the video sequence (also the size sequence), there is a pattern of $N$ picture types which repeats indefinitely.

The objective of smoothing is to eliminate rate fluctuations that are a consequence of interframe coding in MPEG. One way to accomplish this is to buffer pictures (at the sending side of a transport protocol) so that each picture within the same pattern can be transmitted at the same rate. To illustrate, consider a video sequence with $M = 3$ and $N = 9$. The repeating pattern is

I B B P B B P B B .

Let $S_i, S_{i+1}, \ldots, S_{i+8}$ be the picture sizes of a particular pattern in the sequence. Let $\tau$ denote the picture period (that is, the picture rate is $1/\tau$). Thus the objective of smoothing is to send each picture in this pattern at the following rate

$$\frac{S_i + S_{i+1} + \ldots + S_{i+8}}{9\tau}$$

That is, the large I picture is transmitted at a smaller rate while the small B pictures are transmitted at a higher rate. Note that this averaging of rates is carried out on a pattern by pattern basis to smooth out picture-to-picture rate fluctuations. However, the rate of the coded bit stream still fluctuates from pattern to pattern. Such fluctuations, however, are inherent characteristics of the video sequence (scene complexity and amount of motion), which cannot be reduced without sacrificing visual quality or incurring unacceptably long delays.

We will refer to the above method as *ideal* smoothing. The ideal method has two disadvantages. First, if the video sequence is generated by a live capture (using a camera), the size of each picture is not known until it has been captured, digitized, and encoded. With the ideal method, the pictures in the same pattern would have to be buffered until all have been encoded—and the rate calculated for the pattern—before the first picture in the pattern can be transmitted. In this case, the buffering delay would be very large, and unacceptable for live video. Second, the ideal method described above does not ensure that the buffering delay of each picture is less than $D$, an upper bound which can be specified.

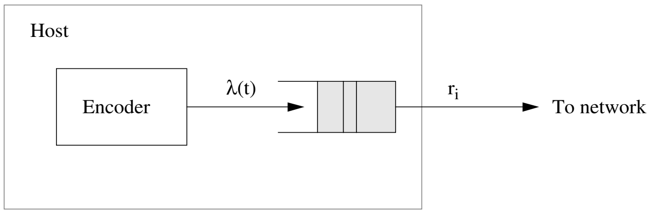


Figure 1: System model for rate smoothing.

In the next section, we design an algorithm for smoothing MPEG video with the objective that the delay incurred by each picture in the video sequence is less than $D$, a parameter that can be specified.

# 4 Smoothing Algorithm

Consider a video sequence that is to be displayed at the rate of $1/\tau$ pictures per second. $\tau$ is called the *picture period*. We assume that the encoding (decoding) time of any picture in the video sequence is less than or equal to $\tau$ seconds. We use $S_i$ to denote the size of picture $i$, $i = 1, 2, 3, \ldots$, which is the number of bits representing picture $i$ in the coded bit stream.

Both the system model and the algorithm described in this section can be used for compressed video in general. (The model and analysis presented in this section are from [5].) The presence of a repeating pattern in an MPEG video sequence is used to estimate the sizes of pictures that have not been encoded; it is, however, not needed in the system model, nor in the algorithm.

### 4.1 System model

The model for rate smoothing is a FIFO queue (with some modifications). Input to the queue is from the output of an encoder (see Figure 1). At time $t$, let $\lambda(t)$ denote the output rate of the encoder (same as input rate of the queue) in bits/s. We do not know $\lambda(t)$ as a time function. It suffices to assume that the $S_i$ bits encoding picture $i$ arrive to the queue during the time interval from $(i-1)\tau$ to $i\tau$.

The server of the queue represents a channel (physical or logical) which sends the bits of picture $i$ to a network at the rate of $r_i$ bits/s. This rate is calculated for picture $i$ by an algorithm (which is to be designed and specified) whenever the server can begin sending picture $i$.

The algorithm has three parameters that can be specified:

- $K$ — required number of complete pictures buffered in queue before the server can begin sending the next picture ($0 \leq K \leq N$); specifically, the server can begin sending picture $i$ only if pictures $i$ through $i + K - 1$ have arrived (each has been completely encoded)
- $D$ — maximum delay specified for every picture in video sequence (seconds)

| | |
|---|---|
| $H$ | lookahead interval, in number of pictures, used by algorithm |

Note that if $K$ is specified to be $N$, the algorithm has knowledge of all picture sizes needed for ideal smoothing.[10]

The *delay of a picture* is defined to be the time of arrival of its first bit to the queue to the time of departure of its last bit from the queue. (The delay, so defined, includes the picture's encoding delay, queueing delay, and sending delay.) Note that the delay bound $D$ must be specified such that

$$D \geq (K+1)\tau \tag{1}$$

in order for the bound to be satisfiable.

The case of $K=0$ means that the server can begin sending the bits of picture $i$ buffered in the queue before the entire picture $i$ has arrived. We allow $K=0$ to be specified for the algorithm. However, using $K=0$ in an actual system gives rise to two problems. First, buffer underflow is possible unless the encoder is sufficiently fast. Second, the algorithm can ensure that picture delays are bounded by $D$ only if $K \geq 1$ (actually, if and only if $K \geq 1$; see Theorem 1 in Section 4.2).

The parameter, $H$, is for improving algorithm performance by looking ahead (even though only the pattern is known, but not necessarily picture sizes). Its meaning will become clear in Section 4.3.

We next define the following notation:

| | |
|---|---|
| $t_i$ | time when server can begin sending picture $i$ |
| $d_i$ | departure time of picture $i$ (the server has just sent the last bit of picture $i$) |

Additionally, at time $t_i$, the algorithm calculates the rate $r_i$. To simplify notation, and without loss of generality, the calculation is assumed to take zero time. The following equation defines the meaning of parameter $K$,

$$t_i = \max\{d_{i-1}, (i-1+K)\tau\} \tag{2}$$

That is, the server can begin sending picture $i$ only after picture $i-1$ has departed and pictures $i, i+1, \ldots, i-1+K$, have arrived (i.e., encoded and picture sizes are known).

The departure time of picture $i$ is

$$d_i = t_i + (S_i/r_i) \tag{3}$$

and the delay of picture $i$ is

$$\text{delay}_i = d_i - (i-1)\tau \tag{4}$$

Note that in an actual system, the encoding of picture $i-1+K$ may be complete at time $y$, such that $(i-2+K)\tau < y \leq (i-1+K)\tau$. Also the first bit of picture $i$ may arrive at time $x$, such that $(i-1)\tau < x < i\tau$. We use $(i-1+K)\tau$ in Eq. (2) and $(i-1)\tau$ in Eq. (4) because $\lambda(t)$ is unknown. If either $x$ or $y$ were known and used instead, the delay of each picture may be smaller than the value calculated using (2)–(4), but the difference would be negligible.

[10] Some modification is needed to ensure that the delay of each picture is less than $D$.

## 4.2 Upper and lower bounds on rate

We present an upper bound and a lower bound on the rate $r_i$ that can be selected by an algorithm for sending picture $i$ at time $t_i$, for all $i$. The lower bounds are used to ensure that the delay of each picture is less than or equal to $D$. We say that the algorithm *satisfies delay bound* $D$ if for $i = 1, 2, \ldots$

$$\text{delay}_i \leq D$$

The upper bounds on rates are used to ensure that the server works continuously. If rates are too large, then the server may send bits faster than the encoder can produce them, forcing the server to idle, i.e., the server cannot send the next picture because the queue does not have $K$ complete pictures.[11] We say that the algorithm *satisfies continuous service* if for $i = 1, 2, \ldots$

$$t_{i+1} = d_i$$

It might be argued that the delay bound is a more important property than the continuous service property. However, there is no need to choose, because Theorem 1 below shows that both properties can be satisfied. An assumption of Theorem 1 is that $S_i$ is known at time $t_i$, which can be guaranteed by specifying $K$ to be greater than or equal to 1. If $S_i$ is not known at time $t_i$ (i.e., $K$ is specified to be 0), it is easy to construct examples such that the delay bound cannot be satisfied.

**Theorem 1** If $S_i$ is known at $t_i$, and $r_i$ is selected for $i = 1, 2, \ldots, n$ such that conditions (5) and (6) hold,

$$r_i \geq \frac{S_i}{D + (i-1)\tau - t_i} \tag{5}$$

$$r_i \leq \frac{S_i}{(i+K)\tau - t_i} \qquad \text{if } t_i < (i+K)\tau \tag{6}$$

then for $i = 1, 2, \ldots, n$, (7), (8) and (9) in the following hold:

$$\text{delay}_i \leq D \tag{7}$$

$$t_{i+1} < i\tau + D \tag{8}$$

$$t_{i+1} = d_i \tag{9}$$

Theorem 1 is proved by induction on $n$. A proof can be found in [5, 6].

We use $r_i^L$ and $r_i^U$ to denote the lower bound in (5) and the upper bound in (6), respectively. For these upper and lower bounds, we say that a bound is *well defined* if its denominator is positive; see (5) and (6). In Theorem 1, (8) guarantees that the lower bounds are all well defined. As for the upper bounds, many in (6) may not be well defined. These are defined as follows:

$$r_i^U = \infty \qquad \text{if } t_i \geq (i+K)\tau.$$

Because of (1), the following corollary is immediate.

**Corollary 1** For all $i = 1, 2, \ldots, n$,

$$r_i^L \leq r_i^U \tag{10}$$

Corollary 1 implies that both the delay bound and the continuous service property can be satisfied.

[11] For $K = 0$, buffer underflow may occur.

## 4.3 Lookahead to improve algorithm performance

Theorem 1 requires that the rate for picture $i$ be chosen from the interval $[r_i^L, r_i^U]$, which may be large if $D > (K+1)\tau$. This flexibility can be exploited to reduce the number of rate changes over time. Suppose the sizes of pictures $i, i+1, i+2, \ldots$ are known. The algorithm can be designed to find a rate for sending pictures $i$ through $i+h$, for as large a value of $h$ as possible.

In our system model, however, the size of picture $j$, $j > i+K-1$, may not be known at time $t_i$. Specifically, for $K=1$, it is likely that $S_j$, $j > i$, has to be estimated. Fortunately, Theorem 1 requires only $S_i$ to be known at $t_i$. Sizes of pictures arriving in the future may be estimated without affecting Theorem 1.

In what follows, we derive a set of upper bounds and a set of lower bounds from $S_i, S_{i+1}, S_{i+2}, \ldots$, where $S_j$, $j > i+K-1$, may be an estimate. There are many ways to estimate the size of a picture from past information. In the experiments described in Section 5, the size of picture $j$, if not known at $t_i$, was estimated to be $S_{j-N}$. This is a simple estimate which uses the fact that pictures $j-N$ and $j$ are of the same type (I, B or P) in MPEG video. They are about the same size unless there is a scene change in the picture sequence from $j-N$ to $j$.

If all pictures in the future are sent at the rate $r_i$, the (approximate) delay of picture $i+h, h = 0, 1, 2, \ldots,$ is

$$t_i + \frac{\sum_{m=0}^{h} S_{i+m}}{r_i} - (i-1+h)\tau \tag{11}$$

Requiring the above to be $\leq D$, we have

$$r_i \geq \frac{\sum_{m=0}^{h} S_{i+m}}{D + (i-1+h)\tau - t_i} \tag{12}$$

where the lower bound on $r_i$ will be denoted by $r_i^L(h)$.

The (approximate) departure time of picture $i+h$ is

$$d_{i+h} = t_i + \frac{\sum_{m=0}^{h} S_{i+m}}{r_i}$$

The continuous service property requires that $d_{i+h} \geq (i+h+K)\tau$, which can be satisfied by requiring

$$r_i \leq \frac{\sum_{m=0}^{h} S_{i+m}}{(i+h+K)\tau - t_i} \qquad \text{if } t_i < (i+h+K)\tau \tag{13}$$

where the upper bound on $r_i$ will be denoted by $r_i^U(h)$ if $t_i < (i+h+K)\tau$; else, $r_i^U(h)$ is defined to be $\infty$.

Note that $r_i^L(0)$ and $r_i^U(0)$ are equal to the lower bound $r_i^L$ and upper bound $r_i^U$, respectively, given in Theorem 1. Also, only $r_i^L(h)$ and $r_i^U(h)$ for $h = 0, 1, \ldots, K-1$ are accurate bounds; the others, calculated using estimated picture sizes, are approximate.

A strategy to reduce the number of rate changes over time is to first find the largest integer $h^*$ such that

$$\max_{0 \leq h \leq h^*} r_i^L(h) \leq \min_{0 \leq h \leq h^*} r_i^U(h) \tag{14}$$

The rate $r_i$ for picture $i$ is then selected such that for $h = 0, 1, \ldots, h^*$

$$r_i^L(h) \leq r_i \leq r_i^U(h)$$

Note that for $K \geq 1$, the selected rate satisfies

$$r_i^L = r_i^L(0) \leq r_i \leq r_i^U(0) = r_i^U$$

Therefore, the hypothesis of Theorem 1 holds and the delay bound $D$ as well as the continuous service property are satisfied even though picture sizes (namely, $S_j$, $j > i$) are estimated.

To minimize delay, we would like to use $K=1$ in the algorithm, in which case most picture sizes are estimated. For this reason, the smoothing algorithm in Section 4.4 is designed with a parameter $H$ which can be specified. Instead of searching for the largest $h^*$ satisfying (14), the search is limited to a maximum value of $H-1$. For MPEG video, we conjecture that there is no advantage in having $H$ greater the size of a pattern ($N$) because picture sizes are estimated using past information. We conducted experiments to study this conjecture and found that it is supported by experimental data; see Section 5.

## 4.4 Algorithm design and specification

The smoothing algorithm is designed using (2)–(4), (12)–(14), Theorem 1, and Corollary 1. A specification of the *basic algorithm* is given in Figure 2. The following are assumed to be global variables:

> *pic_size*: **array** [*index*] **of** *integer*;
>
> *seq_end*: *boolean*;
>
> *tau*: *real*;

The value of *pic_size*[*i*] is $S_i$ in the system model, the value of *tau* is the picture period, and *seq_end*, initially *false*, is set to *true* when the algorithm reaches the last picture of a video sequence.

There are three functions in the specification: *max*, *min*, and *size*. In particular, *size*$(j,t)$ returns, at time $t$, either the actual size of picture $j$ or an estimated size (in number of bits). For the experimental results presented in Section 5, we used the following simple estimation based upon the fact that a known pattern of $N$ picture types repeats indefinitely in MPEG video:

> **if** $(t \geq j * tau)$ **then return** *pic_size*[*j*]
> **else return** *pic_size*[$j-N$]

For the initial part of a video sequence, where *pic_size*[$j-N$] is not defined, each I picture is estimated to be 200,000 bits, each P picture 100,000 bits, and each B picture 20,000 bits. These estimates are far from being accurate for some video sequences. But by Theorem 1, they do not need to be accurate.

An MPEG encoder may change the values of $M$ and $N$ adaptively as the scene in a video sequence changes. Note

```
procedure smooth(H, K: integer; D: real);
var   i, h, sum: integer;
      depart, time, rate, delay,
      lower, upper, lower_old, upper_old: real;
begin i := 0; depart := 0.0; seq_end := false;
  repeat i := i + 1;
    time := max(depart, (i − 1 + K) ∗ tau);
                              {time to begin sending picture i}
    h:=0; sum:=0; lower:=0.0; upper:=∞;
    repeat
      sum := sum + size(i + h, time);
      lower_old := lower; upper_old := upper;
      lower := sum/(D + (i − 1 + h) ∗ tau− time);
      if (time ≥ (K + i + h) ∗ tau) then upper:=∞
        else upper := sum/((K + i + h) ∗ tau− time);
      lower := max(lower, lower_old);
      upper := min(upper, upper_old);
      h := h + 1;
    until (lower > upper) or (h ≥ H);

    if (lower > upper) then
      if (lower > lower_old)
        then rate:=upper            {upper=upper_old}
        else rate := lower
               {lower = lower_old, upper < upper_old}
    else                                    {h = H}
      if (i = 1) then rate:=(lower+upper)/2;
                                {rate for first picture}
        else                {possible modification here}
          if (rate>upper) then rate:=upper
            else if (rate < lower)
              then rate := lower;
    notify(i, rate);
                {notify transmitter the rate for picture i}
    depart := time + pic_size[i]/ rate;
                            {departure time of picture i}
    delay := depart −(i − 1) ∗ tau     {delay of picture i}
  until seq_end
end; {smooth}
```

Figure 2: Specification of basic algorithm.

that the basic algorithm does not depend on $M$, and it uses $N$ only in picture size estimation.

Lastly, we use *notify*$(j, r)$ to denote a communication primitive which notifies a transmitter that picture $j$ is to be sent at rate $r$.

Note that the inner **repeat** loop calculates the bounds in (14). The loop has two exit conditions. The exit condition, (*lower* > *upper*), corresponds to $h^*$ in (14) being less than $H-1$; when this happens (called *early exit*), it can be proved that one of these two conditions holds:

- *lower* > *lower_old* and *upper* = *upper_old*
- *lower* = *lower_old* and *upper* < *upper_old*

The selection of $r_i$ in each case is designed to minimize the number of rate changes over time.

The second exit condition corresponds to $h^*$ in (14) being larger than or equal to $H - 1$ (called *normal exit*); in the algorithm, the search for $h^*$ stops at $h = H - 1$ because the lookahead interval is limited to $H$ pictures. Upon normal exit, $r_i$ is selected to be the same as $r_{i-1}$, i.e., no rate change unless the current value of *rate* is larger than *upper* or smaller than *lower*. This selection strategy is designed to minimize the number of rate changes.

We also investigated a variation of the basic algorithm such that the moving average calculated using

$$rate := sum/(N * tau) \qquad (15)$$

is selected for $r_i$ (unless the moving average is larger than *upper* or smaller than *lower*). To modify the algorithm, the assignment statement in (15) replaces the comment "{possible modification here}" in procedure *smooth*. The modified algorithm produces numerous small rate changes over time, but its rate $r(t)$, as a function of time, tracks the rate function of ideal smoothing more closely than the basic algorithm. In particular, the area difference (a performance measure defined in Section 5) is smaller.

# 5 Experiments

To show that the smoothing algorithm is effective and satisfies the correctness properties given in Theorem 1, we performed a large number of experiments using four MPEG video sequences. Some of our experimental results are shown in Figures 4–8 and discussed below. For all experiments, the picture rate is 30 pictures/s.

## 5.1 MPEG video sequences

Driving1 ($N = 9$, $M = 3$) and Driving2 ($N = 6$, $M = 2$)

This video was chosen because we thought that it would be a difficult one to smooth. There are two scene changes in the video. Initially, the scene is that of a car moving very fast in the countryside. The scene then changes to a close-up of the driver, and then changes back to the moving car. This video is encoded twice, using different coding patterns, to produce two MPEG video sequences. Note, from Figure 3, that the scene changes give rise to abrupt changes in picture sizes. In particular, P and B pictures in the driving scenes are much larger than P and B pictures in the close-up scene. The pictures were encoded with a spatial resolution of 640 × 480 pixels.

Tennis ($N = 9$, $M = 3$)

This video shows a tennis instructor initially sitting down and lecturing. He then gets up to move away. There is no scene change in the video. But as the instructor gets up, his motion gives rise to increasingly large P and B pictures. These changes in picture sizes are gradual. However, there are two isolated instances of large P pictures in the first half of the sequence. The pictures were encoded with a spatial resolution of 640 ×480 pixels.

Backyard ($N = 12$, $M = 3$)

There are also two changes of scene in this video. Initially, the scene is that of a person in a backyard. The scene

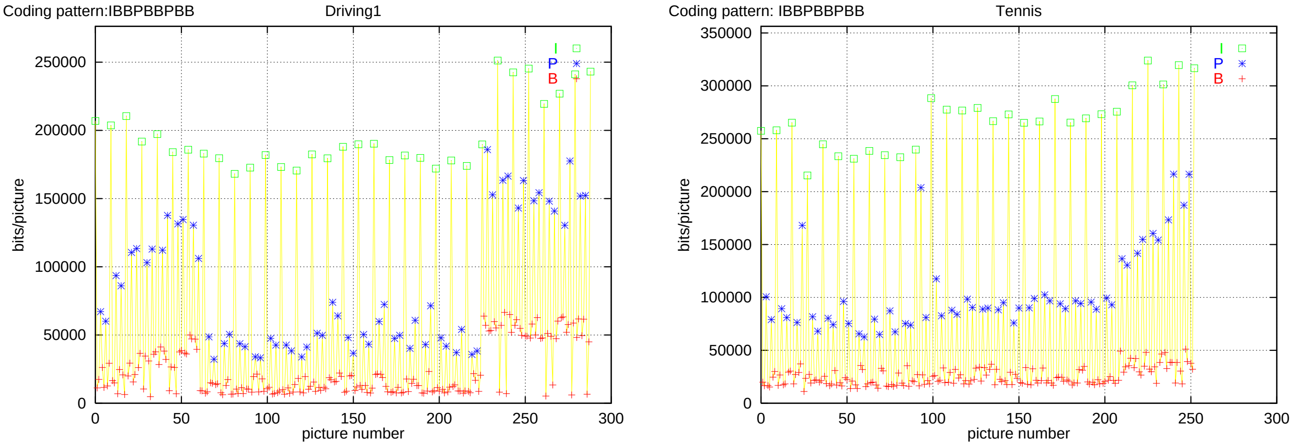


Figure 3: Two MPEG video sequences.

changes to two other people in another area of the backyard, and then changes back to the first person. The backgrounds of both scenes are complex with many details. While there are movements, the motion is not rapid. The pictures were encoded with a spatial resolution of $352 \times 288$ pixels.

Due to space limitation, only Driving1 and Tennis are shown in Figure 3.

## 5.2 Performance of the basic algorithm

For the Driving1 sequence, Figure 4 shows bit rate as a function of time for $K = 1$, $H = 9$, and four values of the delay bound $D$. In each case, we compare the rate function from the basic algorithm, denoted by $r(t)$, with the rate function from ideal smoothing, denoted by $R(t)$. From Figure 4, we see that the "smoothness" of $r(t)$ improves as the delay bound is relaxed. (We will define some quantitative measures of smoothness below.)

For $D = 0.1$ second, $r(t)$ does not look smooth at all, even though it is a lot smoother than the rate function $\lambda(t)$ of the MPEG encoder output. (If $\lambda(t)$ is not smoothed, the largest I picture would require over 7.5 Mbps to send in 1/30 second.) Note that the improvement in smoothness from $D = 0.2$ second to $D = 0.3$ second is not significant. Therefore, $D = 0.2$ second would be an excellent parameter value to use if a delay of up to 0.2 second (which includes encoding delay) is an acceptable price to pay for a smoothed output.

Note that the smoothed rate function of the Driving1 sequence varies from about 1 Mbps to 3 Mbps. These variations are due to differences in the content and motion of scenes. These rates depend upon the spatial resolution ($640 \times 480$) and quantizer scales (4 for I, 6 for P, and 15 for B) specified for encoding the sequence.

For the Driving1 sequence, Figure 5 shows the delays of pictures for two comparisons. In the graph on the left, we compare these three cases:

- $D = 0.1$ second, $K = 1$, $H = 9$, basic algorithm
- $D = 0.3$ second, $K = 1$, $H = 9$, basic algorithm
- ideal smoothing

As shown, the delays of pictures are bounded by 0.1 second and 0.3 second as specified for the basic algorithm. For ideal smoothing, picture delays are large, due to the requirement that pictures in the same pattern are buffered until all have arrived before the first picture in the pattern can be transmitted.

In the graph on the right of Figure 5, we compare these three cases

- $K = 1$, $H = 9$, $D = 0.1333 + (K+1)/30$ second, basic algorithm
- $K = 9$, $H = 9$, $D = 0.1333 + (K+1)/30$ second, basic algorithm
- ideal smoothing

For $K = H = N = 9$, the smoothing algorithm does not estimate picture sizes. In this case, the basic algorithm is very similar to ideal smoothing.[12]

A comparison of the delays for the two cases, $K = 1$ and $K = 9$, shows the desirability of using $K = 1$. The *slack* in the delay bound is chosen to be the same, 0.1333 second, so that the smoothness of $r(t)$ is about the same in both cases (see discussion on Figure 8 below).

No "delay bound violation" has been observed in any of our experiments where $K \geq 1$. This is not surprising, since the absence of delay bound violation is guaranteed by Theorem 1 if $K \geq 1$. For $K = 0$, however, we did observe some delay bound violations when the slack in the delay bound was deliberately made very small.

Different quantitative measures can be defined to characterize the effectiveness of smoothing. We use four of them to study algorithm performance as each of the parameters, $D$, $H$, $K$, varies. The first measure is defined as follows:

$$\text{Area difference} = \frac{\int_0^T [r(t) - R(t + (N-K)\tau)]^+ \, dt}{\int_0^T R(t + (N-K)\tau) \, dt} \quad (16)$$

where $T$ denotes the time duration of the video sequence. Note that with ideal smoothing, picture 1 begins transmission $(N-K)\tau$ seconds later than if the basic algorithm were

[12] They are not identical, because ideal smoothing as described in Section 3.2 does not try to keep the delay of each picture less than a specified bound $D$.

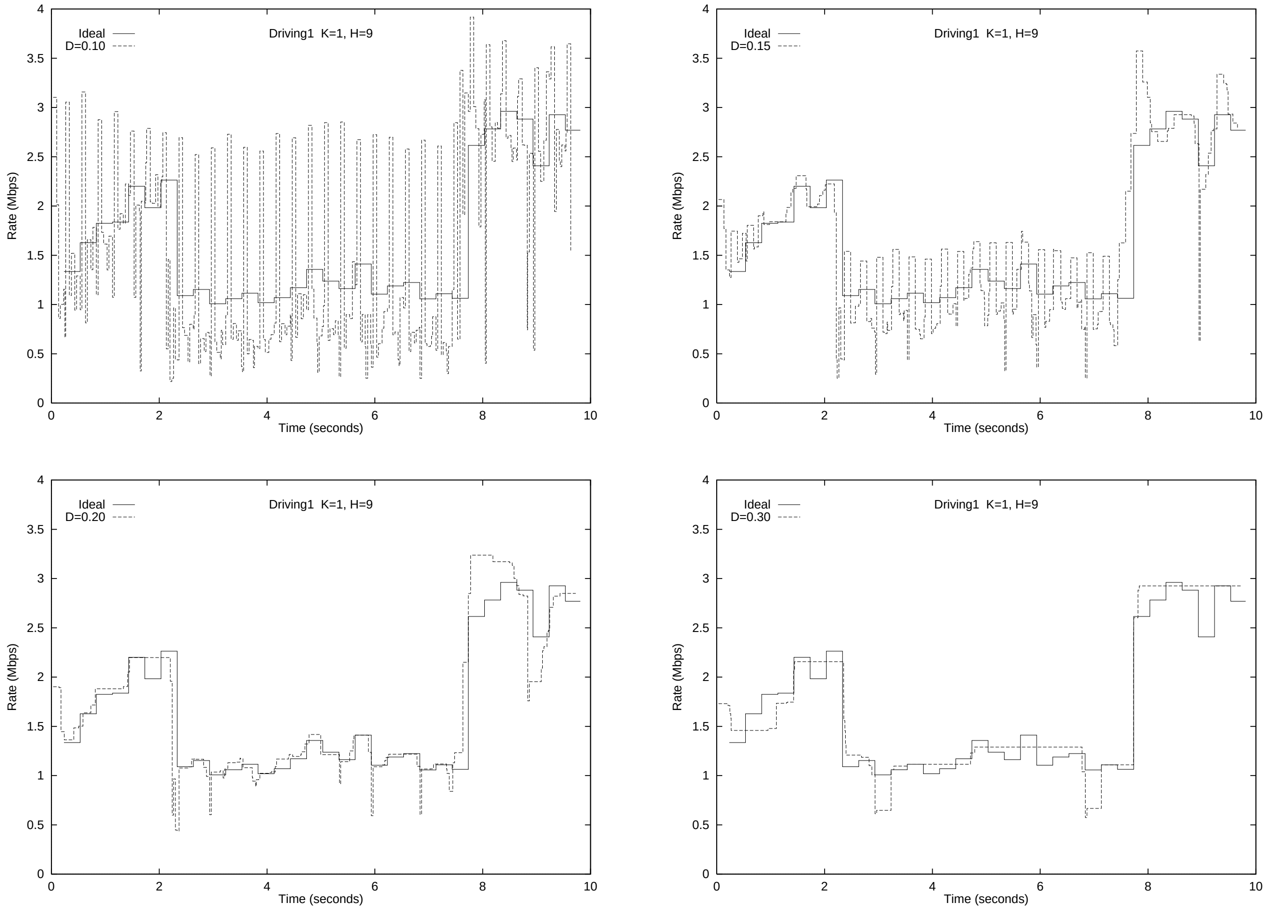


Figure 4: Rate as a function of time for four delay bounds (Driving1 sequence, basic algorithm).

used. Therefore the rate function from ideal smoothing is shifted by this much time in (16). Only the positive part of the difference between $r(t)$ and $R(t)$ is used in (16) because of the following:

$$\int_0^T [r(t) - R(t + (N - K)\tau)]\, dt = 0$$

We use three other measures:

- the number of times $r(t)$ is changed by the algorithm over $[0, T]$
- the maximum value of $r(t)$ over $[0, T]$
- the standard deviation (S.D.) of $r(t)$ over $[0, T]$

Figure 6 shows the four quantitative measures as a function of delay bound $D$ for the four MPEG video sequences. All four measures indicate that as the delay bound is increased (relaxed), the rate function $r(t)$ becomes more smooth. The Backyard sequence appears to be the easiest to smooth. For the three MPEG video sequences encoded at a spatial resolution of 640 $\times$ 480 pixels, the maximum smoothed rate is about 3 Mbps. For the Backyard sequence encoded at a spatial resolution of 352 $\times$ 288 pixels, the maximum smoothed rate is about 1.5 Mbps, which is about the target rate of the MPEG standard. The maximum (smoothed) rate versus $D$ curves in Figure 6 represent a valuable design tradeoff made possible by lossless smoothing.

Figure 7 shows the quantitative measures as a function of the lookahead interval, $H$, for the four MPEG video sequences. In Section 4.3, we conjectured that because most picture sizes are estimated using past information, there is no advantage in having $H$ larger than the size of the repeating pattern ($N$). Our experimental data support this conjecture. In Figure 7, the area difference, standard deviation of rate, and maximum rate do not show any noticeable improvement for values of $H$ larger than $N$. In fact, the number of rate changes increases as $H$ increases.

$K$ should be as small as possible to reduce picture delay. Theorem 1 requires $K \geq 1$. We conducted experiments to investigate whether there is any improvement in smoothness of $r(t)$ from using $K > 1$. Figure 8 shows that there is a small improvement as $K$ increases, but barely noticeable. Note that the delay bound is $D = 0.1333 + (K+1)/30$, with a constant slack of 0.1333 for all cases. We conclude that $K = 1$ should be used.

# 6 Conclusions and Related Work

As part of our research project on the design of transport and network protocols for multimedia applications, we studied

MPEG video. We found that interframe compression techniques, such as the ones specified by MPEG, give rise to a coded bit stream in which picture sizes differ by a factor of 10 or more. As a result, some buffering is needed to smooth the picture-to-picture rate fluctuations in the coded bit stream; otherwise, the very large fluctuations would make it very difficult to allocate a communication channel (based upon either packet switching or circuit switching) with appropriate quality-of-service guarantees.

Some researchers have described techniques for controlling the output rate of VBR encoders [2, 4, 9]. Most of these techniques are lossy, and are inappropriate for smoothing picture-to-picture rate fluctuations that are a consequence of interframe compression. For alleviating network congestion, the lossy techniques may be used *in addition to* lossless smoothing. However, they should be used only as a last resort, because both the spatial and temporal redundancy present in video are greatly reduced in MPEG bit streams. On the other hand, a lossless algorithm, such as the one presented in this paper, should always be used to smooth out rate fluctuations from interframe compression.

Our algorithm is designed to satisfy a delay bound, $D$, which is a parameter that can be specified. The algorithm is characterized by two other parameters, $K$, the number of pictures with known sizes, and, $H$, a lookahead interval for improving algorithm performance. We also presented a theorem which states that if $K \geq 1$, then our algorithm satisfies both the delay bound $D$ and a continuous service property.

Although our system model and algorithm, as well as Theorem 1, were motivated by MPEG video, they are applicable to compressed video in general. We make use of the assumption that there is a known pattern of picture types which repeats indefinitely in the video sequence, to estimate picture sizes. Such size estimates are used in a lookahead strategy to improve algorithm performance.

The problem of smoothing was analyzed by Ott et al. [8], where picture sizes in a video sequence are assumed to be known a priori. The parameter $K$ is absent in their model, and there is no notion of a repeating pattern [8]. From a practical point of view, $K$ is a crucial parameter for any smoothing algorithm. Furthermore, Theorem 1 shows that there is no need to assume all picture sizes to be known a priori. Instead, we use a known pattern and estimated picture sizes in our algorithm.

We conducted a large number of experiments using statistics from four MPEG video sequences to study the performance of our algorithm. We found that it is effective in smoothing rate fluctuations, and behaves as described by Theorem 1. Experimental data suggest that the following choice of parameters provides a smooth rate function: $K = 1$, $H = N$, and $D = 0.2$ second. The delay bound includes the encoding delay of each picture. A larger delay bound does not seem to provide any noticeable improvement in the smoothness of the resulting rate function. In the case that a multimedia application requires a smaller delay bound, the rate fluctuations would be noticeably larger.


# Acknowledgements

We thank Thomas Y. C. Woo for his technical comments and his help in the preparation of figures. We also thank the referees for their constructive comments.


# References


[1] M. Anderson. VCR quality video at 1.5 Mbits/s. In *National Communication Forum*, October 1990.

[2] L. Delgrossi, C. Halstrick, D. Hehmann, R. G. Herrtwich, O. Krone, J. Sandvoss, and C. Vogt. Media scaling for audiovisual communication with the Heidelberg transport system. In *Proceedings of ACM Multimedia '93*, pages 99–104, August 1993.

[3] D. Le Gall. MPEG: A video compression standard for multimedia applications. *CACM*, 34(4):46–58, April 1991.

[4] H. Kanakia, P. Mishra, and A. Reibman. An adaptive congestion control scheme for real-time packet video transport. In *Proceedings of ACM SIGCOMM '93*, pages 20–31, September 1993.

[5] S. S. Lam. A model for lossless smoothing of compressed video, January 1994. Unpublished manuscript (presented at Experts on Networks Symposium, 60th birthday of Professor Leonard Kleinrock, UCLA, June 1994).

[6] S. S. Lam, S. Chow, and D. K. Y. Yau. An algorithm for lossless smoothing of MPEG video. Technical Report TR-94-04, Department of Computer Sciences, University of Texas at Austin, February 1994.

[7] Coding of moving pictures and associated audio, November 1991. SC29/WG11 committee (MPEG) draft submitted to ISO-IEC/JTC1 SC29.

[8] T. Ott, T. Lakshman, and A. Tabatabai. A scheme for smoothing delay-sensitive traffic offered to ATM networks. In *Proceedings of INFOCOM '92*, pages 776–785, 1992.

[9] P. Pancha and M. El Zarki. Bandwidth requirements of variable bit rate MPEG sources in ATM networks. In *Proceedings of INFOCOM '93*, pages 902–909, March 1993.

[10] A. Reibman and A. Berger. On VBR video teleconferencing over ATM networks. In *Proceedings of GLOBECOM '92*, pages 314–319, 1993.

[11] D. Reininger, D. Raychaudhuri, B. Melamed, B. Sengupta, and J. Hill. Statistical multiplexing of VBR MPEG compressed video on ATM networks. In *Proceedings of INFOCOM '93*, pages 919–925, March 1993.

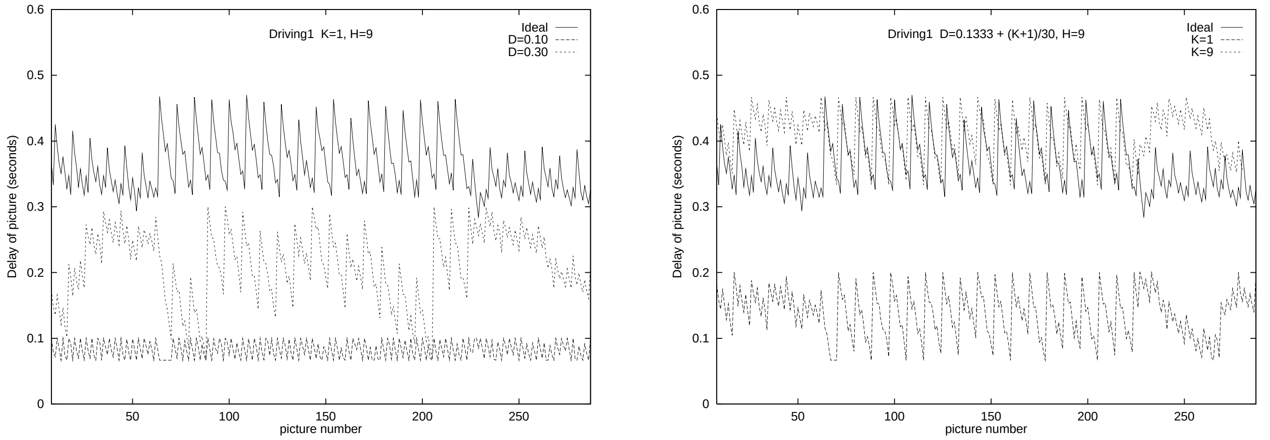


Figure 5: Delays of pictures in Driving1 sequences (basic algorithm).

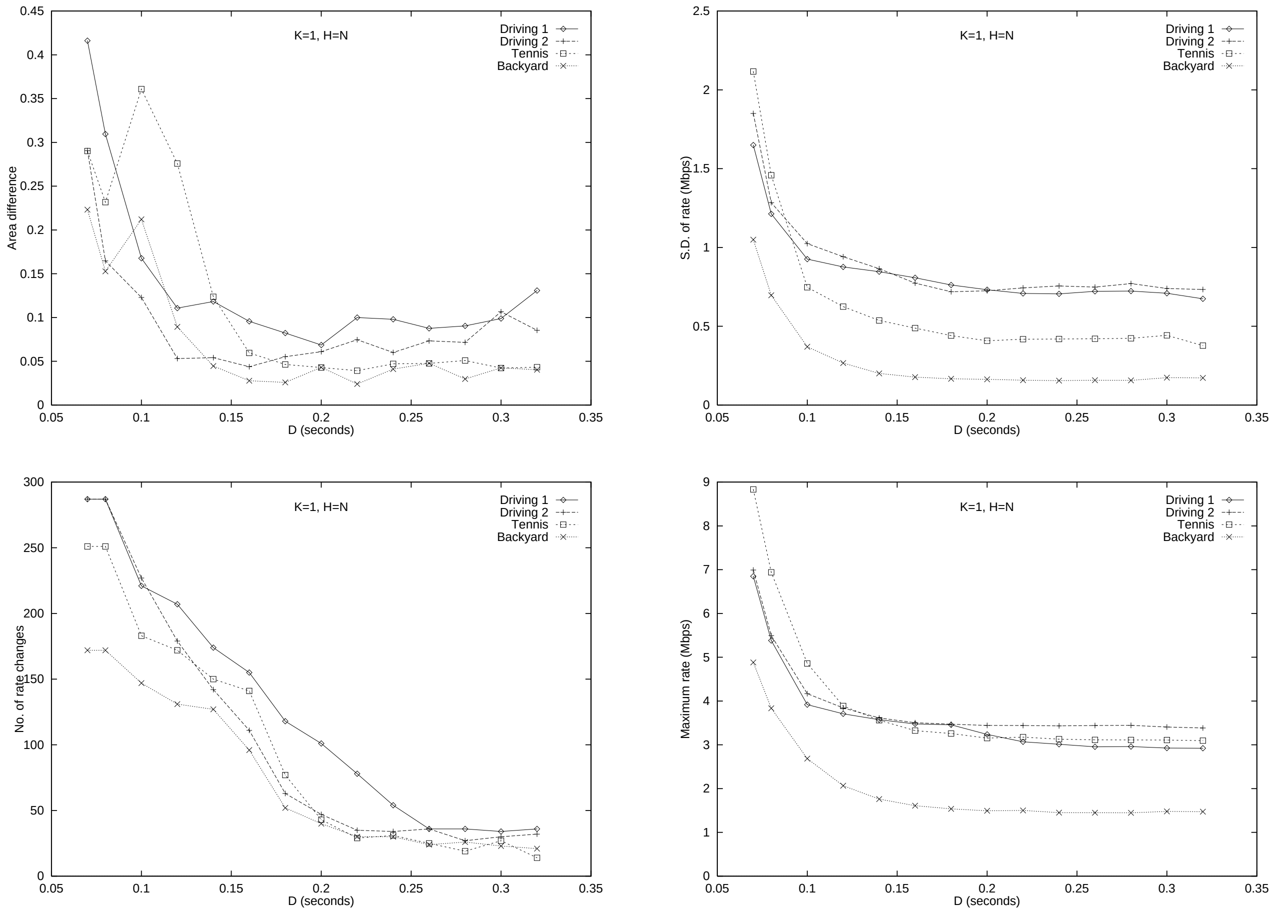


Figure 6: Performance of basic algorithm as a function of delay bound $D$.

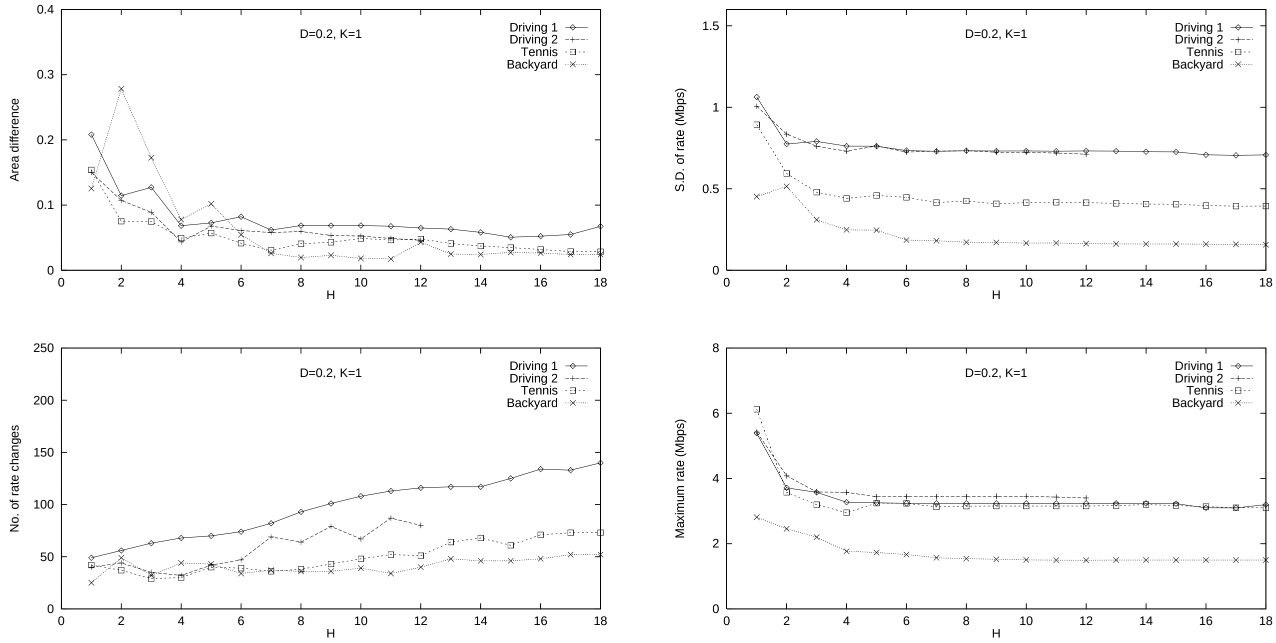


Figure 7: Performance of basic algorithm as a function of parameter $H$.

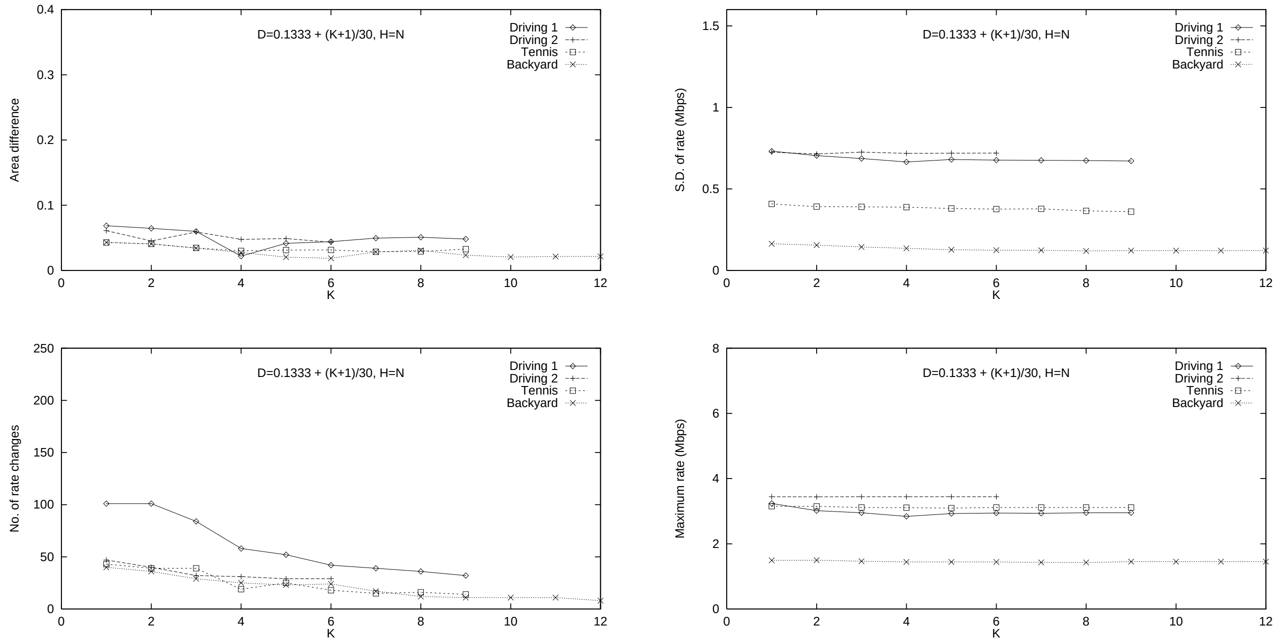


Figure 8: Performance of basic algorithm as a function of parameter $K$.