A Scalable and Adaptive Temporal Segmentation Algorithm for Video Coding

S. H. Kwok*

Department of Electrical and Electronic Engineering, Imperial College of Science, Technology and Medicine, Exhibition Road, London SW7 2BT, England; and Department of Electronic Engineering, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong

W. C. Siu†

Department of Electronic Engineering, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong

and

A. G. Constantinides‡

Department of Electrical and Electronic Engineering, Imperial College of Science, Technology and Medicine, Exhibition Road, London SW7 2BT, England

Received November 20, 1995; revised February 14, 1997; accepted March 4, 1997

1. INTRODUCTION

Video coding has gained increasing interest in the past few years. The temporal decimation technique is regarded as a powerful tool for increasing compression for video coding. Spatial decimation algorithms can also be used in conjunction with other video processing techniques such as using the motion vector estimation technique [1, 2] for a further improvement of the compression ratio. The temporal decimation approach became mature recently when the advancement of this approach using Olstad’s adaptive temporal decimation [3] made some significant contributions. The approach potentially could achieve very low bit-rate video coding.

Obviously, a simple temporal decimation can be achieved by periodically dropping whole frames from image sequences. However, this may cause objectionable temporal aliasing artifacts, especially in regions where temporal activity is high. Olstad’s algorithm adaptively selects important blocks for decimation, resulting in reconstructed sequences of better quality.

Although temporal decimation algorithms can achieve very high compression ratio, they are prone to some problems. These include (1) no adaptive control on the number of segments; (2) not being able to preserve all important blocks; (3) portability; (4) ineffective coding of the Block Position Map (BPM); and (5) too many Intra frames.

In fact, an adaptive temporal decimation algorithm is similar in some sense to algorithms which preserve facial...
position and perform accurate coding on that particular area. The conventional facial position or background detection algorithms, such as the ones in Refs. [4, 5], locate the human body and the facial position in “head and shoulders” images using geometric properties for videophone systems. These algorithms work if and only if the images are involved. No extensive report for a dynamic background has been described in the literature. Another closest approach is segmented video coding [6], which performs nonlinear filtering on images such that the background and the stationary objects are highly correlated between two successive frames. Although this could be effective for extraction of the dynamic background and is able to control the bit rate, spatial filtering causes serious distortion over the images if very low bit rate is used. Hence, it appears that spatial segmentation for video coding is not as efficient as temporal segmentation. Olstad’s temporal discontinuity detection is sophisticated enough to identify the background in an adaptive manner. However, this is subject to the size of the time window \( N \) and the number of remaining frames \( K \). Our proposed algorithm using \( N \)-frames decimation is able to identify the background area even when the video has no permanent background.

Many of the above drawbacks have been reduced or eliminated for our scalable and adaptive temporal segmentation algorithm (SATS). Our algorithm can optimize video sequences according to criteria defined by users. The criteria such as the bit rate, the objective quality measure, and even some specific image features are formulated in the form of a cost function. The number of discontinuity segments in the time window can be reduced by recursive operation on this cost function until it meets a specific requirement. Therefore this approach can be used for various applications, ranging from very low bit-rate coding to pattern recognition processing if the edge feature can be included in the cost function.

2. OLSTAD’S ADAPTIVE TEMPORAL DECIAMATION ALGORITHM

Recently Olstad has proposed an adaptive temporal decimation algorithm [3] for video coding and claimed that a very low bit-rate coding scheme could be achieved. The main idea of the algorithm is to select adaptively \( K \) important blocks in the temporal domain and to fill them up with the corresponding remaining frames. Results of experimental work showed that the degradation of the reconstructed sequence could be kept to a minimum. All remaining frames as well as the additional block position maps were coded and transmitted for the reconstruction.

Figure 1 shows a graphical representation of the temporal decimation algorithm. Figure 1a indicates a time window while Fig. 1b explains the spatial temporal decimation approach. A fixed number of image frames from a sequence are grouped as the time window as shown in Fig. 1a. The size \( N \) of the time window is predefined. This value is gradually optimized to \( K \) frames (which is equal to the total number of Inter frames) for this temporal decimation approach. The value of \( K \), which is an unknown before the process, gives the total number of remaining frames. Each time window is spatially subdivided into block sequences as shown in Fig. 1b. The block size has been selected as \( 8 \times 8 \) pixels in our verification of the algorithm.

Let us also assume that the size of each frame is \( x_{\max} \) by \( y_{\max} \) pixels; hence there are \((x_{\max}/8) \times (y_{\max}/8)\) blocks in each frame. The block located at the \( m \)th row and \( n \)th column with the frame at time \( t \) can be referred to as \( B(m, n, t) \), where

\[
\begin{align*}
m &= 0, 1, \ldots, (x_{\max}/8) - 1 \\
n &= 0, 1, \ldots, (y_{\max}/8) - 1 \\
t &= 0, 1, \ldots, N - 1,
\end{align*}
\]

and \( N \) is the number of frames in the time window. Hence, the intensity of each element within any block, \( B(m, n, t) \), can be written as

\[
I(8m + i, 8n + j, t) \quad \text{for } i, j = 0, 1, \ldots, 7.
\]
For temporal decimation, a number of blocks within the block sequence have to be identified and discarded. The blocks to be eliminated contain relatively low activity in the block sequence. They are known as low activity blocks. On the other hand, blocks remaining in the resulting block sequence are high activity blocks. The number of removed blocks in a block sequence is \( N - K \). After the elimination of the low activity blocks, the block sequence can be considered as composed of segments. Let us define a discontinuity plane to be the first block of a segment. Then all discontinuity segments are filtered with respect to their segments by an averaging filter, which is defined as follows.

Let

\[
B(m, n) = \{ \cdots, B_e(m, n, 1), B_e(m, n, 2), B_e(m, n, 3), B_e(m, n, 4), B_e(m, n, 5), B_e(m, n, 6), B_e(m, n, 7), \cdots \}
\]

be a possible sequence of frames at a certain position \((m, n)\) in the temporal domain, where the subscript \(e\) stands for blocks to be eliminated and \(r\) refers to remaining blocks. The average filter will replace \(B_e(m, n, 4)\) with \(B_e(m, n, 4)'\), for which an element in \(B_e(m, n, 4)’\) is the average of the corresponding elements in \(B_e(m, n, 4), B_e(m, n, 5), B_e(m, n, 6), B_e(m, n, 7), \cdots \)

Three prime components, Intra, Inter, and BPM, are required for the reconstruction of the complete block sequence. The first block in the block sequence, which is called the Intra block, acts as a reference for the rest of the blocks. Blocks other than the Intra block are Inter blocks. In order to reconstruct the complete block sequences, a mechanism, the block position map (BPM), is needed to indicate the positions of the discarded blocks. In our example in Fig. 1b, zeros indicate the missing blocks.

3. POSSIBLE IMPROVEMENTS OF THE ADAPTIVE TEMPORAL DECIMATION ALGORITHM

The adaptive temporal decimation algorithm decimates a video sequence in the temporal direction. It can outperform the nonadaptive decimation approaches in terms of the quality of the reconstructed image sequences when the bit rate is kept fixed. The \( K \) most important blocks based on the Mean Absolute Difference (MAD) criterion are packed in appropriate positions in the \( K \) remaining frames.

There are a number of problems in using adaptive temporal decimation algorithms for video coding, although it has been claimed that they could be used for very low bit-rate applications.

1. No adaptive control on the number of segments. Blocks with few or little spatial activities are compulsively decimated into a specified number of segments; however, we expect that very few bits should be used to represent them. On the other hand, blocks with significant changes are confined to a limited number of segments, but more bits should be required to represent them. Hence distortion is inevitable. The number of segments in a block sequence is affected by the number of remaining frames, \( K \). In other words, the bit rate should depend heavily upon the input sequence.

2. Not being able to preserve all important blocks. High activity blocks may be anywhere in the time window. In typical cases, continuous motions often occur in certain block sequences. They should all be retained in order to produce a faithful reconstruction. Adaptive temporal decimation algorithms fail to handle this situation. They result in temporal aliasing artifacts occurring in the reconstruction sequence. This effect is illustrated in Fig. 2, which consists of two time windows, one with more high activity blocks (indicated by ones), the other with more low activity blocks (indicated by zeros).

Assume that \( N = 8 \) and \( K = 3 \); it means that the total number of frames is equal to 8, including the Intra frame. Since \( K \) is 3, three blocks have to be selected in both cases. For the time window with more high activity blocks, two high activity blocks are missing, while the window with more low activity blocks contains an extra remaining block.

3. Problem of portability. There are no guidelines for determining the size of the time window \( N \). The value of \( N \) varies for different video sequences.

4. Ineffective coding of the block position map (BPM). The entropy of the BPM is rather high for any image sequence due to the fixed number of ones in the map and the relatively small compression ratio for variable length coding (VLC).

A simple exercise can give a clear demonstration of the problem.

Consider a CIF video sequence of size \( 355 \times 288 \) with a frame rate of 30. The size of each block sequence is 64 pixels. The size of the BPM before compression is \( 355 \times 288 \times 30 / 64 \approx 48 \) Kbps if \( N = 30 \). The compression ratio (CR) depends mainly on the total number of ones in the BPM, while the number of ones depends on the ratio of \( N \) and \( K \). Obviously, the greater the ratio the better the compression.
an adaptive decimation algorithm is to keep high activity blocks and to abandon low activity blocks in the temporal direction as much as possible. In Fig. 3, there are three sets of block sequences and their corresponding BPMs for each case. Using the three sets of block sequences, we apply various approaches on them. By observing the numbers of discontinuity planes, we can evaluate their performance with respect to the objective. We use Olstad’s algorithm, mentioned in Section 2; the results are shown in Fig. 3a. An N-frames decimation can be used to include all high activity blocks and to keep them in the remaining frames. Figure 3b illustrates the situation of keeping all six high activity blocks. A K-frames decimation is also shown to exclude \( N - K \) low activity blocks from the time window. Due to this specification, one high activity block in the second block sequence has been left out and has not been taken into consideration for reconstruction, as shown in Fig. 3c. These two alternatives fulfill the primary objective of the adaptive algorithm. However, they both have a compatibility problem with well-known frame-based standards, H.261 and MPEG, for which they code videos on a frame-to-frame basis.

Comparing the three algorithms shown in Fig. 3 in terms of image quality, it is certain that the N-frames decimation provides the best quality, since it preserves all high activity blocks. There should be no visual distortion from reconstruction if it is used in conjunction with the visually lossless quantization. K-frames decimation and Olstad’s algorithm could have equivalent visual quality. However, from the coding point of view, the number of coded blocks is quite expensive than Inter frames. Each time window needs one Intra frame to eliminate all quantization errors propagating across the window boundaries.

For an Intra frame of the “Miss America” sequence at a resolution of 355 × 288 pixels, the size of the JPEG compressed image is

16 Kbits (CR = 52, marginal image quality)

and

25.3 Kbits (CR = 32, acceptable image quality)

More Intra frames are required for the 30 frames; hence very high compression on Inter frame is required if limited bandwidth of the channel is assumed.

We provide two possible improvements in this paper. Figure 3 gives an illustration of our approaches and Olstad’s algorithm. Recall that the ultimate objective of

\[ N=8, K=3 \]

5. Too many Intra frames. Intra frames are much more expensive than Inter frames. Each time window needs one Intra frame to eliminate all quantization errors propagating across the window boundaries.

For an Intra frame of the “Miss America” sequence at a resolution of 355 × 288 pixels, the size of the JPEG compressed image is

16 Kbits (CR = 52, marginal image quality)

and

25.3 Kbits (CR = 32, acceptable image quality)

More Intra frames are required for the 30 frames; hence very high compression on Inter frame is required if limited bandwidth of the channel is assumed.

We provide two possible improvements in this paper. Figure 3 gives an illustration of our approaches and Olstad’s algorithm. Recall that the ultimate objective of

\[ N=8, K=3 \]

4. VISUALLY LOSSLESS QUANTIZATION

We propose here a simple but efficient method for detecting discontinuity of segments in block sequences for our adaptive decimation algorithm. Let us sample a sequence with a fixed time window of N frames. The time windows eventually have to be decimated into windows to have K remaining frames. Each time window is spatially
subdivided into block sequences. The block size is selected as $8 \times 8$ pixels in the experiments of this paper.

For a typical $b$-bit gray-scale image representation, the number of intensity levels is $2^b$, where usually $6 \leq b \leq 8$. It is noteworthy that quantization levels less than 64 will introduce false contours or edges to the human visual system [7]. Therefore, a quantization process using 6 bits representation appears to be lossless to the HVS. The number of representation levels is not so important for our approach because we only make use of this representation to detect temporal continuity within the block sequences. The process of uniform quantization and compact representation of an image at a resolution of $x_{\text{max}} \times y_{\text{max}}$ pixels can be described by the following equation:

$$I_q(x, y, t) = \text{int} \left[ \frac{I(x, y, t) + k}{S} \right]$$

for $x = 0, 1, \ldots, x_{\text{max}} - 1$

$y = 0, 1, \ldots, y_{\text{max}} - 1$

$t = 0, 1, \ldots, N - 1.$

Note that $(x, y)$ gives the location of the pixel and $t$ stands for the frame number in the temporal direction. The expression $\text{int}[a]$ extracts the integer portion of $a$, while $S$ is the quantizer step. The parameter $k$ takes the value of 1 for quantization with rounding to the nearest integer, and 0 for quantization with truncation. For using a 6-bit representation, $S$ is equal to 4 if the initial range of $I(x, y, t)$ is from 0 to 255.

Let us apply the above quantization process to the current time window. The mean absolute difference (MAD) matching criterion is used to identify the discontinuity segments. Due to the fact that the dynamic range of the gray levels has been greatly reduced, a continuity segment in the block sequence has very small MAD with adjacent blocks in the temporal domain,

$$\text{MAD}(m, n, t) = \sum_{i=0}^{7} \sum_{j=0}^{7} |I(8m + i, 8n + j, t) - I(8m + i, 8n + j, t - 1)|$$

for $t = 1, 2, \ldots, N - 1,$

where $(m, n)$ is the spatial location of the block;

$$\text{MAD}(m, n, t) = \begin{cases} \text{continuous} & \text{if } \text{MAD}(m, n, t) \leq \text{thr} \\ \text{discontinuous} & \text{otherwise.} \end{cases}$$

The constant $\text{thr}$ is a predefined threshold which is equal to 1 for the representation using six bits. All temporal activities are preserved by the corresponding segments. The topology of the temporal segments is kept in the BPM, while low activity blocks are indicated by 0 and high activity blocks are indicated by 1.

5. SCALABLE AND ADAPTIVE TEMPORAL SEGMENTATION ALGORITHM

The temporal segmentation is a useful means which can transform the input video signals into another representa-
tion. It contains a smaller number of discontinuity segments but the visual quality is good and it is lossless to HVS. Since it maintains the same format of the output as for the input, there is no problem of compatibility with the H.261 and MPEG standards. Temporal decimation is also a very powerful method for reducing data to be processed in the temporal domain of the coding system. The scalable and adaptive temporal segmentation (SATS) algorithm, which is a combination of temporal decimation and temporal segmentation, is basically a refined version of \( N \)-frames decimation. It preserves all high activity blocks and is capable of optimizing the bit-rate and the format of the output is the same as the input. With the additional scalable and adaptive features as well as the advantages of the \( N \)-frames decimation, surely our proposed algorithm is appropriate for various video processing applications.

Let us consider a video sequence with time window \( N = 4 \) as shown in Fig. 4. Recall that the BPM can be used to register the positions of the remaining blocks of the time window. Let \( b_p \) be an indicator to mark a particular block in a block sequence within a time window; i.e., when \( b_p = 1 \), the block is to be kept and to be evaluated by a cost function; when \( b_p = 0 \), the block is to be ignored. The remaining blocks are rearranged for further process and they are stored in a packed BPM expressed by \( b_p' \). The variable \( b_p' \) is used to effect the cost evaluation of segments. The cost function, \( C \), is a customizing function for measuring user’s specified criterion between two consecutive segments. Criteria that can be represented mathematically such as similarity and dissimilarity between segments are possible cost functions. By recursively removing the least cost segment, the system specifications, such as the bit rate of the channel and the visual quality of the reconstruction and so on, can be met. Time windows are overlapped by one frame, except the first time window. The last reconstructed frame of the last time window is used as a reference for the current time window.

Figure 4 illustrates the situation of using the \( N \)-frames decimation, when \( N = 4 \). The first frame of the entire time window is regarded as the reference of the whole time window and to be coded using an intra frame coding. The rest of the frames are coded as motion-compensated prediction errors. In our example, different values of \( b_p \), indicated by \( b_{p_{tw}}(m, n, t) \) of the current time window (tw) for blocks at spatial location \((m, n)\) and at \( t \in (1, 2, \ldots, N - 1) \) are shown such as follows:

\[
\begin{align*}
    b_{p_{tw}}(m, n, 1) &= 1 \\
    b_{p_{tw}}(m, n, 2) &= 0 \\
    b_{p_{tw}}(m, n, 3) &= 1,
\end{align*}
\]

where \( m \in \{0, \ldots, (x_{\text{max}}/8) - 1\} \) and \( n \in \{0, \ldots, (y_{\text{max}}/8) - 1\} \). Note that \( x_{\text{max}} \) and \( y_{\text{max}} \) are the width and height of the image and they are multiples of 8. If \( b_{p_{tw}}(m, n, t) = 1 \), this indicates that it is a discontinuity plane, i.e., the first block of a segment, and is to be used for the cost evaluation. For the sake of convenience, the value of \( b_p \) is rearranged in such a way that all discontinuity planes are packed to the left hand side while zeroes are added to the remaining positions to indicate that they are not used, and finally they become \( b_{p_{tw}'} \). Therefore,

\[
\begin{align*}
    b_{p_{tw}'}(m, n, 1) &= 1 \\
    b_{p_{tw}'}(m, n, 2) &= 1 \\
    b_{p_{tw}'}(m, n, 3) &= 0.
\end{align*}
\]
The cost between two consecutive discontinuity segments has to be measured for optimization. The measurement is taken between two consecutive discontinuity segments; i.e., their $bp_{tw}(m, n, t) = 1$. The algorithm measures the costs by making a raster scanning of the block sequences in the time window. The measurement is ended when the first block reaches $bp_{tw}(m, n, t) = 0$. The index $k$ which is used to count the number of segments is increased by one when a new measurement is taken. The number of ones in $bp_{tw}$ is equal to the total number of discontinuity segments. The cost or the weight of a segment is formulated as

$$\text{Seg}(k) = \begin{cases} \text{Cost}(B_{tw}(m, n, t), B_{tw}(m, n, t - 1)); k = k + 1 \\ \infty \\ \text{if } bp_{tw}(m, n, t) = 1 \\ \text{otherwise} \end{cases} \ orall m, n,$$

where $B(m, n, t)$ is the block at the $m$th row, $n$th column, and temporal location $t$. Thus

$$\text{Total cost} = \sum_{n=0}^{k-1} \text{Seg}(n).$$

Let us define $\text{Cost}(a, b)$, which is a function of $a$ and $b$, where $a$ and $b$ are blocks with $8 \times 8$ pixels. The cost function can be any function which could give an indication for discrimination, identification or recognition, etc. For example, for the Mean Absolute Difference (MAD) between two Blocks ($B_1$ and $B_2$) of image pixels, a matching criterion can be a cost function, which can be written as

$$\text{Cost}(B_1, B_2) = \frac{1}{64} \sum_{i=0}^{7} \sum_{j=0}^{7} |I_{B_1}(i, j) - I_{B_2}(i, j)|,$$

where $B_1$ and $B_2$ are two consecutive blocks at a certain spatial location with the block size of $8 \times 8$ pixels, and $I_{B_1}(i, j)$ and $I_{B_2}(i, j)$ are pixel intensities of the blocks in a block sequence.

This cost can be used to rank the discontinuity of segments. The segment with the least cost is selected according to the following criterion:

$$\text{least cost segment} = \min\{\text{Seg}(0), \ldots, \text{Seg}(k - 1)\}.$$

Figure 5 describes the block diagram of the Scalable Adaptive Temporal Segmentation. Presumably the initial BPM and the time window for the input sequence are obtained by using the $N$-frames decimation. Then they are fed into the system. An Inter block measurement is needed for all discontinuity segments and the results are passed for cost evaluation. The required number of bits and the MAD between blocks are extremely useful information. The costs of segments in the time window are then sorted and the least among them is located and removed by resetting its flag at $bp$ to zero. Assume that the block sequence has been segmented into $s$ discontinuity segments. Each of these segments is represented by an $8 \times 8$ image block that holds the temporal averages within the given time segment for each spatial location. If the stopping criterion reaches, an optimal number of segments is achieved. Otherwise, the system recursively removes the next segment with the least cost.

After the optimization of the bit rate, the Intra frame and the decimated frames (Inter frames) are coded by the JPEG coding and the transform coding of the H.261 respectively. The BPM is then coded by the VLC. Shown below is a pseudo-code program for the realization of the algorithm.

**The Scalable and Adaptive Temporal Segmentation Algorithm.**

Input : a time window (tw) of $N$ frames of $M_{\text{max}} \times N_{\text{max}}$ blocks in resolutions,

a BPM (bp), and

the system specifications

Output : $(N - 1)$ decimated frames and an optimal BPM

1. To initialize $k = 0$.
2. while the system specifications NOT fulfilled
3. $bp_{tw}(m, n, t) = bp_{tw}(m, n, t)$ (converting bp map, see Fig. 4)
4. for $n = 0 \ldots (N_{\text{max}} - 1)$
5. for $m = 0 \ldots (M_{\text{max}} - 1)$, $t = 1$
6. while $bp_{tw}(m, n, t) \neq 0$ and $t < N$
7. $\text{Seg}(k) = \text{Cost}(B_{tw}(m, n, t), B_{tw}(m, n, t - 1))$
8. $k = k + 1$, $t = t + 1$
9. locating the least cost of discontinuity segment,
FIG. 7. The 24th frames of Miss America at different bit-rates.
where $I_{R_1}(i, j)$ and $I_{R_2}(i, j)$ are intensities of elements inside blocks $B_1$ and $B_2$, respectively.

Figure 6 shows the performance of the SATS algorithm for $N = 14$, for situations with increasing bit rates. The bit rate ranges from 42 K bits to 84.5 K bits. The SATS algorithm performs well, and for the objective evaluation, it gives a SNR of over 35 dB for all frames at all different bit rates. A careful examination of the reconstruction sequence, it also shows very good subjective qualities for all frames with such low bit rates. The objective qualities of the reconstructed sequence shown in Fig. 6 are considered to be very good for human perception although the signal-to-noise ratio (SNR) drops slightly in the latter part of the sequence due to its high spatial activity. For videophone sequences, most of the frames are highly correlated to the previous frame. Our algorithm can handle them efficiently in terms of performance and it provides high SNR and low bit rate in many cases. For the case of using 64 Kbit videophone coding, our proposed algorithm gives a SNR of 37 dB for the overall output with the 60.7 Kbit for the coding. The SNR of the 24th frame is relatively low. Therefore, we present the 24th frames of the images for different bit rates as shown in Fig. 7.

At lower bit rates, such as 41.9 and 48.4 Kbits, obvious aliasing occurs at the left eyebrow of the lady. Apart from that, the above frames do not have obvious visual differences.

The performances of the SATS algorithm with different sizes of the time window $N$ is shown in Fig. 8. Three $N$ values, 7, 10, and 14, were used to explore how the system responses. The bit rate is fixed to the range of 46 to 50 Kbits. The quality of the reconstruction sequences varies with different $N$. This is caused by the limited bit rate of a time window. All time windows with various $N$ values are shown in Fig. 9. The first 10 frames of the sequence are with low activity and consequently the corresponding SNRs (at $tw = 1$) are relatively high. Frames with higher activity, i.e., from frame 15 to frame 30, result in degradation of the SNR.

Let us return to a comparison of our approach with the Olstad’s algorithm. Our algorithm outperforms the Olstad’s adaptive temporal decimation algorithm, by giving higher compression ratio on the BPM and comparable objective perceptual quality of the reconstructed frames. Table 1 shows details of the comparison. Note also that our algorithm can tune the output bit rate for various bandwidth requirements and it is also useful for various video processing applications since the cost function can be used to optimize different image features.

### 6. EXPERIMENTAL RESULTS

A number of experiments have been carried out to test the performance of our algorithm. In particular, the Miss America sequence (CIF format with $352 \times 288$ bytes per frame and with 30 frames) was used. The motion compensation is based on a brute search with the mean absolute difference matching criterion. The motion compensation is only necessary for all discontinuity planes. The motion compensated prediction errors are finally coded with the DCT. The BPM is coded by an arithmetic coder [8, 9]. Results of our concern include the overall bit-rate of the sequence, apart from the JPEG compression of the Intra frame, since only one Intra frame is necessary for the entire time window. This could be omitted in our evaluation. The cost function used for all experiments is

$$
\text{Cost}(B_1, B_2) = \frac{1}{64} \sum_{i=0}^{7} \sum_{j=0}^{7} |I_{B_1}(i, j) - I_{B_2}(i, j)|
$$

| Table 1 Comparison between Olstad’s and the Proposed Algorithms ($N = 14$) |
|-----------------|-----------------|
| Olstad’s algorithm (K = 3) | Proposed algorithm |
| BPM (Kbits) | 39.312 Kbits | 7.192 Kbits |
| Decimated frames (Kbits) | 243.534 Kbits | 35.748 Kbits |
| SNR (average of 30 frames) | 44.5 dB | 38.23 dB |

### 7. CONCLUSIONS

A versatile temporal segmentation algorithm is proposed in this paper. The Scalable and Adaptive Temporal
Segmentation (SATS) algorithm converts the input video signals into a more efficient representation which contains a smaller number of discontinuity segments but the visual quality is lossless to HVS. The approach can be applied to various applications, such as video coding and videophone coding. The cost function is a multifunction which implements any features and criteria for optimization. Visually lossless quantization is also introduced in conjunction with the SATS algorithm. A distinctive advantage of the proposed algorithm is that the quality of the reconstructed output can maintain very high, especially for “head and shoulder” sequences at very low bit rate. A reinforcement of the quality is needed for high motion sequence but the reinforcement would not be as frequent as the MPEG and other general coding algorithms. This is a definite advantage for videophone applications.

It has also been found that the size of the time window \( N \) be an important factor to the output. Due to the limitation of the bit rate within the time window, high motions could affect the resulting quality of the picture. Results of

FIG. 9. Different time windows of the three \( N \) values, 7, 10, and 14.
our experiments show that when the time window, \( N \), is doubled, the bit rate is not equivalent to two successive \( N/2 \) time windows because the optimization is not linear and the cost function is a simple spatially dependent function. A spatiotemporal cost function is able to improve the overall performance.

The proposed algorithm has a proven quality and it outperforms the closest coding algorithm, Olstad’s algorithm, in terms of both the coding efficiency of the BPM and the decimated frames, when \( N \) is fixed. The visual quality of the output also remains high.

ACKNOWLEDGMENT

This research was supported by the British Council through Grant JRS 93/35.

REFERENCES