MERL – A MITSUBISHI ELECTRIC RESEARCH LABORATORY
http://www.merl.com

Drift Compensation for Reduced Spatial Resolution Transcoding

Peng Yin Anthony Vetro Bede Liu Huifang Sun

TR-2002-47 August 2002

Abstract

This paper discusses the problem of reduced-resolution transcoding of compressed video bitstreams. An analysis of drift errors is provided to identify the sources of quality degradation when transcoding to a lower spatial resolution. Two types of drift error are considered: a reference picture error, which has been identified in previous works, and error due to the non-commutative property of motion compensation and down-sampling, which is unique to this work. To overcome these sources of error, four novel architectures are presented. One architecture attempts to compensate for the reference picture error in the reduced resolution, while another architecture attempts to do the same in the original resolution. We present a third architecture that attempts to eliminate the second type of drift error and a final architecture that relies on an intra block refresh method to compensate all types of errors. In all these architectures, a variety of macroblock level conversions are required, such as motion vector mapping and texture down-sampling. These conversions are discussed in detail. Another important issue for the transcoder is rate control. This is especially important for the intra refresh architecture since it must find a balance between number of intra blocks used to compensate errors and the associated rate-distortion characteristics of the low-resolution signal. The complexity and quality of the architectures are compared. Based on the results, we find that the intra refresh architecture offers the best trade-off between quality and complexity, and is also the most flexible.

This work may not be copied or reproduced in whole or in part for any commercial purpose. Permission to copy in whole or in part without payment of fee is granted for nonprofit educational and research purposes provided that all such whole or partial copies include the following: a notice that such copying is by permission of Mitsubishi Electric Information Technology Center America; an acknowledgment of the authors and individual contributions to the work; and all applicable portions of the copyright notice. Copying, reproduction, or republishing for any other purpose shall require a license with payment of fee to Mitsubishi Electric Information Technology Center America. All rights reserved.

Copyright © Mitsubishi Electric Information Technology Center America, 2002
201 Broadway, Cambridge, Massachusetts 02139

Published in IEEE Transaction on Circuits and Systems for Video Technology, vol. 12, no. 11, pp. 1009-1020, November 2002.

Drift Compensation for Reduced Spatial Resolution Transcoding

Peng Yin, *Student Member, IEEE,* Anthony Vetro*, *Member, IEEE*, Bede Liu, *Fellow, IEEE,* and Huifang Sun, *Fellow, IEEE*

Contact Author Information: Dr. Anthony Vetro, Mitsubishi Electric Research Labs, 558 Central Ave, Murray Hill, NJ 07974. Tel: +1-908-363-0504 Fax: +1-908-363-0550, Email: avetro@merl.com.

Manuscript received October 2001; revised June 2002.
P. Yin and B. Liu are with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08540.
A. Vetro and H. Sun are with are with Mitsubishi Electric Research Laboratories, Murray Hill, NJ 07974.

Abstract

This paper discusses the problem of reduced-resolution transcoding of compressed video bitstreams. An analysis of drift errors is provided to identify the sources of quality degradation when transcoding to a lower spatial resolution. Two types of drift error are considered: a reference picture error, which has been identified in previous works, and error due to the non-commutative property of motion compensation and down-sampling, which is unique to this work. To overcome these sources of error, four novel architectures are presented. One architecture attempts to compensate for the reference picture error in the reduced resolution, while another architecture attempts to do the same in the original resolution. We present a third architecture that attempts to eliminate the second type of drift error and a final architecture that relies on an intra block refresh method to compensate all types of errors. In all these architectures, a variety of macroblock level conversions are required, such as motion vector mapping and texture down-sampling. These conversions are discussed in detail. Another important issue for the transcoder is rate control. This is especially important for the intra refresh architecture since it must find a balance between number of intra blocks used to compensate errors and the associated rate-distortion characteristics of the low-resolution signal. The complexity and quality of the architectures are compared. Based on the results, we find that the intra refresh architecture offers the best trade-off between quality and complexity, and is also the most flexible.

Keywords

Transcoding, MPEG-2 to MPEG-4, Reduced Resolution, Drift Compensation

I. INTRODUCTION

In the context of coding and transmission, there is an increasing need to perform many types of conversions to accommodate terminal constraints, network limitations or preferences of a user. On one side there exists a rich set of content that is rapidly growing by the day, and on the other, we have terminals with varying capabilities. Furthermore, these two entities may be connected through a variety of different network configurations. It should be clear that the transmission and playback of content in such a diverse and dynamic environment may require many types of different conversions.

In this paper, we focus our attention to the problem of reduced resolution transcoding. Specifically, we consider techniques and architectures that convert a compressed video bitstream that has been encoded at one spatial resolution to an output bitstream with half the spatial resolution. The methods presented in this paper aim to minimize the amount of drift in the reconstructed sequence. The transcoding methods themselves can be applied within the same syntax format or between different syntax formats. Since MPEG-4 is adopted as the solution for mobile multimedia communications and a large amount of MPEG-1/2 content is available, we often focus our discussion on MPEG-2 to MPEG-4 transcoding. This is particularly

important due to the limited display size of mobile terminals. However, details on the differences between syntax formats are not elaborated on.

Actually, transcoding has been an extensively studied topic in recent years. In earlier works, bit-rate reduction techniques were explored to meet available channel capacity [1], [2], [3]. These architectures were further refined in [4] to reduce overall complexity. Additionally, researchers have investigated conversions between constant bit-rate (CBR) streams and variable bit-rate (VBR) streams to facilitate more efficient transport of video [5]. To improve the performance, techniques to refine the motion vectors have also been proposed [6].

Several papers have addressed the problem of resolution conversion [7]-[10]. The primary focus of the work in [7] was on motion vector scaling techniques. In [8], the authors propose to use DCT-domain down-scaling and motion compensation for transcoding, while also considering motion vector scaling and coding mode decisions. With the proposed two-loop architecture, computational savings of 40% have been reported with a minimal loss in quality. In [9], [10], a comprehensive study on the transcoding to lower spatio-temporal resolutions and to different encoding formats has been provided based on the reuse of motion parameters. In this work, a full decoding and encoding loop were employed, but with the reuse of information, processing time was improved by a factor of 3.

In this paper, we provide a different approach to the problem of spatial resolution transcoding by providing a detailed analytical analysis of the overall architecture. In doing so, we give significant attention to the degradation of reconstructed macroblocks in reduced resolution pictures. Based on this analysis, several new architectures for compressed-domain transcoding are developed and analyzed. We also address the issue of constructing a reduced-resolution macroblock from macroblocks in the original bitstream that have different coding modes. We refer to this problem as the *mixed block problem* and present several way to overcome it.

The organization of the paper is as follows. We shall focus on the development of drift compensation architectures with transcoding performed in the compressed-domain. Since B-frames do not introduce any drift propagation, our discussion is focused on P-frames only. Before presenting the various architectures themselves, an analysis of the drift problem for reduced-resolution transcoding is given in section II. Here, we introduce a reference architecture and compare it to a simple open-loop transcoding architecture. The open-loop architecture is used to identify the sources of drift error with respect to the reference. In section III, the various types of macroblock-level conversions that are needed for reduced resolution transcoding are considered. Based on the results of our analysis, we present four transcoding architectures that attempt

to overcome the types of drift errors that have been identified. These architectures are presented in section IV. In section V, rate control issues are discussed. Finally, experimental results, including quality and complexity comparisons, are presented and discussed in section VI, and our concluding remarks are given in section VII.

II. Drift Error Analysis

In video coding, drift error refers to the continuous decrease in picture quality when a group of Motion Compensated (MC) inter-frame pictures are decoded. In general, it is due to an accumulation of error in the decoder's MC loop, however the cause of drift is due to many different factors. The problem of drift error has been studied for full-resolution transcoding [4], and for multi-layer coding [11], but no explicit analysis has been done for reduced-spatial-resolution transcoding. In this section, we will analyze drift errors by comparing a cascaded closed-loop architecture that is drift-free with an open-loop architecture. The open-loop architecture is the simplest and lowest complexity architecture that one can consider and it is characterized by severe drift errors due to many sources. In Section IV, we will consider new drift-compensating architectures based on the drift error analysis. These new architectures attempt to simplify the drift-free architecture, while improving the quality of the open-loop architecture.

With regard to notation, lowercase variables indicate spatial domain signals, while uppercase variables represent the equivalent signal in the DCT domain. The subscript on the variable indicates time, while a superscript equaling to one denotes an input signal and a superscript equaling to two denotes an output signal. *D* refers to down-sampling and *U* to up-sampling. M_f denotes full-resolution motion compensation and M_r denotes reduced-resolution motion compensation. x indicates a full-resolution image signal, y indicates a reduced-resolution image signal, e indicates a full-resolution residual signal and g indicates a reduced-resolution residual signal. mv_f denotes the set of full-resolution motion vectors and mv_r denotes the set of reduced-resolution motion vectors.

A. Reference Architecture

Figure 1 shows a cascaded closed-loop transcoding architecture, which is referred to as the *Reference*. This architecture is the most complex and costly, but it has no drift errors and provides a basis for drift error analysis and further simplification.

Since there is no motion compensated prediction for I-frames, $x_n^1 = e_n^1$. The reconstructed signal is then down-sampled,

$$y_n^1 = D(x_n^1). \tag{1}$$

Then, in the encoder, $g_n^2 = y_n^1$.

In the case of P-frames, the identity,

$$x_n^1 = e_n^1 + M_f(x_{n-1}^1), \tag{2}$$

yields the reconstructed full-resolution picture. As with the I-frame, this signal is then down-sampled via equation 1. Then, the reduced-resolution residual is generated according to,

$$g_n^2 = y_n^1 - M_r(y_{n-1}^2), \tag{3}$$

which is equivalently expressed as,

$$g_n^2 = D(e_n^1) + D(M_f(x_{n-1}^1)) - M_r(y_{n-1}^2). \tag{4}$$

The signal given by equation 4 represents the reference signal that is free of drift errors. Based on this equation, we can analyze drift errors.

B. Drift Error Analysis of Open-Loop Architecture

Figure 2 shows the open-loop architecture for a transcoder, referred as *OpenLoop*. The objectives and functions of certain blocks, such as the Mixed-Block Processor, Motion Vector (MV) Mapping and Downs-Sampling will be described later in Section III. At this time, we are mainly concerned with the analysis of drift errors caused by reduced-resolution transcoding. With this *OpenLoop* architecture, the reduced-resolution residual is given by,

$$g_n^2 = D(e_n^1). \tag{5}$$

Compared to equation 4, the drift error, d, can be expressed as,

$$\begin{aligned} d &= D(M_f(x_{n-1}^1)) - M_r(y_{n-1}^2) \qquad (6)\\ &= [D(M_f(x_{n-1}^1)) - M_r(y_{n-1}^1)] + [M_r(y_{n-1}^1) - M_r(y_{n-1}^2)] \\ &= [D(M_f(x_{n-1}^1)) - M_r(D(x_{n-1}^1))] + [M_r(y_{n-1}^1 - y_{n-1}^2)] \\ &= d_r + d_q \end{aligned}$$

where

$$d_q = M_r(y_{n-1}^1 - y_{n-1}^2) \tag{7}$$

and

$$d_r = D(M_f(x_{n-1}^1)) - M_r(D(x_{n-1}^1)). \tag{8}$$

In the above, the drift error has been decomposed into two categories. The first component, d_q, represents an error in the reference picture that is used for MC. This error may be caused by re-quantization, eliminating some non-zero DCT coefficients, or arithmetic error caused by integer truncation. This is a common drift error that has been observed in other transcoding works [4]. In this way, the pictures originally used as references by the transcoder will be different from their counterparts in the decoder, thus creating a mismatch between predictive and residual components. The second component, $d_\downarrow$, is due to the non-commutative property of motion compensation and down-sampling, which is unique to reduced-resolution transcoding. There are two main factors contributing to the impact of $d_\downarrow$: motion vector (MV) mapping and down-sampling. In mapping MV's from the original-resolution to a reduced-resolution, a truncation of the MV is experienced due to the limited coding precision. In down-sampling to a lower spatial resolution in the compressed domain, block constraints are often observed to avoid filters that overlap between blocks. Due to this effort to reduce complexity, the quality of the down-sampling process must be compromised and some errors are typically introduced. Regardless of the magnitude of these errors for a single frame, the combination of these two transformations generally creates a further mismatch between the predictive and residual components that will increase with every successively predicted picture.

To illustrate this mismatch between predictive and residual components due to the non-commutative property of motion compensation and down-sampling, we consider an example with 1-D signals and neglect any error due to requantization (or d_q). Let b denote the reconstructed block, a denote the reference block, and e denote the error (residual) block, all in the original-resolution. Furthermore, let h_v denote a full-resolution motion compensation filter and $h_{v/2}$ denote a reduced resolution motion compensation filter. Then, the reconstructed block in the original-resolution is given by,

$$b = h_v a + e. \tag{9}$$

If we apply a down-conversion process to both sides, we have,

$$D(b) = D(h_v a) + D(e). \tag{10}$$

The quality produced by the above expression would not be subject to the drift errors included in $d_\downarrow$. However, this is not the signal that is produced by the reduced-resolution transcoder. The actual reconstructed signal is given by,

$$\tilde{D}(b) = h_{v/2} D(a) + D(e). \tag{11}$$

Since $D(h_v a)$ does not usually equal $h_{v/2} D(a)$, there is a mismatch between the reduced-resolution predictive and residual components. To achieve the quality produced by equation 10, either or both of the

predictive and residual components would need to be modified to match each other. In the Reference architecture, this mismatch is eliminated with the second (encoder) loop that computes a new reduced-resolution residual. With this second-loop, the predictive and residual components are re-aligned. Our objective in the following section is to consider alternative ways to compensate for drift with reduced complexity. Before that, we will first explain several issues related to reduced-resolution transcoding.

III. Macroblock Conversions

In this section, we will discuss some issues related to reduced-resolution transcoding, which correspond to three functional units: Mixed Block Processing, MV Mapping and Texture Down-sampling. These blocks are presented in the *OpenLoop* architecture, which is shown in Figures 2, and will also be used in the new drift compensating architectures that will be presented in the next section.

A. Mixed Block Processing

In transcoding compressed video to a lower spatial resolution, a group of four macroblocks in the original video corresponds to one macroblock in the transcoded video. So, the purpose of the Mixed-Block Processor is to pre-process selected groups of macroblocks to ensure that the down-sampling process will not generate an output macroblock in which its sub-blocks have different coding modes, e.g., both inter- and intra sub-blocks within a single macroblock. Mixed coding modes within a macroblock are not supported by any known video coding standards.

The function of the Mixed-Block Processor includes producing motion vectors and corresponding residues for inter-frames. Because inter-frames contain both intra- and inter-macroblocks, we need first to decide the macroblock (MB) mode in the output bitstream from the MB modes of the four corresponding macroblocks in original bitstream. If the four input MB modes are intra, the output MB mode is intra. Similarly, if the four input MB modes are inter, the output MB mode will also be inter. If neither of these cases is true, we encounter a group of *mixed blocks*, i.e, a group of macroblocks that contain both intra- and inter-coded macroblocks. In this case, we need to decide on a consistent mode and modify the associated motion vectors and DCT coefficients accordingly so that a non-mixed block is produced in the output. We propose three methods to accomplish this mixed-block processing.

In the first method, *ZeroOut*, the MB modes of the mixed macroblocks are all modified to inter-mode. The MV's for the intra-macroblocks are reset to zero and so are corresponding DCT coefficients. In this way, the input macroblocks that have been converted are replicated with data from corresponding blocks in the reference frame.

The second method is called *IntraInter*. In this method, the MB modes are also modified to be inter-mode, but the motion vectors for the intra-macroblocks are predicted. The prediction is based on the data in neighboring blocks, which can include both texture and motion data. In our experiments, the following estimate of the new motion vector is used,

$$mv_{pred} = \sum w_i \cdot mv_{ni}, \qquad \sum w_i = 1, i = 1 \ldots \tag{12}$$

where mv_{pred} is the predicted motion vector for the intra-macroblock that needs to be converted, mv_{ni} is the neighboring inter-macroblocks, w_i is the corresponding weighting factor based on the DCT energy of the inter-macroblocks. As an alternative, we can simply set the motion vector to be zero, depending on which produces less residual. In an encoder, the mean absolute difference of the residual blocks are typically used for mode decision. The same principles can be applied here. Based on the predicted motion vector, a new residual for the modified macroblock is calculated.

In the third method, *InterIntra*, the MB modes are all modified to intra-mode. In this case, there is no motion information associated with the reduced-resolution macroblock, therefore all associated motion vector data is reset to zero. This is necessary to perform in the transcoder because the motion vectors of neighboring blocks are predicted from the motion of this block. To ensure proper reconstruction in the decoder, the motion vector for the particular group of macroblocks must be reset to zero in the transcoder. The intra-DCT coefficients are generated to replace the inter-DCT coefficients.

It should be noted that to implement the second and third methods, we need a decoding loop to reconstruct a full-resolution picture. The reconstructed data is used as a reference to convert the DCT coefficients from intra-to-inter, or inter-to-intra. For a sequence of frames with a small amount of motion and a low-level of detail, the low complexity strategy of *ZeroOut* can be used. Otherwise, either *IntraInter* or *InterIntra* should be used. The performance of *InterIntra* is a little better than *IntraInter*, because *InterIntra* can stop drift propagation by transforming inter blocks to intra blocks.

B. Motion Vector Mapping

When down-sampling four macroblocks to one macroblock, the associated motion vectors have to be mapped. Several methods suitable for frame-based motion vector mapping have been described in past work [7], [12]. To map from four frame-based motion vectors, i.e., one for each macroblock in a group, to one motion vector for the newly formed macroblock, a weighted average or median filters can be applied. This is referred to as a 4:1 mapping.

However, with certain compression standards, such as MPEG-4 and H.263, there is support in the

syntax for advanced prediction modes that allow one motion vector per 8×8 block, i.e., sub-block motion. In this case, each motion vector is mapped from a 16×16 macroblock in the original resolution to an 8×8 block in the reduced resolution macroblock with appropriate scaling by 2. This is referred as a 1:1 mapping.

1:1 mapping usually performs better than 4:1 mapping if no motion compensation is performed [12]. But, it is inefficient to always use the 1:1 mapping, because more bits are used to code four motion vectors. We propose a mixed mapping method based on the variance of the four input motion vectors. If the variance of these motion vectors is greater than a threshold, then use the 1:1 mapping method presented in [12], otherwise use 4:1 mapping.

Because MPEG-2 supports interlaced video, we also need to consider field-based MV mapping. We first convert the field-based MV's to a frame-based MV by using the top-field MV as the default [13]. However, if the bottom field MV is used to predict the top field, the top-field and bottom-field motion vectors are averaged. Then, we divide the horizontal MV by 2 (the vertical MV is already halved by using the top-field MV). It should be pointed out that there is a relation between the methods used for texture down-sampling (discussed below) and MV mapping. This is especially true when converting from an interlace source to a progressive output. We have tried various combinations of down-conversion filters and MV mappings, and have verified experimentally that this particular MV mapping method works slightly better than if the top-field MV is always used.

C. Texture Down-Sampling

For the purpose of down-sampling texture data, this paper mainly relies on the filtering concepts that we have developed a few years ago for memory saving decoders. A review of this work can be found in [14] and references therein. For the transcoder, there are some new aspects that must be considered. However, since the derivation of new filters is beyond the scope of this paper, we only provide a brief explanation of this process and expressions for particular filters used in our experiments.

In our earlier work, the concept of frequency synthesis has been proposed to transform an input DCT macroblock consisting of four 8×8 DCT blocks into an single 8×8 DCT block. The computations are actually performed on the rows and columns of the macroblock using separable 1D filters. Let $\underline{A}$ and $\underline{B}$ denoted the input vectors of size N. Then, the output data, $\underline{E}$, also of size N, is computed according to,

$$\underline{E} = f_1 \cdot \underline{A} + f_2 \cdot \underline{B} \tag{13}$$

where

$$\begin{aligned} f_1(k,p) &= \sum_{i=0}^{N-1} \psi_p^N(i) \cdot \psi_k^{2N}(i) \\ f_2(k,p) &= \sum_{i=0}^{N-1} \psi_p^N(i) \cdot \psi_k^{2N}(i+N) \end{aligned} \tag{14}$$

and

$$\psi_k^N(i) = \sqrt{\frac{2}{N}}\alpha(k)\cos\left(\frac{2i+1}{2N}k\pi\right) \tag{15}$$

and $\alpha(k) = 1/\sqrt{2}$ for $k = 0$, and 1 for $k \neq 0$. The above down-conversion filters can be applied in both the horizontal and vertical directions, and to both frame-DCT and field-DCT blocks. In our experiments, we aim to produce a progressive (MPEG-4) output. Therefore, if the input is a frame-DCT block, we apply the above filters to both fields. On the other hand, if the input is a field-DCT block, a different set of filters are used to simultaneously perform a field-to-frame conversion and down-conversion in the DCT domain. These filters are given by,

$$\begin{aligned} f_1(k,p) &= \sum_{i=0}^{N-1} \psi_p^N(i) \cdot \psi_k^{2N}(2i) \\ f_2(k,p) &= \sum_{i=0}^{N-1} \psi_p^N(i) \cdot \psi_k^{2N}(2i+N/2) \end{aligned} \tag{16}$$

We would also like to point out that all of the above DCT filters can easily be transformed to spatial-domain filters, which are mathematically equivalent [14]. In fact, such filters are used in our reference architecture to obtain a pure comparison of the architecture and avoid any differences due to down-sampling.

IV. Drift Compensation Architectures

In Section II, the drift errors related to reduced-resolution transcoding were analyzed by comparing the signals produced by the *Reference* and *OpenLoop* architectures. The *OpenLoop* architecture is the simplest, but the drift error can be very severe. On the other hand, the *Reference* architecture is completely drift free, but the complexity is high. Therefore, it is desirable to approximate the quality, while achieving significant complexity reduction. In this section, we will present four new architectures that provide different means of drift error compensation. The first two architectures can be viewed as extensions to the full-resolution transcoding architectures presented in [4]. The signal given by equation 4 represents the reference signal that the architectures described here approximate. Since I-frames do not cause drift and the processing of I-frames is simple and mature [7], we will focus our discussion on P-frames only.

We note that several methods have been proposed to perform motion compensation in the frequency domain, e.g., [4]. In this way, some DCT/IDCT computation can be avoided, but they involve matrices multiplications that may be computationally complex. In the architectures presented in this paper, we adopt spatial-domain motion compensation for simplicity. However, corresponding frequency-domain approaches can easily be applied.

A. Drift Compensation in Reduced Resolution

Figure 3 shows the first closed-loop architecture that performs drift compensation in the reduced resolution. This architecture will be referred to as *DriftLow*. The reduced resolution residual signal in Figure 3 is expressed as,

$$g_n^2 = D(e_n^1) + M_r(y_{n-1}^1 - y_{n-1}^2), \tag{17}$$

which can be generated from equation 4, with the approximation,

$$D(M_f(x_{n-1}^1)) \approx M_r(D(x_{n-1}^1)) = M_r(y_{n-1}^1). \tag{18}$$

The drift compensation is performed on a macroblock-by-macroblock basis using the set of reduced-resolution motion vectors. This architecture attempts to eliminate drift errors caused by d_q. It should be noted that the Mixed Block Processor would be implemented as an MC loop with frame memory if the *IntraInter* or *InterIntra* configurations discussed in section III-A are chosen. Only the *ZeroOut* configuration would avoid this additional loop.

B. Drift Compensation in Original Resolution

Figure 4 presents the second closed-loop architecture that performs drift compensation in the original resolution. This architecture will be referred to as *DriftFull*. The reduced resolution residual signal in Figure 4 is expressed as,

$$g_n^2 = D(e_n^1 + M_f(x_{n-1}^1 - x_{n-1}^2)), \tag{19}$$

which can be generated from equation 4, with the approximation,

$$M_r(y_{n-1}^2) \approx D(M_f(U(y_{n-1}^2))) = D(M_f(x_{n-1}^2)). \tag{20}$$

This architecture requires up-sampling in the DCT domain, which is briefly discussed below. In this architecture, the drift compensation is performed on a macroblock-by-macroblock basis using the set of original-resolution motion vectors. This architecture attempts to eliminate drift errors caused by d_q, as

well as drift errors caused by d_r. As with the *DriftLow* architecture, an additional MC loop with frame memory may be substituted for the Mixed Block Processor depending on the particular configuration that has been chosen, i.e., *ZeroOut*, *IntraInter*, or *InterIntra*.

The up-sampling operation required by this transcoding architecture is unique from past up-sampling operations described in [14] in that the sub-division back to four 8×8 blocks has not been considered. In our previous work, the up-sampling filters were derived based on a least-squares approach that minimized the difference between the original and reconstructed data in the spatial domain. Specifically, the optimal up-sampling filter, X^+, is given by, $X^+ = X^T(XX^T)^{-1}$, where $X = [f_1,\ f_2]$ denotes a concatenated down-conversion filter matrix based on the filters given in equation 14. In this way, an $N \times N$ DCT block, $\underline{E}$, can be up-sampled to a $2N \times 2N$ DCT block, $\underline{C}$ according to,

$$\underline{C} = X^+ \cdot \underline{E} \tag{21}$$

However, with the above, the sub-block structure of a macroblock is not recovered, i.e., a single 16×16 DCT block will be obtained, which is clearly not suitable for the difference of four 8×8 DCT blocks that need to be computed after the up-sampling operation. Without proof, we claim that two reconstructed $N \times N$ DCT blocks, $\underline{\hat{A}}$ and $\underline{\hat{B}}$, can be obtained through the following expressions,

$$\begin{aligned} \underline{\hat{A}} &= y_1 \cdot \underline{E} \\ \underline{\hat{B}} &= y_2 \cdot \underline{E} \end{aligned} \tag{22}$$

where

$$\begin{aligned} y_1(k,p) &= X^+(k,p) \cdot \sum_{i=0}^{N-1} \psi_p^{2N}(i) \cdot \psi_k^N(i) \\ y_2(k,p) &= X^+(k,p) \sum_{i=0}^{N-1} \psi_p^{2N}(i+N) \cdot \psi_k^N(i) \end{aligned} \tag{23}$$

Applying the expression given in equation 22 to rows and columns of the down-sampled DCT block allows us to reconstruct four 8×8 DCT blocks. These blocks can be then be subtracted from the DCT blocks prior to encoding to yield the difference signal used for compensating the drift in the original resolution.

C. Partial_Encode Architecture

Figure 5 shows the third closed-loop architecture that performs drift compensation in the reduced resolution. We shall refer to this architecture as *PartiaL Encode*. The reduced resolution residual signal

in Figure 5 is expressed as,

$$g_n^2 = D(e_n^1) + D(M_f(x_{n-1}^1)) - M_r(D(x_{n-1}^1)), \tag{24}$$

which can be generated from equation 4, with the approximation,

$$y_{n-1}^2 \approx y_{n-1}^1 = D(x_{n-1}^1). \tag{25}$$

The *Partial_Encode* architecture attempts to eliminate drift errors caused by d_r. It should be noted that since full-resolution decoding is performed with this architecture, there is no mixed block problem.

D. Intra_Refresh Architecture

As presented in Section III-A, the *InterIntra* and *IntraInter* configurations of the Mixed Block Processor require an additional MC loop to reconstruct a full-resolution picture. In this sub-section, we exploit this structure by considering methods to control or minimize the drift. We will present a new method using intra-refresh to stop the drift, which can be applied in both open-loop and close-loop architectures.

In the Mixed-Block Processor, *InterIntra* provides a way to convert an inter-coded block to intra-coded block. Because intra-coded blocks are not subject to drift, this kind of conversion stops the drift propagation, for both d_q and d_r. We refer to this technique of using *InterIntra* for drift compensation as *Intra_Refresh*. This method has successfully been applied to error-resilience coding schemes [15], and we find the principle is also very useful for reducing drift in a transcoder. The *Intra_Refresh* architecture is illustrated in Figure 6. As seen from the figure, this architecture only has one motion compensated loop, which is used to reconstruct a full-resolution picture to convert the DCT coefficients from inter-to-intra. As we will show, this architecture is very flexible in that it can adapt itself based on the input signal characteristics and transcoding condition. Besides providing an effective, yet simple means to combat both types of drift errors, d_r and d_q, it is also capable of correcting any errors that may be caused by incorrect MV mapping. These properties make this architecture very appealing. At this time, however, we focus our discussion on the ability to correct drift errors only and do not consider any other sources of error.

In the following, we consider two steps for the intra-refresh process. First, the amount of drift in the compressed bitstream is estimated. The drift can be estimated based on data pertaining to the amount of motion vector truncation, residue energy, motion activity or re-quantization error. Second, the estimated value of drift is translated to an intra-refresh rate, β, that is used as input to the Mixed-Block Processor. β is the percentage of intra-coded macroblocks in one frame. One point to keep in mind about the

intra-refresh technique is that more bits are usually required for coding intra-blocks. For this reason, the intra-refresh process and the rate control must be considered jointly. The specific issues involved will be discussed in more detail in Section V.

V. Rate Control

In this section, we first review the rate control framework that is common to all the drift compensation architectures presented in this paper. Then, special considerations for the rate control that must be accounted for in the *Intra_Refresh* architecture are discussed. These special considerations arise from the fact that an increase in the number of intra-coded blocks should be considered in determining the quantizer used to code the frame.

A. Common Framework

We begin our discussion with a brief overview of the quadratic texture model that we use for rate control, which was proposed by Chiang and Zhang in [16]. Let R represent the texture bits spent for a frame, Q denote the quantization parameter, (X_1,X_2) the first and second-order model parameters and S the encoding complexity. The relation between R and Q is given by:

$$R = S \cdot \left(\frac{X_1}{Q} + \frac{X_2}{Q^2} \right). \tag{26}$$

In an encoder, the complexity measure, S, is typically given by the mean absolute difference of residual blocks. However, for compressed-domain transcoding architectures, this measure must be computed from the DCT coefficients. As a result, we adopt a DCT-based complexity measure, $\tilde{S}$, which was presented in [17] and is given by,

$$\tilde{S} = \frac{1}{M_c} \sum_{m \epsilon \mathcal{M}} \sum_{i=1}^{63} \rho(i) \cdot |B_m(i)|^2 \tag{27}$$

where $B_m(i)$ are the AC coefficients of a block, m is a macroblock index in the set $\mathcal{M}$ of inter-coded blocks, M_c is the number of blocks in that set and $\rho(i)$ is a frequency dependent weighting, e.g., a quantizer matrix.

Given the above model and complexity measures, the rate control works in two main stages: a pre-encoding and post-encoding stage. In the pre-encoding stage, the target estimate for the frame is obtained based on available bit-rate and buffer fullness, and the complexity is also computed. Then, the value of Q is determined based on these values and the current model parameters. After encoding, the post-encoding stage is responsible for calibrating the model parameters based in the actual bits spent and corresponding Q. This can be done by linear regression using the results of the past n frames.

B. Intra Refresh Considerations

Theoretically, the effect of intra-refresh can be characterized by the operational rate-distortion (R-D) function $D(\beta, R)$: i.e., the average distortion D is expressed as a function of the average bit-rate, R, and intra-refresh rate, β. To illustrate the behavior of this function, consider the Foreman sequence at CIF resolution, encoded at 2Mbps with $N = 30$ and $M = 1$. This bitstream is transcoded with a number of different fixed quantizer scales and various values of β. The R-D curves for each β are shown in Figure 7. The intra-refresh scheme that we used to generate these results is similar to that of H.263 [15]. In this scheme, each macroblock is assigned a counter that is increased if the macroblock is encoded in inter-frame mode. If the counter reaches a threshold, $T = 1/\beta$, which denotes the update interval, the macroblock is encoded in intra-mode and the counter is reset to zero. By assigning a different initial offsets to each macroblock, the updates of individual macroblocks can be spread over time.

Figure 7 shows that overall quality is decreased at low bit-rates when the intra-refresh rate is high. The reason is that too many bits are consumed by the intra-coded blocks without a sufficient increase in quality. The opposite is observed at higher bit-rates. With a larger amount of bits that can be spent per frame, the overall quality is increased with more intra-coded blocks. Since the goal of the intra-refresh techniques is to minimize the effect of drift, we should point out that at lower bit-rates, d_q is likely to dominate the overall error, while at higher bit-rates, the impact of d_q is significantly less and d_r is likely to be more dominant.

In the *Intra_Refresh* architecture, it is important that the intra-refresh process be adaptive to account for the above characteristics, and also that the outcome of the process, i.e., the number of intra blocks to be coded in a frame, be accounted for in certain aspects of the rate control. Specifically, the quantizer selection and model parameter calculation. In the following, we propose an Adaptive Intra-Refresh (AIR) technique which is simple, but effective. In general, the objective of this technique is to dynamically determine the blocks to be converted from inter-to-intra. The particular scheme that we describe is adaptive according to available bit-rate and block attributes. After describing the technique itself, we explain how the outcome is accounted for by the rate control.

As analyzed in Section II, drift errors come from two mismatches, d_q and d_r. Through observation, we find that a large drift error always correlates to inter-coded blocks with large residual energy or motion activity. Consequently, AIR decides that a group of (four) macroblocks need to be intra-coded if the sum of residual energy in this group of macroblocks is larger than a threshold, T_r, or if the sum of motion vector variance in this group of macroblocks is larger than a threshold, T_m. Initial values for the

thresholds are determined experimentally through a simple linear relationship with the distortion (MSE). Specifically, the relations are given by, $MSE = \alpha T_r$, and $MSE = \beta T_m$, where the parameters α and β are fitted based on several training sequences[1]. After each frame is encoded, the thresholds are dynamically adjusted according to the difference between the target bit rate and actual bit rate. If the difference is positive, it implies that the target quality is higher than the bits that have actually been spent and we set the thresholds lower. On the other hand, if the difference is negative, the thresholds are set higher. Since this AIR decision may ignore the inter-coded boundary blocks of a moving object, we further expand the intra-refresh boundary to the left or right to cover an object boundary. It should be noted that this procedure is not optimal, but as experimental results will show, it works quite well. The main purpose of this scheme is to provide some means of adaptive intra block conversion to illustrate the concepts and strengths of the Intra Refresh architecture. An optimal scheme that considers rate-distortion trade-offs would be a topic for further study.

With the intra-refresh procedure, the total number of intra blocks may be high and must be accounted for in the rate control. For the quantizer selection, a single quantization parameter is selected for the frame and is applied to both intra and inter-coded blocks. With this scheme, we consider a hybrid complexity measure that accounts for both inter and intra DCT coefficients. In other words, Equation 27 is extended to include normalized intra DCT coefficients as well. Specifically,

$$\tilde{S} = \frac{1}{M}\left(\sum_{k \epsilon \mathcal{K}} \sum_{i=1}^{63} \rho_1(i) \cdot |B_k(i)|^2 + \sum_{l \epsilon \mathcal{L}} \sum_{i=1}^{63} \rho_2(i) \cdot |B_l(i)|^2\right), \tag{28}$$

where $B_l(i)$ are the AC coefficients of an intra-coded block, l is a macroblock index in the set $\mathcal{L}$ of intra-coded blocks, M is the total number of non-skipped blocks in a frame, and $\rho_1(i)$ and $\rho_2(i)$ are a frequency dependent weights for inter and intra-coded blocks, respectively. This hybrid complexity measure is used to calculate the updated model parameters after coding. With these small modifications to the rate control, we have found that a better fit between the rate-quantizer model and the actual data is achieved.

VI. Experimental Results

In this section, we present a comparative study of the quality and complexity for the architectures under consideration. We simulated all the proposed architectures in software by transcoding MPEG-1 and MPEG-2 bitstreams to quarter-resolution MPEG-4 bitstreams. The machine we used to run in the simulations is an Ultra 60 workstation with processor speed 360 MHz, model UltraSPARC-II, 4-MB L2

[1] The training sequences were chosen outside of the sequences used for the experimental results presented in section VI.

Cache and 512 MB memory, and the source code used in these experiments has not been optimized for speed.

Independent of these above platform specifications, Table I shows the main blocks contributing to the complexity of each implementation. The table also provides a relative indication of the expected amount of drift error and the primary source of that error. With regard to the complexity of the architectures that employ inter-to-intra or intra-to-inter conversions, it should be noted that certain blocks involved in the conversion process are not invoked for every macroblock. For instance, the DCT block in the *Intra_Refresh* architecture is only used for blocks that require a conversion, which may be only a fraction of the total number of blocks in a given frame, e.g., 10%. This is especially important for software implementations that rely mainly on average processing times. Also, in our simulations, the *InterIntra* configuration for the Mixed-Block Processor has been adopted for the *DriftLow* and *DriftFull* architectures since this leads to the best quality. The numbers in Table I take this configuration into account, and will be lower if the *ZeroOut* configuration is used.

The tests are conducted with various input sequences at different resolutions, formats and encoding pattern (GOP structure). All the sequences are coded at 30fps. In the *Intra-Refresh* simulations, we set initial thresholds: $T_r = 100000$ and $T_m = 6000$. During transcoding, both thresholds are adaptively adjusted as described in Section V. For comparison, we also simulate an *Optimal* architecture, which consists of an MPEG-2 decoder, followed by spatial-domain sub-sampling, and a full MPEG-4 encoder with motion re-estimation. This surely has the best quality, but has at least 37 times more computational cost than the *Reference* [7].

We first conduct some experiments using input bitstreams that have been encoded with periodic I-frames. The primary purpose of this experiment is to analyze the quality and complexity of all the presented architectures under common conditions that are somewhat tolerant to the accumulation of drift, but not completely forgiving. The original sequence is CIF resolution (352×288) and coded as an MPEG-1 bitstream with $N = 15$ and $M = 3$.

In the first experiment, we consider the *Akiyo* sequence, which has low motion and a low-level of detail. The source bit-rate is 512Kbps and the target rates for the transcoder are 32, 64 and 96Kbps. The simulation shows that even the simplest open-loop architecture *ZeroOut* can achieve reasonably good quality and higher quality can be reached by other architectures with higher complexity. Table II shows processing time in seconds, as well as the mean and variance of the frame-based PSNR values (in dB) for all the architectures. We can see from the table that *InterIntra*, *IntraInter*, *IntraRefresh*, *PartiaL Encode*

and *DriftLow* have comparable quality with the *Reference*. Also, for these architectures, we have recorded speed-up factors of 3.59, 3.59, 3.4, 1.68 and 1.36, respectively, in comparison to the *Reference*.

For the second experiment, the *Foreman* sequence, which has a medium amount of motion and a medium level of detail, is considered. The original bit rate is 2Mbps. The target rates for the transcoder are 128, 384 and 512Kbps. Through simulation, we observed that artifacts can be found in *Zeroout*, *IntraInter*, *InterIntra* and *DriftLow*. We believe these artifacts are the result of d_r becoming more dominant in d with increased motion. However, such artifacts were not observed in *Intra_Refresh* and *Partial_Encode*. For the range of bit-rates tested, these two architectures were able to achieve similar quality to the *Reference*, while decreasing the time by a factor of 1.7 and 1.4, respectively. We note that the speed-up for *Intra_Refresh* has decreased in this simulation compared to the simulation with *Akiyo*. This is due to the increased need in Foreman for inter-to-intra conversion, which mainly requires an added DCT for each converted block. Summarized results for all the other architectures are provided in Table III.

To truly test the drift compensation capabilities of the *Intra_Refresh* and *Partial_Encode* architecture, we test two sequences that have been coded with $N = 100$ and $M = 1$. The first sequence is *Foreman* with CIF resolution and coded as MPEG-1 bitstream at a bit rate of 2Mbps. The target rates for the transcoder are 1Mbps, 512kbps and 256kbps. The second sequence is *Football*, which has fast motion and a high-level of detail. The original sequence is a CCIR601 format (720×480, interlace) and coded as an MPEG-2 bitstream at a rate of 6Mbps. The target rates for the transcoder are 3Mbps, 1.5Mbps and 750kbps. Figures 8 and 9 provide frame-based PSNR plots of the transcoded results using select architectures. From these plots, we can see that *Intra_Refresh* compensates for drift quite well in both sequences, even with such a long series of successive predictions. However, the results are not so consistent for *Partial_Encode*, i.e., a strong decay over time is observed for *Foreman* and almost no decay for *Football*. There are two explanations for this occurrence. The first reason is that for *Foreman*, d_q plays a more critical role as N increased, while for *Football*, d_r is always more influential than d_q due to the high motion. The second reason is due to the large number of intra blocks in P-frames that were used to encode *Football*, which also relates to the high motion complexity in the sequence. This result indicates that drift propagates less in such sequences.

¿From the experiments overall, we notice that *DriftFull* with *InterIntra* is even more complex than the *Reference*. Therefore, this architecture is not recommended for any application. For simple sequences with low motion, low-level of detail, low bit rate, short GOP and small frame size, *ZeroOut* can be used to get reasonably good quality. If higher quality is required, other architectures, such as *InterIntra*,

IntraInter, *IntraRefresh*, *Partial_Encode* and *DriftLow* should be considered. For sequences with medium to fast motion, artifacts can be found in *Zeroout*, *IntraInter*, *InterIntra* and *DriftLow*. Compared to the *Reference*, only *Intra_Refresh* can achieve similar quality with less complexity. *Partial_Encode* can only be used in sequences with a short GOP, or sequences with a long GOP but with a high amount of intra-coded blocks.

Through the simulations, we observe that the *Intra_Refresh* architecture can achieve quality comparable to the *Reference*, especially for sequences with larger motion, long GOP and large frame size. As the scene complexity increases, we have observed that the speed-up over the *Reference* decreases, i.e., by a factor of 3.4 for *Akiyo*, 1.7 for *Foreman*, and 1.44 for *Football*. These results demonstrate the flexibility of the *Intra_Refresh* architecture and the balance that it provides in terms of quality and complexity.

VII. Summary & Future work

This paper presented several architectures and methods for reduced-resolution transcoding. Through an analysis of the drift error for this type of transcoding, we have identified two distinct sources of drift: one type due to errors in the reference picture, which have been studied in other works and arise mainly from re-quantization errors, and a second type that is due to the non-commutative property of motion-compensation and down-sampling. With these sources of error in mind, four drift compensating architectures that attempt to approximate the quality of the reference and reduce its complexity have been proposed. The first architecture, *DriftLow*, attempted to compensate for the reference picture error in the reduced resolution, while the second architecture, *DriftFull*, attempted to do the same in the original resolution. The third architecture, *Partial_Encode* targeted the second type of drift error only and the final architecture, *Intra_Refresh*, relied on a novel intra-block refresh method to partially compensate for all types of errors.

¿From the experiments we have conducted, we believe that the *Intra_Refresh* architecture offers the best trade-off between quality and complexity. In most cases, the quality provided by this transcoding method is very similar to that of the reference. It is also a flexible and adaptable architecture that can easily be scaled in terms of its complexity and quality to meet any number of system needs.

With regards to future work, a better rate control model to combine the intra-refresh process with the quantizer selection requires more investigation. Another interesting topic is to combine temporal resolution reduction with spatial resolution reduction for video transcoding. Finally, we think there are some interesting studies to be done regarding the relation between MV mapping and texture down-sampling, especially within the context of interlace-to-progressive transcoding.

REFERENCES

[1] H. Sun, W. Kwok, and J. Zdepski, "Architectures for MPEG compressed bitstream scaling," *IEEE Trans. Circuits Syst. Video Technol.*, April 1996.

[2] Y. Nakajima, H. Hori, and T. Kanoh, "Rate conversion of MPEG coded video by re-quantization process," *Proc. IEEE Int'l Conf. Image Processing*, vol. 3, pp. 408-411, Washington, DC, Oct. 1995.

[3] N. Yeadon, F. Garcia, D. Hutchison and D. Shepherd, "Continuous Media Filters for Heterogeneous Internetworking," *Proc. SPIE Multimedia Computing & Networking*, 1996.

[4] P. Assuncao and M. Ghanbari, "A frequency-domain video transcoder for dynamic bit-rate reduction of MPEG-2 bit streams," *IEEE Trans. Circuits Syst. Video Technol.*, vol.8, Dec. 1998.

[5] M. Yong, Q.F. Zhu, and V. Eyuboglu, "VBR transport of CBR-encoded video over ATM networks," *Proc. Sixth Int'l Packet Video Workshop*, Sept. 1994.

[6] J. Youn, M. T. Sun, and C. W. Lin, "Motion vector refinement for high performance transcoding," *IEEE Trans. Multimedia*, vol. 1, no. 1, pp. 30-40, Mar. 1999.

[7] B. Shen, I.K. Sethi and V. Bhaskaran, "Adaptive Motion Vector Resampling for Compressed Video Down-scaling," *Proc. IEEE Int'l Conf. Image Processing*, Santa Barbara, CA, Oct. 1997.

[8] W. Zhu, K. H. Yang and M. J. Beacken, "CIF-to-QCIF Video Bitstream Donw-Conversion in the DCT Domain," *Bell Labs Technical Journal*, 3(3), July-September, 1998

[9] T. Shanableh and M. Ghanbari, "Heterogeous Video Transcoding to Lower Spatio-Temporal Resolutions and Different Encoding Formats," *IEEE Trans. Multimedia*, vol.2, no.2, June 2000.

[10] T. Shanableh and M. Ghanbari, "Transcoding Architectures for DCT-domain Heterogeneous Video Transcoding," *Proc. IEEE Int'l Conf. Image Processing*, Thessaloniki, Greece, Oct. 2001.

[11] R. Mokry and D. Anastassiou, "Minimal Error Drift in Frequency Scalability for Motion-Compensated DCT coding," *IEEE Trans. Circuits Syst. Video Technol.*, Vol.4, Aug. 1994.

[12] P. Yin, M. Wu and B. Liu, "Video Transcoding By Reducing Spatial Resolution," *Proc. IEEE Int'l Conf. Image Processing*, Vancouver, BC, Oct. 2000

[13] S. J. Wee, J. G. Apostolopoulos and N. Feamster, "Field-To-Frame Transcoding with Spatial and Temporal Downsampling," *Proc. IEEE Int'l Conf. Image Processing*, Kobe, Japan, Oct. 1999

[14] A. Vetro, H. Sun, P. DaGraca and T. Poon, "Minimum drift architectures for three-layer scalable DTV decodig," *IEEE Trans. Consumer Electronics*, vol.44, no.3, Aug. 1998.

[15] K. Stuhlmuller, N. Farber, M. Link and B. Girod, "Analysis of Video Transmission over Lossy Channels," *J. Select Areas of Commun.*, June 2000.

[16] T. Chiang and Y-Q. Zhang, "A new rate control scheme using quadratic rate-distortion modeling," *IEEE Trans. Circuits Syst. Video Technol.*, Feb 1997.

[17] A. Vetro, H. Sun and Y. Wang, "Object-based transcoding for adaptable video content delivery," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 3, March 2001.

LIST OF TABLES

TABLE I

COMPARISON OF TRANSCODING ARCHITECTURES

(*:INTERINTRA IS USED FOR MIXED-BLOCK PROCESSOR)

Transcoding Architecture	DCT/IDCT	MC Loop	Frame Buffer	Upsample	Drift Source	Drift Amount
Reference	4	2	2	No	No	Low
OpenLoop	0	0	0	No	$d_r + d_q$	High
DriftLow*	4	2	2	No	d_r	Medium
DriftFull*	4	2	2	Yes	d_r	Medium
Partial_Encode	2	2	2	No	d_q	Low
Intra_Refresh	2	1	1	No	partial	Low

TABLE II

EXPERIMENTAL RESULTS FOR AKIYO

	time	32 Kbps		64 Kbps		96 Kbps	
		mean	var	mean	var	mean	var
Optimal	/	32.13	0.23	34.83	0.67	36.12	0.82
Reference	17.6	31.77	0.57	34.78	0.55	36.26	0.96
Intra_Refresh	5.2	31.30	2.84	34.31	1.87	35.34	2.28
Partial_Encode	10.5	31.83	2.25	34.40	1.79	35.66	1.92
DriftLow	12.9	31.12	0.99	34.02	0.97	35.08	1.86
DriftFull	21.7	31.52	0.69	34.73	0.44	36.09	0.88
IntraInter	4.9	31.46	3.20	34.16	2.89	35.06	2.81
InterIntra	4.9	31.46	3.46	34.22	2.51	35.11	2.71
ZeroOut	3.6	31.51	3.33	33.91	3.27	34.81	3.62

TABLE III

EXPERIMENTAL RESULTS FOR FOREMAN

	time	128 Kbps		384 Kbps		512 Kbps	
		mean	var	mean	var	mean	var
Optimal	/	30.91	0.59	32.17	1.37	32.36	1.63
Reference	20.8	29.76	1.56	32.02	1.09	32.32	1.29
Intra_Refresh	12.2	29.28	2.72	31.33	3.43	31.69	3.11
Partial_Encode	14.7	28.25	2.69	30.90	1.51	31.54	1.64
DriftLow	16.2	28.88	2.40	30.13	3.71	30.30	4.08
DriftFull	26.9	29.61	1.53	31.03	2.44	31.25	2.78
IntraInter	7.7	28.00	3.69	29.56	3.85	29.85	4.18
InterIntra	7.9	28.34	3.26	29.86	3.98	30.14	4.40
ZeroOut	5.8	25.79	13.01	26.46	14.75	26.58	14.93

LIST OF FIGURES

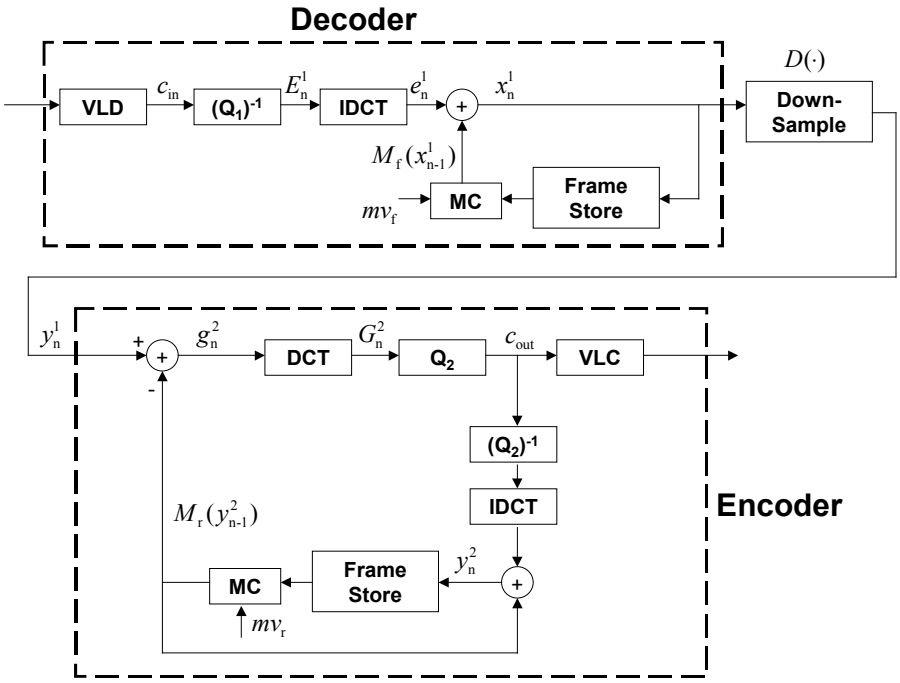

Fig. 1. Reference Architecture

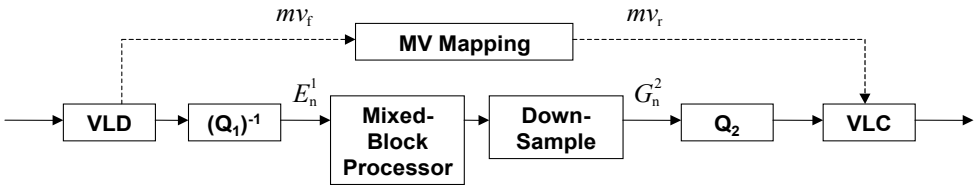

Fig. 2. Open-Loop Architecture

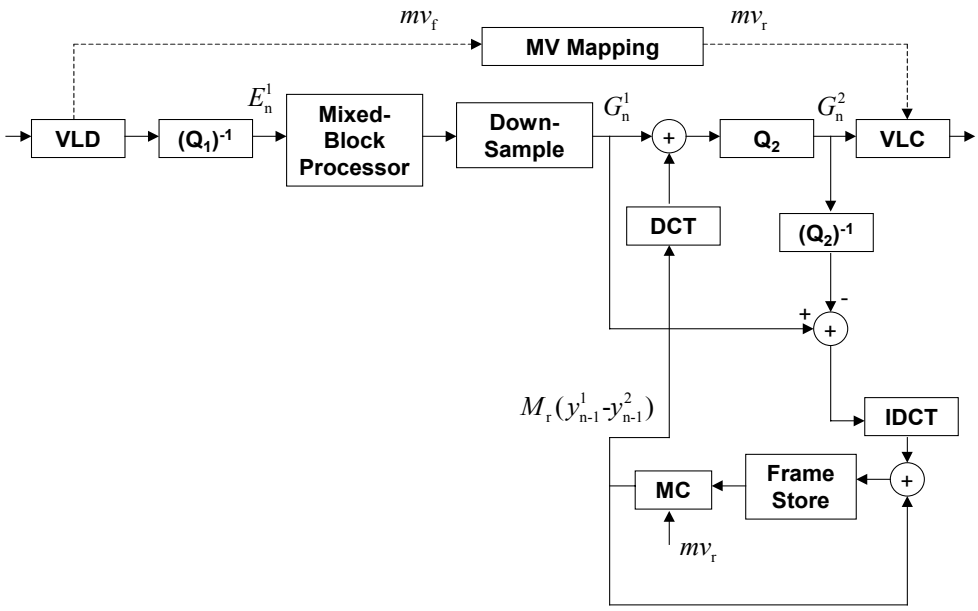

Fig. 3. Drift Compensation in Reduced Resolution

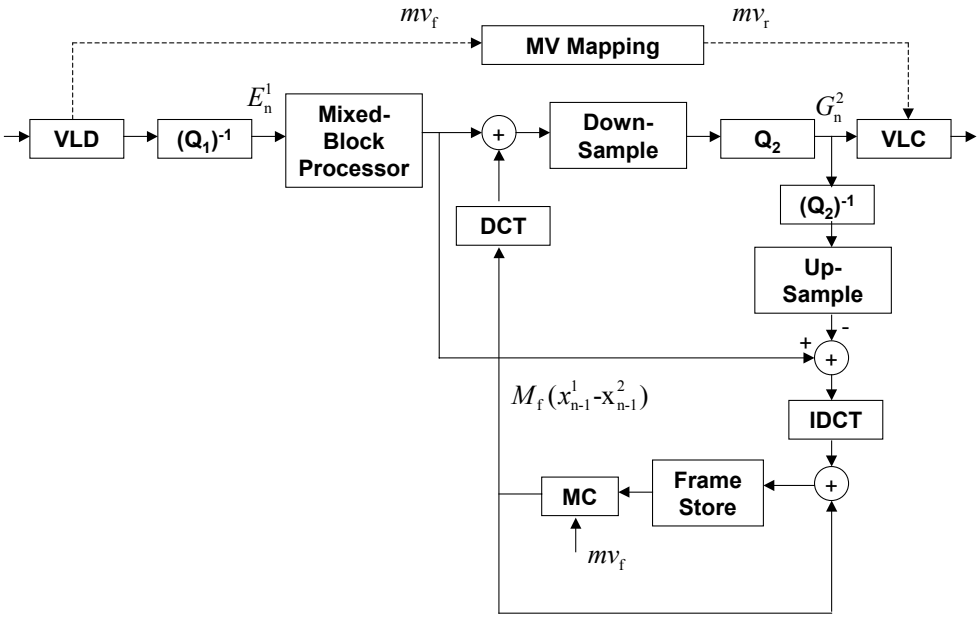

Fig. 4. Drift Compensation in Original Resolution

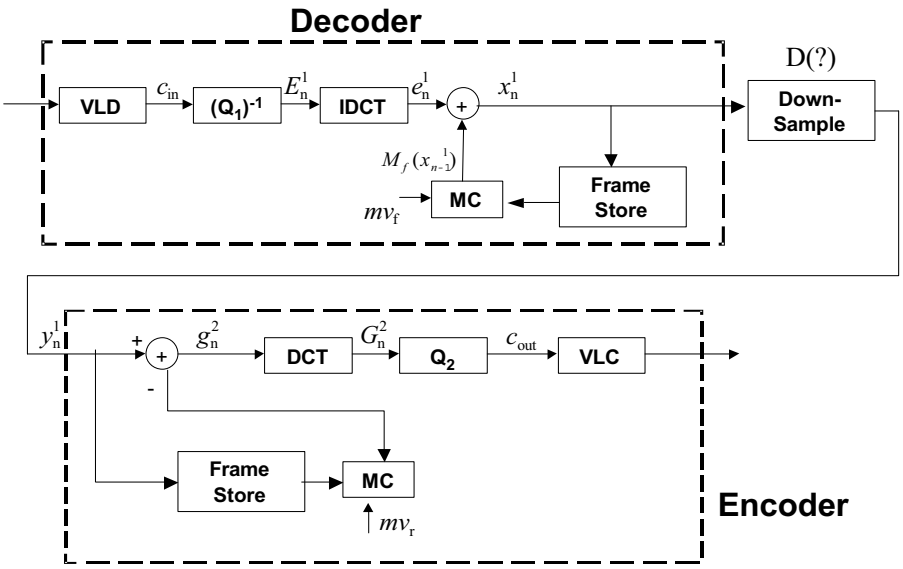

Fig. 5. *Partial_Encode* Architecture

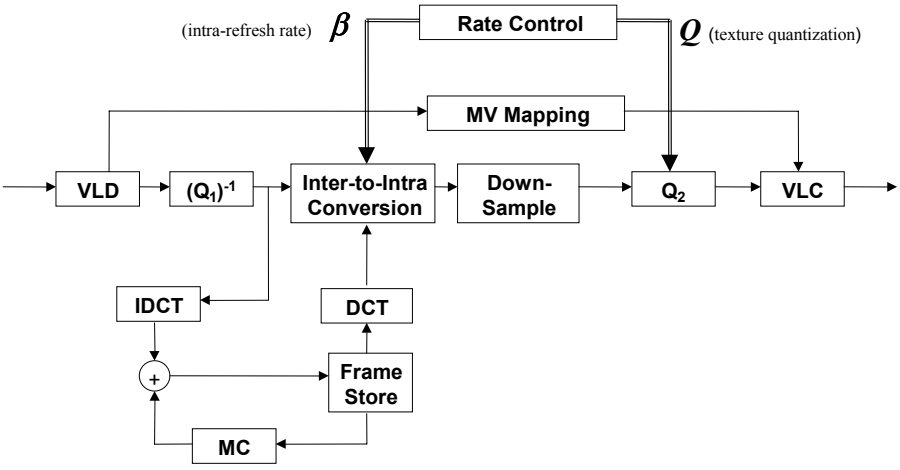

Fig. 6. *Intra_Refresh* in *OpenLoop* Architecture

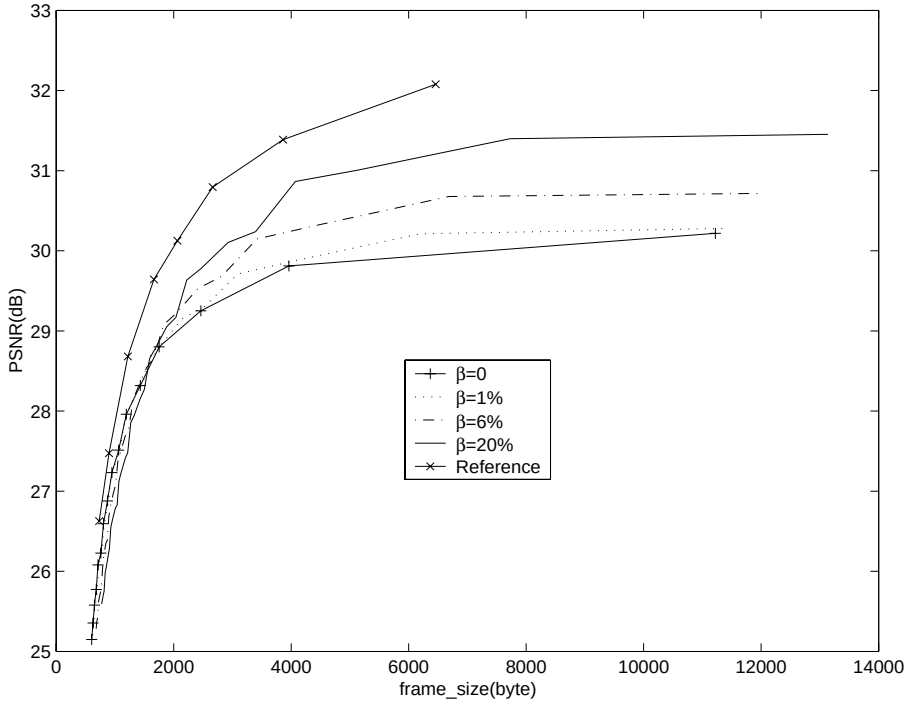

Fig. 7. RD Curve of Foreman for *Intra_Refresh* Architecture

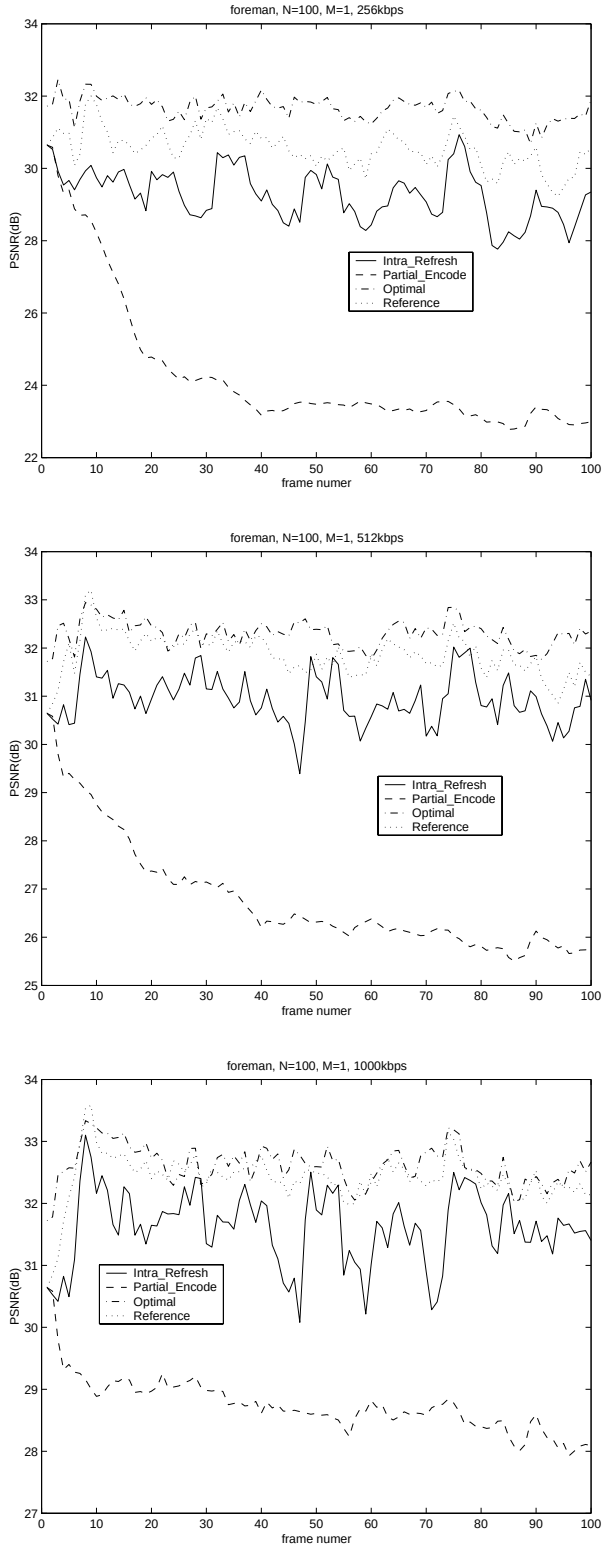

Fig. 8. Experimental Result for Reduce-spatial-resolution Transcoding by 2 with Various Output Rate for Foreman

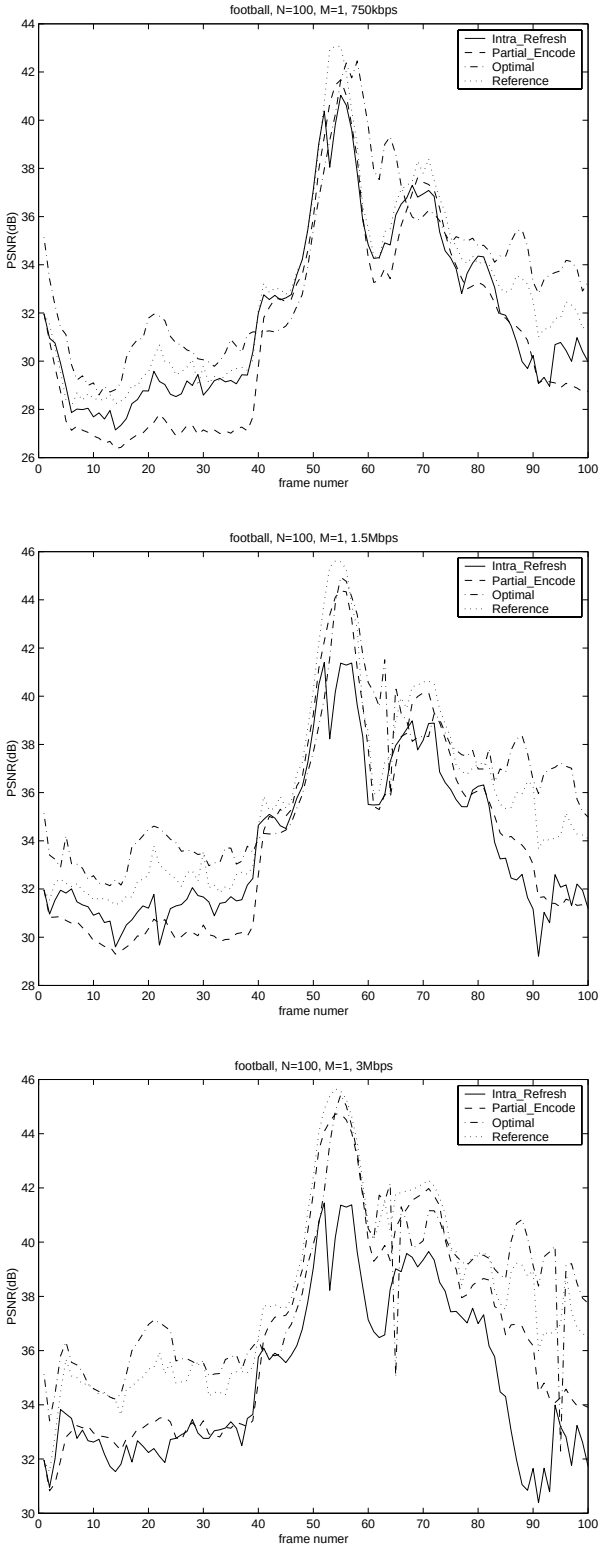

Fig. 9. Experimental Result for Reduce-spatial-resolution Transcoding by 2 with Various Output Rate for football