Video Transcoding to Support Random Access in Scalable Video Coding

Zhenyu Wu, Hongyang Yu, Bin Tang
Electronic Engineering Department
University of Electronic Science and Technology of China
Chengdu, Sichuan
China
zywu813cn@hotmail.com

Abstract: - In video applications including video broadcasting, interactive TV and IPTV, video streams are transmitted over various networks. In order to fulfill the requirement of video devices to cut-in broadcasting, start playback at a random location and jump to another location, the video service provider has to implement the random access functionality. Furthermore, the random access points can also enable the user devices to refresh the decoding process in error-resilient transmission. In this paper, we propose a novel transocding algorithm to insert random access points in the pre-encoded scalable video streams. Experiments show that the proposed algorithm can get 0.5~2.1dB PSNR gain over full decode and recode (FDR) transcoder and get 0.8~4dB PSNR gain over cascade pixel domain transcoder(CPDT). Simulation results also display that the proposed transcoding algorithm can reduce the computational complexity significant compared with FDR and CPDT method.

Key-Words: - scalable video coding, transcoding, random access

1 Introduction

The emerging home-networking and broadband convergence network technologies facilitate the application of scalable video coding (SVC) to heterogeneous receiving devices. The-state-of-art SVC standard offers desired temporal, spatial and quality scalabilities generalization of mobile communication services [1]. In SVC, the most fundamental information to reconstruct the video frame is referred to as the base layer (BL). The other layers, called enhancement layers (ELs), produce more refined information when combined with lower layers. Usually, playback, error-resilient transmission and devices cut-in requirements all need the random access points. Random access refers to the decoder's ability to start decoding a stream at a point other than the beginning of the stream, and recover an exact or approximate representation of the decoded pictures. Random access points enable seek, fast forward, fast backward, and stream switching operations on video streams. Furthermore, random access points enable tune-in to a broadcast or multicast session.

Conventionally the intra picture can be used as a random access point in the coded bit stream. However, due to the introduction of multiple reference pictures for inter prediction, the intra picture may not be sufficient for random access in the JVT coding standard [2][3]. For example, the decoded picture before an intra picture may be used as a reference for the pictures decoded after the intra picture. A particular type of the intra picture, called an instantaneous decoding refresh (IDR) picture, is specified in SVC as well as H.264/AVC. The IDR picture causes the decoder to mark all reference pictures as "unused for reference" immediately after decoding the IDR picture, i.e., the decoder clears the reference picture buffer when it receives the IDR picture. Once the IDR picture is decoded, all the following pictures can be decoded without using the pictures decoded prior to the IDR picture. Therefore, IDR pictures can be used as random access points instead of ordinary intra pictures.

To implement such random access in SVC, we must take layer switching random access into account. This kind of random access is a layer-dependent property. It enables the layer switching operation from another layer, which is important in scalable stream adaptation. Any layer switching operation can be performed at an access NAL unit where all the layer representations are IDR pictures. Decoding of an IDR picture and the following layer representations of that layer do not depend on any earlier access unit. However, relying on such access unit either causes reduced coding efficiency due to frequent coding of such access NAL units, or no prompt stream adaptation when such access NAL units are not coded frequently enough. To solve this problem, SVC allows encoding of IDR pictures independently for each layer. See [6] and [7] for more discussions and the coding efficiency difference of the two coding schemes. This flexibility in IDR coding makes the concepts of IDR access unit and coded video sequence dependent on the target layer for output, as mentioned earlier. An IDR picture in an enhancement layer may be predicted from the lower-layer representations in the same access unit. One property (popular) of single-loop decoding is that the IDR pictures in enhancement layers with a non-IDR base layer picture can also be used as layer random access point as well as layer switching point.

So implementation of absolutely random access in SVC is focused on IDR inserting in the base layer (dependency_id and quality_level all equal to zero). In the base layer, SVC streams usually have several temporal scalable layers. The details are described in section 2. Strictly, the entire sequence should be re-encoded to insert additional IDR pictures. Even when a single IDR picture is inserted in the pre-encoded bit stream, the re-encoding of all the pictures is required, according to the SVC standard [8]. However, we can adopt transcoding method instead of absolutely re-encoding the whole sequences. The structures of transcoders are presented in section 3.

Rest of the paper is organized as follows: section 2 overviews the temporal scalability of SVC, section 3 displays 5 structures of video transcoding, section 4 presents our proposed random access point inserting scheme in SVC, section 5 describes the motion compensation method in frequency domain, while the experiment results are given out in section 6. Finally, section 7 concludes the whole paper.

2 Overview of the SVC Temporal Scalable

A bit stream provides temporal scalability when the set of corresponding access units can be partitioned into a temporal base layer and one or more temporal enhancement layers with the following property. Let the temporal layers be identified by a temporal layer identifier T, which starts from 0 for the base layer and is increased by 1 from one temporal layer to the next. Then for each natural number k, the bit stream that is obtained by removing all access units of all temporal layers with a temporal layer identifier T greater than k forms another valid bit stream for the given decoder. For hybrid video codecs, temporal scalability can generally be enabled by restricting motion-compensated prediction to reference pictures with a temporal layer identifier that is less than or equal to the temporal layer identifier of the picture to be predicted. H.264/AVC allows the coding of picture sequences with arbitrary temporal dependencies, which are only restricted by the maximum usable decoded picture buffer (DPB) size.

A typical hierarchical prediction structure with 4 dyadic hierarchy stages for enabling temporal scalability is depicted in Fig. 1(a). Its GOP length is 8 and structural delay equals to 7. The first picture of a video sequence is encoded as the IDR picture. The temporal base layer pictures, which are called key pictures, are encoded in regular intervals. The set of pictures between two successive key pictures together with the succeeding key picture is considered to build a GOP, which is showed in Fig. 1(a). Fig. 1(b) illustrates a nondyadic hierarchical prediction structure, which provides 2 independently decodable subsequences with 1/9th and 1/3rd of the full frame rate. While fig. 1(c) shows a structure, which does not employ motion compensated prediction from pictures in the future. There is no delay in this structure but its coding efficiency is decreased heavily.

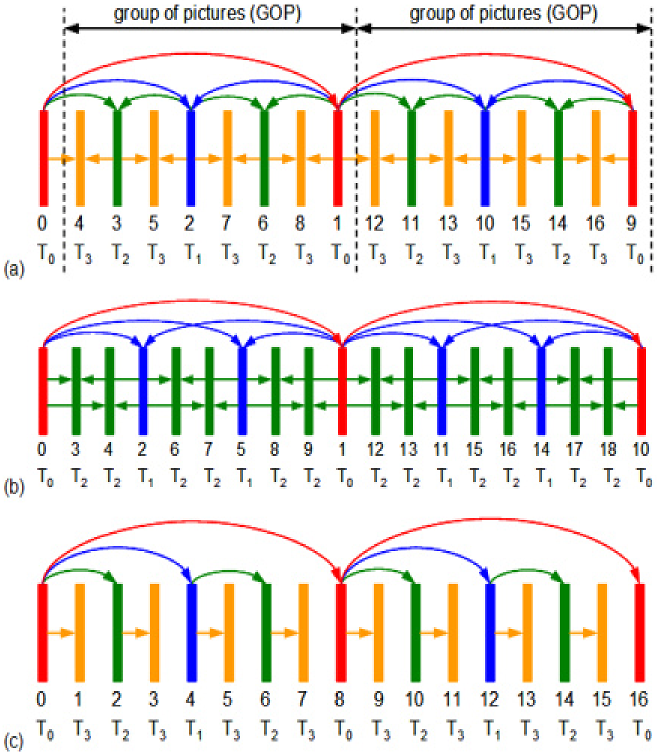

group of pictures (GOP)
group of pictures (GOP)
0 4 3 5 2 7 6 8 1 12 11 13 10 15 14 16 9
T_0 T_3 T_2 T_3 T_1 T_3 T_2 T_3 T_0 T_3 T_2 T_3 T_1 T_3 T_2 T_3 T_0
(a)
0 3 4 2 6 7 5 8 9 1 12 13 11 15 16 14 17 18 10
T_0 T_2 T_2 T_1 T_2 T_2 T_1 T_2 T_2 T_0 T_2 T_2 T_1 T_2 T_2 T_1 T_2 T_2 T_0
(b)
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
T_0 T_3 T_2 T_3 T_1 T_3 T_2 T_3 T_0 T_3 T_2 T_3 T_1 T_3 T_2 T_3 T_0
(c)

Fig.1 Hierarchical prediction structures for enabling temporal scalability. (a) Coding with hierarchical B-pictures. (b) Nondyadic hierarchical prediction structure. (c) Hierarchical prediction structure with a structural encoding/decoding delay of zero.

3 Structures of Transcoding Scheme

Typically, there are five transcoding architectures according to pioneers work: FDR (full decode and recode), CPDT(Cascade pixel domain transcoder), FPDT(Fast pixel domain transcoder), CFDT(Cascade Frequency domain transcoder) and FFDT(Fast Frequency domain transcoder).

FDR is the most straightforward and accurate way to transcoding by re-encoding frames absolutely. But it is most time consuming and complex. According to Marpe‘s work[14], ME occupies about 70% of the total encoding computations in H.264/AVC. Integer transform, inverse integer transform, quantization and inverse quantization process is approximately 8-16%, depending on the motion activities. CPDT reuses MV information, MB modes and partitions which can reduce the complexity greatly[4][5]. However, it will worsen the video quality up to 4dB compared with FDR at the same time[9][10]. FPDT combines frame’s reconstruction between the decoder and encoder. It simplifies the transcoding structure furthermore and halves frame storage buffer. FPDT method is based on the assumptions that motion compensation and loop filter are linear. Those assumptions have already been proved to be incorrect [11][12]. So there will introduce drift error in inter frames transcoding which will propagate through the whole GOP. Instead of fully decoding frames into pixel domain, motion estimation can also be done in the frequency domain[13]. So similar to CPDT and FPDT, CFDT and FFDT have been proposed

4 The Proposed Random Access Point Inserting Algorithm

To trade off the computation complexity and the quality of the whole sequences, our proposed random access point inserting algorithm is based on transcoding method and the transcoding is performed within a single GOP in which the low-pass frame ((I_n/P_n) is converted into IDR picture (IDR_n).

Since the low-pass frame (I_{n-1}) is deleted from the reference buffer due to the newly inserted IDR frame, we should transcode all the bilateral predicted frames which are forward predicted by I_{n-1} and backward predicted by IDR_n into only backward predicted ones. To implement our proposed scheme, there are two steps should be followed.

4.1 Selection of the pictures to be transcoded

Let us define the nth GOP by GOP_n. If the GOP consists of N frames, the display order of GOP_n is H^1_n……I_n. When I_n has been changed to IDR frame, each of the high-pass frames in GOP_n should be identified whether it is predicted from I_{n-1} or not. If H^a_n (0<a<N) is predicted from I_{n-1}, it should be included in the set T_n of the frames to be transcoded. For three typical structures of hierarchical prediction shown in Fig. 1, the pictures need to be transcoded are determined as follows:

Type 1. Dyadic structure

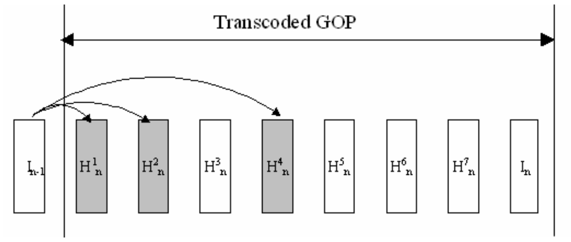


Fig. 2 Dyadic hierarchical structure with 4 temporal levels

The transcoding set is $T_n = \{H_n^{2^0}, H_n^{2^1}, ..., H_n^{2^m}\}$

where m is the non-negative integer and 2^m<N. Fig. 2 shows an example where N is 8 and m is 2. Note that there are only m+1 frames should be transcoded in the GOP sized N. The transcoding ratio $\frac{m+1}{N}$ decreases with raising GOP size. After determining T_n, all H^a_ns in the set are transcoded in the descending order of a.

Type 2 Non-dyadic structure

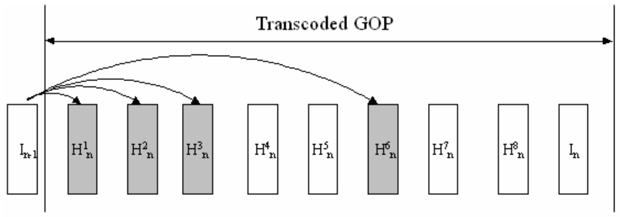


Fig. 3 Non-dyadic hierarchical coding structure with 3 temporal levels

The transcoding set is $T_n = \{1, 2, 3, ..., H_n^{k-1}, H_n^{k}\}$

where k is a non-negative integer and $k = 2 \times (k-1) \quad (k \le \frac{1}{2}N)$. For example, Showing

in figure 3, the GOP size N is 9 and k is 6. There are totally 4 frames need to be transcoded.
Type 3 Zero coding delay structure

Since there is no frames have not been decoded when IDR frame inserting, it needs not to transcode.

4.2 The proposed transcoding algorithm

In SVC, a B picture is composed of one or more B-slices and each MB in the B-slice is encoded as either INTER,INTRA,DIRECT or SKIP mode. Since there is no necessary to transcode the INTRA, DIRECT and SKIP MBs, we only focus on the INTER MBs in frames labeled as H. For those MBs, all of the transcoding architectures presented in section 3 can be adopted. But all of them may change the reconstructed frames entirely. It must result in the mismatching of the MVs and residuals of B frames in other temporal layers predicted from those HB frames. So the transcoding structures motioned in section 3 will not be the good choice for our proposed random inserting scheme.

With the purpose of our transcoding which is to keep the transcoded HB frames as exactly as the input ones, we should reuse the MVs and prediction modes in the original input SVC streams. As our assumption that the current bi-predicted MB's backward predicted MV and partition are the best prediction choice from the backward reference frame alone, we reuse them. So in our proposed transcoding algorithm, we need only to recalculate the residual of each MB by backward prediction.

Let us calculate the residual of an INTER MB first.

$$\begin{aligned} residual(x,y) = B_{B_n}(x,y) - \frac{1}{2}[&B_{I_{n-1}}(x+mv_{x-for}, y+mv_{y-for}) \\ &+ B_{I_n}(x+mv_{x-back}, y+mv_{y-back})] \end{aligned} \quad (1)$$

where $B_{B_n}(x,y)$ is the MB value at start point *(x,y)*, $B_{I_{n-1}}(x+mv_{x-for}, y+mv_{y-for})$ and $B_{I_n}(x+mv_{x-back}, y+mv_{y-back})$ are the forward and backward predicted value for this INTER MB.

When transcoding into backward predicted block, the residual will become:

$$\begin{aligned} residual'(x,y) &= B_{B_n}(x,y) - B_{I_n}(x+mv_{x-back}, y+mv_{y-back}) \\ &= \frac{1}{2}[B_{I_{n-1}}(x+mv_{x-for}, y+mv_{y-for}) \\ &\quad - B_{I_n}(x+mv_{x-back}, y+mv_{y-back})] + residual(x,y) \end{aligned} \quad (2)$$

Based on the linearity of integer transform, we can get the transcoded residual in frequency domain as:

$$\begin{aligned} T(residual'(x,y)) = T(residual(x,y)) + \frac{1}{2}\{&T(B_{I_{n-1}}(x+mv_{x-for}, y+mv_{y-for})) \\ &- T(B_{I_n}(x+mv_{x-back}, y+mv_{y-back}))\} \end{aligned} \quad (3)$$

where T( ) represents the integer transform.

After quantization, we will get the backward predicted MB value as:

$$\begin{aligned} Q(T(residual'(x,y))) &= Q(T(residual(x,y))) + \\ &round\{\frac{1}{2}[Q(T(B_{I_{n-1}}(x+mv_{x-for}, y+mv_{y-for}))) \\ &- Q(T(B_{I_n}(x+mv_{x-back}, y+mv_{y-back})))]\} \\ &= Z_{cur} + round\{\frac{1}{2}[Q(T(B_{I_{n-1}}(x+mv_{x-for}, y+mv_{y-for}))) \\ &- Q(T(B_{I_n}(x+mv_{x-back}, y+mv_{y-back})))]\} \end{aligned} \quad (4)$$

Re-writing function (4), we will get the transcoded backward predicted residual after integer transform and quantization as:

$$Z_{cur}{}' = Z_{cur} + round\{\frac{1}{2}(Z_{for} - Z_{back})\} \quad (5)$$

Fig. 4 shows us the proposed structure for inter MBs transcoding. The stored I frames are the reconstructed key frames in the frequency-domain of the continued two GOPs which displayed in fig. 5.

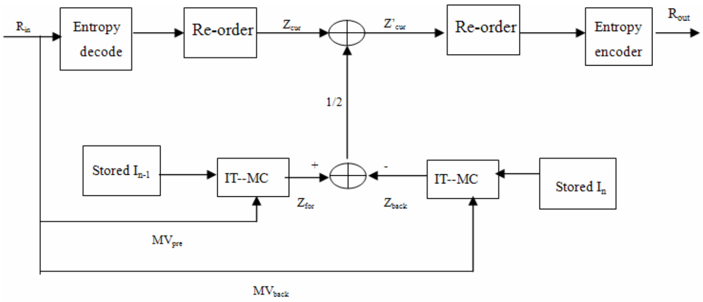


Fig. 4 the proposed structure for inter MBs transcoding

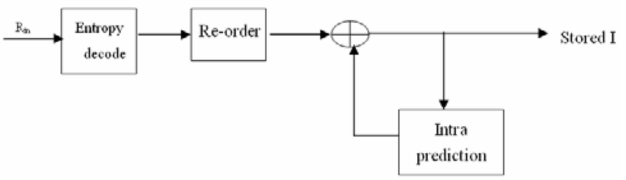


Fig. 5 the process to reconstruct stored I frame

Similar with 2-D DCT, the 2-D integer transform in H.264/AVC can also be performed by two 1-D integer transforms following the detrive as function (6).

$$Y = T(X) = HXH^T = H(HX)^T \quad (6)$$

Adopting the fast algorithm idea described in [15], we can simplify the computation further. Firstly, we split integer transform matrix into as follows:

$$H = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{pmatrix} = P_4 \begin{pmatrix} I_2 & J_2 \\ J_2 & -I_2 \end{pmatrix} \qquad (7)$$

where $P_4 = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 2 & 0 & 1 & 0 \\ 1 & -1 & 0 & 0 \\ 1 & 0 & -2 & 0 \end{pmatrix}$, I_2 is the identity matrix of order 2, J_2 is the matrix by reversing the rows of I_2 .Then we use butterfly structure to calculate 1-D integer transform value shown in Fig. 6.

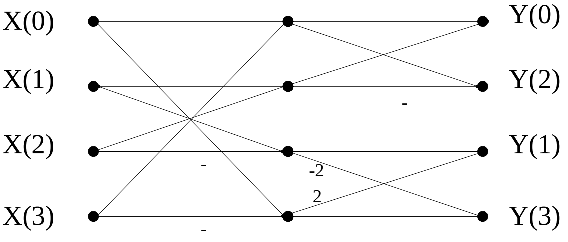


Fig. 6 1-D integer transform fast algorithm

Our proposed transcoding algorithm for random access points inserting in SVC can be described as following 7 steps:

I. Determine the transcoding frame set T_n following the rules in section 4.1

II. Select the MB subset Ω to be transcoded as follows:

- If MB_k is INTRA, DIRECT or SKIP, then $MB_k \notin \Omega$
- If MB_k is INTER, but I_{n-1} is not used as reference, then $MB_k \notin \Omega$
- If MB_k is INTER, and I_{n-1} is used as reference, then $MB_k \in \Omega$

III. Reconstruct I_{n-1} and I_n frames in frequency domain as shown in Fig. 5

IV. 4. Decode MBs in subset Ω to get MVs and Z_{cur}

V. 5. Do motion compensation in frequency domain to get Z_{for} and Z_{back} .

VI. 6. Calculate the transcoded backward predicted MB residual Z'_{cur} by function (5).

VII. 7. Modify the header information and output the transcoded SVC video stream.

The flow chart of our proposed random access point inserting scheme is shown in figure 7.

Through the discussion above, the key technique in our proposed random access points inserting algorithm should be focused on getting the predicted block value Z_{for} and Z_{back} by doing motion compensation (MC) in frequency domain. The next section presents our proposed MC method in frequency domain forH.264/AVC.

5. Motion Compensation in Frequency Domain

Motion estimation is based on blocks. When doing motion compensation in Frequency domain, the reference blocks do not always superpose the DCT blocks in the reference frames. So the idea of doing MC in DCT domain is to represent a reference block as a summation of horizontally and/or vertically displaced anchor blocks. Then the DCT value of the target block is constructed using the precomputed DCT values of the shifting matrices, according to the linearity of integer transform operation. In this section, we focus on implement motion compensation in frequency domain for H.264/AVC.

For simplicity of the representation, we use the following symbols to representation:
B_{xx}: integer sample
b_{xx}: half-pixel sample
bb_{xx}: quarter-pel sample

5.1 integer pixel MC in Frequency domain

The general setup is shown in fig. 7. Here, B_p is the current block of interest, B_1 …. B_4 are the four original neighboring blocks from which B_p is derived, and the motion vector is MV=(⊿x,⊿y)=(w,h). The shaded regions in B_1 …. B_4 are moved by MV. We are thus interested in obtaining the DCT representation of block B_p given the DCT representation of B_1 …. B_4 and motion vector MV. If we represent each block as a 4x4 matrix, then we can describe in the spatial domain through matrix multiplications (see function (8)).

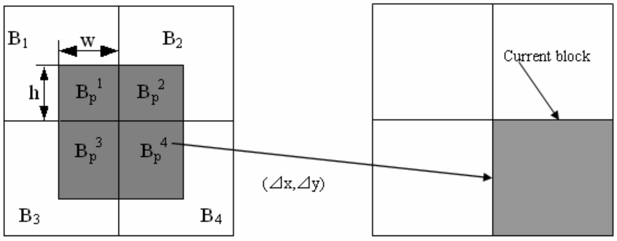


Fig. 8 Integer pixel position predicted block in reference frame

$$B_P = \begin{pmatrix} B_P^1 & B_P^2 \\ B_P^3 & B_P^4 \end{pmatrix} = \sum_{i=1}^{4} V_i B_i H_i \tag{8}$$

where $V_1 = V_2 = \begin{pmatrix} 0 & I_h \\ 0 & 0 \end{pmatrix}_{4\times4}$ $V_3 = V_4 = \begin{pmatrix} 0 & 0 \\ I_{4-h} & 0 \end{pmatrix}_{4\times4}$

$H_1 = H_3 = \begin{pmatrix} 0 & 0 \\ I_w & 0 \end{pmatrix}_{4\times4}$ $H_2 = H_4 = \begin{pmatrix} 0 & I_{4-w} \\ 0 & 0 \end{pmatrix}_{4\times4}$

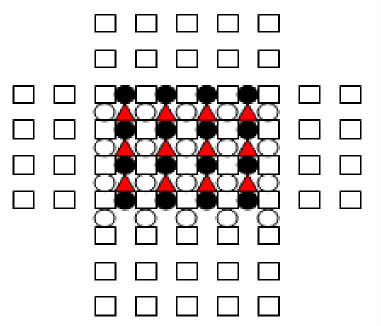
I_n is the identity matrix of order n and 0 is zero matrix.

Given the linearity of integer transform, we can get the predicted value of block B_p in frequency domain as in function (9).

$$T(B_P) = \sum_{i=1}^{4} T(V_i)T(B_i)T(H_i) \tag{9}$$

5.2 Fractional pixel MC in Frequency domain

In H.264/AVC, the interpolation process is depicted in Fig. 9 and 10 which can be subdivided into two steps. At first, the half-pixel positions (circle positions in Fig.9) are calculated using a horizontal or vertical 6-tap Wiener filter (1/32, -5/32, 20/32, 20/32, -5/32, 1/32), respectively. Using the same Wiener filter applied at half-pixel positions the fractional-pixel positions (triangular positions in Fig. 9) are computed. In the second step, which is described in Fig. 10, the remaining quarter-pel positions are obtained by bilinear filter applied at already calculated half-pixel positions and existing full-pixel positions.

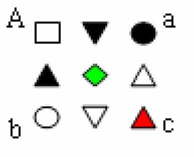


● is horizontal interpolated by integer samples.

○ is vertical interpolated by integer samples.

▲ is horizontal or vertical interpolated by other half-pel samples.

Fig. 9 half -pixel position interpolation



▼ is horizontal interpolated by mixtual integer and half -pixel samples (eg. (A+a)/2)

▲ is vertical interpolated by mixtual integer and half- pixel samples (eg.(A+b)/2)

▽ is horizontal interpolated by mixtual two types of half - pixel samples (eg. (b+c)/2)

△ is vertical interpolated by mixtual two types of half - pixel samples (eg. (a+c)/2)

◆ is diagonal interpolated by half-pixel samples (eg. (a+b)/2)

Fig. 10 quarter-pel position interpolation

5.2.1 half -pixel MC in DCT domain for H.264

According to Fig. 9, there are 3 types of half-pixel interpolation. We display them as following.

Type 1. horizontal interpolating by integer pixel

Since the half-pixel interpolation filter is a 6-tap Winner filter, the size of reference block should be 4x9, shown in Fig.11.

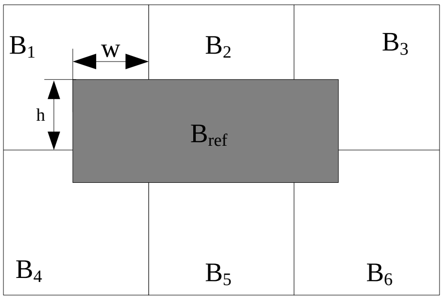


Fig. 11 half-pixel position predicted block in reference frame (horizontal prediction)

Once getting the reference block, we can calculate the target block's pixel values by filtering every raw through the 6-tap filter. The matrix multiplication is shown in function (10).

$$b_p = B_{ref} \times H_{pre} \quad （10）$$

where
$$H_{pre} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -5 & 1 & 0 & 0 \\ 20 & -5 & 1 & 0 \\ 20 & 20 & -5 & 1 \\ -5 & 20 & 20 & -5 \\ 1 & -5 & 20 & 20 \\ 0 & 1 & -5 & 20 \\ 0 & 0 & 1 & -5 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$B_{ref} = \sum_{i=1}^{6} V_i B_i H_i \qquad (11)$$

The final integer transform values of block b_p are got from function (12).

$$T(b_p) = \sum_{i=1}^{6} T(V_i)T(B_i)T(H_i \times H_{pre}) \qquad (12)$$

where $V_1 = V_2 = V_3 = \begin{pmatrix} 0 & I_h \\ 0 & 0 \end{pmatrix}_{4\times 4}$

$$V_4 = V_5 = V_6 = \begin{pmatrix} 0 & 0 \\ I_{4-h} & 0 \end{pmatrix}_{4\times 4}$$

$$H_1 = H_4 = \begin{pmatrix} 0 & 0 \\ I_w & 0 \end{pmatrix}_{4\times 9}$$

$$H_2 = H_5 = \begin{pmatrix} 0_{4\times w} & I_{4\times 4} & 0_{4\times(5-w)} \end{pmatrix}_{4\times 9}$$

$$H_3 = H_6 = \begin{pmatrix} 0 & I_{5-w} \\ 0 & 0 \end{pmatrix}_{4\times 9}$$

Type 2 vertical interpolating by integer pixel

It is similar with type 1, except the size of reference block is 9x4, shown in Fig.12.

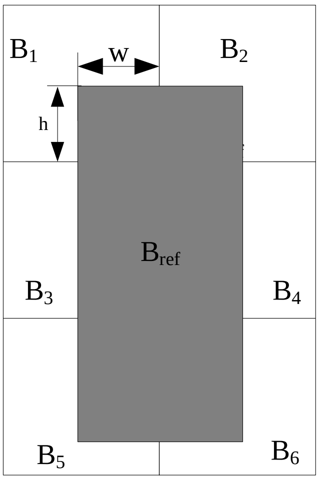


Fig. 12 half-pixel position predicted block in reference frame (vertical prediction)

The value of predicted block b_p is gotten as function (13)-(14).

$$\begin{aligned} b_p &= V_{pre} \times B_{ref} \\ &= V_{pre} \times (\sum_{i=1}^{6} V_i B_i H_i) \end{aligned} \qquad (13)$$

$$T(b_p) = \sum_{i=1}^{6} T(V_{pre} \times V_i) T(B_i) T(H_i) \qquad (14)$$

where
$$V_{pre} = \begin{pmatrix} 1 & -5 & 20 & 20 & -5 & 1 & 0 & 0 & 0 \\ 0 & 1 & -5 & 20 & 20 & -5 & 1 & 0 & 0 \\ 0 & 0 & 1 & -5 & 20 & 20 & -5 & 1 & 0 \\ 0 & 0 & 0 & 1 & -5 & 20 & 20 & -5 & 1 \end{pmatrix}$$

$$V_1 = V_2 = \begin{pmatrix} 0 & I_h \\ 0 & 0 \end{pmatrix}_{9\times 4}$$

$$V_3 = V_4 = \begin{pmatrix} 0_{h\times 4} & I_{4\times 4} & 0_{(5-h)\times 4} \end{pmatrix}^T_{9\times 4}$$

$$V_5 = V_6 = \begin{pmatrix} 0 & 0 \\ I_{5-h} & 0 \end{pmatrix}_{9\times 4}$$

$$H_1 = H_3 = H_5 = \begin{pmatrix} 0 & 0 \\ I_w & 0 \end{pmatrix}_{4\times 4}$$

$$H_2 = H_4 = H_6 = \begin{pmatrix} 0 & I_{4-w} \\ 0 & 0 \end{pmatrix}_{4\times 4}$$

Type 3 interpolating from other half-pixel position

These half-pixel positions can be interpolated either horizontally or vertically. We choose them by the principle to minimum the total computation.

(1) horizontal

We can get transcoded half-pixel value by horizontal interpolation in triangular positions as function (15) after integer transforming

$$T(b'_p) = \sum_{i=1}^{6} T(V_i)T(b_i)T(H_i \times H_{pre}) \qquad (15)$$

where V_i, H_i and H_{pre} are the same in type 1.
b_i is half-pixel sample interpolated from vertical integer pixels.
(2) Vertical

The final value of half-pixel sample by vertical interpolation in triangular positions is calculated as function (16)

$$T(b_p^{'})=\sum_{i=1}^{6}T(V_{pre}\times V_i)T(b_i)T(H_i) \quad (16)$$

where V_i, H_i and V_{pre} are the same in type 2. b_i is half-pixel sample interpolated from horizontal integer pixels.

5.2.2 quarter pixel MC in DCT domain for H.264

In H.264/AVC, bilinear filter is adopted in quarter pixel interpolation. According to Fig. 10, there are 5 types for quarter pixel interpolation.

Type 1 horizontal interpolating by mixed integer and half-pixel positions

There are 2 cases to interpolate the 4x4 block in quarter-pel position.

1) $B_{ref\,4\times3}$, $b_{ref\,4\times2}$

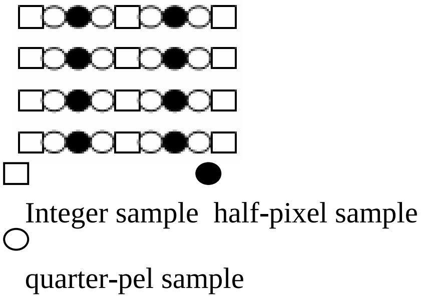


Fig.13 quarter-pel position compensation type1.1

The interpolated quarter-pel can be represented as function (17).

$$bb_p=\frac{1}{2}\{B_{ref}\times K_1+b_{ref}\times L_1\} \quad (17)$$

where $K_1=\begin{pmatrix}1&0&0&0\\0&1&1&0\\0&0&0&1\end{pmatrix}, L_1=\begin{pmatrix}1&1&0&0\\0&0&1&1\end{pmatrix}$,

$B_{ref\,4\times3}=\sum_{i=1}^{4}V_i\times B_i\times H_i$ and

$b_{ref\,4\times2}=\sum_{i=1}^{4}V_i\times b_i\times h_i$

We adopt the same idea to get $B_{ref\,4\times3}$ and $b_{ref\,4\times2}$ as described in previous parts. To simplify the computation, we make the b_{ref} within only one b_i. For example, we let w=2 and h=4 to enclose b_{ref} inside b_1: $b_{ref\,4\times2}=b_1\times h_1$. The final transcoded value is calculated by function (18)

$$T(bb_p)=\frac{1}{2}\{\sum_{i=1}^{4}\left[T(V_i)\left(T(B_i)T(H_iK_1)+T(b_1)T(h_1L_1)\right)\right]\} \quad (18)$$

2) $B_{ref\,4\times2}$, $b_{ref\,4\times3}$

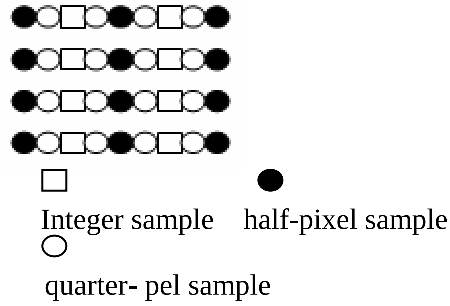


Fig.14 quarter-pel position compensation type1.2

$$bb_p=\frac{1}{2}\{b_{ref}\times K_1+B_{ref}\times L_1\} \quad (19)$$

$$B_{ref\,4\times2}=\sum_{i=1}^{4}V_i\times B_i\times h_i$$

$$b_{ref\,4\times3}=\sum_{i=1}^{4}V_i\times b_i\times H_i$$

Similar with case (1) we let w=3 and h=4 to enclose b_{ref} inside b_1. Then $b_{ref\,4\times3}=b_1\times H_1$ and the final value is:

$$T(bb_p)=\frac{1}{2}\{\sum_{i=1}^{4}\left[T(V_i)\left(T(b_1)T(H_1K_1)+T(B_i)T(h_iL_1)\right)\right]\} \quad (20)$$

Type 2 vertical interpolate by mixed integer and half-pixel positions

1) $B_{ref\,3\times4}$, $b_{ref\,2\times4}$

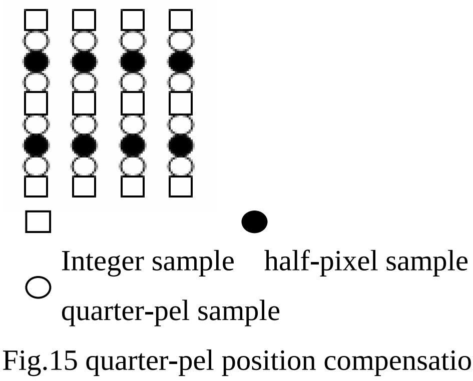


Fig.15 quarter-pel position compensation type2.1

$$bb_p=\frac{1}{2}\{K_2\times B_{ref}+L_2\times b_{ref}\} \quad (21)$$

where $K_2=\begin{pmatrix}1&0&0\\0&1&0\\0&1&0\\0&0&1\end{pmatrix}, L_2=\begin{pmatrix}1&0\\1&0\\0&1\\0&1\end{pmatrix}$

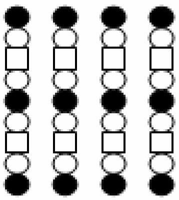
$$B_{ref\,3\times4}=\sum_{i=1}^{4}V_i\times B_i\times H_i$$

$$b_{ref\,2\times4}=\sum_{i=1}^{4}v_i\times b_i\times H_i$$

We let w=4,h=2 to enclose to enclose b_{ref} inside b_1. $b_{ref\,2\times4}=v_1\times b_1$

$$T(bb_p)=\frac{1}{2}\{\sum_{i=1}^{4}\left[\left(T(K_2V_i)T(B_i)+T(L_2v_1)T(b_1)\right)T(H_i)\right]\}\quad(22)$$

2) $B_{ref\,2\times4,}$ $b_{ref\,3\times4}$



□ Integer sample ● half-pixel sample

○ quarter-pel sample

Fig.16 quarter-pel position compensation type2.2

$$bb_p=\frac{1}{2}\{K_2\times b_{ref}+L_2\times B_{ref}\}\quad(23)$$

$$B_{ref\,2\times4}=\sum_{i=1}^{4}v_i\times B_i\times H_i$$

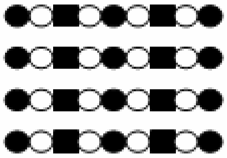
$$b_{ref\,3\times4}=\sum_{i=1}^{4}V_i\times b_i\times H_i$$

We let w=4,h=3 to enclose to enclose b_{ref} inside b_1. Then $b_{ref\,3\times4}=V_1\times b_1$

$$T(bb_p)=\frac{1}{2}\{\sum_{i=1}^{4}\left[\left(T(K_2V_1)T(b_1)+T(L_2v_i)T(B_i)\right)T(H_i)\right]\}\quad(24)$$

Type 3 horizontal interpolate by mixed 2 types of half-pixel (type 1and 3 or type 2 and 3 in section 5.2.1) positions

1) $b_{ref\,4\times3,}$ $b'_{ref\,4\times2}$



● half-pixel sample got by integer sample b_{ref}

■half-pixel sample got by half-pixel sample b'$_{ref}$

○ quarter- pel sample

Fig.17 quarter-pel position compensation type3.1

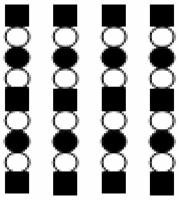
$$bb_p=\frac{1}{2}\{b_{ref}\times K_2+b'_{ref}\times L_2\}\quad(25)$$

$$b_{ref\,4\times3}=\sum_{i=1}^{4}V_i\times b_i\times H_i$$

$$b'_{ref\,4\times2}=\sum_{i=1}^{4}V_i\times b'_i\times h_i$$

$$T(bb_p)=\frac{1}{2}\{\sum_{i=1}^{4}\left[T(V_i)\left(T(b_i)T(H_iK_1)+T(b'_i)T(h_iL_1)\right)\right]\}\quad(26)$$

2) $b_{ref\,4\times2,}$ $b'_{ref\,4\times3}$



● half-pixel sample got by integer sample b_{ref}

■ half-pixel sample got by half-pixel sample b'$_{ref}$

○ quarter- pel sample

Fig.18 quarter-pel position compensation type3.2

$$bb_p=\frac{1}{2}\{b'_{ref}\times K_1+b_{ref}\times L_1\}\quad(27)$$

$$b_{ref}=\sum_{i=1}^{4}V_i\times b_i\times H_i(2)$$

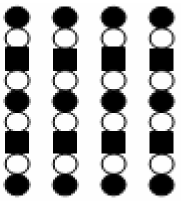
$$b'_{ref}=\sum_{i=1}^{4}V_i\times b'_i\times H_i(3)$$

$$T(bb_p)=\frac{1}{2}\{\sum_{i=1}^{4}\left[T(V_i)\left(T(b'_i)T(H_iK_1)+T(b_i)T(h_iL_1)\right)\right]\}\quad(28)$$

The computation can also be reduced by calculate b_{ref} similar with type 2.

Tye 4 vertical interpolate by mixtual 2 types of half-pixel positions (type 1and 3 / type 2 and 3)

1) $b_{ref\,3\times4,}$ $b'_{ref\,2\times4}$



● half-pixel sample got by integer sample b_{ref}

■ half-pixel sample got by half-pixel sample b'_{ref}

○ quarter- pel sample

Fig.19 quarter-pel position compensation type4.1

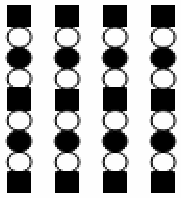
$$bb_p = \frac{1}{2}\{K_2 \times b_{ref} + L_2 \times b'_{ref}\} \qquad (29)$$

$$b_{ref} = \sum_{i=1}^{4} V_i(3) \times b_i \times H_i$$

$$b'_{ref} = \sum_{i=1}^{4} V_i(2) \times b'_i \times H_i$$

$$T(bb_p) = \frac{1}{2}\{\sum_{i=1}^{4}\left[\left(T(K_2V_i)T(b_i) + T(L_2v_i)T(b'_i)\right)T(H_i)\right]\} \qquad (30)$$

2) $b_{ref\,2\times 4}$, $b'_{ref\,3\times 4}$



● half-pixel sample got by integer sample b_{ref}

■ half-pixel sample got by half-pixel sample b'_{ref}

○ quarter- pel sample

Fig.20 quarter-pel position compensation type4.2

$$bb_p = \frac{1}{2}\{K_2 \times b'_{ref} + L_2 \times b_{ref}\} \qquad (31)$$

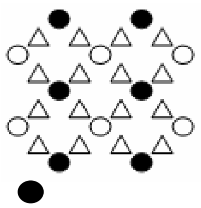
$$b_{ref} = \sum_{i=1}^{4} V_i(2) \times b_i \times H_i$$

$$b'_{ref} = \sum_{i=1}^{4} V_i(3) \times b'_i \times H_i$$

$$T(bb_p) = \frac{1}{2}\{\sum_{i=1}^{4}\left[\left(T(K_2V_i)T(b'_i) + T(L_2v_i)T(b_i)\right)T(H_i)\right]\} \qquad (32)$$

Type 5 diagonal interpolation

1) b_{ref1} 3x2, b_{ref2} 2x3



● ○

half-pixel sample b_{ref1} half-pixel sample b_{ref2}

△ quarter-pel sample

Fig.21 quarter-pel position compensation type5.1

$$bb_p = \frac{1}{2}\{A \times b_{ref1} \times K_3 + L_3 \times b_{ref2} \times A^T\} \qquad (33)$$

where $K_3 = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}, L_3 = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix}$

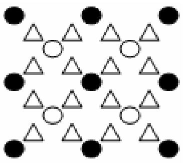
and $A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

If b_{ref1} *is* got from half-pixel position interpolation type 1 then b_{ref2} is got from half-pixel position interpolation type 2, vice versa.

$b_{ref1/2}$ are got from half-pixel position MC, described in part B.

$$T(bb_p) = \frac{1}{2}\{\sum_{i=1}^{4}[T(AV_i)T(b_i)T(H_iK_3)] + \sum_{l=1}^{4}[T(L_3V_l)T(b_l)T(H_lA^T)]\} \qquad (34)$$

2) b_{ref_1} 3x3, b_{ref_2} 2x2



○ ●

half-pixel sample b_{ref2} half-pixel sample b_{ref1}

△ quarter-pel sample

Fig.22 quarter-pel position compensation type5.2

$$bb_p = \frac{1}{2}\{A \times b_{ref1} \times K_4 + L_3 \times b_{ref2} \times C\} \qquad (35)$$

where $K_4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$, $C = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$

$$T(bb_p)=\frac{1}{2}\{\sum_{i=1}^{4}[T(AV_i)T(b_i)T(H_iK_4)] + \sum_{l=1}^{4}[T(L_3V_l)T(b_l)T(H_lC)]\} \quad (36)$$

6. Experimental Results

In this section, we present the experimental results comparison of our proposed transcoding algorithm, FDR and CPDT. We simulate these algorithms with Joint Scalable Video Model (JSVM) 9.17 reference software. The experiments are performed on several scalable sequences, with QCIF@15fps for Base layer and CIF@30 for Extension layer. Table 1 shows the input streams parameter for dyadic and non-dyadic structures.

Table 1 Input stream coding parameters

Parameters	value	
	Dyadic	Non-dyadic
GOP size	8	9
Frame number	320	270
QP	28	28
FGS	No	No
Number of layers	2	2
Inter-layer prediction	Adaptive	Adaptive

The random access points inserting period is designed to 2 GOP size. So the IDR period is 16 for dyadic structure while 18 for non-dyadic structure.

We use the following notation to identify the result streams.

- B_o: The bitstream without IDR pictures encoded by JSVM
- B_f: The bitstream transcoded whole bitstream using FDR structure at the same QP value
- B_c: The bitstream transcoded whole bitstream using CPDT structure at the same QP value
- B_p: The bitstream transcoded using our proposed algorithm at the same QP value

Table 2 shows the average PSNR(dB) and bitrate(k bits/s) comparison of typical sequences for dyadic hierarchical prediction structure with four dyadic hierarchy stages in temporal scalability in base layer. Compared with B_o, all of the transcoded bitstreams are showing lower PSNR and higher bitrate. It is because when an IDR frame is encoded, the frames prior to this IDR frame are cleared in the reference picture buffer. The reduction of the reference frames will cause encoder's performance dropping. Transcoding using either the structures mentioned in section 3 or our proposed algorithm will also worsen the original streams' quality inevitably.

The performance of the FDR algorithm has shown that re-encoding the whole stream absolutely by inserting IDR frames increased less bitrate (5~13%) than our proposed method. The reason is that, during transcoding by FDR method, the residuals are got from decoded frames instead of the original one, thus will result to smaller number of bits allocated to the transcoded MB [16]. Re-encoding must cause quality reduction. Obviously, there is a 0.75~1.97dB drop in FDR method. CPDT method simplified the re-encoding procedure of FDR method by re-using MVs and partitions. Its performance is worse than FDR method. Compared with our proposed scheme, it has a 1.1~3.7dB degradation.

Our proposed transcoding algorithm just needs to transcode 3 frames in the IDR frames inserting GOPs. Focusing on keeping the reconstructed frames as the same as those in the input streams after transcoding, the PSNRs of our proposed algorithm are much higher than FDR and CPDT methods. There are only less than 0.4dB decrease with the original streams. However, the bitrate increased 7~18.5%, for reducing the number of reference frames.

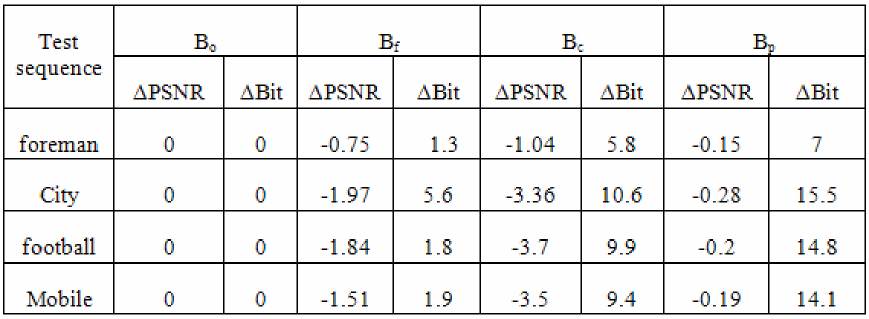
Table 3 shows the average PSNR(dB) and bitrate(k bits/s) comparison of typical sequences for non-dyadic hierarchical prediction structure with three non-dyadic hierarchy stages in temporal scalability in base layer. From it, the similar conclusion can be drawn as dyadic one. The bitrate increasing is a little bit more than dyadic one for our proposed algorithm, since there are one more frame needs to be transcoded in each IDR period.

Table 4 and table 5 display the comparison of computation complexity between our proposed algorithm and FDR method and between proposed algorithm and CPDT method for both dyadic and non-dyadic hierarchical prediction structures. According to these tables, it is obviously that our proposed transcoding algorithm for random access points inserting can save 93~98% transcoding time compared with FDR method and save 88~95% coding time with CPDT method.

7. Conclusion

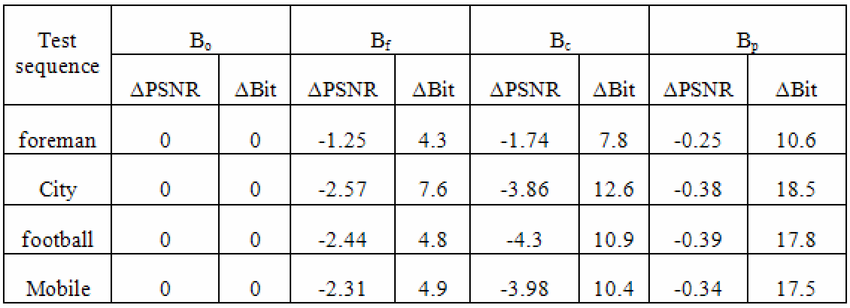
In this paper, we propose a novel transcoding algorithm, which enables the playback, broadcasting cut-in at random access points. Experimental results showed that the proposed algorithm outperformed the FDR and CPDT scheme quite well in terms of rate-distortion performance as well as computational complexity. With ignorable computation addition, our proposed algorithm can be efficiently applied to video streaming service over various networks. The fractional pixel motion compensation method in frequency domain for H.264/AVC proposed in section V can also be adopted in other H.264/AVC based transcoders.

Table 2 AverageΔPSNR(dB) ΔBitrate (%)



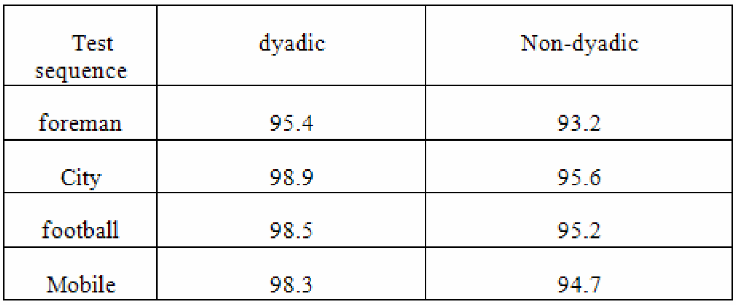
Test sequence	B_o		B_f		B_c		B_p	
	ΔPSNR	ΔBit	ΔPSNR	ΔBit	ΔPSNR	ΔBit	ΔPSNR	ΔBit
foreman	0	0	-0.75	1.3	-1.04	5.8	-0.15	7
City	0	0	-1.97	5.6	-3.36	10.6	-0.28	15.5
football	0	0	-1.84	1.8	-3.7	9.9	-0.2	14.8
Mobile	0	0	-1.51	1.9	-3.5	9.4	-0.19	14.1

Table 3 AverageΔPSNR(dB) ΔBitrate (%)



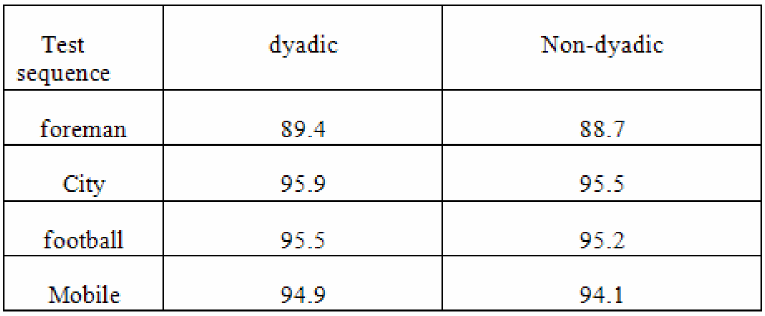
Test sequence	B_o		B_f		B_c		B_p	
	ΔPSNR	ΔBit	ΔPSNR	ΔBit	ΔPSNR	ΔBit	ΔPSNR	ΔBit
foreman	0	0	-1.25	4.3	-1.74	7.8	-0.25	10.6
City	0	0	-2.57	7.6	-3.86	12.6	-0.38	18.5
football	0	0	-2.44	4.8	-4.3	10.9	-0.39	17.8
Mobile	0	0	-2.31	4.9	-3.98	10.4	-0.34	17.5

Table 4 Complexity reduction (%) between FDR and Proposed method



Test sequence	dyadic	Non-dyadic
foreman	95.4	93.2
City	98.9	95.6
football	98.5	95.2
Mobile	98.3	94.7

Table 5 Complexity reduction (%) between CPDT and Proposed method



Test sequence	dyadic	Non-dyadic
foreman	89.4	88.7
City	95.9	95.5
football	95.5	95.2
Mobile	94.9	94.1

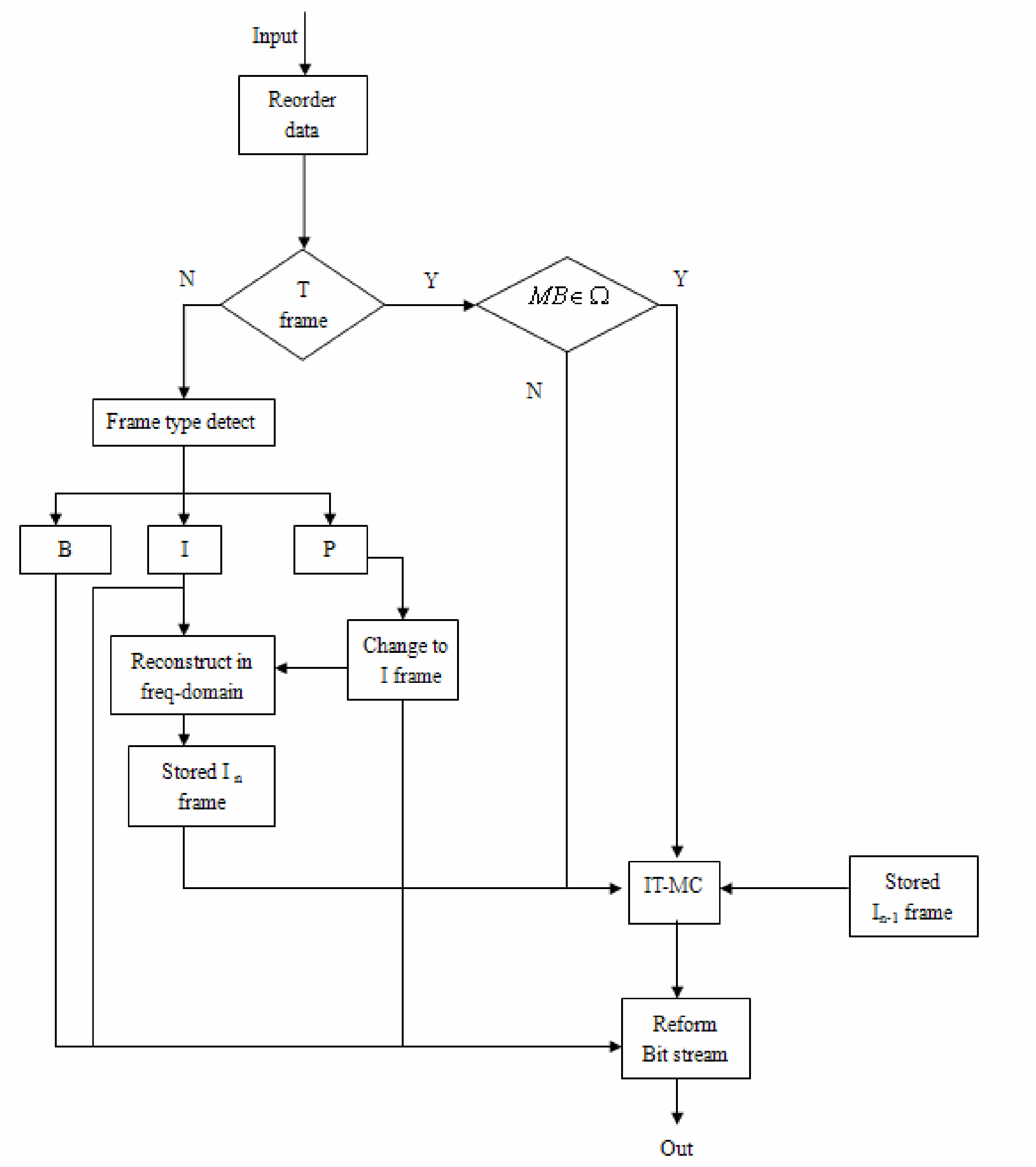


Fig. 7 flow chart of our proposed random access point inserting scheme

References

[1] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/ AVC Standard", IEEE Trans. Circuits. Systs. Video Tech., vol. 17, no. 9, Sept. 2007, pp.1103-1120.

[2] ITU-T and ISO/IEC JTC1, "Advanced video Coding for Generic Audiovisual Services", ITU-T Recommendation H.264 – ISO/IEC 14496-10 AVC, 2003

[3] M. M. Hannuksela, Y. –K. Wang, and M. Gabbouj, "Random access using isolated regions", Proc. Of IEEE International Conference on Image processing (ICIP), Sept. 2003

[4] Y. G. Lee, J.B. RA, "Fast motion estimation robust to random motions based on a distance prediction", IEEE Trans. Circuits. Systs. Video Tech., vol. 16, no. 7, Jul. 2006, pp. 869-875

[5] Y.-W. Huang, B.-Y. Hsieh, T.-C. Chen, and L.-G. Chen, "Analysis fast algorithm and VLSI

architecture design for H.264/AVC intra frame coder", IEEE Trans. Circuits. Systs. Video Tech., vol. 15, no. 3, Mar. 2005, pp.378-401

[6] Q. Shen, H. Li, "CE6: Enhancement-layer IDR (EIDR) picture", ISO/IEC JTC1/SC29/WG11, Doc. JVT- R053, Jan., 2006

[7] Y. Wang, M. M. Hannuksela, "Enhancement-layer IDR (EIDR) picture", ISO/IEC JTC1/SC29/WG11, Doc. JVT- Q065, Oct. 2005

[8] T. Wiegand et al., "Joint Draft 10 of SVC Amendment", ISO/IEC JTC1/SC29/WG11, Doc. JVT-W201, Apr, 2007

[9] D. Lefol, D. Bull, N. Canagarajah, "Performance Evaluation of Transcoding Algorithms for H.264" , IEEE trans. Consumer Electronics Volume 52, Issue 1, Feb. 2006 pp:215 – 222

[10] Z. Wu, H. Yu, B. Tang, H. Hu, "Performance Evaluation of Transcoding Algorithms for MPEG-2 to AVS-P2", Proc. Of International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP'06), Dec. 2006

[11] P. A. A. Assuncao and M. Ghanbari, "Frequency-domain video transcoder for dynamic bit-rate reduction of MPEG-2 bit streams," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, 1998 pp. 953-967.

[12] J. Youn, M.-T. Sun, and J. Xin, "Video transcoder architectures for bit rate scaling of H.263 bit streams," presented at Proceedings of the 1999 7th International Multimedia Conference - ACM MULTIMEDIA '99, Oct 30-Nov 5 1999, Orlando, FL, USA, 1999.

[13] S.-F. Chang, and D. G. Messerschmitt, "Manipulation and Compositing of MC-DCT Compressed Video", IEEE Journal of Selected Areas in Communications (JSAC), Special Issue on Intelligent Signal Processing, Jan. 1995, pp. 1-11.

[14] D. Marpe, H. Schwarz, and T. Wiegand, "Content-based adaptive binary arithmetic coding in the H.264/AVC video compression standard", IEEE Trans. Circuits. Systs. Video Tech., 2003,13(7),pp.620-636

[15] Y.H. Zeng, L.Z. Cheng, G.Bi,A, C.Kot, "Integer DCTs and Fast Algorithms", IEEE trans. Signal proc. Vol. 49,no. 11, Nov, 2001

[16] W. X. Guo et al., "Mismatch MB Retrieval for MPEG-2 to MPEG-4 Transcoding", Lecture notes in computer science, vol. 2195, Oct. 2001, pp. 86-93