

Research Article

Video Transcoder in DCT-Domain Spatial Resolution Reduction Using Low-Complexity Motion Vector Refinement Algorithm

Tsung-Han Tsai, Yu-Fun Lin, and Hsueh-Yi Lin

Department of Electrical Engineering, National Central University, Jhongli, Taoyuan County 32001, Taiwan

Correspondence should be addressed to Hsueh-Yi Lin, davidlin409@dsp.ee.ncu.edu.tw

Received 26 February 2008; Revised 30 June 2008; Accepted 2 September 2008

Recommended by Moon Kang

We address the topic of spatial-downscaling video transcoder in DCT-domain. The proposed techniques include the hierarchical fast motion resampling (HFMR) with accuracy motion resampling, the fast refinement for nonintegral (FRNI) motion vector (MV), and the dynamic regulating search (DRS) with low-complexity motion vector refinement. Two kinds of motion vector refinement algorithms in DRS are designed for different architectures and applications. Based on brute-force motion compensation in DCT-domain (MC-DCT), FRNI can provide better quality than nonrefine MV and reduce the complexity. DRS can utilize the filter for half-pixel MV in MC-DCT and it is an efficient method for extracting MC-DCT block to improve the performance further. From the experiments, the proposed algorithms can improve the entire quality and also reduce the complexity for DCT-domain video transcoder.

Copyright © 2008 Tsung-Han Tsai et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

1. INTRODUCTION

Recently, the masses have brought up all kinds of multimedia services to more and more demands on digital video, such as video on demand, distance learning, and video surveillance. In these applications, compressed digital video is a major component of the multimedia data. Video coding technology makes those applications feasible. In video coding, an analog video signal is digitized and then compressed to fit into desirable bandwidth.

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 user. There exists a rich set of contents that are rapidly growing by the day. On the other hand, we have terminals with varying capabilities. Connection between the terminals requires conversions to adapt between heterogeneous network configurations and the differences between terminal constraints. Among existing techniques, video transcoding has allowed user to convert a previously compressed bit-stream into another format to meet various multimedia services. The relative work is a process of converting a previously compressed video bit-stream into

another bit-stream with a lower bitrate, a different resolution (e.g., downscaling), a different coding format (e.g., the conversion between MPEG-x and H.26x, or adding error resilience), and so forth.

H.264 and H.263 play different roles in current industry. H.264 is computationally expensive in software. Hardware realization can enhance the computation speed to achieve real-time goal. However, the real-time requirement on software is hard to achieve in H.264. If simple methods are applied, resulting performance might not be satisfactory. When real-time software is desired without considering too much performance gain, H.263 becomes an alternative. Especially in the age with wider Internet capacity, occupying larger bandwidth might be acceptable. Nowadays, most of existing mobile phones are capable of watching movies from 3GPP or MV4 format. Video clips from the phone-embedded camera are stored in 3GPP format. According to the general description, 3GPP is version of H.263 for streaming and mobile applications. Since it is widely adopted in mobile devices, transcoding between mobile phone can be conducted by H.263 transcoding.

Video transcoding has allowed users to convert a previously compressed bit-stream into another format to meet

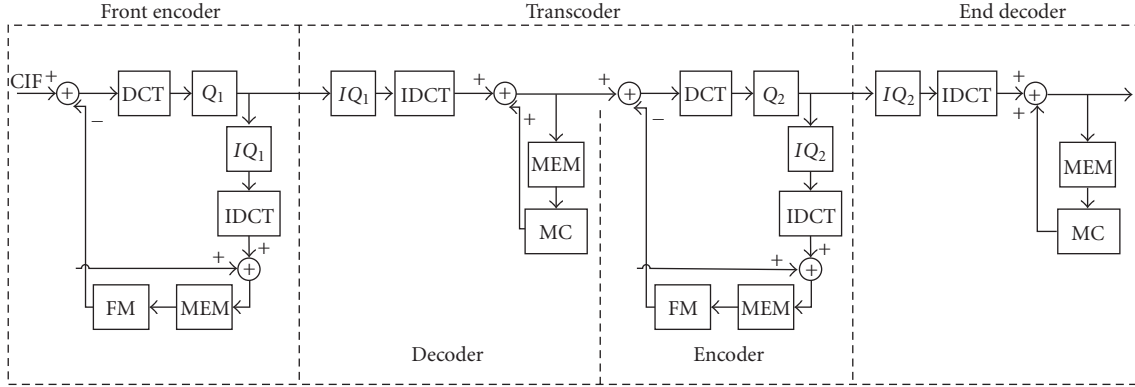


FIGURE 1: A straightforward realization of video transcoder.

$$\begin{array}{c}
 \left(\begin{array}{c}
 \left(\begin{array}{cccc}
 a_{11} + a_{12} + a_{21} + a_{22} & \cdots & \cdots & \cdots \\
 \vdots & \ddots & & \\
 & & a_{77} + a_{78} + a_{87} + a_{88} & \\
 0 & & & 0
 \end{array} \right) + \left(\begin{array}{cccc}
 b_{11} + b_{12} + b_{21} + b_{22} & \cdots & \cdots & \cdots \\
 \vdots & \ddots & & \\
 & & b_{77} + b_{78} + b_{87} + b_{88} & \\
 0 & & & 0
 \end{array} \right) + \\
 \left(\begin{array}{cccc}
 c_{11} + c_{12} + c_{21} + c_{22} & \cdots & \cdots & \cdots \\
 \vdots & \ddots & & \\
 & & c_{77} + c_{78} + c_{87} + c_{88} & \\
 0 & & & 0
 \end{array} \right) + \left(\begin{array}{cccc}
 d_{11} + d_{12} + d_{21} + d_{22} & \cdots & \cdots & \cdots \\
 \vdots & \ddots & & \\
 & & d_{77} + d_{78} + d_{87} + d_{88} & \\
 0 & & & 0
 \end{array} \right) \\
 \hline
 \begin{array}{c}
 \uparrow 8 \\
 \text{Down} \\
 \text{sampled} \\
 \text{block} \\
 \downarrow 8 \\
 \leftarrow 8 \rightarrow
 \end{array}
 \end{array}
 \right)
 \end{array}$$

FIGURE 2: Pixel averaging and down-sampling performed on 8×8 block basis.

various multimedia services. It is classified into three categories: spatial transcoding, temporal transcoding, and special application transcoding. Among those considerations, the display resolution of terminal constraint is especially important [1–7]. Therefore, we focus our attention on the problem of reduced resolution transcoding. Specifically, we consider techniques and architectures to convert a compressed video bit-stream with one spatial resolution to the output with half of original spatial resolution.

A straightforward realization of video transcoder is to cascade a decoder followed by an encoder directly, as shown in Figure 1. However, the computational complexity is time-consuming so that it is not suitable for real-time applications. To avoid drift error [2] and reduce computational complexity, the close loop DCT-domain architecture becomes the main stream in video transcoding. The DCT-domain approach reduces computational complexity by 40% than the pixel-domain one, meanwhile preserving comparable picture quality with little degradation [3]. To construct the DCT-domain architecture, different approaches are applied, such as those in [5–7].

In this article, the novel video transcoder method in DCT-domain spatial resolution reduction is proposed. It includes the fast motion resampling method, called Hierarchical fast motion resampling (HFMR), and two motion vector refinement (MVR) algorithms, called fast refinement for nonintegral (FRNI) motion vector and dynamic regulation search (DRS). Based on brute-force motion compensation in DCT-domain (MC-DCT), FRNI can provide better quality than nonrefine motion vector and reduces the complexity. In DRS, it utilizes the filter for half-pixel motion vector in

MC-DCT [8] and efficient method for extracting MC-DCT block [9, 10] to further improve the performance. Two kinds of MVR algorithms in DRS are further designed for different architectures. This paper is organized as follows: in Section 2, a brief introduction and related works are introduced; in Section 3, our proposed algorithms are proposed; in Section 4, we show our experimental result for all proposed algorithms; and finally, our concluding remarks are given in Section 5.

2. FUNDAMENTALS IN DCT-DOMAIN VIDEO TRANSCODING

2.1. DCT-domain down-conversion

In pixel-domain transcoder, the downsampled video is composed of summation of four pixels into a new one. When DCT is removed from original coding flow, modification is necessary to achieve the same functionality. The framework of extending the simple pixel averaging and down-sampling to the DCT-domain was introduced by Chang and Messerschmitt [10] and subsequently optimized for fast processing by Merhav, Bhaskaran [11], and Merhav [12]. In DCT-domain down-sampling, pixel averaging and down-sampling are performed with the smallest unit of 8×8 block of pixels, as shown in Figure 2. The a, b, c, d in the figure indicate four neighboring blocks, respectively. Afterward, all matrices are transformed into DCT-domain.

Downscaling is separated into three stages. At the first stage, four adjacent pixels in b_i are summed up to create a new pixel. This implies that the input block is replaced by

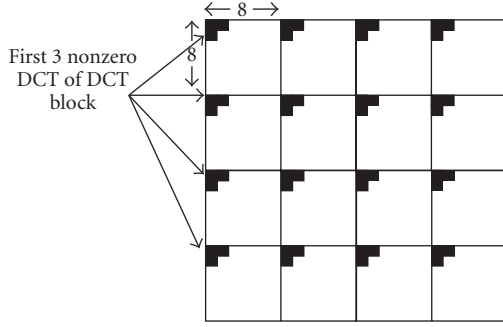


FIGURE 3: WLF-max motion vector composition algorithm.

4×4 pixels in the top left corner; and the rest of the block is padded with zeros. At the second stage, the adjacent blocks are shifted according to the location of the underlying block b_i . That is, the 4×4 pixels are shifted to the top left corner in b_1 and to the top right in b_2 and so forth. Finally, the four new blocks are added and divided by four to generate the down-sampled block b . These steps are formulated by (1). In the figure, Q_1 and Q_2 are defined by (2) and (3). However, Q'_1 and Q'_2 indicate the transpose of Q_1 and Q_2 , while q_1 and q_2 are defined by two matrices, as formulated in (4) and (5):

$$B = \frac{1}{4}(Q_1 B_1 Q'_1 + Q_2 B_2 Q'_2 + Q_3 B_3 Q'_3 + Q_4 B_4 Q'_4), \quad (1)$$

$$Q_1 = \text{DCT}(q_1), \quad (2)$$

$$Q_2 = \text{DCT}(q_2), \quad (3)$$

$$q_1 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad (4)$$

$$q_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}. \quad (5)$$

Owing to the unitary property of the DCT, (1) is performed in the DCT-domain by simply transforming all participating blocks to the transform-domain. This method is hereby referred to as pixel averaging and down-sampling.

2.2. Fast motion resampling (FMR)

Fast motion resampling involves generation of new motion vector from existing 4 blocks. When the predicted motion vectors are accurate, less refinement points are searched. Lots

of algorithms are designed for accurate approximation. Brief descriptions of the algorithms are described as follows.

- (1) The average. This is the most straight-forward approach by averaging existing motion vectors, which is formulated by (6):

$$V(x) = \frac{1}{2} \times \left(\frac{1}{4} \sum_{i=1}^4 V_i(x) \right), \quad (6)$$

$$V(y) = \frac{1}{2} \times \left(\frac{1}{4} \sum_{i=1}^4 V_i(y) \right).$$

- (2) The median. Let V be the set of four adjacent motion vectors (v_1, v_2, v_3, v_4). The algorithm is to calculate the sum of the mutual distances, as described in:

$$v = \frac{1}{2} \arg \left(\min_{V \in \{v_1, v_2, v_3, v_4\}} \left(\sum_{j=1, j \neq i}^4 \|v_i - v_j\| \right) \right). \quad (7)$$

- (3) WLF-MAX. The objective is to estimate optimal motion vector with reduced complexity. Visual quality matrix (VQM) [13], as described in Table 1, is utilized to weight individual AC coefficients. To further reduce the complexity, adopted coefficients are reduced from 64 to first three nonzero coefficients with zigzag scan order. The algorithm is formulated according to:

$$\text{ACT}_i = \sum_{k=1}^4 \text{Abs}(\text{DCT}(m, n)) \times \text{VQM}(m, n), \quad (8)$$

$$i, m, n = 1, 2, 3, 4,$$

$$v = \frac{1}{2} \text{mv}_i \mid \mid \text{ACT}_i \text{ is maximum},$$

where “ v ” is the composed motion vector for down-sized video, while “ mv_i ” denotes the motion vector of block i in a macroblock, and $\text{DCT}(m, n)$ denotes the DCT coefficient in the m th row and n th column, relative to VQM matrix. Overall process is illustrated in Figure 3.

- (4) ACT-weighted. In this method, the distance between each vector and the rest is calculated as sum of the activity-weighted distances by (9). The activity is the squared or absolute sum of DCT coefficients, or simply the DC value. The work in [14] adopts the squared sum to measure the activity. Optimal motion vector is obtained with the least distance from all

$$d_i = \frac{1}{\text{ACT}} \sum_{j=1, j \neq i}^4 \|v_i - v_j\|. \quad (9)$$

For composing the new motion vectors to the down-sized version, the four techniques were compared in [14]. It turns out that the ACT-weighted scheme outperforms the other techniques.

TABLE 1: Visual quantization matrix for images.

Row/Col	1	2	3	4	5	6	7	8
1	1	0.9962	0.985	0.9668	0.9423	0.9122	0.8775	0.8393
2	0.9944	0.9906	0.9795	0.9615	0.9372	0.9073	0.873	0.8351
3	0.9778	0.9741	0.9633	0.9458	0.9221	0.8931	0.8596	0.8227
4	0.9511	0.9476	0.9373	0.9206	0.8979	0.8701	0.8381	0.8026
5	0.9158	0.9125	0.9028	0.8871	0.8658	0.8396	0.8094	0.7759
6	0.8734	0.8704	0.8615	0.8469	0.8272	0.8029	0.7748	0.7437
7	0.826	0.8232	0.8151	0.8018	0.7837	0.7615	0.7358	0.7072
8	0.7752	0.7727	0.7654	0.7534	0.7371	0.7171	0.6937	0.6678

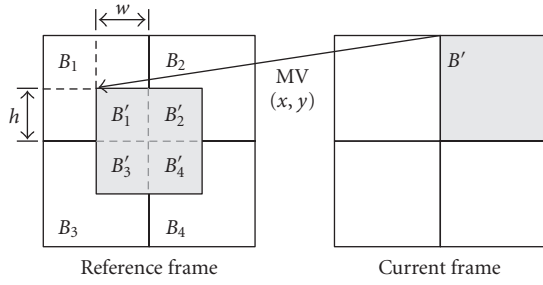
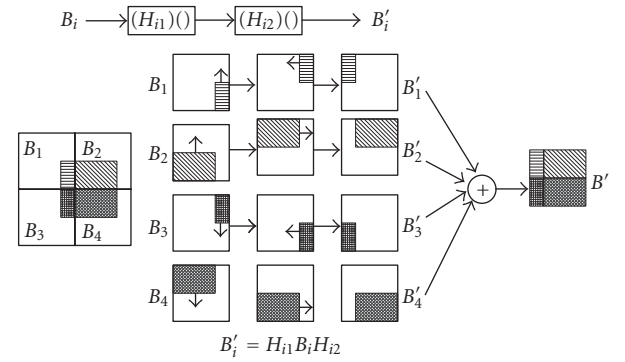


FIGURE 4: DCT-domain motion compensation.

FIGURE 5: A new image block, B' , consisting of contributions (B'_1 , B'_2 , B'_3 , and B'_4) from four original neighboring blocks (B_1 , B_2 , B_3 , and B_4).

2.3. DCT-domain motion compensation (MC-DCT)

MC-DCT is the process to manipulate motion compensation in DCT-domain approaches. From Figure 4, a motion vector specified as (x, y) means that the block B' is predicted from the reference block with corresponding displacement. The reference block occupies four blocks in the previous frame. Thus, proper manipulation is applied to obtain the prediction/compensation content B .

It is separated into three steps, as illustrated in Figure 5. The first step is to extract corresponding region from reference blocks. The second is to shift the pixels to respective locations. Finally, the results are summed up to obtain the prediction content. According to the shift distance, corresponding matrices are applied (H_{i1} and H_{i2}) to obtain the final result. H_{i1} is used for horizontal translation, and H_{i2} is used for vertical translation. Since four blocks are involved in composing single prediction block, four matrices, as described in Table 2, are obtained. However, I_w and I_h are identity matrices with size $w \times w$ and $h \times h$, respectively, while h is number of rows extracted, and w is the number of rows extracted.

DCT operation means to remove DCT with proper modifications. According to the derivation in (10), the matrices H_{i1} and H_{i2} are transformed and stored first:

$$\begin{aligned} \text{DCT}(B'_i) &= \text{DCT}(H_{i1} \cdot B_i \cdot H_{i2}) \\ &= \text{DCT}(H_{i1})\text{DCT}(B_i)\text{DCT}(H_{i2}). \end{aligned} \quad (10)$$

TABLE 2: Matrices of H_{i1} and H_{i2} .

Subblock	Position	H_{i1}	H_{i2}
B_1	Low right	$\begin{pmatrix} 0 & I_{h1} \\ 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 \\ I_{w1} & 0 \end{pmatrix}$
B_2	Lower left	$\begin{pmatrix} 0 & I_{h2} \\ 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & I_{w2} \\ 0 & 0 \end{pmatrix}$
B_3	Upper right	$\begin{pmatrix} 0 & 0 \\ I_{h3} & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 \\ I_{w3} & 0 \end{pmatrix}$
B_4	Upper left	$\begin{pmatrix} 0 & 0 \\ I_{h4} & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & I_{w4} \\ 0 & 0 \end{pmatrix}$

Afterward, simple matrix multiplication is applied to allocate part of predicted coefficients. Since B' is the summation of B'_i , overall prediction coefficients are formulated in:

$$\begin{aligned} \text{DCT}(B') &= \text{DCT}(B'_1 + B'_2 + B'_3 + B'_4) = \sum_{i=1}^4 \text{DCT}(B'_i) \\ &= \sum_{i=1}^4 \text{DCT}(H_{i1})\text{DCT}(B_i)\text{DCT}(H_{i2}). \end{aligned} \quad (11)$$

The block numbering order is shown in Figure 4.

Since matrix multiplications are required for prediction, efficient search algorithm becomes a key during refinement.

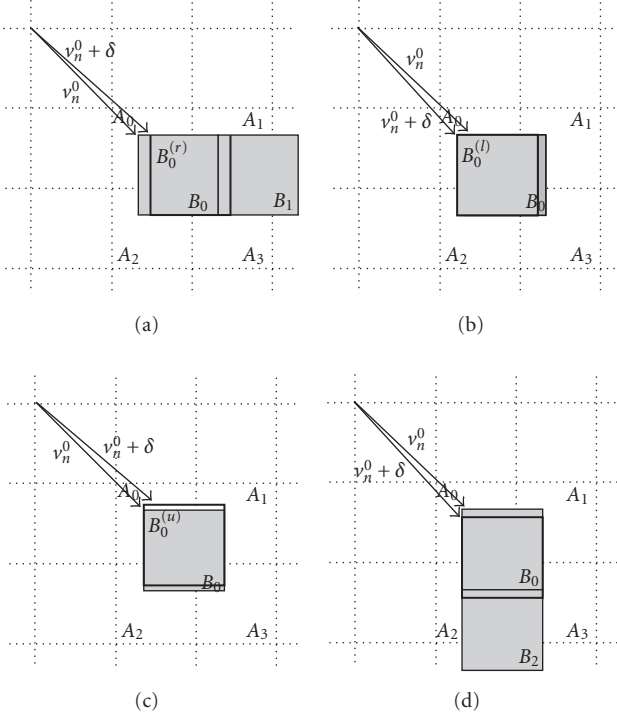


FIGURE 6: Overlap property of MC block predicted by $+\delta$, where $\delta = \{(1, 0), (-1, 0), (0, 1), (0, -1)\}$. (a) Overlap property of B_0 predicted by $v_n^0 + (1, 0)$. (b) Overlap property of B_0 predicted by $v_n^0 + (-1, 0)$. (c) Overlap property of B_0 predicted by $v_n^0 + (0, 1)$. (d) Overlap property of B_0 predicted by $v_n^0 + (0, -1)$.

2.4. Efficient method for extracting MC-DCT block in MVR

To apply the criterion function of (17), the motion compensated DCT macroblock with motion vector of $(v_n^0 + \delta)$ is to be extracted from the reference frame. Let the MC-DCT macroblock \widehat{M}_n (predicted by v_n^0 of $\widehat{\Omega}_n$) be composed of four blocks $\widehat{B}_0, \widehat{B}_1, \widehat{B}_2,$ and \widehat{B}_3 according to relative locations (top left, top right, bottom left, bottom right), respectively. The extraction is to find the MC-DCT block from the intersecting blocks A_i ($i = 0, 1, 2, 3$) pointed by a motion vector (MV) in the reference frame \widehat{f}_{k-1} . One possible solution is illustrated in (11). However, it is too heavy for real-time application.

The work in [9] exploited the overlapping property of consecutive prediction. As shown in Figure 6(a), the MC-DCT block, predicted by $v_n^0 + (1, 0)$, is displaced from B_0 by one pixel in the right. The superscript “ r ” in \widehat{B}_0^r denotes the right displacement by one pixel. (Similarly, the superscripts “ l ,” “ u ,” and “ d ” denote the left, upward, and downward displacement by one pixel, resp.) Thus, \widehat{B}_0^r overlaps B_0 by 8×7 pixels and B_1 by 8×1 pixels (Figure 6(a)). To extract \widehat{B}_0^r , (13) is proposed, where

$$R = \begin{pmatrix} 0 & 0 \\ I_7 & 0 \end{pmatrix}, \quad S = \begin{pmatrix} 0 & I_1 \\ 0 & 0 \end{pmatrix}. \quad (12)$$

I_k is an identity matrix with size $k \times k$:

$$\begin{aligned} W_r &= \begin{pmatrix} 0 & T_{mx} \\ 0 & 0 \end{pmatrix}, \\ \widehat{B}_0^r &= \widehat{B}_0 \widehat{R} + \widehat{B}_1 \widehat{S}, \\ \widehat{B}_1^r &= \widehat{B}_1 \widehat{R} + \sum_{i=1,3} \widehat{P}_i \widehat{A}_i \widehat{W}_r, \\ \widehat{B}_2^r &= \widehat{B}_2 \widehat{R} + \widehat{B}_3 \widehat{S}, \\ \widehat{B}_3^r &= \widehat{B}_3 \widehat{R} + \sum_{j=1,3} \widehat{P}_j \widehat{A}_j \widehat{W}_r. \end{aligned} \quad (13)$$

For the case \widehat{B}_0^l predicted by $v_n^0 + (-1, 0)$, \widehat{B}_0^l overlaps B_0 by 8×7 pixels and partially overlaps A_0 and A_2 (Figure 6(b)). To extract \widehat{B}_0^l , [9] proposes (14), where \widehat{R}^T is $\text{DCT}(R^T)$,

$$\begin{aligned} W_l &= \begin{pmatrix} 0 & 0 \\ U_{8-mx} & 0 \end{pmatrix}, \\ \widehat{B}_0^l &= \widehat{B}_0 \widehat{R}^T + \sum_{i=0,2} \widehat{P}_i \widehat{A}_i \widehat{W}_l, \\ \widehat{B}_1^l &= \widehat{B}_1 \widehat{R}^T + \widehat{B}_0 \widehat{S}^T, \\ \widehat{B}_2^l &= \widehat{B}_2 \widehat{R}^T + \sum_{j=0,2} \widehat{P}_j \widehat{A}_j \widehat{W}_l, \\ \widehat{B}_3^l &= \widehat{B}_3 \widehat{R}^T + \widehat{B}_2 \widehat{S}^T. \end{aligned} \quad (14)$$

U_k is a matrix with size $k \times k$, where only $(0, 0)$ th component is 1 with others being zero. Similarly, [9] propose (15) and (16) to extract \widehat{B}_0^u (predicted by $v_n^0 + (0, 1)$) and \widehat{B}_0^d (predicted by $v_n^0 + (0, -1)$), respectively,

$$\begin{aligned} W_u &= \begin{pmatrix} 0 & U_{8-my} \\ 0 & 0 \end{pmatrix}, \\ \widehat{B}_0^u &= \widehat{R} \widehat{B}_0 + \sum_{i=0,1} \widehat{W}_u \widehat{A}_i \widehat{Q}_i, \\ \widehat{B}_1^u &= \widehat{R} \widehat{B}_1 + \sum_{j=0,1} \widehat{W}_u \widehat{A}_j \widehat{Q}_j, \\ \widehat{B}_2^u &= \widehat{R} \widehat{B}_2 + \widehat{S} \widehat{B}_0, \\ \widehat{B}_3^u &= \widehat{R} \widehat{B}_3 + \widehat{S} \widehat{B}_1, \\ W_d &= \begin{pmatrix} 0 & 0 \\ T_{my} & 0 \end{pmatrix}, \\ \widehat{B}_0^d &= \widehat{R}^T \widehat{B}_0 + \widehat{S}^T \widehat{B}_2, \\ \widehat{B}_1^d &= \widehat{R}^T \widehat{B}_1 + \widehat{S}^T \widehat{B}_3, \\ \widehat{B}_2^d &= \widehat{R}^T \widehat{B}_2 + \sum_{i=2,3} \widehat{W}_d \widehat{A}_i \widehat{Q}_i, \\ \widehat{B}_3^d &= \widehat{R}^T \widehat{B}_3 + \sum_{j=2,3} \widehat{W}_d \widehat{A}_j \widehat{Q}_j. \end{aligned} \quad (15)$$

For the case $\widehat{B}_1^{\text{dir}}$, where $\text{dir} \in \{r, l, u, d\}$, a similar algorithm is applied to extract the DCT block predicted by $v_n^0 + \delta$ using

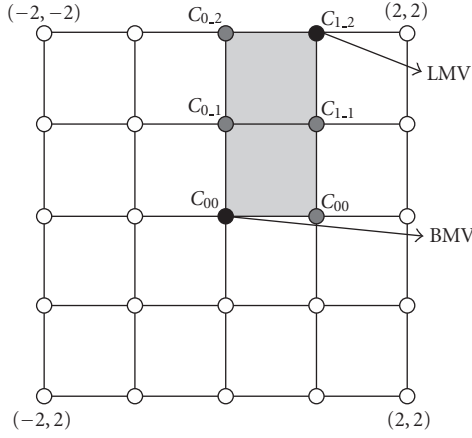


FIGURE 7: Fast searching pattern using LMV.

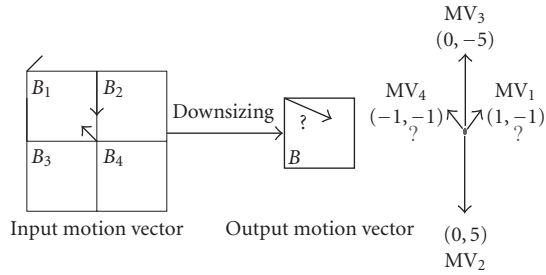


FIGURE 8: Example of the different direction motion vectors having the same value.

the overlap information. If the four intersected blocks are denoted as A_i ($i \in \{0, 1, 2, 3\}$), the required equations for extracting the desired DCT block are the second equations of (13), (14), (15), and (16). T_k is a matrix with size $k \times k$, where only $(k-1, k-1)$ th component is one and other components are zero. For the case of \hat{B}_2^{dir} , where $\text{dir} \in \{r, l, u, d\}$, the third equations are applied. For the case of \hat{B}_3^{dir} , where $\text{dir} \in \{r, l, u, d\}$, the fourth equations are applied. For the case of \hat{B}_i^{dir} , where $i \in \{1, 2, 3\}$ and $\text{dir} \in \{r, l, u, d\}$, if the displaced block is fully overlapped with previously obtained MC-DCT blocks, the form of the equation is like the first equations of (13) and (16). However, if the displaced block is partially overlapped with the intersected blocks A_i , the equation form will be like the first equations of (14) and (15). Therefore, to extract a DCT macroblock displaced by one pixel in any direction from the previously obtained DCT macroblock, the required computation is largely reduced.

2.5. DCT-domain block matching criterion

From energy conservation property, the signal energy in the DCT-domain is equal to the energy in the pixel-domain. The base motion vector results in local motion variation in current macroblock. In order to efficiently capture the variation, we define a localized search area. Since the base motion vector v_n^0 is available, block matching between the $(k-1)$ th frame (f_{k-1}) and the k th frame (f_k) amounts to

find the refinement vector δ_n for each target $\hat{\Omega}_n$ by MSE. However, DCT-domain MSE is defined by:

$$\delta_n = \arg \left(\min_{\delta \in S_L} \left(\sum_{p \in \hat{\Omega}_n} |\hat{f}_k(p) - \hat{f}_{k-1}(p + v_n^0 + \delta)|^2 \right) \right), \quad (17)$$

where \hat{f}_k is the DCT-domain version of k th frame f_k , $\hat{\Omega}_n$ is the n th DCT macroblock in \hat{f}_k , v_n^0 is the base motion vector of $\hat{\Omega}_n$, p is the position vector, δ is the delta vector, and S_L is the local search area (LSA) depending on original motion vectors. The refined motion vector for $\hat{\Omega}_n$ is defined by:

$$v_n = v_n^0 + \delta_n. \quad (18)$$

As the nonzero DCT coefficients statistically concentrate on the neighborhood of DC component, only few coefficients are considered in the new criterion. This will alleviate the burden in DCT matching.

2.6. Fast search (FS) algorithm

Seo and Kim [9] proposed that base motion vector from the median method is good enough to achieve the small search window $(-2, +2)$. However, it is still not suitable for the MVR in the DCT-domain. Thus, it is highly desired to reduce the search area as much as possible. For this purpose, [9] introduces a localization motion vector (LMV). The LMV is detected by calculating the average of three motion vectors except the base one. Figure 7 shows an example of the proposed fast search algorithm. In this example, the LMV points at $(1, -2)$. The shaded area is called the localized search area, which corresponds to S_L in (17). The checkpoints in the localized search area are considered for MVR.

If the LMV points at the vertical or horizontal axes, number of checkpoints is significantly reduced. In this case, a maximum of 3 points are checked for MVR. The example is shown in Figure 7, with 6 points checked. First, C_{00} is checked. C_{10} displaced from C_{00} is checked by using overlapped 16×15 pixels. Second, $C_{0,1}$ and $C_{1,1}$ are checked by using the overlapped 15×16 pixels with the obtained C_{00} and C_{10} , respectively. Similarly, $C_{0,2}$ and $C_{1,2}$ are checked by using the overlapping information with the obtained $C_{0,1}$ and $C_{1,1}$, respectively. Through extensive simulations, it was established that if LMV points outside the search window $(-2, +2)$, motion correlation between the four original MVs is low. Thus, MVR effect may be poor or meaningless. In this case, MVR is not performed. Instead, the macroblock type is determined as "INTER_4v." This macroblock type allows four motion vectors for each 8×8 block forming the 16×16 macroblock [15]. To generate four motion vectors per macroblock, the incoming motion vectors are scaled down by half to reflect the spatial resolution transcoding. Since the approach

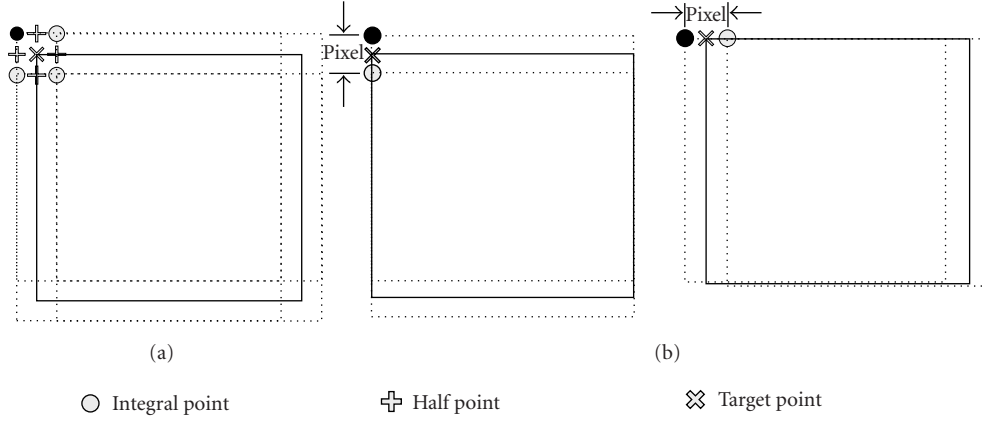


FIGURE 9: (a) MV_x and MV_y are all nonintegral (b) MV_x or MV_y is nonintegral.

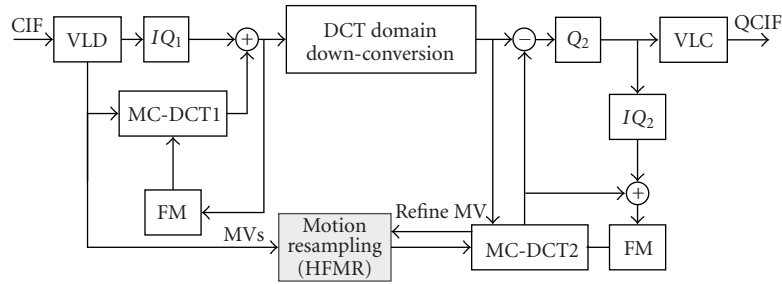


FIGURE 10: Using FRNI in CDDT.

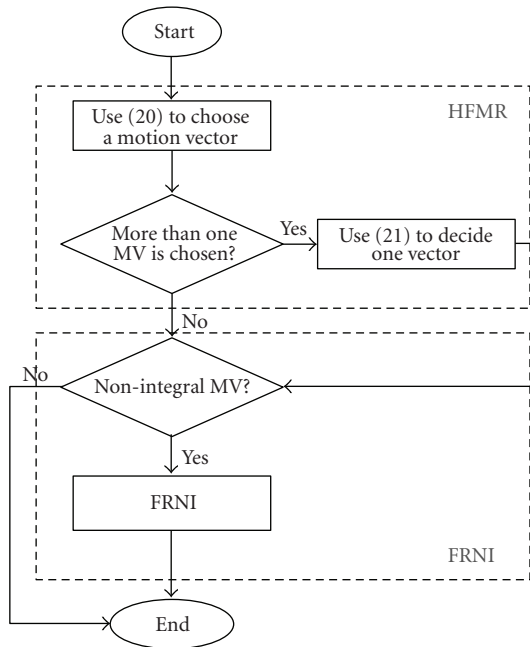


FIGURE 11: Flow chart of the entire proposed HFMR and FRNI.

can apply refinement with fewer point (compared with traditional refinement), it is thereby referred to as fast search (FS).

3. PROPOSED ARCHITECTURES

It is obvious that three problems occurred in DCT-domain video transcoding.

- (1) It is elaborate to extract macroblock in DCT-domain.
- (2) It is difficult to refine motion vector in DCT-domain. Therefore, transcoding in DCT-domain needs more accurate motion vector than pixel domain.
- (3) However, nonrefined motion vector is not suitable for video transcoder in DCT-domain.

In order to solve these problems, we propose the following algorithms.

3.1. Hierarchical fast motion resampling (HFMR)

Fast motion resampling (FMR) is always performed by simple operations (such as average, median filtering, and weighting) to reduce computation complexity and get new motion vector. It is based on the property of motion vectors and macroblock activity. Among those operations, median filtering can reach general performance for all sequences as in:

$$v = \frac{1}{2} \arg \min_{\bar{v}_i \in \{v_1, v_2, v_3, v_4\}} \sum_{j=1, j \neq i}^4 \|v_i - v_j\|. \quad (19)$$

However, transcoding in DCT-domain needs more accurate motion vector than pixel-domain and different direction

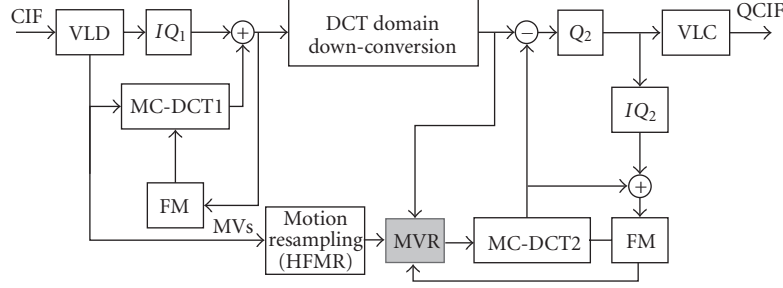
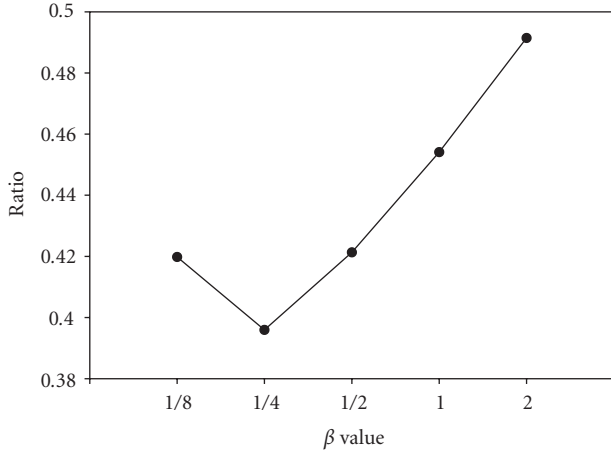


FIGURE 12: The architecture of CDDT with MVR.

FIGURE 13: β and ratio.

motion vectors have the same value under usual circumstances. For example, considering four motion vectors: $(1, -1)$, $(0, 5)$, $(-1, -1)$, and $(0, -5)$, both motion vectors $(1, -1)$ and $(-1, -1)$ have the minimum value as shown in Figure 8. Therefore, we must further evaluate two motion vectors.

The number of nonzero coefficient is related to the residue energy. When fewer nonzero DCT coefficients are presented, less residue energy indicates that the predicted motion vector is more accurate. According to the observation, the number of nonzero DCT coefficients is applied to decide motion vector in this situation. As in (20), we choose MV by minimum value of A_{vj} . However, v_j is detected by (19), and all v_j are of the equivalent minimum value. A_{vj} denotes the number of nonzero DCT coefficients in macroblock and detected by v_j :

$$v = \frac{1}{2} \arg \min_{v_j \in \{v_1, v_2, v_3, v_4\}} A_{vj}. \quad (20)$$

3.2. Fast refinement for nonintegral MV (FRNI)

As mentioned above, nonrefined motion vector is not suitable for video transcoder in DCT-domain. However, it is difficult to refine motion vector in DCT-domain. In MC-DCT, we have to extract more than one block for nonintegral

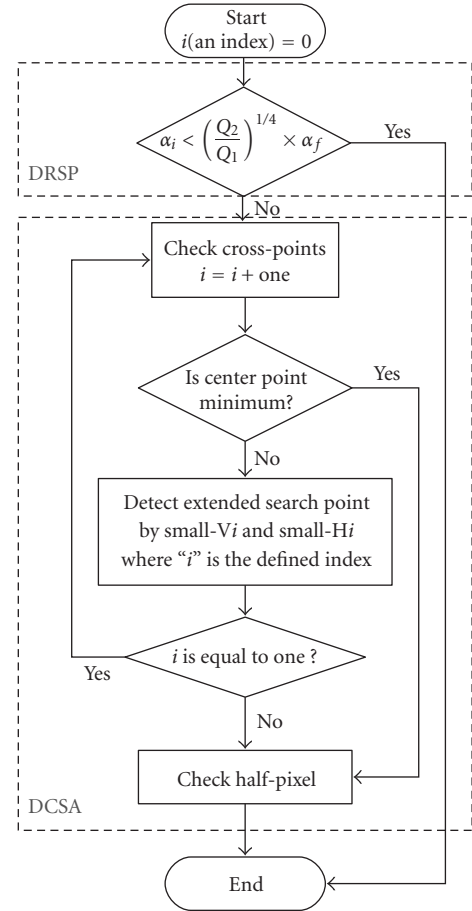


FIGURE 14: Flow chart of the proposed DRS.

motion vector which generated from FMR. As shown in Figure 9, we must extract all blocks of integral point to compose the block of target points which are detected by motion vector. Unfortunately, this motion vector generated by FMR is not always accurate enough.

Therefore, we propose the fast algorithm, FRNI motion vector, to only refine nonintegral motion vector by extracted block. Our proposed FRNI is based on cascaded DCT-domain transcoder (CDDT) shown in Figure 10. The main concept is data reusing in MC-DCT to refine the generated half-pixel motion vector. The difference between the data

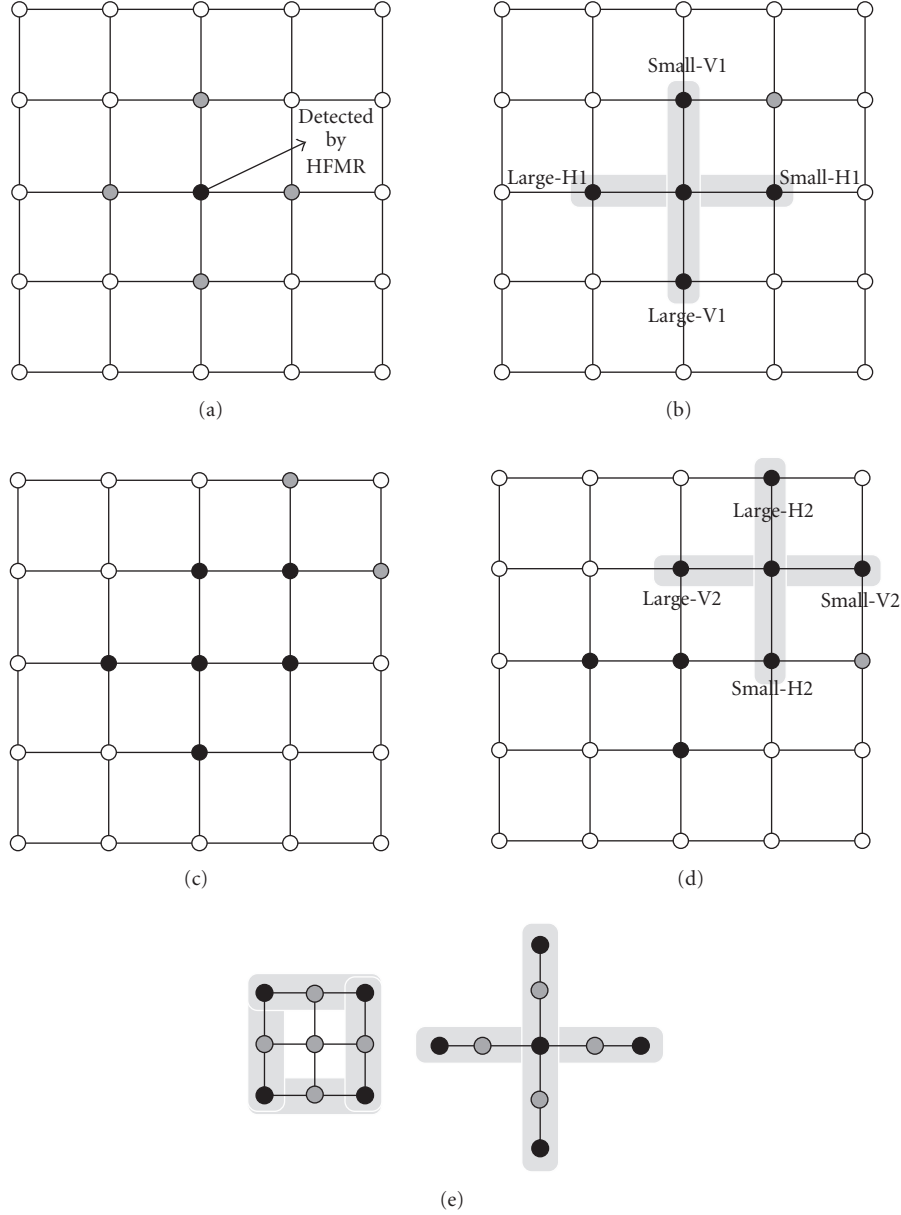


FIGURE 15: Steps in double cross-search algorithm: (a) Step 1, (b) Step 2, (c) Step 3, (d) Step 4, (e) Step 5 of double cross-search algorithm.

reuse in [9] and the approach is from the fact that the basic motion vector decision, as called resampling, is different.

Due to this concept, we can get more useful data from MC-DCT without paying any additional operation. Furthermore, the proposed algorithm is easily combined with conventional architecture to get a more efficient architecture. Based on our analysis, the computational complexity of extracted DCT block is larger than the criterion function. Therefore, it is essential to utilize the extracted blocks efficiently. As shown in Figure 9, we realize that there are nine additional checkpoints if MV_x and MV_y are all nonintegral in Figure 9(a), and three additional checkpoints if MV_x or MV_y is nonintegral in Figure 9(b). Because the blocks of integral points have been extracted, we can

obtain the additional checkpoints directly or by computing the average from extracted blocks. The black point in Figure 9 is decided by rounding nonintegral motion vector. Afterward, we use the absolute sum of DCT coefficients to determine the refined motion vector in (21), where MV_{offset} is the offset motion vector, S is the checkpoint detected by original motion vector, δ is the current checkpoint, MB_δ is the residue block detected by δ , a is the refinement motion vector distance, and $MB_{\delta-a}$ is the extracted block from DCT-domain with refined motion vector. The refined MV is defined as $MV_{\text{Refined}} = MV_{\text{non-refined}} + MV_{\text{offset}}$:

$$MV_{\text{offset}} = \arg \min_{\delta \in S} \sum_{a=0}^3 \sum_{i=0}^{63} \| \text{abs}(MB_{\delta-a}(\text{DCT}_i)) \|. \quad (21)$$

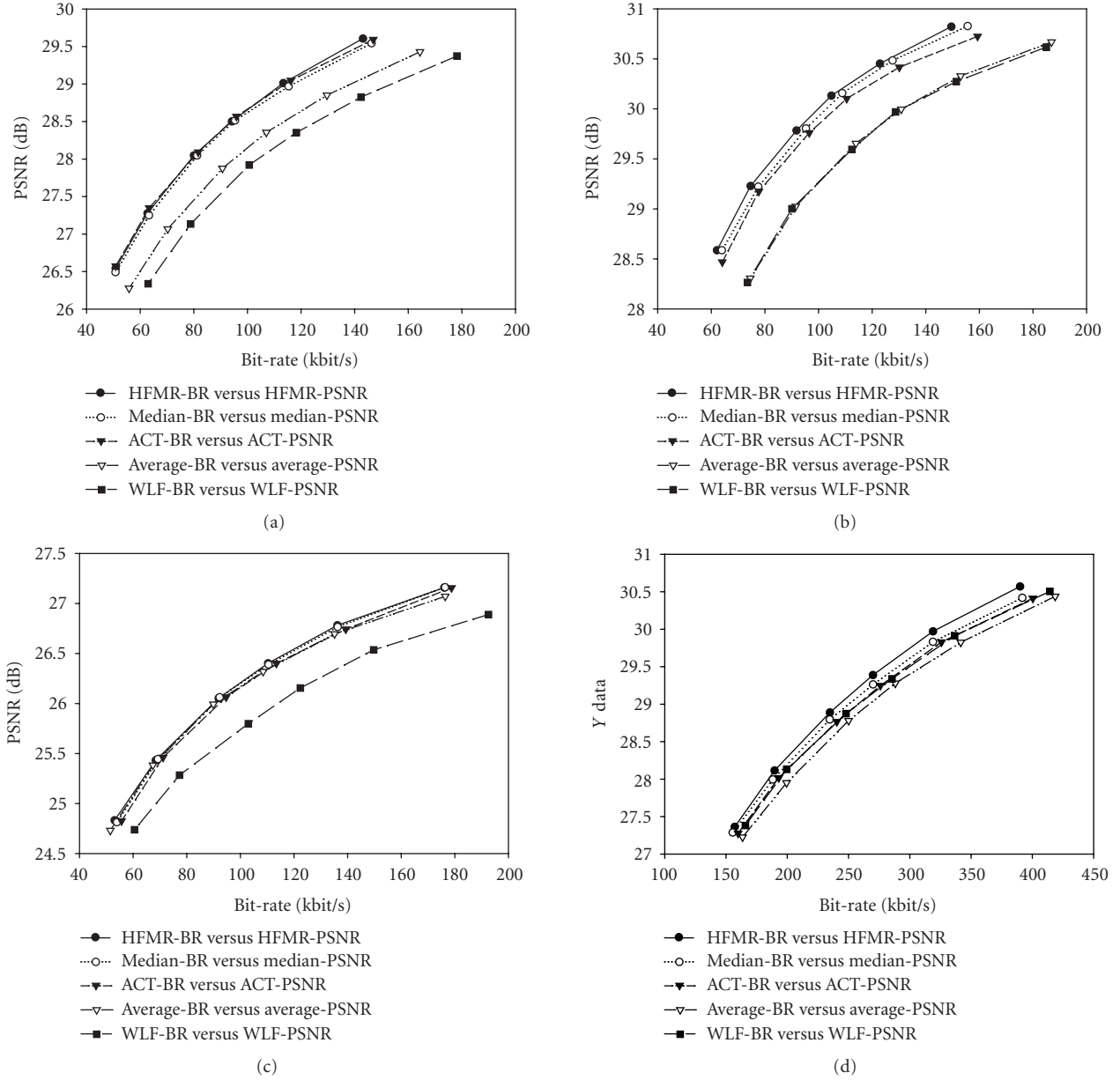


FIGURE 16: The R-D curves for different FMR algorithm. (a) Foreman. (b) TableTennis. (c) Coastguard. (d) Football.

By using FRNI, we can get three advantages.

- (1) Considering the rate distortion, we can obtain more suitable motion vector.
- (2) FRNI does not increase the computation complexity for extracting MBs in DCT-domain.
- (3) MV refinement can be separated into integer and half-pixel process. Since half-pixel refinement can be achieved by the integer search with some additions and decisions (based on the definition of FRNI), the complexity on MC-DCT can be reduced. This is obvious in DCT-domain since the compensation and prediction cannot be preformed directly from simple arithmetic.

The flow chart of the entire proposed algorithm is shown in Figure 11. It is constructed by two individual parts. For HFMR, it provides more accurate motion vector. In FRNI, we can get more suitable refined motion vector and reduce complexity of nonintegral motion vector in MC-DCT. Furthermore, the flow chart of the entire proposed algorithm is only performed in luminance component. As chrominance components, the coded type is decided based on its corresponding luminance component.

3.3. Dynamic regulating search (DRS)

According to experiment and analysis, the fast search algorithm [9] can control search range in $(-2, 2)$ efficiently. However, the complexities of MVR in different bit-streams

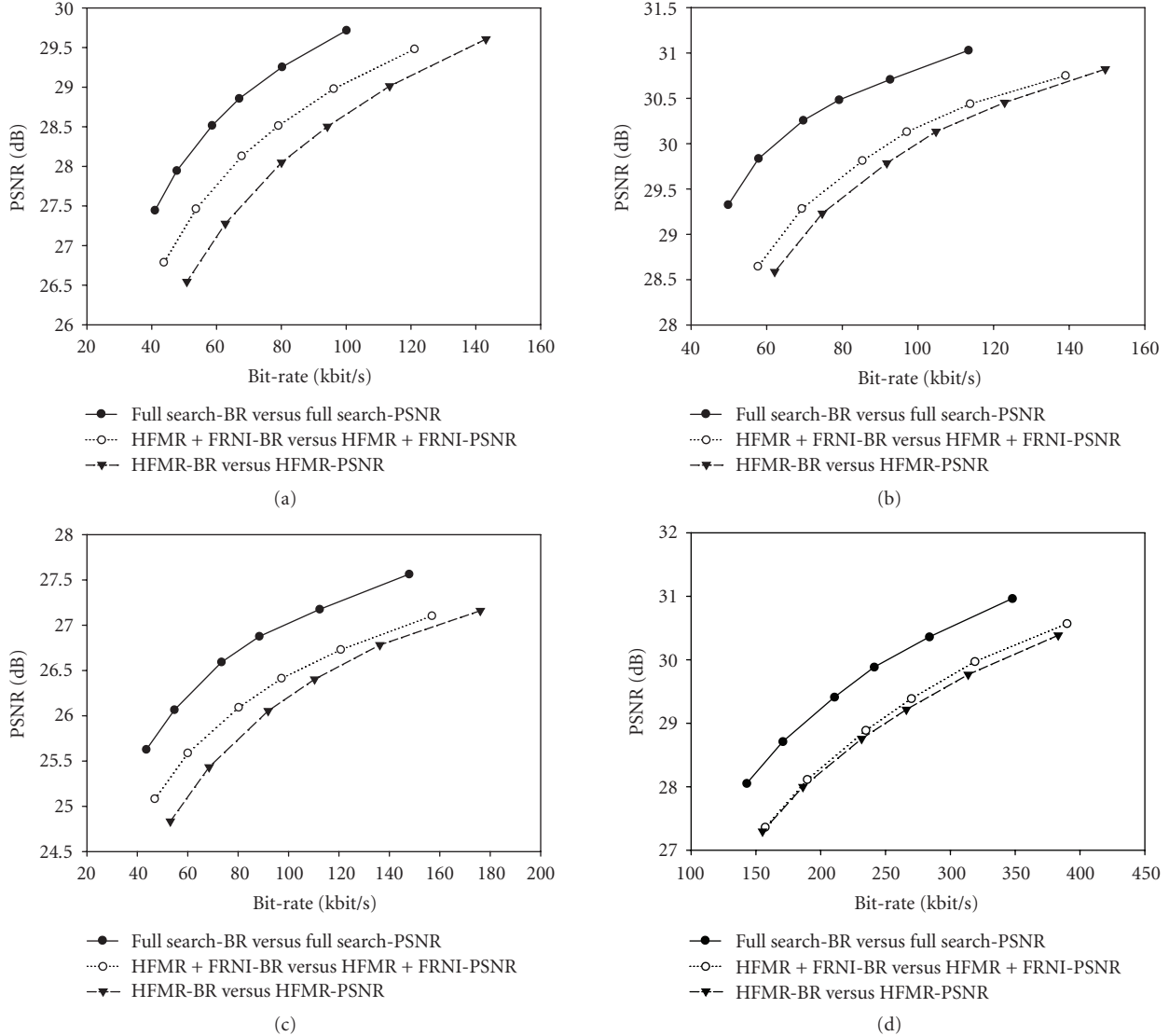


FIGURE 17: The R-D curves for our proposed FRNI. (a) Foreman. (b) TableTennis. (c) Coastguard. (d) Football.

are the same, since the search range is only decided by four input motion vectors. In video coding technique, quantization error dominates video coding distortions in low bit-stream saturation. Therefore, it is essential to reduce search points dynamically in low bit-rate. Furthermore, we also propose a novel fast MVR algorithm, double cross-search algorithm (DCSA), to improve the performance of MVR. The proposed algorithms are based on the architecture, as shown in Figure 12.

3.3.1. Dynamic reduced search points in MVR (DRSP)

Most of the refinement algorithms try to reduce computational complexity by search range reduction or fast search algorithm. However, there are still heavy computational complexities since all the motion vectors in the frame are re-estimated in the refinement process. Therefore, we utilize the variance of α and $(Q_2/Q_1)^\beta \times \alpha_f$ to be a criterion. α_i

is the absolute sum of DCT coefficients of current block and α_f is the average value of blocks of previous frame. Q_1 is the quantization parameter of input stream and Q_2 is the quantization parameter of output stream. β is a parameter for refinement threshold definition. To investigate the effect of various β on refinement points, an experiment is conducted with 5 patterns (1/8, 1/4, 1/2, 1, 2) for β and the result is presented in Figure 13. The experiment is performed on various sequences (including *Foreman*, *Coastguard*, and *TableTennis*). Since the objective is to minimize refinement point, a criterion function is defined by:

$$\text{ratio} = \frac{(\text{refined points})_{\beta=0} - (\text{refined points})_{\beta}}{(\text{search point})_{\beta=0} - (\text{search point})_{\beta}}. \quad (22)$$

The pseudocode of DRSP is described by (23). Based on the simulation result, 1/4 is the most appropriate value for β . According to the setup for $\beta(1/4)$ and α , (23) can be

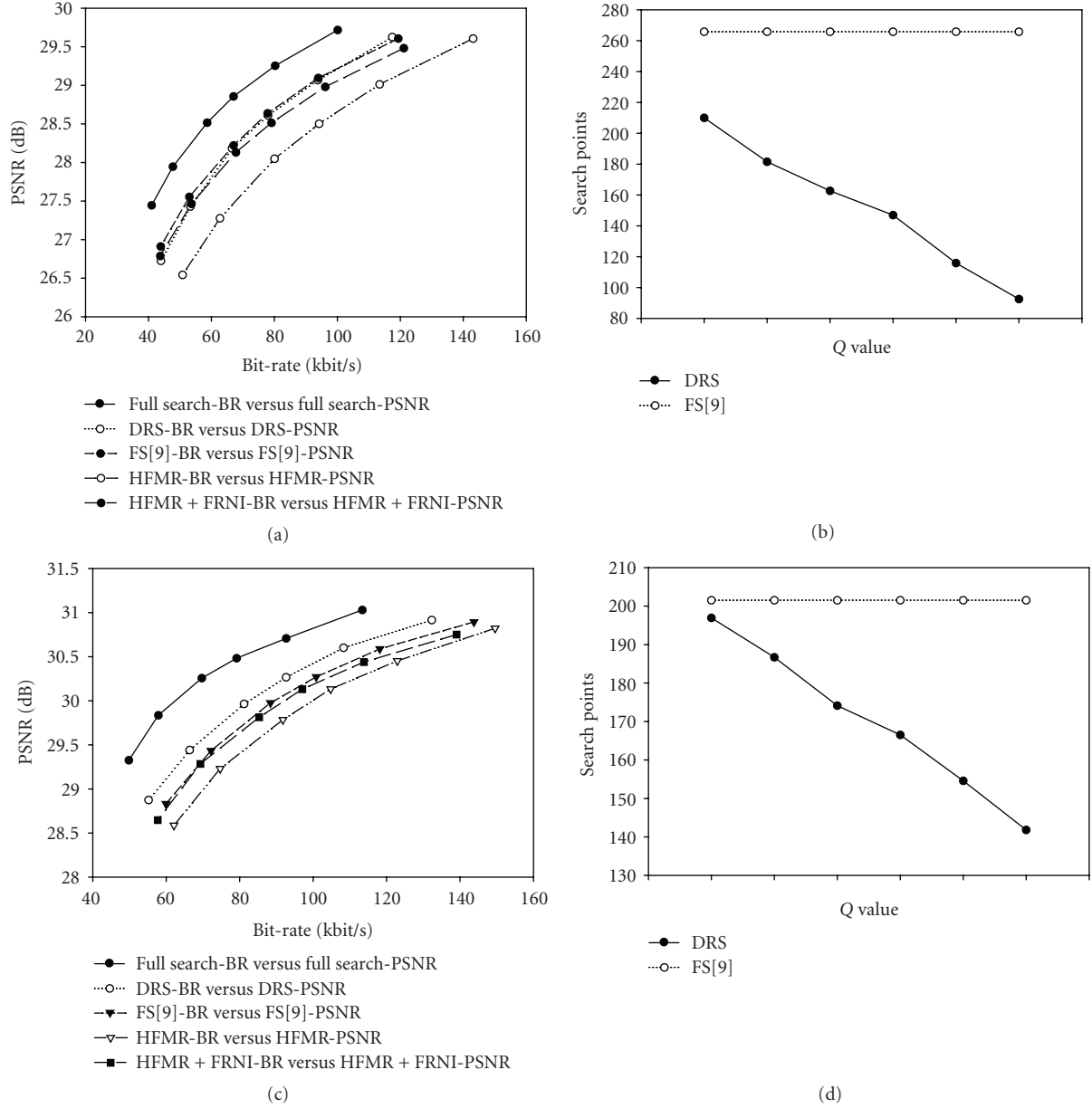


FIGURE 18: Simulation results for DRS. (a) The R-D curves of Foreman. (b) Average search points per frame for Foreman. (c) The R-D curves of TableTennis. (d) Average search points per frame for TableTennis.

implemented to fulfill the goal in search point reduction:

$$\text{if } \left(\alpha_i < \left(\frac{Q_1}{Q_2} \right)^\beta \times \alpha_f \right), \text{ then Non-refinement,} \quad (23)$$

else, Refinement.

3.3.2. Double cross-search algorithm

After DRSP, we perform double cross-search algorithm for refined blocks. Overall flowchart is presented in Figure 14. When the criterion is matched in DRSP, double cross-search is performed. Initial search is performed on a “cross-like” area, and the shape is demonstrated in Figure 15(a). At the same time, an index “ i ” is defined and set to one. The index

is to number the iteration. If cost function is minimal at the center, refinement is stopped. Else, the search point is expended by the cross shape based on smaller horizontal and vertical motion vector, defined as small-H1 and small-V1. The refinement process repeated if the extension is performed. Since the loop is terminated when index “ i ” is equal or larger than 2, maximum iteration number is 2.

The following is an example for the proposed algorithm. *Step 1.* We use (21) to check the 4-neighboring points (up, down, left, and right). If the checkpoints are smaller than the center point, we expand search point in Step 2; otherwise, stop the search (Step 5). The process is shown in Figure 15(a).

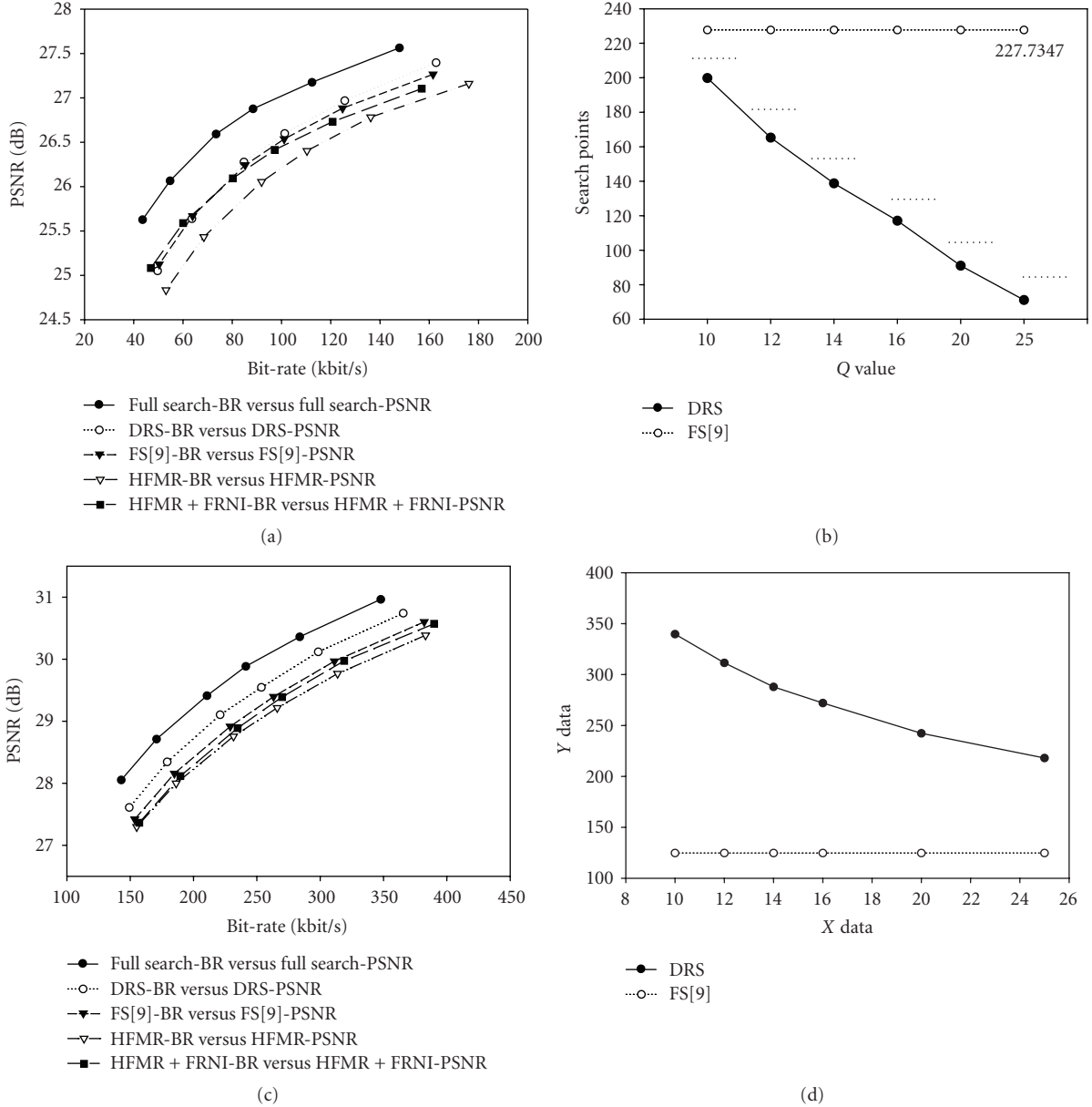


FIGURE 19: Simulation results for DRS. (a) The R-D curves of Coastguard. (b) Average search points per frame for Coastguard. (c) The R-D curves of Football. (d) Average search points per frame for Football.

Step 2. The expanded point is decided by Small-V1 and Small-H1. If the expanded point is smaller than Small-V1 and Small-H1, we expand search points in Step 3 further; otherwise, stop the search (Step 5). The process is as shown in Figure 15(b).

Step 3. As shown in Figure 15(c), we expand horizontal and vertical search points. If the prediction error of any checkpoint is smaller than the center one, we expand search point in Step 4; otherwise, stop search (Step 5).

Step 4. The expanded point is decided by Small-V2 and Small-H2, as shown in Figure 15(d).

Step 5. We perform search in half-pixel according to final search point. As shown in Figure 15(e), there are two possible patterns.

In addition to improving the efficiency of search, the proposed DRS has several advantages. First, we can control search range in $(-2, 2)$ efficiently in Step 2. Second, we can search half-pixel points efficiently. Finally, we can reduce complexity further by combining with the efficient method in [9] for block extracting.

3.4. Overall design

In previous subsections, techniques are proposed including the HFMR with accuracy motion resampling, the FRNI for MV, and the DRS with low-complexity MVR. Two kinds of MVR algorithms are designed for different architectures. Based on brute-force MC-DCT, FRNI can provide better

TABLE 3: Experimental results for three simulated sequences at QP = 10.

Sequence name	FMR	Bitrate	PSNR
Foreman	HFMR	143.12	29.6063
	Median	146.31	29.5446
	ACT-weighted	146.95	29.5928
	Average	164.38	29.4295
	WLF	178.17	29.3728
TableTennis	HFMR	149.55	30.822
	Median	155.63	30.8285
	ACT-weighted	159.34	30.7248
	Average	186.92	30.668
	WLF	185	30.6177
Coastguard	HFMR	176.09	27.1596
	Median	176.41	27.1623
	ACT-weighted	178.77	27.1558
	Average	176.42	27.0714
	WLF	192.57	26.8894

TABLE 4: Reduced ratio of nonintegral in MC-DCT.

Sequence name	Number of extracted DCT blocks for nonintegral MV (4 luminance blocks and 2 chrominance blocks)		Reduced ratio
	Nonrefined	Refined	
Foreman	$4 \times 6544 + 2 \times 9114$	$4 \times 6544 + 2 \times 7220$	8.50%
Table	$4 \times 6560 + 2 \times 8976$	$4 \times 6560 + 2 \times 6696$	11.30%
Coastguard	$4 \times 6100 + 2 \times 7904$	$4 \times 6100 + 2 \times 6738$	5.8%

quality than nonrefine motion vector and reduce the complexity. DRS can utilize the filter for half-pixel motion vector in MC-DCT [8] and efficient method for extracting MC-DCT block [9] to improve the performance.

4. EXPERIMENTAL RESULTS

4.1. Experiment environment

We simulate the proposed algorithms in software by transcoding H.263 CIF (352×288) bit streams to QCIF (176 by 144). The simulated four well-known sequences are *Foreman*, *Coastguard*, *TableTennis*, and *Football*. *Foreman* and *TableTennis* is with median motion activity. *Coastguard* is with low motion activity. *Football* is with high motion activity. Since the performance is deeply related with motion activity, the sequences can show the performance under various circumstances. In order to get better comparison with the other methods, the coding style of each input stream is IPPPPP with 50 frames and 30 frames/s. The test sequences are first encoded at quantization parameter (QP) of 6. Afterward, six transcoders (with QP= 10, 12, 14, 16, 20, and 25) are applied for transcoding, respectively. The full search range is $(-2, 2)$.

4.2. Experiment result of HFMR

We compare with the other FMR algorithms, ACT-weight, median filter, average, and WLF-max. The rate-distortion (R-D) curves of test sequences, *Foreman*, *TableTennis*, *Coastguard*, and *Football*, are shown in Figures 16(a), 16(b), 16(c), and

16(d), respectively. In Table 3, we also show the average PSNR and bit-rate at QP = 10 for three sequences. The objective of HFMR is to ensure consistent quality among various video sequences. Thus, slight modification is made on conventional approaches. According to the experimental results, HFMR is with highest quality among all. The reason for similar performance is from the fact that resampling applies simple method to obtain a rough motion vector. Since it is not refined extensively, the effect on performance is limited.

4.3. Experiment result of FRNI

In this subsection, we show the experiment results of FRNI for the test sequences. It is quite clear that the performance for all test sequences (*Foreman*, *TableTennis*, *Coastguard*, and *Football*) is improved at various QPs after combining with FRNI shown in Figures 17(a), 17(b), 17(c), and 17(d), respectively.

In Table 4, we show the total numbers of extracted DCT blocks for half-pixel motion vectors and the reduced ratios. The results prove that the proposed FRNI will not induce extra computational complexity in MC-DCT. However, it can get about 8.5% of reduction with the proposed HFMR in average. Note that the reduced operation of extracted block is larger than the checked point.

4.4. Experiment result of DRS

The experiment results of DRS for four test sequences are shown in Figures 18, 19(a), and 19(c). The comparisons of

search points between [9] and DRS are shown in Figures 18, and 19(b), and 19(d). Quantization parameters for (a)–(c) and (b)–(d) are the same, as illustrated in (b)–(d). For the best case (*TableTennis*) DRS can improve performance and reduce search points in all different QPs. The reduction of search points is about 29%, especially at QP of 25. As to the normal cases, we do not only preserve the quality but also reduce 21% ~ 65% and 12% ~ 69% of search points for *Forman* and *Coastguard*, respectively. The only exception falls in the sequence *Football*. However, the performance of DRS is obviously higher than FS. From the results, proposed DRS can achieve higher performance (compared with FS) while the search point is reduced in average. When the search point is higher, performance of DRS is relatively strong compared with other algorithms.

The search point for HFMR-FRNI is smaller than DRS. However, the performance in DRS is better than HFMR-FRNI. Since it is a tradeoff between search point and performance, the adoption of specific approach is considered based on the given constraint. If faster approach is desired, HFMR-FRNI is a better choice. If better performance is desired, DRS is the best candidate.

5. CONCLUSION

This article presented several techniques for spatial resolution reduction for video transcoder in DCT-domain. For motion vector resampling, our proposed HFMR modified the median filter to improve the entire quality further. As to MVR, we proposed FRNI and DRS for different architectures. FRNI is based on brute-force MC-DCT to only refine half-pixel motion vector. DRS reduced search points dynamically and achieved efficient refinement by DRSP and DCSA, respectively.

From the experiments, we believe that the proposed HFMR outperforms the other techniques for all simulated sequences. Besides, two MVR algorithms also improved the entire quality efficiently, especially DRS. For the best case, DRS can improve performance and reduce search points in all different QPs. Compared with [9], the reduction of search points is about 29%. As to the normal cases, it does not only preserved the quality but also reduced 16% ~ 67% of search points.

REFERENCES

- [1] A. Vetro, C. Christopoulos, and H. Sun, "Video transcoding architectures and techniques: an overview," *IEEE Signal Processing Magazine*, vol. 20, no. 2, pp. 18–29, 2003.
- [2] P. Yin, A. Vetro, B. Liu, and H. Sun, "Drift compensation for reduced spatial resolution transcoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 11, pp. 1009–1020, 2002.
- [3] W. Zhu, K. H. Yang, and M. J. Beack, "CIF-to-QCIF video bitstream down-conversion in the DCT domain," *Bell Labs Technical Journal*, vol. 3, no. 3, pp. 21–29, 1998.
- [4] B. Shen, I. K. Sethi, and B. Vasudev, "Adaptive motion-vector resampling for compressed video downscaling," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, no. 6, pp. 929–936, 1999.
- [5] K.-T. Fung and W.-C. Siu, "DCT-based video downscaling transcoder using split and merge technique," *IEEE Transactions on Image Processing*, vol. 15, no. 2, pp. 394–403, 2006.
- [6] Y.-R. Lee and C.-W. Lin, "DCT-domain spatial transcoding using generalized DCT decimation," in *Proceedings of IEEE International Conference on Image Processing (ICIP '05)*, vol. 1, pp. 821–824, Genova, Italy, September 2005.
- [7] Y.-R. Lee and C.-W. Lin, "Visual quality enhancement in DCT-domain spatial downscaling transcoding using generalized DCT decimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 8, pp. 1079–1084, 2007.
- [8] G. Cao, Z. Lei, J. Li, N. D. Georganas, and Z. Zhu, "A novel DCT domain transcoder for transcoding video streams with half-pixel motion vectors," *Real-Time Imaging*, vol. 10, no. 5, pp. 331–337, 2004.
- [9] K.-D. Seo and J.-K. Kim, "Fast motion vector re-estimation for transcoding MPEG-1 into MPEG-4 with lower spatial resolution in DCT-domain," *Signal Processing: Image Communication*, vol. 19, no. 4, pp. 299–312, 2004.
- [10] S.-F. Chang and D. G. Messerschmitt, "A new approach to decoding and compositing motion-compensated DCT-based images," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP '93)*, vol. 5, pp. 421–424, Minneapolis, Minn, USA, April 1993.
- [11] N. Merhav and V. Bhaskaran, "Fast algorithms for DCT-domain image downsampling and for inverse motion compensation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 3, pp. 468–476, 1997.
- [12] N. Merhav, "Multiplication-free approximate algorithms for compressed-domain linear operations on images," *IEEE Transactions on Image Processing*, vol. 8, no. 2, pp. 247–254, 1999.
- [13] K. N. Ngan, D. Chai, and A. Millin, "Very low bit rate video coding using H.263 coder," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 3, pp. 308–312, 1996.
- [14] Y.-R. Lee, C.-W. Lin, and C.-C. Kao, "A DCT-domain video transcoder for spatial resolution downconversion," in *Proceedings of the 5th International Conference on Recent Advances in Visual Information Systems (VISUAL '02)*, vol. 2314 of *Lecture Notes in Computer Science*, pp. 207–218, Hsin Chu, Taiwan, March 2002.
- [15] A. K. Jain, *Fundamentals of Digital Image Processing*, Prentice Hall, Englewood Cliffs, NJ, USA, 1989.