Deblocking Algorithms in Video and Image Compression Coding

Wei-Yi Wei
E-mail: r97942024@ntu.edu.tw
Graduate Institute of Communication Engineering
National Taiwan University, Taipei, Taiwan, ROC

Abstract

Blocking artifact is one of the most annoying artifacts in video and image compression coding. In order to improve the quality of the reconstructed image and video, several deblocking algorithms have been proposed. In this paper, we will introduce these deblocking algorithms and classify them into several categories. On the other hand, the conventional PSNR is widely adopted for estimating the quality of the compressed image and video. However, PSNR sometimes does not reveal the quality perceived by human visual system. In this paper, we will introduce one measurement to estimate the blockiness in the compressed image and video.

1. Introduction

Block-based transform coding is popularly used in image and video compression standards such as JPEG, MPEG and H.26x because of its excellent energy compaction capability and low hardware complexity. These standards achieve good compression ratio and quality of the reconstructed image and video when the quantizer is not vey coarse; however, in very low bit rate, the well-known annoying artifact in image and video compression coding come into existence and degrade the quality seriously. This artifact is called **Blocking Artifact**, which results from coarse quantization that discards most of the high frequency components of each segmented macroblock of the original image and video frame and introduces severe quantization noise to the low frequency component. One example is shown in Fig. 1 to illustrate this.

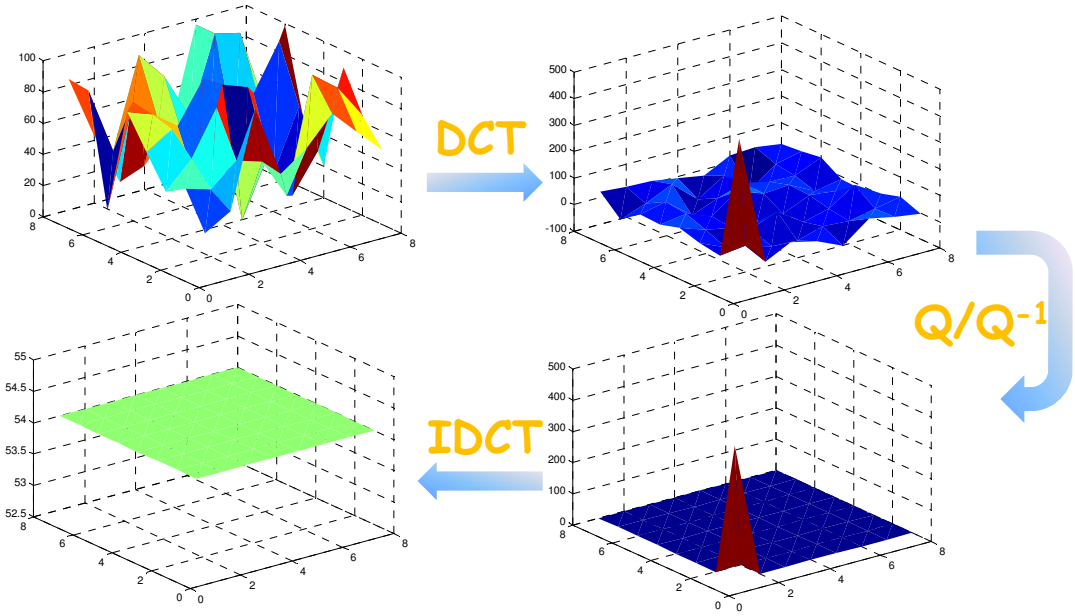


Fig. 1 The highly compressed image block

The blocking artifact incurs discontinuity across the block boundaries in the reconstructed image and video. Fig. 2 (a) and (b) shows the original image and the compressed image, respectively. As can be seen from Fig. 1, there are many square blocks in the highly compressed image.

Fig. 2 (a) The original image (b) The highly compressed image

In order to reduce the annoying blocking artifacts, several deblocking algorithms had been proposed. We can classify the deblocking algorithms into four types: in-loop filtering, post-processing, pre-processing and overlapped block methods. The in-loop filtering algorithm inserts deblocking filter into the encoding and decoding loop of the video CODEC, and one example of this method is adopted in H.264/AVC. The

post-processing algorithms apply some post-processing lowpass filters and algorithms after the image and video has been decoded to improve the image and video quality. The pre-processing algorithms pre-process the original image and video so that the quality of the reconstructed image and video can be the same as that without being processed under lower bit rate. The overlapped block methods include lapped orthogonal transform (LOT) whose transform bases are overlaid to each other and overlapped block motion compensation (OBMC) which consider the neighboring blocks for motion estimation and motion compensation in video coding.

The remained part of this paper is organized as follows. In Section 2, the observations on blocking artifacts will be introduced. In Section 3, we will introduce the in-loop filtering algorithm for removing the blocking artifact in video coding only. In Section 4, several post-processing algorithms in image and video coding will be described. In Section 5, the pre-processing method will be introduced. In Section 6, we will give a brief description of the LOT and OBMC method for reducing the blocking artifact. In section 7, we will introduce some blockiness measurement. In Section 8, we will compare the pros and cons of the algorithms introduced in this paper. In Section 9, we will give some conclusions and issues of the future word.

2. Observations of Blocking Artifacts

There are three major observations on blocking artifacts could be noted in block-based transform coding (BTC).

- **I.** Because of the masking effect of the human visual system (HVS), there are different sensitivity of the HVS to areas of the image and video with different complexity. The blocking artifacts are more noticeable in flat areas than in complex areas.
- **II.** The deblocking filter can remove some high frequency discontinuity over the block boundaries; however, it may result into blurring the real edges in the original image or video frames.
- **III.** The motion compensation prediction (MCP) propagates the blocking artifacts into the next frame in video coding.

Several deblocking algorithms are mainly based on these three observations, so we describe them in this section in advance. In the following sections, we will enter the primary issues of the deblocking algorithms.

3. In-loop Deblocking Filter

As mentioned in the previous section, we know that the blocking artifact will propagate into the later frames in video coding, which will degrade the visual quality and thus affect the compression ratio. Based on this observation, we can achieve higher compression ratio and better visual quality if we can effectively eliminate the blocking artifacts. Therefore, H.264/AVC and H.263+ add the deblocking filter into the coding loop to improve the visual quality and the accuracy of MCP. In the following two sub-sections, we will introduce the H.264/AVC in-loop filter and one novel in-loop filter in [4].

3.1 H.264/AVC In-loop Deblocking Filter

In order to enhance the visual quality and coding performance, H.264/AVC adopts the in-loop filter in its coding loop [1-3]. Fig. 3-1 shows the encoding architecture of H.264/AVC. As can be seen from the figure, the previously reconstructed frame passes the loop filter before motion estimation. Because the filtered frame is more similar to the original frame, we can obtain motion vectors with higher accuracy.

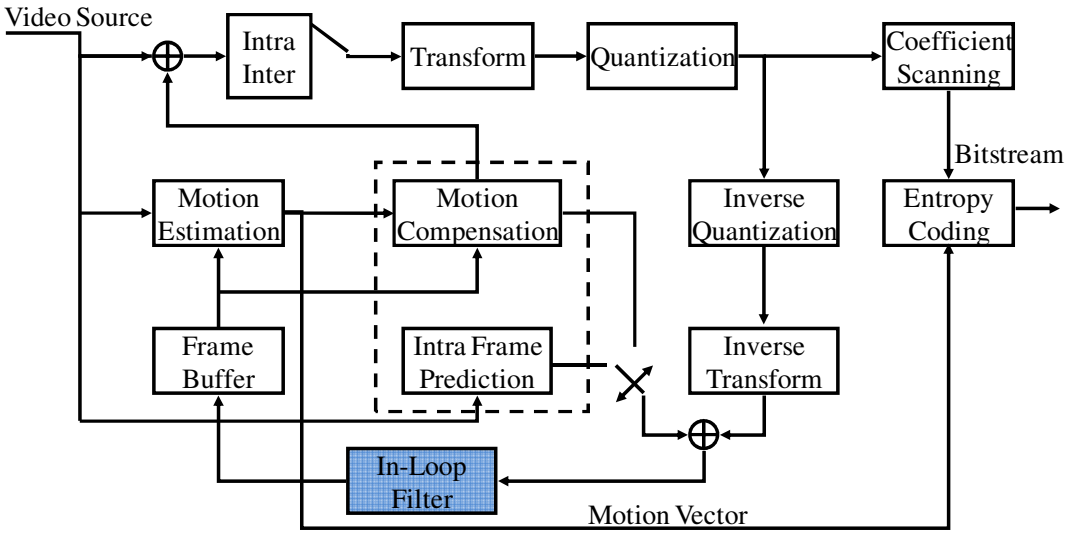


Fig. 3-1 The encoder architecture of H.264/AVC

Now we have the fundamental concept about the deblocking filter in the H.264/AVC encoder, we will briefly describe how the H.264/AVC deblocking works. Fig. 3-2 shows the model of two dequantized blocks across the boundaries in H.264/AVC. p_0, p_1, p_2 and p_3 denote the pixels in the left (top) 4x4 block while

q_0, q_1, q_2 and q_3 denote the pixels in the right (down) 4x4 block. Because only low frequency components are reserved after coarse quantization, the discontinuity, which seems to be a newly high frequency component, comes into existence across the block boundary. Therefore, we must apply the lowpass filter to discard the new high frequency components. The orange curve indicates the pixels after filtering.

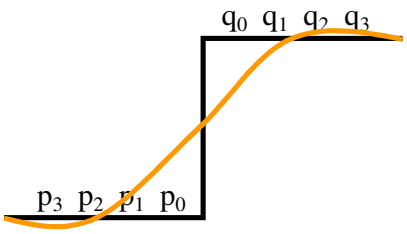


Fig. 3-2 The model of blocking artifact across the block boundary in H.264/AVC

Because there are inter-frame and intra-frame coding in video compression, H.264/AVC apply the filters with different strength based on the type of coding frames. Thus, H.264/AVC encoder must determine the boundary strength (BS) before filtering. The BS parameters are determined according to Table 3-1.

Table 3-1 The coding mode based decision for the parameter BS

Block modes and conditions	BS	Pixels to be modified
At least one of the blocks is Intra coded and the edge is a macroblock edge	4	p_0, p_1, p_2 q_0, q_1, q_2
Both of the blocks are Intra coded, but the boundary is not a macroblock boundary	3	p_0, p_1 q_0, q_1
Neither of the two blocks are intra coded, and the two blocks contain inter-coded coefficients (That is, both blocks refer to the same frame)	2	p_0, p_1 q_0, q_1
Neither of the two blocks are intra coded and inter coded	1	p_0, p_1 q_0, q_1
Otherwise	0	No filtering is applied

On the other hand, H.264/AVC defines two parameters α and β to determine what type of the filter will be applied. Because there is no room for listing all the filter coefficients of each filter, the reader can refer to [3] for more details about this.

3.2 Optimal Post-Process/In-Loop Filtering

In the previous sub-section, we introduced the H.264/AVC in-loop filter for removing the blocking artifact. However, the filter coefficients are fixed so that they may not be the best solution to all the macro blocks. In [4], Kim et al. proposed a new adaptive in-loop filtering algorithm for removing the blocking artifacts and recovering the compressed video as much as possible. Fig. 3-3 shows the overall block diagram of the encoder in [4].

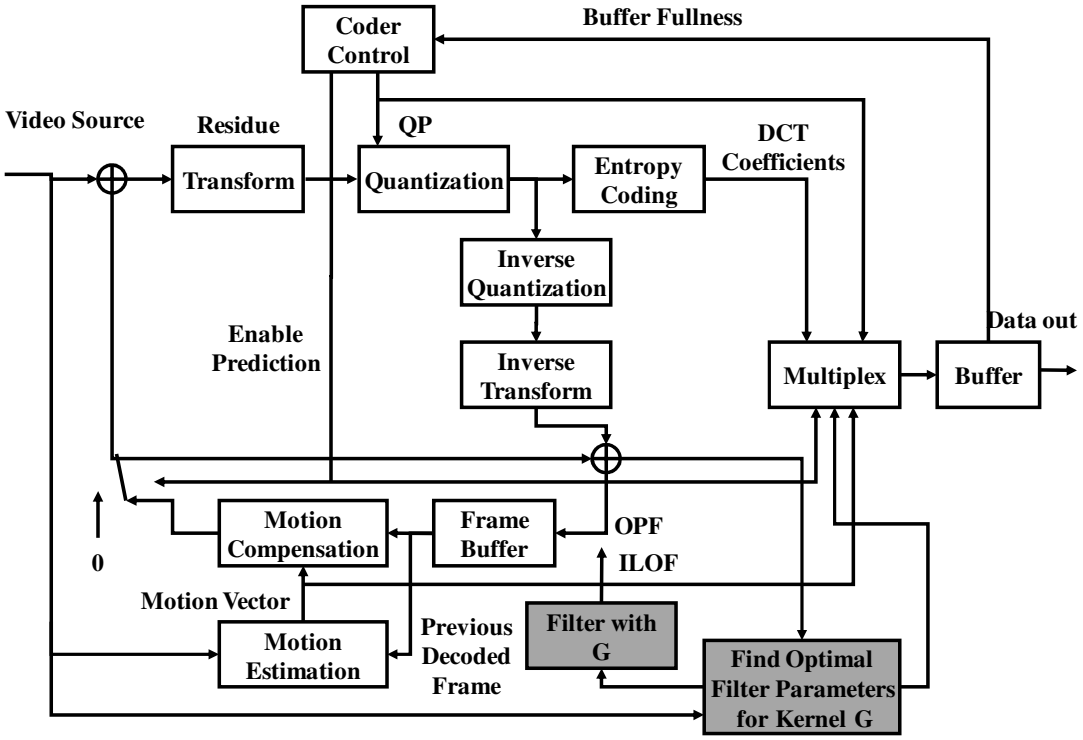


Fig. 3-3 The overall block diagram of the encoder in

In contrast to the H.264/AVC in-loop filter, the algorithm adaptively determines the filter coefficients according the new input frame. We find the optimal filter coefficients which minimize the difference between the previous reconstructed frame and the new input frame. In this approach, the in-loop optimal filter (ILOF) is applied, and the optimal filtered frames are used in the coding loop. In another approach called OPF, the optimal filtered frames are not used in the coding loop. Until now we have a brief knowledge about the overall encoder architecture in [4]. We will introduce how to obtain the filter coefficients in the remained part of this sub-section.

We denote the new input frame by **I**, which is a *W*-by-*H* matrix. The filter kernel is expressed as *l*-by-*l* matrix G and the reconstructed frame is expressed as a *W*-by-*H* matrix $\hat{\mathbf{I}}$. The relationship of these three matrixes can be expressed as

$$\mathbf{I} = \hat{\mathbf{I}} * \mathbf{G} \tag{3-1}$$

If we can find one matrix **G** satisfies (3-1), the optimal filter can be obtained. That is, the objective is to find the filter matrix satisfies

$$\arg \min_{\mathbf{G}} \| \mathbf{I} - \hat{\mathbf{I}} * \mathbf{G} \| \tag{3-2}$$

For simplicity, we convert the filter and the original image matrix to row-stacked form vectors as (3-3) and (3-4)

$$\textit{The filter kernel coefficients}: \quad \mathbf{g}(m+nl) = \mathbf{G}(m,n), \ 0 \le m,n \le l-1 \tag{3-3}$$

$$\textit{The original image values}: \quad \mathbf{b}(i+jl) = \mathbf{I}(i,j), \ 0 \le i \le W-1 \text{ and } 0 \le j \le H-1 \tag{3-4}$$

Here we define a new matrix called *Window Matrix* **A** with size *W*-by-*H*, whose element $\mathbf{a_{i,j}}$ is composed of the *l*-by-*l* matrix which is a window centered around the pixel location (i,j) of $\hat{\mathbf{I}}(i,j)$ as shown in Fig. 3-4. The *l*-by-*l* matrix is not right the element of $\mathbf{a_{i,j}}$, it must be converted to the row-stacked form vector with size l^2. In order to be consistent with the previous expression, the matrix **A** is converted to (3-5).

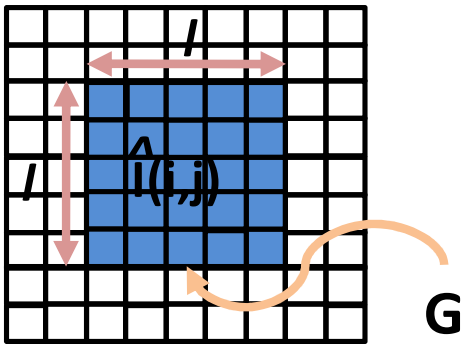


Fig. 3-4 The element of the window matrix

$$\textit{Window matrix}: \mathbf{A} = \begin{bmatrix} a_{0,0}^T \\ a_{0,1}^T \\ \vdots \\ a_{W,H}^T \end{bmatrix}_{(WH)\times(1)} \tag{3-5}$$

Now we can express (3-1) as (3-6)

$$\mathbf{Ag} = \mathbf{b} \tag{3-6}$$

In order to solve (3-6) using the linear algebra methods, we must obtain the square matrixes. Therefore, we multiply (3-6) by $\mathbf{A}^T$ in both sides and obtain the new equality (3-7)

$$\overline{\mathbf{A}}\mathbf{g} = \overline{\mathbf{b}} \tag{3-7}$$

where $\overline{\mathbf{A}}_{l\times l} = \mathbf{A}^{\mathrm{T}}\mathbf{A}$ and $\overline{\mathbf{b}}_{l\times 1} = \mathbf{A}^{\mathrm{T}}\mathbf{b}$

If the inverse matrix of $\overline{\mathbf{A}}$ can be obtained, we can find the filter coefficients g that minimize (3-2). However, the computation cost to find the filter coefficients is high because the size of the matrix $\overline{\mathbf{A}}$ is very large and it is hard to find its inverse matrix. In [4], the authors found that (3-7) can be solved by an iterative algorithm called *iterative preconditioned conjugate gradients algorithm* [5], which can save large computation. The algorithm is not introduced here, the interested reader can refer to [4] for more details.

4. Post-processing Deblocking Algorithms

Post-processing algorithms are the most popular methods for improving the quality of the image and video and eliminate the annoying blocking artifact. On the other hand, the post-processing algorithms can achieve deblocking without the original image and video, so the standard need not to be modified. In this section, we will introduce several post-processing algorithm for removing blocking artifact.

4.1 Reduction of Blocking Artifacts in DCT Domain

In this subsection, we introduce a post-processing filtering algorithm in DCT domain [6-8]. We define the block $b_{m,n}^{k,l}(u,v)$ and $B_{m,n}^{k,l}(u,v)$ first. $b_{m,n}(u,v)$ is the (m,n)-th 8x8 block in the compressed image, and $B_{m,n}(u,v)$ is the DCT coefficients of $b_{m,n}(u,v)$. $b_{m,n}^{k,l}(u,v)$ is the shifted block with displacement k pixel in the x direction and displacement l pixels in the y direction with respective to block $b_{m,n}(u,v)$, and $B_{m,n}^{k,l}(u,v)$ is the DCT coefficients of the block $b_{m,n}^{k,l}(u,v)$. One example is shown in Fig. 4-1.

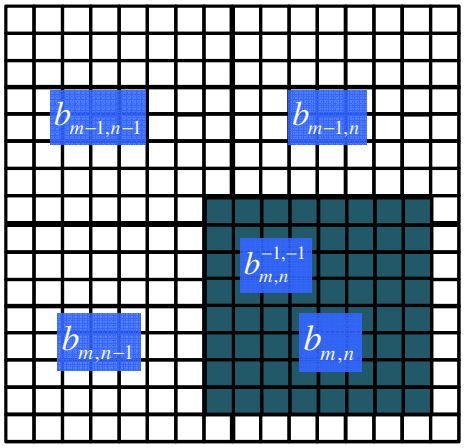


Fig. 4-1 Example of shifted block $b_{m,n}^{k,l}(u,v)$

In the original image, the neighboring DCT coefficients at the same frequency are very similar and do not vary radically within a small range. Thus, we can apply lowpass filter to the DCT coefficients at each frequency to filter the high frequency parts resulting from blocking effect. However, this method may blur the real edges in the original image, so we must have the mechanism to detect activity of the block and apply the filter with corresponding strength.

DCT-domain filtering is applied to revise the block $B_{m,n}(u,v)$ to obtain the new DCT coefficients $\hat{B}_{m,n}(i,j)$.

$$\hat{B}_{m,n}(i,j) = \frac{1}{W}\sum_{k=-h}^{h}\sum_{l=-h}^{h} w_{k,l} B_{m,n}^{k,l}(u,v) \tag{4-1}$$

$$W = \sum_{k=-h}^{h}\sum_{l=-h}^{h} w_{k,l} \tag{4-2}$$

The post-filtering works in different ways for the blocks with different activities.

For blocks with low activity, the blocking artifact is more noticeable, so we apply strong filtering to smooth the high frequency components. The filter coefficients are defined in (4-3).

$$w_{k,l} = 1,\ k,l = -2,...,2 \tag{4-3}$$

For blocks with high activity, the blocking artifact is less noticeable, so we apply filtering with less strength to smooth blocking artifact and preserve the real edge. The filter coefficients are defined in (4-4).

$$w_{k,l} = \begin{cases} 3, & \text{for } (k,l) = (0,0) \\ 1, & otherwise \end{cases} \tag{4-4}$$

There are several functions to detect the activity. However, we do not list the lengthy equations in this paper, the interested reader can refer to [6-8] for more details.

4.2 Deblocking Using Weighted Sums of Symmetrically Aligned Pixels

In [9], Averbuch et al. proposed a new deblocking using the symmetrical pixels across the block boundaries. The deblocking algorithm using weighted sums of symmetrically aligned pixels is abbreviated as WSSAP.

Denote the image with size $R \times C$ as follows

$$I = \{ p_{i,j} \},\ 0 \le i \le R\text{-}1,\ 0 \le j \le C\text{-}1 \tag{4-5}$$

We divide the input image into several 8×8 sub-blocks $B_{r,c}$

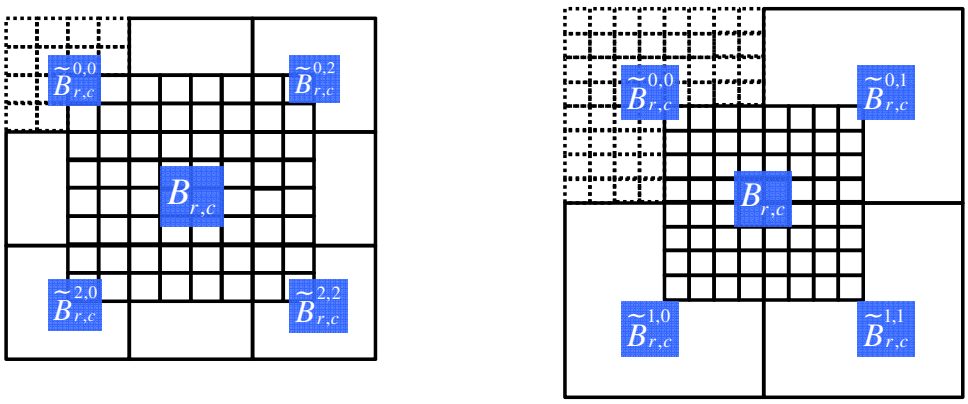
$$B_{r,c} = \{ p_{8r+i,8c+j} \}_{i,j=0,\ldots,7},\ 0 \le r \le \frac{R}{8},\ 0 \le c \le \frac{C}{8} \tag{4-6}$$

We define a *deblocking frame* $\mathbf{B}(S_f)_{r,c}$ whose size is $S_f \times S_f$.

$$\mathbf{B}(S_f)_{r,c} = \{ \tilde{B}^{m,n}_{r,c} \},\ 0 \le m \le \frac{8}{S_f},\ 0 \le n \le \frac{8}{S_f} \tag{4-7}$$

where $\tilde{B}^{m,n}_{r,c} = \{ p_{8r+(m-\frac{1}{2})S_f+i,8c+(n-\frac{1}{2})S_f+j} \},\ 0 \le i \le S_f - 1,\ 0 \le j \le S_f - 1$

Two examples of (4-6) are shown in Fig. 4-2.



(a) S_f=4 (b) S_f=8

Fig. 4-2 Two examples of the deblocking frames $\mathbf{B}(S_f)_{r,c}$

For simplicity, the $8r+(m-(1/2))S_f$ and $8c+(n-(1/2))S_f$ terms in (4-7) are ignored, and we rewrite (4-7) as (4-8).

$$\widetilde{B}_{r,c}^{m,n} = \{p_{i,j}\}_{i,j=0,\ldots,S_f'}, \text{where } S_f' = S_f - 1 \tag{4-8}$$

The pixel to be filtered is denoted by $p_{i,j}$, and the filtered pixel $p_{i,j}'$ is defined in (4-9). The pixels $p_{S_f'-i,j}$ and $p_{i,S_f'-j}$ lie symmetrically to $p_{i,j}$, and the pixel $p_{S_f'-i,S_f'-j}$ lie symmetrically to $p_{i,j}$ with respect to the center of the deblocking frame. The four weights $\alpha_{i,j}, \beta_{S_f'-i,j}, \gamma_{i,S_f'-j}$, $\delta_{S_f'-i,S_f'-j}$ are the filter coefficients.

$$p_{i,j}' = \alpha_{i,j} \cdot p_{i,j} + \beta_{S_f'-i,j} \cdot p_{S_f'-i,j} + \gamma_{i,S_f'-j} \cdot p_{i,S_f'-j} + \delta_{S_f'-i,S_f'-j} \cdot p_{S_f'-i,S_f'-j} \tag{4-9}$$

In [9], the authors obtain the filter coefficients by extending the 1D case to the 2D case. How the filter coefficients are obtain is not included in this paper, please refer to [9] for more details. In the 1D case, we have two filter coefficients only for each pixel, and the filter coefficients can be classified into linear solution and quadratic solution, which are defined in (4-10) to (4-12).

1) Linear Solution

Let the linear solution ω_L be of the form $\omega_L(x) = ax + b$. Then the solution is

$$\omega_L^{[0,k-1]}(x) = \frac{\theta - \eta}{k-1} x + \eta, \text{ if } x \in [0, k-1] \tag{4-10}$$

$$\omega_L^{[0,k-1]}(x) = \frac{\eta - \theta}{k-1} x + \frac{(2\theta - \eta)k - \theta}{k-1}, \text{ if } x \in [k, 2k-1] \tag{4-11}$$

2) Quadratic Solution

Let the linear solution ω_Q be of the form $\omega_Q(x) = ax^2 + bx + c$. Then the solution is

$$\omega_Q(x) = \frac{(\theta - \eta)}{k(k-1)} x^2 + \frac{(\theta - \eta)(1-2k)}{k(k-1)} x + \eta \tag{4-12}$$

where $k = \frac{S_f}{2}$ and $\theta \in [0,1]$

Now, we can obtain the 2D filter coefficients based on (4-10) to (4-12). The filter coefficients $\alpha_{i,j}, \beta_{S_f'-i,j}, \gamma_{i,S_f'-j}$, $\delta_{S_f'-i,S_f'-j}$ are defined in (4-13) to (4-16).

$$\alpha_{i,j} = \omega(i) \cdot \omega(j) \tag{4-13}$$

$$\beta_{S_f^{'}-i,j} = \psi(S_f^{'} - i, j) \cdot \omega(j) \tag{4-14}$$

$$\gamma_{i,S_f^{'}-j} = \omega(i) \cdot \psi(i, S_f^{'} - j) \tag{4-15}$$

$$\delta_{S_f^{'}-i,S_f^{'}-j} = \psi(S_f^{'} - i) \cdot \psi(S_f^{'} - j) \tag{4-16}$$

where $\omega(\cdot)$ and $\psi(\cdot)$ are both linear or quadratic

Until now, we obtain the filter coefficients, so we can achieve deblocking through (4-9).

4.3 Deblocking Using Offset and Shift Technique

We have introduced two post-processing deblocking algorithms. However, the computation cost of the DCT domain deblocking method is high and the WSSAP algorithm may excessively blur the image. In [10], Kim et al. proposed a new deblocking algorithm that saves large computation cost and prevents from excessively blurring the real edge.

The algorithm is described as follows. In order to check the direction the blocking artifact locates, we need some measurements. First, we define a deblocking block (DB) as a squared of adjacent pixels as shown in Fig. 4-3.

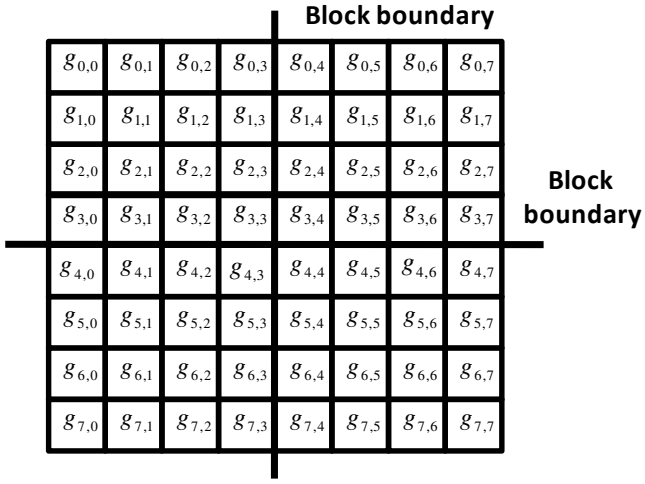


Fig. 4-3 The Deblocking Block

To classify the type of the DB, we define the horizontal activity and the vertical activity as follows:

$$ACH_H = \sum_{k=0}^{6} \Delta C_k \tag{4-17}$$

$$ACH_V = \sum_{k=0}^{6} \Delta R_k \tag{4-18}$$

where ΔC and ΔR are the difference in the horizontal and vertical direction, respectively. They are define as

$$\Delta C_k = \sum_{k=0}^{7} | q_{i,k} - q_{i,k+1} | \tag{4-19}$$

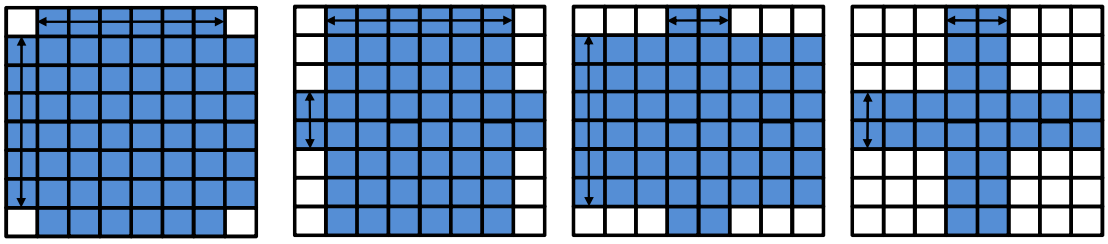
$$\Delta R_k = \sum_{k=0}^{7} | q_{k,j} - q_{k,j+1} | \tag{4-20}$$

After the activity variables are derived, we can classify the DB into four types according to Table 4-1. The term UDB indicates *Uniform Deblocking Block*. The two terms HDB and VDB indicate *Horizontal Deblocking Block* and *Vertical Deblocking Block*, respectively. The term CDB indicates *Complex Deblocking Block*.

Table 4-1 Classification of the deblocking block

	UDB	HDB	VDB	CDB
ACT_H	< T	> T	< T	> T
ACT_V	< T	< T	> T	> T

After the block classification, we apply different filtering method on each DB type as shown in Fig. 4-4.



(a)UDB (b)HDB (c)VDB (d)CDB

Fig. 4-4 Filtering direction and strength for each DB type

In order to find the amount of the modification, we define a variable “offset” as

$$offset = p_3 - p_4 \tag{4-21}$$

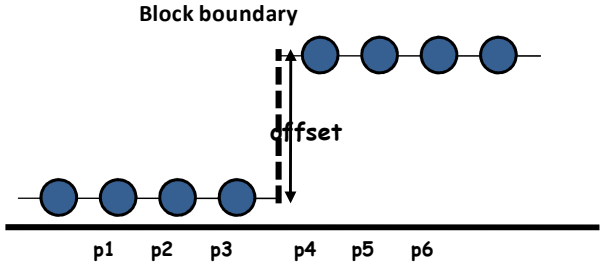
1) Filtering for Uniform Deblocking Blocks

Because the whole block is uniform, we must apply the strong filter to remove the blocking artifact. The filtered pixels are defined as (4-22) and (4-23). One example is shown in Fig. 4-5.

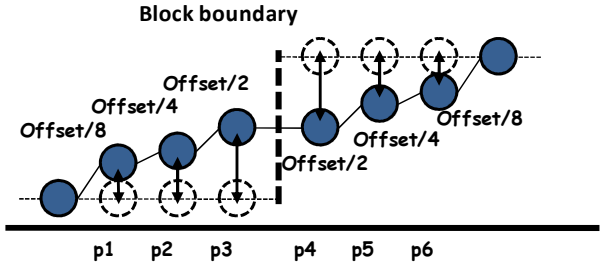
$$p_i^{'} = p_i - sign(offset) \cdot (\frac{|offset|}{\alpha_i}), \text{ for i=1,2,3} \tag{4-22}$$

$$p_i^{'} = p_i + sign(offset) \cdot (\frac{|offset|}{\alpha_i}), \text{ for i=4,5,6} \quad (4\text{-}23)$$

where $\alpha_i = \{8, 4, 2, 2, 4, 8\}$



(a)



(b)

Fig. 4-5 The one-dimensional representation of the UDB (a) Before filtering (b) After filtering

2) Filtering for Directional Deblocking Blocks

The directional deblocking block (DDB) includes the HDB and VDB.

- If both the top and bottom blocks are regarded uniform, the current block is regarded as HDB. For HDB, we perform weaker filtering along the horizontal direction and stronger filtering along the vertical direction.
- If both the left and right blocks are regarded uniform, the current is regarded as VDB. For VDB, we perform the weaker filtering along the vertical direction and stronger filtering along the horizontal direction.

The filtering with less strength is defined in (4-24) and (4-25).

$$p_3^{'} = p_3 - sign(offset) \cdot (\frac{|offset|}{4}) \quad (4\text{-}24)$$

$$p_4^{'} = p_4 + sign(offset) \cdot (\frac{|offset|}{4}) \quad (4\text{-}25)$$

3) Filtering for Complex Deblocking Blocks

If the DB is classified to CDB, we only apply weaker filtering along the block boundary and preserve the inter texture as shown in Fig. 4-4 (d). After the block boundary is filtered, a 2-D mask is used to remove blocking artifact while preserving

the real edges. The filter is defined in (4-26).

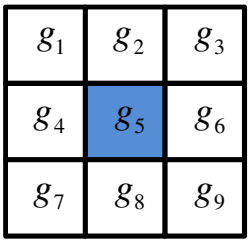


Fig. 4-6 The filter kernel

$$g_5^{'} = w_0 \cdot g_5 + w_1 \cdot \frac{g_1 + g_9}{2} + w_2 \cdot \frac{g_3 + g_7}{2} + w_3 \cdot \frac{g_2 + g_8}{2} + w_4 \cdot \frac{g_4 + g_6}{2} \quad (4\text{-}26)$$

$$w_1 = f(|\, g_1 - g_9 \,|) \quad (4\text{-}27)$$
$$w_2 = f(|\, g_3 - g_7 \,|) \quad (4\text{-}28)$$
$$w_3 = f(|\, g_2 - g_8 \,|) \quad (4\text{-}29)$$
$$w_4 = f(|\, g_4 - g_6 \,|) \quad (4\text{-}30)$$

where $f(\Delta)W = 0.25 \cdot e^{-\beta\Delta}$, where $\beta = 0.04$

5. Pre-processing Algorithms

We can improve the quality of the compressed image and video by applying the deblocking algorithms. The deblocking algorithms described above are all post-precessing methods. In [13], Wang et .al proposed a pre-processing algorithm to enhance the coding efficiency and to achieve deblocking at the same time. Fig. 5-1 shows the results of reconstructed image with and without pre-processing. Fig. 5-1(a) shows the original image S without pre-processing, its compressed frame $S^{'}$ and its reconstructed frame $S^{''}$. Fig. 5-1(b) is the frame after pre-processing S_p, its compressed frame $S_p^{'}$ and its reconstructed frame $S_p^{''}$. From the reconstructed results, we can observe that both of the two reconstructed frames $S^{''}$ and $S_p^{''}$ are very similar to the original frame S, but the data quantity of $S_p^{'}$ is much less than that of $S^{'}$.

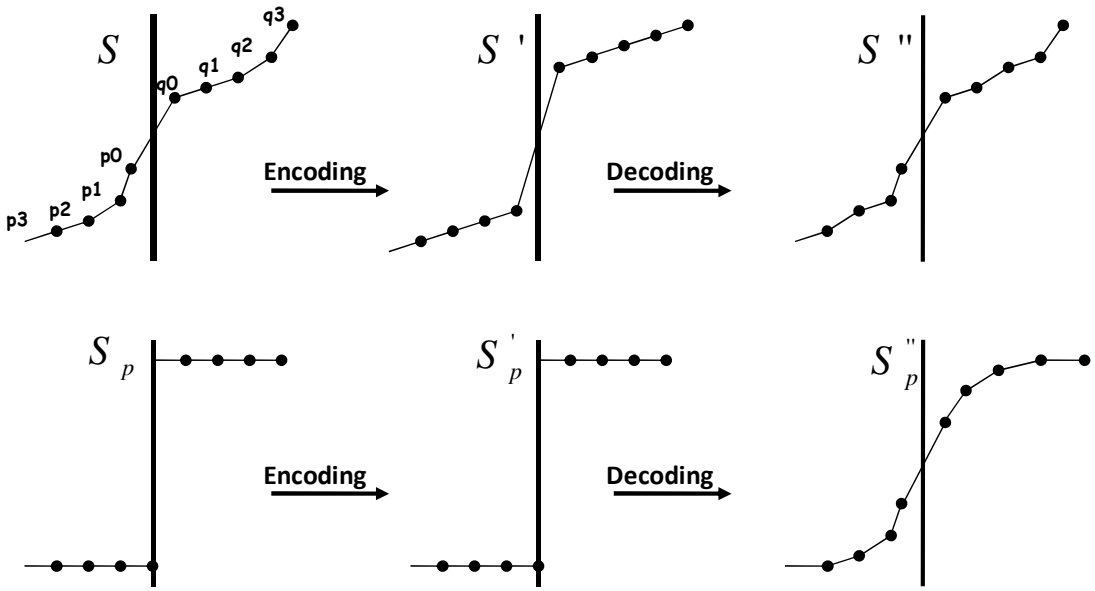


Fig. 5-1 The reconstructed block boundary with and without pre-processing

We introduce one pre-processing method combined with H.264 inloop deblocking filter. Denote the 8-point block boundary in original image as I, the block boundary after pre-processing as R and the filtered block boundary as V. The deblocking filter only works on the center four points q_0, q_1, q_2, q_3. The three block boundary is defined in (5-1) to (5-3).

$$\text{Original(I)}: (p_3, p_2, p_1, p_0, q_0, q_1, q_2, q_3) \tag{5-1}$$

$$\text{Pre-processed(R)}: (p_3^{'}, p_2^{'}, p_1^{'}, p_0^{'}, q_0^{'}, q_1^{'}, q_2^{'}, q_3^{'}) \tag{5-2}$$

$$\text{Filtered(V)}: (p_3^{'}, p_2^{'}, p_1^{''}, p_0^{''}, q_0^{''}, q_1^{''}, q_2^{'}, q_3^{'}) \tag{5-3}$$

Our objective is to minimize the difference between I and V. We define the difference ε in (5-4), and we can obtain the minimum by take the derivative of ε.

$$\varepsilon = \|V - I\| \tag{5-4}$$

The function have four variables $Q = (q_0^{'}, q_1^{'}, q_2^{'}, q_3^{'})$, and the equation is expressed as (5-5).

$$\frac{\partial \varepsilon}{\partial Q} = \frac{\partial \|V - I\|}{\partial Q} = 2(V - I)\left[\frac{\partial V}{\partial Q}\right]^T = 0 \tag{5-5}$$

By solving (5-5), we can obtain the pre-processing formulas which are defined in (5-6) to (5-9).

$$q_0^{'} = \frac{1}{49}(-9 + 40p_0 + 20p_1 + 40q_0 + 16q_1 - 8q_2 - 49p_0^{'} - 10p_2^{'}) \quad (5\text{-}6)$$

$$q_1^{'} = 4 - 4p_0 + 4q_0 + p_1^{'} \quad (5\text{-}7)$$

$$q_2^{'} = \frac{1}{49}(-4 - 4p_0 - 2p_1 - 4q_0 + 18q_1 + 40q_2 + p_2^{'}) \quad (5\text{-}8)$$

$$q_3^{'} = q_3 \quad (5\text{-}9)$$

6. Overlapped Block Methods

As mentioned earlier, the block-based transform and quantization is adopted to achieve decorrelation in many existing image and video compression standards. However, this kind of transform does not take the correlation of pixels across the block boundaries into account, so the block artifact appears in the reconstructed image and video. On the other hand, in video compression coding, the motion compensation will propagate the blocking artifact into next frames. In order to overcome these problems, the *overlapping transform* and *overlapped block motion compensation* methods are proposed. In the remained parts of this sub-section, we will introduce these two methods.

6.1 Lapped Orthogonal Transform

In conventional block-transform, the image is segmented into several non-overlapping blocks and each of these blocks is transformed and quantized separately. In Lapped Orthogonal Transform [14], the blocks overlap slightly, so the redundant information is transmitted for the samples in the block boundaries. The LOT (Lapped Orthogonal transform) is introduced as follows.

Assume the 1D discrete time signal is a sequence of *MN* samples, where *N* is the block size, and *M* is the number of the segmented blocks. We denote the original signal by x_0 and the transform coefficients as y_0. The relationship between x_0 and y_0 is expressed in (6-1).

$$y_0 = Tx_0 \quad (6\text{-}1)$$

where T is the transform matrix with size *MN*×*MN*

$$T = \begin{bmatrix} P_1 & & & & 0 \\ & P_0 & & & \\ & & \ddots & & \\ & & & P_0 & \\ 0 & & & & P_2 \end{bmatrix} \tag{6-2}$$

where P_0, P_1 and P_2 are matrices with size *L*x*N*. The matrices P_1 and P_2 are different from P_1 because of the two boundary block of the original have only one neighboring block. Thus, they are defined in a different way.

We show the concept of LOT in Fig. 6-1. In conventional block-based transform, we code current block individually. However, in LOT, we take the current block and its neighboring block for decorrelation. We should note that we can obtain the transform coefficients with size *N* after taking LOT of the block with length *L*.

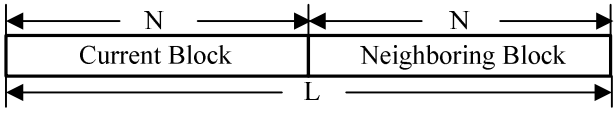


Fig. 6-1 The current block and its neighboring block for decorrelation in LOT

The autocorrelation matrix of the input signal is denoted by R_{xx}.

$$R_{xx} = \begin{bmatrix} 1 & \rho & \rho^2 & \cdots & \rho^L \\ \rho & 1 & & & \rho^{L-1} \\ \vdots & & \ddots & & \vdots \\ \rho^{L-1} & & & 1 & \rho \\ \rho^L & \cdots & \rho^2 & \rho & 1 \end{bmatrix} \tag{6-3}$$

An optimal LOT should maximize the energy compaction measurement G_{TC}

$$G_{TC} = \frac{\frac{1}{N}\sum_{i=1}^{N}\sigma_i^2}{\left(\prod_{i=1}^{N}\sigma_i^2\right)^{1/N}} \tag{6-4}$$

where σ_i^2 is the *i*th diagonal entry of the matrix R_0

$$R_0 = P_0^T R_{xx} P_0 \tag{6-5}$$

After some math manipulation, we can obtain the transform matrix P_0 in (6-6)

$$P_0 = \frac{1}{2}\begin{bmatrix} D_e & D_o & 0 & 0 \\ 0 & 0 & D_e & D_o \end{bmatrix}\begin{bmatrix} I & I & & \\ I & -I & & \\ & & I & I \\ & & I & -I \end{bmatrix}\begin{bmatrix} 0 & 0 \\ I & I \\ I & -I \\ 0 & 0 \end{bmatrix}\begin{bmatrix} I & 0 \\ 0 & Z \end{bmatrix} \qquad (6\text{-}6)$$

where $Z = T_1 T_2 \cdots T_{N/2-1}$. D_e and D_0 are the *N*x*N/2* matrices containing the even and odd DCT functions, respectively.

The plane rotation matrix T is defined as

$$T_i = \begin{bmatrix} I & 0 & 0 \\ 0 & Y(\theta_i) & 0 \\ 0 & 0 & I \end{bmatrix} \qquad (6\text{-}7)$$

where $Y(\theta_i) = \begin{bmatrix} \cos\theta_i & \sin\theta_i \\ -\sin\theta_i & \cos\theta_i \end{bmatrix}$

. Now we have the transform matrix (6-6), so we can present the implementation here. We take block size 8 for introduction, and the flow diagram of the fast LOT is shown in Fig. 6-2.

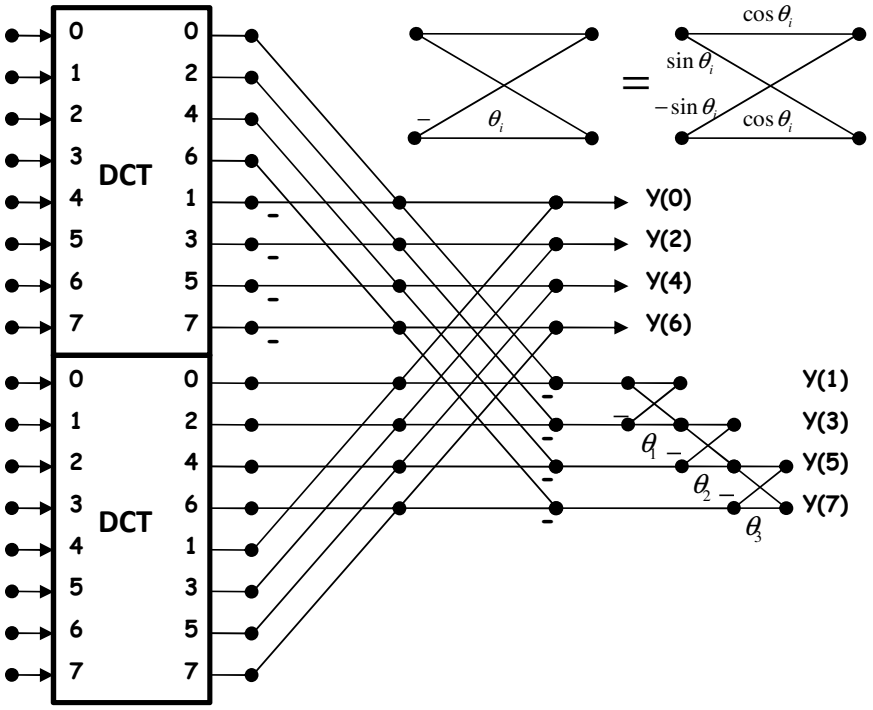


Fig. 6-2 Fast LOT with block size 8

Fig. 6-3 shows the overall LOT of the whole image. As can be seen from this figure, the first and last blocks do not have the complete neighboring blocks. Thus, we reflect the data at these two boundaries, which can form the transform matrix P_1 and

P_2 in (6-2).

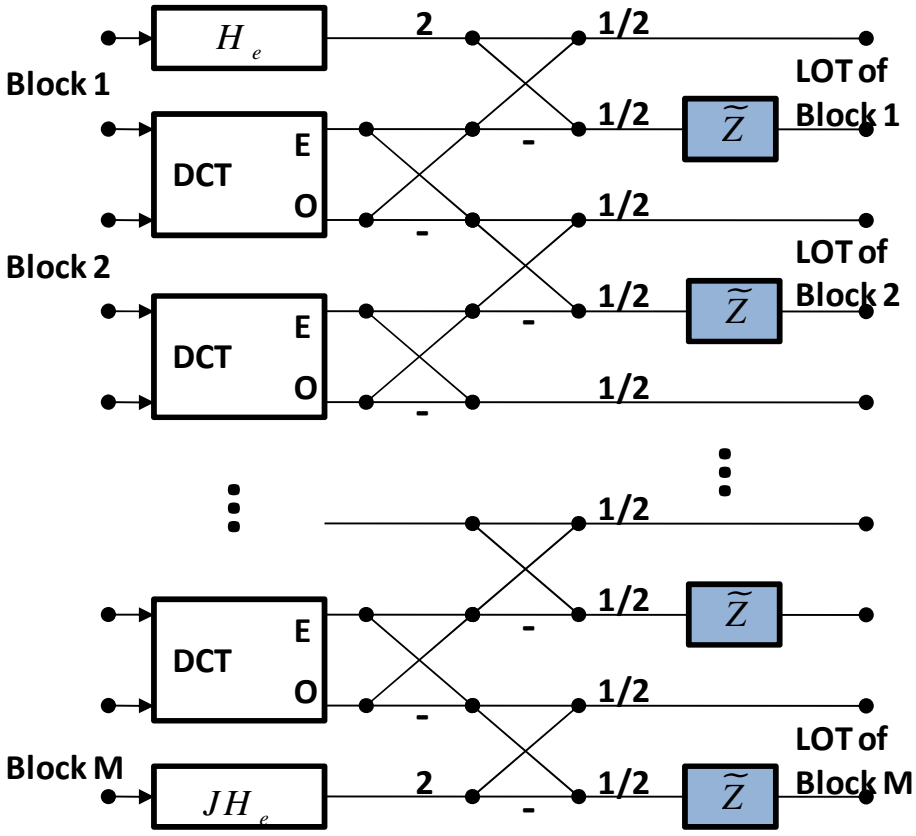


Fig. 6-3 Flow diagram of the fast LOT for the full data sequence

The notation J in Fig. 6-3 is the "counter-identity" which is define as

$$J = \begin{bmatrix} 0 & & & 1 \\ & & \iddots & \\ & 1 & & \\ 1 & & & 0 \end{bmatrix} \tag{6-8}$$

With the fast LOT algorithm, we can compute the DCT coefficients for each blocks and then applying the +/- and Z. Thus, the computation cost of LOT is not very far from that of DCT, and we can reduce the blocking artifact effectively.

6.2 Overlapped Block Motion Compensation

The conventional motion estimation and compensation algorithm produce block edges in the compensated frame because the motion vectors between the neighboring blocks are not always the same. These block edges will decrease the coding efficiency because the energy of the residue signal increase. When the LOT is applied to the residue, the block edges within the overlapped block will introduce high frequency components to the residue. Therefore, overlapped block motion estimation and

compensation algorithm [15] is proposed to reduce the high frequency components in the residue

A) Overlapped block Motion Estimation

Fig. 6-4 shows the overlapped block motion estimation scheme. The reference frame and the current frame are segmented into several overlapped blocks. The target block $P_{v_i}(x, y)$ is predicted from an enlarged block size *N*x*N* (which is larger than the original block size). The difference block for each target block is defined as

$$E_{v_i}(x, y) = P_{v_i}(x, y) - S(x, y) \tag{6-9}$$

where $S(x, y)$ is the block in the original frame.

In order to reduce the high frequency components in the difference blocks, a window function $W(x, y)$ is operated to the prediction error signal. The window function is designed to decay toward the block boundaries. Therefore, we can obtain a new weighted difference block $E_{v_i W}(x, y)$ defined as

$$E_{v_i W}(x, y) = E_{v_i}(x, y) \times W(x, y) \tag{6-10}$$

We search for the candidate motion vector by using $E_{v_i W}(x, y)$, and the mean absolute error (MAE) is redefined in (6-11)

$$MAE = \frac{1}{N^2} \sum_{x=1}^{N} \sum_{y=1}^{N} \left| E_{v_i W}(x, y) \right| \tag{6-11}$$

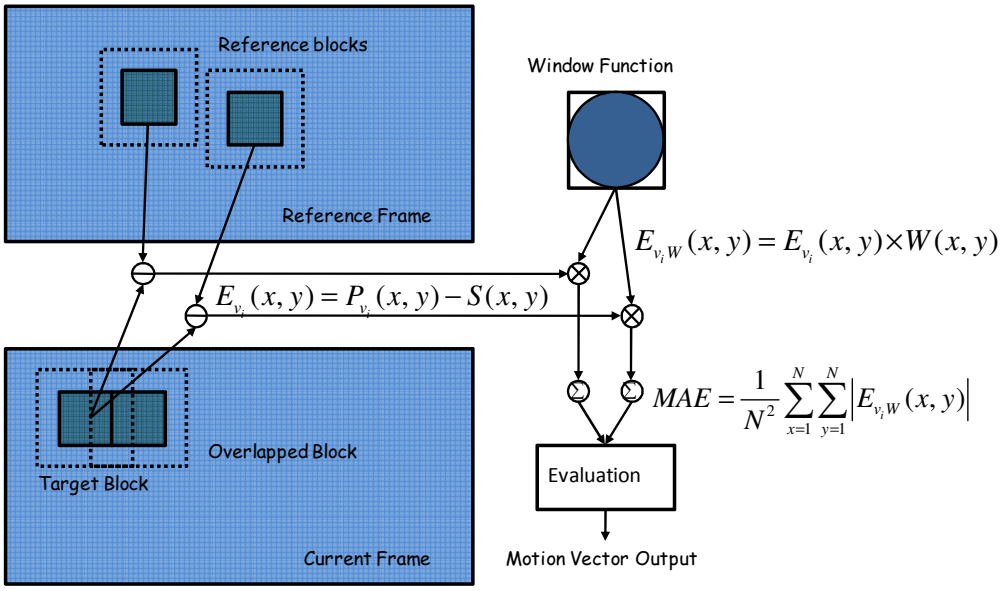


Fig. 6-4 Overlapped Block Motion Estimation

B) Overlapped block Motion Compensation

After the motion vector $P_v(x, y)$ for each target block $P_{v_i}(x, y)$ is determined, the compensated block is also generated from the enlarged reference blocks. The overall overlapped block motion compensation scheme is shown in Fig. 6-5.

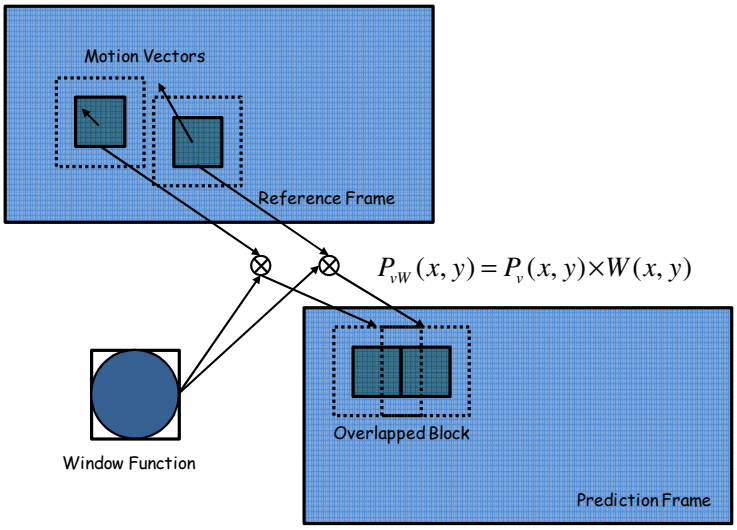


Fig. 6-5 Overlapped Block Motion Compensation

The prediction block picks up the most similar block by the motion vector. The enlarged block is also multiplied with the window function $P_v(x, y)$ to form the weighted prediction block $P_{vW}(x, y)$.

$$P_{vW}(x, y) = P_v(x, y) \times W(x, y) \tag{6-12}$$

The whole prediction frame is generated by summing all the weighted prediction blocks. With the overlapped block motion estimation and motion compensation algorithm, the reconstructed frames without severe block edges can be obtained.

7. Block-Edge Impairment Metric

Although PSNR (Peak Signal-to-Noise Ratio) is widely adopted to express the quality of the reconstructed image and video, it does not always reveal the real quality perceived by the HVS. On the other hand, we can improve the quality of the compressed image and video by means of deblocking algorithms and the PSNR will increases proportional quality. However, in some condition, the improving scale of PSNR is not high, but large amount of the blocking artifacts are reduced. Therefore, several block-edge impairment metrics [16-17] have been proposed.

In this section, we introduce one of these block-edge impairment metrics called Generalized Block-Edge Impairment Metric, abbreviated GBIM. Assume the reconstructed image or video frame is $\mathbf{f}$ defined as

$$\mathbf{f} = \left\{ \mathbf{f}_{c1}, \mathbf{f}_{c2}, ..., \mathbf{f}_{cN_c} \right\} \tag{7-1}$$

where $\mathbf{f}_{cj}$ is the *j*th column and N_c is the width of the image $\mathbf{f}$.

The interpixel difference between the block boundaries in the horizontal direction is defined as

$$D_c\mathbf{f} = \begin{bmatrix} \mathbf{f}_{c8} - \mathbf{f}_{c9} \\ \mathbf{f}_{c16} - \mathbf{f}_{c17} \\ \vdots \\ \mathbf{f}_{c(Nc-8)} - \mathbf{f}_{c(Nc-7)} \end{bmatrix} \tag{7-2}$$

where each of the block is a 8 x 8 block.

The metric to measure the horizontal blockiness is defined as

$$\begin{aligned} M_h &= \left\| \mathbf{W} D_c f \right\| \\ &= \left[\sum_{k=1}^{N_c/8-1} \left\| \mathbf{w}_k \left[\mathbf{f}_{c(8\times k)} - \mathbf{f}_{c(8\times k+1)} \right] \right\| \right]^{1/2} \end{aligned} \tag{7-3}$$

where $\mathbf{W} = diag\left[\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_{N_c/8-1} \right]$ is the diagonal weighting matrix which takes the spatial characteristics into account. $\mathbf{w}_k = diag\left[w_{1,j}, w_{2,j}, ..., \mathbf{w}_{Nr,j} \right]$ where N_r is the height of the image, and $j = 8\times k$ for $k = 1, 2, ..., N_c/8 - 1$.

The weighting function $w_{i,j}$ is defined as

$$w_{i,j} = \begin{cases} \lambda \ln\left(1 + \dfrac{\sqrt{\mu_{i,j}}}{1+\sigma_{i,j}} \right), \text{if } \mu_{i,j} \le \zeta \\ \ln\left(1 + \dfrac{\sqrt{255-\mu_{i,j}}}{1+\sigma_{i,j}} \right), \text{ otherwise} \end{cases} \tag{7-4}$$

where ζ is the average luminance value which is proportional to the blockiness and distortion, and $\lambda = \dfrac{\ln(1+\sqrt{255-\zeta})}{\ln(1+\sqrt{\zeta})}$.

For horizontal blocks, the mean $\mu_{i,j}$ and variance $\sigma_{i,j}$ is defined as the average of the two adjacent blocks

$$\mu_{i,j} = \frac{\mu_{i,j}^L + \mu_{i,j}^R}{2} \tag{7-5}$$

where $\mu_{i,j}^L = \frac{1}{8}\sum_{n=j-7}^{j} f_{i,n}$ and $\mu_{i,j}^R = \frac{1}{8}\sum_{n=j+1}^{j+8} f_{i,n}$ and $f_{i,n}$ represents an individual pixel at row *i* and column n of the image $\mathbf{f}$.

$$\sigma_{i,j} = \frac{\sigma_{i,j}^L + \sigma_{i,j}^R}{2} \tag{7-6}$$

where $\sigma_{i,j}^L = \left[\frac{1}{8}\sum_{n=j-7}^{j}\left(f_{i,n} - \mu_{i,j}^L\right)^2\right]^{1/2}$ and $\sigma_{i,j}^R = \left[\frac{1}{8}\sum_{n=j+7}^{j+8}\left(f_{i,n} - \mu_{i,j}^R\right)^2\right]^{1/2}$

We normalize M_h by the average interpixel difference E to obtain M_{hGBIM}.

$$M_{hGBIM} = M_h / E \tag{7-7}$$

$$E = \frac{1}{7}\sum_{n=1}^{7} S_n \tag{7-8}$$

where $S_n = \left[\sum_{k=1}^{N_c/8-1} \left\| \mathbf{w}_k \left[\mathbf{f}_{c(8\times k+n)} - \mathbf{f}_{c(8\times k+n+1)}\right]\right\|\right]^{1/2}$

The M_{hGBIM} is used to describe the horizontal blockiness, we can obtain the vertical blockiness metric M_{vGBIM} by similar process. Finally, we can obtain the GBIM M_{GBIM} which is defined as

$$M_{GBIM} = \alpha M_{hGBIM} + \beta M_{vGBIM} \tag{7-9}$$

In order to show the performance of GBIM, we give two examples shown in Fig. 7-1. As can be seen from the simulation result, the PSNR of the compressed image and the filtered one is not far different from each other, but the blockiness is reduced and the M_{GBIM} reveals the fact.

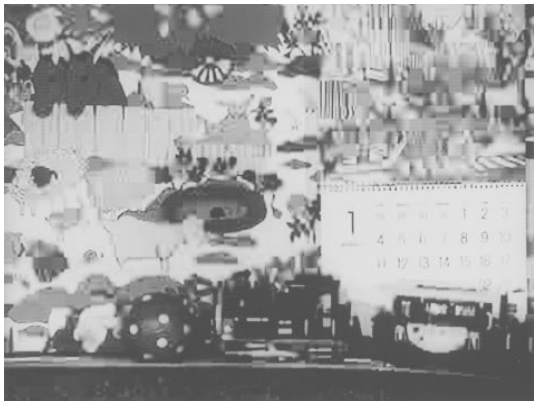


Fig. 7-1 (a) Reconstructed Image	Fig. 7-1 (b) Reconstructed image after deblocking
M_{GBIM} =3.11, PSNR=21.75 dB	M_{GBIM} =1.05, PSNR=21.86 dB

Fig. 7-1 The PSNR and GBIM of two compressed images before and after deblockin, respectively

8. Comparison

In this section, we discuss the pros and cons of each algorithm.

A) In-loop Deblocking

	Advantage	Disadvantage
H.264/AVC	1. It adaptively selects the deblocking filter based on the strength of blockiness for deblocking. 2. The complexity is low	The coding efficiency is lower than the OPF/OLF algorithm
Optimal Post-Processing and In-loop Filtering	It can achieve better coding performance and reduce the blocking artifact more effectively than H.264/AVC because it obtains the optimal filter coefficients by referring to the new input frame	The complexity is very high because it must iteratively compute the filter coefficients.

B) Post-processing Deblocking Algorithms

The post-processing algorithms are employed at the decoder output, so they have good potential to be integrated into existing image and video standards.

	Advantage	Disadvantage
Reduction of blocking artifacts in DCT Domain	1. Take the HVS into account 2. Apply the filter with less strength to preserve the real edge 3. Apply the filter with larger strength to reduce the blocking artifact	High complexity and computation time because of large amount of DCT operation
WSSAP	1. The filter coefficients can be obtained in advance	The deblocking operation is applied to the whole image, so it may blur the real edge
Offset and Shift	1. Take the HVS into account	

Technique	2. Apply the filter with less strength to preserve the real edge 3. Apply the filter with larger strength to reduce the blocking artifact 4. Low Complexity 5. Prevent the processed video and image from blurring	

C) Pre-processing Deblocking Algorithms

The pre-processing deblocking algorithms combine the post-processing deblocking algorithms to reduce the data rate, and the quality of the image and video is very close to that without pre-processing.

D) Overlapped Block Methods

All the deblocking algorithms described reduce the blocking artifact can be classified into the **remedy methods**. In contrast to the other method, the overlapped block methods prevents the blocking artifacts from happening in advanced, so it can be classified into **Prophylaxis**. The disadvantage of the overlapped block methods is that they are not compatible with the existing video and image standards.

9. Conclusions and Future Work

In this paper, we have introduced several deblocking algorithms to the reader. In section 2, we briefly describe the characteristics and observations of blocking artifact, and several deblocking algorithms that take advantage of these characteristics and observations to improve the quality of the compressed image and video. In section 3 to 6, we introduced the four categories of deblocking algorithms, and the related cases are given to each category.

The first type is the in-loop filter, which is used to reduce the blocking artifact in compressed video. The advantage of the in-loop filter is that it can achieve better improvement because it can refer to the new input video frame while the conventional deblocking algorithm adopted in image compression is blind to the source image. However, the in-loop filter is not compatible with the existing video coding standard. The second type is the post-processing deblocking algorithm, which is the most popular method because it can be combined with the existing image and video coding standards. The basic idea of the post-processing methods is filtering the sharp edge over the block boundary to smooth the compressed image and video by using the

lowpass filter. The third type is the pre-processing deblocking algorithm, which modifies the source image and video in advance and reduces the bit rate, can achieve the quality close to the direct compressed image and video. The last type is the overlapped method, which is much different from other methods because the rule of it is taking preventive injection instead of putting out the fire. The drawback of the overlapped methods is that it is not compatible with the existing standards, which is the same as the in-loop filter.

In section 7, we introduce one blockiness metric GBIM because the conventional metric PSNR does not always reveal the real quality perceived by the HVS. From the simulation result we can observe that the PSNR of the processed image is very close to that of the un-processed one, but GBIM effectively reveals the difference.

To summarize, the conventional deblocking algorithms smooth the sharp block boundary by lowpass filter. However, many conventional methods do not exploit the characteristics of the blocking artifacts. Nowadays, more and more deblocking algorithms take advantage of the characteristics of blocking artifacts to improve the quality of the compressed image and video. These methods apply strong filter over the flat area and weak filter over the complex area due to the HVS sensitivity to the blocking artifact. As just mentioned, these new deblocking algorithms may not improve the PSNR in a large degree, so we must have some blockiness metric to estimate the performance of these deblocking algorithms. To add all the discussion, the better deblocking algorithm based on the characteristics of the blocking artifacts and the better blockiness metric may be the future trend.

10. References

[A] In-loop Filter

[1] Lain E.G. Richardson, "*H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia*", John Wiley & Sons, Jan. 2004.

[2] Lee, Jae-Beom, Kalva and Hari, "*The VC-1 and H.264 Video Compression Standards for Broadband Video Services*", Springer, 2008.

[3] "Text of ISO/IEC FDIS 14496-10/Draft ITU-T H.264: Information Technology – Coding of Audio-Visual Objects: Advanced Video Coding", International Organization for Standardization, 2003.

[4] Dong-Hwan Kim, Hwa-Yong Oh, O˘guzhan Urhan, Sarp Ertürk and Tae-Gyu Chang, "Optimal Post-Process/In-Loop Filtering for Improved Video Compression Performance", *IEEE Trans. on Consumer Electronics*, vol. 53, no. 4, Nov. 2007.

[5] S. Romero, and L.F. Romero, “An Optimized Preconditioned Conjugate Gradient Algorithm”, Technicah Report No: UMA-DAC-02/11, University of Malaga, Sept. 2002.

[B] Post-Filtering

[6] Tao Chen, Hong Ren Wu and Bin Qiu, “Adaptive Postfiltering of Transform Coefficients for the Reduction of Blocking Artifacts”, *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 11, no. 5, Dec. 2001.

[7] Shizhong Liu and Alan C. Bovik, “Efficient DCT-Domain Blind Measurement and Reduction of Blocking Artifacts”, *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 12, no. 12, May 2002.

[8] Ci Wang, Wen-Jun Zhang and Xiang-Zhong Fang, “Adaptive Reduction of Blocking Artifacts in DCT Domain for Highly Compressed Images”, *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 50, no. 2, May 2004.

[9] A. Z. Averbuch, A. Schclar and D. L. Donoho, “Deblocking of block-transform compressed images using weighted sums of symmetrically aligned pixels,” *IEEE Trans. on Circuits Syst. Video Technology*, vol.14, pp.200-212, Feb. 2005.

[10] Jongho Kim, Minseok Choi, and Jechang Jeong, “Reduction of Blocking Artifact for HDTV using Offset-and-Shift Techniques”, *IEEE Trans. on Consumer Electronics*, vol. 53, no.4, November 2007.

[11] Zixiang Xiong, Michael T. Orchard, and Ya-Qin Zhang, “A Deblocking Algorithm for JPEG-Compressed Images Using Overcomplete Wavelet Representations”, *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 7, no. 2, April 1997.

[12] Jongho Kim and Jechang Jeong, “Adaptive Deblocking Technique for Mobile Video”, *IEEE Trans. on Consumer Electronics*, vol. 53, no. 4, Nov. 2007.

[C] Pre-Filtering

[13] Sheng-Ho Wang, Sung-Wen Wang, Yi-Shin Tung, Ja-Ling Wu, Jau-Hsiung Huang, "Pre-Process for maximizing the effect of in-loop deblocking filtering in H.264/AVC encoding," 5th EURASIP Conference focused on Speech and Image Processing, Multimedia Communications and Services (EC-SIP-M 2005), Smolenice, Slovak, 29 June - 2 July, 2005.

[D] Overlapped Block Methods

[14] H. S. Malvar and D. H. Staelin, “The LOT: transform coding without blocking effects”, *IEEE Trans. on Acoustic., Speech, Signal Processing*, vol. 37, no. 4, pp. 553-559, April 1989.

[15] M. T. Orchard and G. J. Sullivan, “Overlapped block motion compensation: an estimation-theoretic approach”, *IEEE Trans. on Image Processing*, vol. 3, no. 4, pp. 693-699, Sep. 1994.

[E] Block-Edge Impairment Metric

[16] H. R. Wu and M. Yuen, “A Generalized Block-Edge Impairment Metric for Video Coding”, *IEEE Signal Processing Letters*, vol. 4, no. 11, Nov. 1997.

[17] Athanasios Leontaris, Pamela C. Cosman and Amy R. Reibman, “Quality Evaluation of Motion-Compensated Edge Artifacts in Compressed Video”, *IEEE Trans. on Image Processing*, vol. 16, no. 4, April 2007.