國立臺灣大學資訊工程學研究所碩士論文

指導教授：黃肇雄博士

吳家麟博士

整合前處理機制與具感知能力去方塊效應濾波器之 H.264/AVC 壓縮效能提升方法

H.264/AVC coding performance enhancement via incorporating pre-process with perceptual-based in-loop deblocking filters

研究生：王盛禾　撰

學號：R92922007

中華民國九十四年六月

致謝

能夠順利完成本篇論文及取得碩士學位，心中充滿感激與興奮，綜觀兩年以來的研究所生活，首先要感謝恩師黃肇雄教授和吳家麟教授，在他們諄諄不悔的教導之下，讓我不但學習了做研究的方法也體悟了學者的風範。其次，感謝用心栽培我的童怡新博士，在他的提攜之下，讓我對於多媒體壓縮這個領域，從懵懵懂懂的情況下，漸漸的了解此領域及產生興趣，他不但指導我做深入的研究，也同時帶領我參加國際的相關研討會議，拓增了我的視野，在平常生活中，不論是做研究、寫程式或是生涯規劃等相關問題，童博士都竭盡所能地提供協助，他如此不吝惜的付出，讓我深深體會到他是個不可多得的好學長。

還要感謝同組的實驗室夥伴李佳盈、楊雅婷、黃義欽、陳宇皓、劉錦昕，以及其他組的同學如陳昭源、倪英豪、徐采琳、謝名凱、黃建賓、林孟輝等，彼此之間互相討論幫忙，閒暇之餘，一同出遊，為這兩年的研究生活留下了美好的回憶。另外，王頌文學長對於我在資料壓縮領域的指導以及幫助本篇論文的完成，也令我相當感激。與鄭文皇、黃俊翔、朱威達、林佳緯、郭晉豪、謝致仁學長和莊玉如學姊的互動中，吸取了不少研究和生活上的寶貴意見，實在獲益良多。

感謝從大學到研究所以來的同窗好友鄭先廷和李明翰，我們一起做研究所花費的腦力、一起打球所留下的汗水、一起分享生活上的酸甜苦辣，這些美好的點點滴滴都伴隨著我來完成本篇論文。

最後要感謝我的爸爸王家竹、媽媽林美珠、哥哥王盛平以及遠在美國唸書的女朋友蔡宜蓉，沒有父母從小到大對我的照顧和栽培，就沒有我今天的成果，在此向父母致上最深的謝意，感謝哥哥在當兵期間仍然持續關心我的學業。而在美國辛苦唸書的宜蓉寶貝，經常寫書信、明信片來幫我加油打氣，善解人意的體貼行為，更是支持我努力做研究的動力。謝謝各位的扶持與鼓勵，在此獻上這本論文和最誠摯的祝福。

王盛禾謹誌

民國九十四年六月

Abstract

Block-based video coding cooperating with block transform and block motion compensation is the most widely adopted way to reduce the data redundancy in various video coding standards. Although the goal of de-correlations is achieved effectively by this way, the most annoying artifact known as the blocking effect also comes into existence. To both remove this artifact and improve the coding performance simultaneously, the latest video coding standard, H.264/AVC, enforces the deblocking filters inside its coding loop.

In the design of deblocking filters of H.264/AVC, one pair of parameters, OffsetA and OffsetB, are provided, which allow the adaptive control of the deblocking strength in slice level. Thus, finding out better parameters for conducting the deblocking process of H.264/AVC is capable of improving visual quality of reconstructed video. Identifying which edges belong to blocking effect relies on perceptual judgment of human beings. In fact, this subjective assessment may not exactly match existing objective measurements and high PSNR does not always stand for less blocking artifacts. In this thesis, we introduce two new criteria for measuring the blocking distortion by analyzing the perceptual difference between the source and the reconstruction. The experimental results validate the proposed approaches, especially in subjective issues. On the other hand, another implicit advantage of deblocking is ignored by most encoders. It is observed that different coded images may have the same output after applying the mandatory deblocking process. Based on this observation, we integrate this concept into H.264/AVC. For eight different deblocking modes, we first derive the equations to change the input image but do not affect the final output reconstruction. By choosing those of less bitrate consumption, the proposed pre-processing approach successfully improves video coding performance.

Combing advantages of both pre-process and post-process, an enhanced H.264/AVC coding system is implemented which maximizes the effect of deblocking filters. The experimental results demonstrate its improvements for H.264/AVC codec both in objective and subjective evaluations

中文摘要

近年來，數位多媒體內容 (digital multimedia content)的相關技術及創作內容已快速蓬勃地發展，各種不同形式的影音特效是造就多媒體內容如此受大眾歡迎的主因，正由於多媒體資料是如此的豐富且複雜，相關的多媒體資料壓縮技術也已經成為重要的研究方向，其中，區塊壓縮 (block-based) 搭配移動補償 (motion compensation) 的方法已被許多國際壓縮標準所採納且行之多年，例如: JPEG、MPEG1/2/4、H.264/AVC。雖然區塊壓縮可達到不錯的壓縮效率，但它同時也造成了視訊影像的失真，其中最明顯的就是方塊效應 (blocking effect)，為了減低此效應的影響，各個壓縮標準也分別制定其去方塊效應濾波器 (deblocking filer)，以求失真程度降到最低。

目前最新的視訊壓縮標準 (H.264/AVC) 提供編碼器兩個動態調整去方塊效應濾波器的參數，在 H.264/AVC 的標準制定中並沒有規範如何動態調整此組參數，因此，本篇論文利用後處理 (post-processing) 的方法來改進 H.264/AVC 去方塊效應濾波器的能力，根據人類視覺系統 (HVS) 模型，分析視訊影像的內容，進而動態調整去方塊效應濾波器的強度，對於解碼後視訊畫質的提升有很大的幫助。

除了後處理機制之外，本篇論文還利用前處理 (pre-processing) 機制搭配後處理的方法來改善 H.264/AVC 的整體壓縮效能，也就是能夠用較少的資料來表示原本的多媒體資料且仍能呈現出不錯的畫面品質，前處理的想法是先將原始多媒體資料經由適當的低通濾波器 (low pass filter) 處理之後，然後將此低頻訊號傳給 H.264/AVC 編碼器來進行壓縮，如此，整體的視訊壓縮效能將會比直接壓縮原始訊號來得好。

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Chapter 1
Introduction

1.1 Motivation

Digital multimedia contents have been prosperously created in the recent decades due to the attracting perceptual effects of their presentation. A multimedia content often consists of a variety of images, videos, audios and so on. Because of the distinct and complex information in multimedia contents, it usually requires a large space to store these digital contents and the transmission of raw media data on the Internet is impractical. Researches on data compression technologies are made great progress to overcome this obstacle, especially for the large quantity of video contents. Among all video coding schemes, block-based transform coding (BTC) is the most widely adopted approach to reach the goal of compression. The basic concept of BTC is to divide the image into non-overlapped blocks and then apply Discrete Cosine Transform (DCT) or integer transform on each block independently. Many multimedia compression standards such as JPEG (Joint Photographic Experts Group) for still images and MPEG 1/2/4 [1] (Moving Picture Experts Group) for video sequences choose block-based DCT (BDCT) as their common transform kernel. The latest video coding standard, H.264/AVC [2], also uses BTC but with integer transform rather than DCT. The reason for the popularity of BTC is not only its energy compaction and de-correlation properties but also the practical implementation costs in hardware devices. Although BTC brings the benefits of data compression, it also results in annoying artifacts known as blocking effect. This is a phenomenon of pixel value discontinuity across block boundaries. When the bitrate decreases, this artifact becomes more noticeable. Consequently, to remove the blocking effect in the reconstructed frames or images turns into an important research topic in image processing and video coding fields.

1.2 Deblocking Methods

The behavior of eliminating the blocking effect and maintaining the visual quality of the video data within a certain level are often called “deblocking.” A variety of deblocking algorithms such as block boundary filtering, deblocking on the domain of overcomplete wavelet representation, projection onto convex sets (POCS), overlapped motion compensation (OBMC), maximum a posterior (MAP), lapped orthogonal transform (LOT) and weighting sums of symmetrically aligned pixels (WSSAP) have been proposed. According to the processing order and the required time of operating deblocking, these approaches can be classified into two categories. One is post-processing and the other is pre-processing. Post-processing algorithms process the decoded frames or images while pre-processing techniques take the original frames as input. Block boundary filtering, overcomplete wavelet representation, POCS, MAP, OBMC, WSSAP belong to the group of post-processing. These methods try to remove the distortion incurred by the blocking effect as much as possible. On the other hand, pre-processing approaches analyze the behavior of the encoder and modify the pixel value of the original frame to reduce the blocking effect. Fig. 1.1 and Fig. 1.2 depict the diagrams of post-processing and pre-processing, respectively. During the deblocking process, the most difficult task is to distinguish the real object edges from artificial edges caused by the blocking effect. The performance of deblocking algorithms will be enhanced if more accurate models of measuring blocking effect are acquired and better smoothing operations are applied to remove blocking. Numerous quality assessment models have been proposed to estimate the different kinds of distortion in images or videos. Some focus on measuring some specific distortion, e.g. blocking effect or blur degree, while others consider the sum effect caused by multiple types of distortion and evaluate the overall quality. Most of these measurements are derived based on human visual system (HVS) in order to provide analyses in proportion to the feeling of humans.

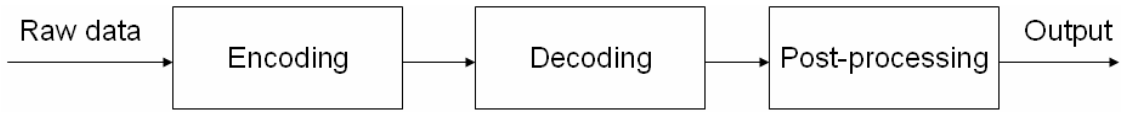


Fig. 1.1: The post-processing deblocking scheme.

Fig. 1.2: The pre-processing deblocking scheme.

1.3 H.264/AVC In-loop Deblocking Filter

ISO/IEC MPEG and ITU-T VCEG co-organized a working group, called the Joint Video Team (JVT), which aims to define a new video coding standard. This task has just completed in March 2003 and finally comes out the newest standard, H.264/AVC. During the development process, a lot of advanced coding tools are considered, and some elaborate combinations of these tools are conducted experimentally. Finally, only those novel and computation-efficient coding tools are adopted. Generally speaking, although the decoding framework of H.264/AVC is analogous to all previous hybrid video coders, the whole process changes dramatically after integrating the new advances into different functional blocks. These new advances include 4×4 block transform, spatial prediction, variable block size motion compensation, long term motion compensation, context-adaptive VLC, context-adaptive arithmetic coding and so on. As a result, H.264/AVC improves the coding performance a lot. It is a general belief that H.264/AVC outperforms the previous standards, such as MPEG-2, MPEG-4 Visual and H.263, by saving about 50% bitrate at the same visual quality. The block diagram of the H.264/AVC encoding process is shown in Fig. 1.3, in which the only major change is in adding an adaptive de-blocking filter inside the coding loop. As mentioned above, H.264/AVC as well as conventional coding standards are all block-based. In this scheme, independent block-based coding can probably lead to noticeable discontinuities between the block boundaries of the reconstructed image or video (a.k.a. "blocking" artifact). Actually, it is

the most annoying defect incurred by block based coding. While some video coding standards regard the deblocking filter as an optional post-processing, H.264/AVC makes it mandatory. The reason why H.264/AVC forces de-blocking filter inside the coding loop is to enhance the visual quality and the coding performance. Furthermore, the de-blocking filters adopted by H.264/AVC provide two control variables to flexibly alter the filter strength or even disable the filter. These control variables are set to different values according to the content characteristics of the processed frame.

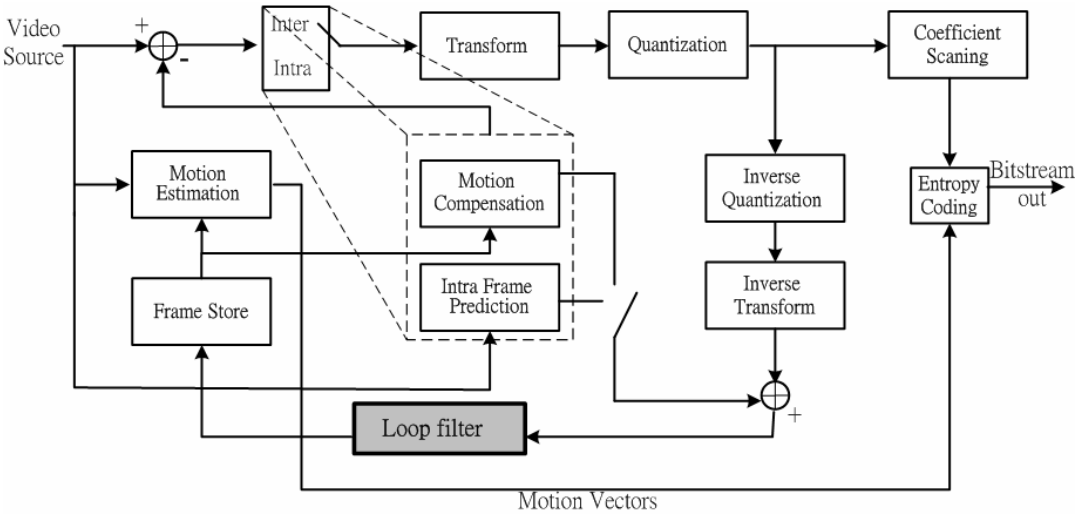


Fig. 1.3: The encoding architecture of H.264/AVC.

1.4 Contributions

This thesis incorporates the spirits of pre-processing and post-processing to improve the coding performance both on the objective metric, *i.e.* rate-distortion performance, and the subjective evaluation. In the pre-processing stage, according to rate-distortion optimization, the formulations of deriving the best candidates for replacing the input pixel values are obtained. In the procedure of post-processing, the goal aims to develop a method for determining the two coding variables which control the ability of in-loop deblocking filters of H.264/AVC, with an emphasis on subjective issues.

1.5 Thesis Organization

The organization of this thesis is laid out as follows. Chapter 2 briefly introduces previous deblocking methods and measurements of blocking effect in the literature. Our proposed system framework jointly considering pre-processing and post-processing is described in Chapter 3. Two main components, post-process and pre-process, proposed to enhance encoding performance are explored in depth in Chapters 4 and 5, respectively. Experimental results and some related discussions are provided in Chapter 6. Finally, Chapter 7 concludes this thesis and points out the directions of our future research.

Chapter 2
Related Works

2.1 Blocking Effect

In the block-based transform scheme, the correlations of neighboring pixels near block boundaries are ignored which incurs the discontinuity between block boundaries. This discontinuity is called blocking effect and is considered to be the most annoying artifacts in the video coding. Blocking effect is mainly resulted from two causes. One is the independent block transform and the other is the irregular block motion. During the block-based coding process, the frame is divided into several non-overlapped blocks, and each is transformed into the frequency domain and then quantized into several quantization bins. Usually, neighboring pixels within a nature scene are highly correlated. If they are separated into two different blocks and then quantized independently, a small difference between the original pixel pair possibly makes them falling into different bins. In other words, the quantization magnifies the error terms in some cases. After accumulating all the effects from different frequencies, a noticeable discontinuity arises. This kind of distortion can be easily found in Fig. 2.1 where blocking effect is shown obviously around each block boundary. Second, the inter coding referenced blocks are compensated from different portions of previous frames. Inevitably, the discontinuity between adjacent motion blocks arises. Current motion estimation algorithms applying equal weight to all pixels enlarge the boundary mismatch. Consequently, the annoying blocking effect becomes more obvious in this case. Fig. 2.2 demonstrates this kind of blocking artifacts.

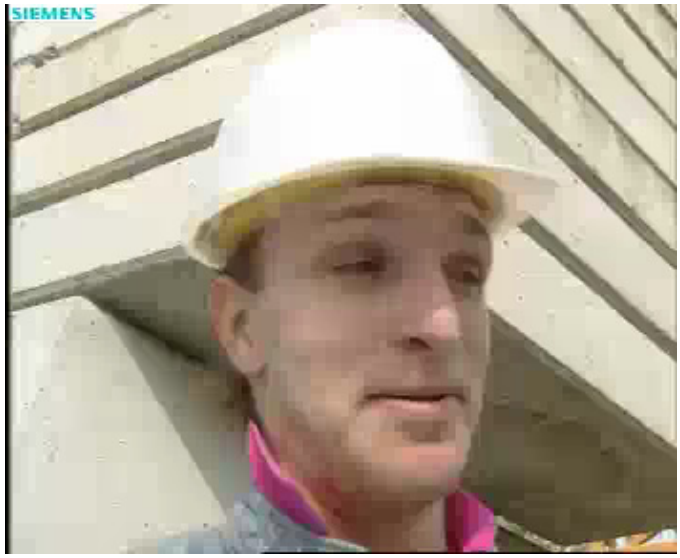


Fig. 2.1: The 4th frame (I frame) of the Foreman sequence in CIF resolution encoded by H.264/AVC with deblocking disabled. All blocks are intra-coded. Quantization parameter is set to 36.

Fig. 2.2: The 4th frame (P frame) of the Foreman sequence in CIF resolution encoded by H.264/AVC with deblocking disabled and "IPPP…." coded. Quantization parameter is set to 36.

2.2 Related Deblocking Methods

To reduce the blocking artifact while retaining the visual quality of images or videos, numerous deblocking algorithms have been proposed and many experimental results also validate their effectiveness. In this section, we briefly describe those deblocking methods developed in the last decade.

(A) Pre-processing

1. The concept of pre-processing is to eliminate the blocking effect in the reconstructed frames by modifying original frames before encoding. The reason why pre-processing can remove the blocking effect is that different input signals may result in similar reconstructed ones when deblocking filters are applied. As a result, encoding the modified version of a frame instead of the original one may get better decoded results. In the work of [3], it employs this property of deblocking filters and proposes an enhanced coding system for MPEG-4 codec. By encoding various sequences under different coding conditions, rate-distortion performance also validates the practical usage of the pre-processing scheme. To improve the coding performance and reduce the blocking effect before encoding simultaneously, a more reliable analysis on behaviors of deblocking filters is needed. The deblocking filters adopted by MPEG-4 are analyzed well in [3] and the optimal modifying formulas for pre-processing are also derived.

(B) Using overlapped block as the basic coding unit

1. *Lapped orthogonal transform* (LOT): Most video and image coding standards adopt non-overlapped block transform and quantization process to attain the goal of de-correlation i.e. removing the redundancy in the spatial domain. Nevertheless, the correlations of pixels across block boundaries are also ignored in this way. As long as this relationship is not taken into account, blocking effect will probably appear in reconstructed frames or images. To get rid of the unpleasant blocking effect and possess the advantage of de-correlation, lapped orthogonal transform (LOT)

[4] is recommended. In LOT, supports of transform bases are overlaid to each other, which is different from the ordinary block-based transform that the support region is confined by the individual block. Thus, pixels located at block boundaries will not lose the relation with their neighboring pixels. In actual implementation, the forward transformation calculation needs its surrounding pixels. With the consideration of neighbors, the relation in block boundary neighborhood can be kept well and the abnormal discontinuity phenomenon will disappear.

2. *Overlapped block motion compensation* (OBMC): In the conventional block motion compensation scheme, the predictions for neighboring blocks are separately copied from the regions of the reference frame that their motion vectors point to. The blocking effect emerged in this circumstance especially when two neighboring motions diverse a lot in the regions of complex motion. This is because block boundaries are discontinuous when their corresponding motion vectors differ much. To solve this problem, overlapped block motion compensation (OBMC) [5, 6] is proposed, which differs from the original scheme in the generation of prediction blocks. The main concept is simultaneously considering motion vectors of neighboring blocks as well as the current block's motion vector when generating a prediction block. The prediction pixel value is a weighted sum of several pixels pointed by the motion vectors of the current block and its neighbors. By this method, large discontinuity between block boundaries is eliminated by applying the smooth transition. However, unwanted blurring effect will be introduced when OBMC is applied on areas with sharp object edges.

(C) Post-processing

1. *Projection onto convex sets* (POCS) is an iterative-based algorithm to solve multiple-constraint problems. The definitions of constraints depend on the various deblocking algorithms. For example, both the quantization rule which confines the reconstructed transform coefficients to fall into the legal range and the variation of pixel values in smooth regions should be small can be one of the constraints. Each constraint

forms a convex set, C_i, and the goal is to find the intersection of these convex sets. The intuitive thought is to solve the intersection directly, but in general cases, it is very difficult to get the answer by this way. Therefore, the concept of POCS is projecting the input data, x, into convex sets one by one and repeats this procedure until the result converges. The projection operator, $P_i(x)$, is to find a pixel which is closest to x and satisfies the constraint C_i, where $0<i\leq m$, and m stands for the number of total constraints and n is the iteration number. When iteration increases, the obtained result x_n get better than those of previous iterations. In usual cases, x_n converges quickly. Various POCS algorithms [7-9] differ in the definitions of constraints. The projection can be formulated as follows.

$$\|x - P_i(x)\| = \min_{f \in C_i} \|x - f\|, \tag{2.1}$$

$$x_n = T^n x, \tag{2.2}$$

where $T = P_m P_{m-1} \ldots\ldots P_1$.

2. *Maximum a posterior* (MAP) [10] based on the assumptions that the correlation of pixels in a natural image follows the stochastic model. By this assumption, it produces images with less blocking effect via Bayesian rule and optimization techniques. Assume that the compressed image is y and z is the reconstruction of y. The goal is to find the most probably decoded image $\hat{z}$ given the compressed data y as shown in Eqn. (2.3).

$$\hat{z} = \arg\max_z P_r(z \mid y). \tag{2.3}$$

According to Bayesian rule, the above formula can be re-written as in Eqn. (2.4). *Pr(y)* can be ignored with respect to the optimization parameter z. The reason for the neglect of *logPr(y|z)* is that an given image will be compressed to the same representation y every time, *i.e. Pr(y|z)=1*. Eventually, the original problem is simplified to find out the best z to

maximize *Pr(z)*. The distribution of z is assumed to be a special form of Gibbs distribution and the solution of z' is derived by the steepest gradient descent method. So, MAP is also a kind of iterative deblocking approach.

$$
\begin{aligned}
z' &= \arg\max_{z}\left\{\log\frac{P_r(y\mid z)P_r(z)}{P_r(y)}\right\} \\
&= \arg\max_{z}\left\{\log P_r(y\mid z)+\log P_r(z)-\log P_r(y)\right\} \\
&= \arg\max_{z}\left\{\log P_r(y\mid z)+\log P_r(z)\right\} \\
&= \arg\max_{z}\left\{\log P_r(z)\right\}
\end{aligned}
\qquad (2.4)
$$

3. *Block boundary filtering* [11-15] is the most widely adopted way to resolve the blocking effect in video coding standards. It often plays the role of post-filter in these standards. The popularity of block boundary filtering can be concluded into the two reasons. One is its low computational complexity and the other is that integrating it into the existing coding standards needs less effort. Because of the low requirement in computational complexity, the hardware implementation of the post-filter is also practical. The MPEG-4 standard [11] provides two filtering modes in its optional deblocking filter. DC offset mode is applied to smooth regions while default mode is used for others. DC offset mode is realized by a Gaussian filter and the filtering process conducted in the default mode is to adjust pixel values of those most closest to the block boundary on both sides. The latest coding standard H.264/AVC [12] makes the deblocking filter inside the main coding loop for improving and ensuring the coding performance. It has the characteristic of adaptive deblocking in three hierarchical levels which are slice, block-edge and pixel (sample) levels. This adaptive ability is reached by appropriately controlling the parameter values in these levels. Although the deblocking filter applied outside the coding loop can be free from extra computation requirements and has the largest extent of freedom to adopt different filtering algorithms, it cannot guarantee that the reconstructed video gets rid of annoying blocking effect. To retain high coding performance, it becomes necessary for the codec designer to consider the inclusion of

deblocking. As a result, H.264/AVC finally decides to adopt content-adaptive deblocking filters, which is thought of having the lowest computation cost among all applicable approaches. The new standard enforces the deblocking inside the coding loop, immediately after the frame reconstruction. Through this way, it also ensures that all reconstructed frames are referred after the blocking effect is eliminated.

4. *Filtering in transform domain* [16-18]: In this type of methods, the deblocking operations are conducted on coefficients in the transform domain instead of pixels values in the spatial domain. A well-known one in this type is the overcomplete wavelet representation [18] in which the decoded image is transformed to the three hierarchies of wavelet subbands and then analyzed the occurrence of block discontinuity on two high-frequency wavelet subbands as depicted in Fig. 2.3. After the analysis, the block discontinuity map of the decoded image is generated and a simple 3-tap low-pass filter is applied along every block discontinuity. Next, the deblocked image is obtained after applying inverse wavelet transform. Finally, the enforcement of DCT quantization constraints and pixel range constraints are ensured if images are encoded by using DCT as their transform kernel.

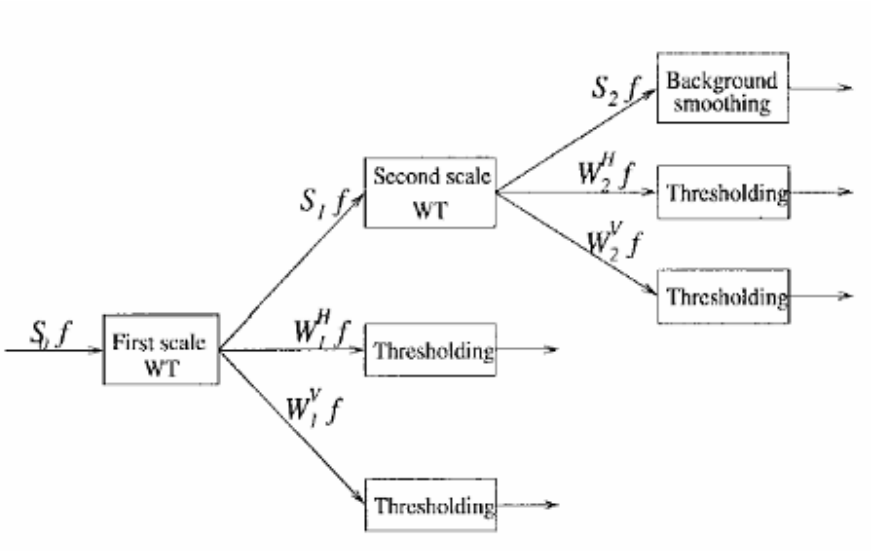


Fig. 2.3: Three-scale wavelet representation

5. *Weighted sums of symmetrically aligned pixels* (WSSAP): Amir Z. Averbuch *et al.* [19] presents a new class of deblocking algorithms named as weighted sums of symmetrically aligned pixels. Because the blocking effect is a phenomenon of the discontinuity between block boundaries, WSSAP removes it via linear or quadratic weights on symmetric pixels. The deblocked value of the pixel $p_{i,j}$ is the sum of linear combination of pixels lying symmetrically to $p_{i,j}$ with respect to the horizontal, vertical central axes and the center of the deblocking frame in size of $S_f \times S_f$, as shown in Eqn. (2.5). Deblocking frame is defined as non-overlapped blocks surrounding and covering the 8×8 blocks. After this procedure, the abnormal discontinuity becomes gradient variation across the block boundary. There are linear or quadratic solutions to obtain the values of weighting factors α, β, γ and δ. One major difference of WSSAP from other methods is that the deblocking process is conducted on all pixels. In the issue of deblocking DC images, WSSAP performs much better than other deblocking approaches.

$$p'_{i,j} = \alpha_{i,j} p_{i,j} + \beta_{S_f - i,j} p_{S_f - i,j} + \gamma_{i,S_f - j} p_{i,S_f - j} + \delta_{S_f - i,S_f - j} p_{S_f - i,S_f - j} \quad (2.5)$$

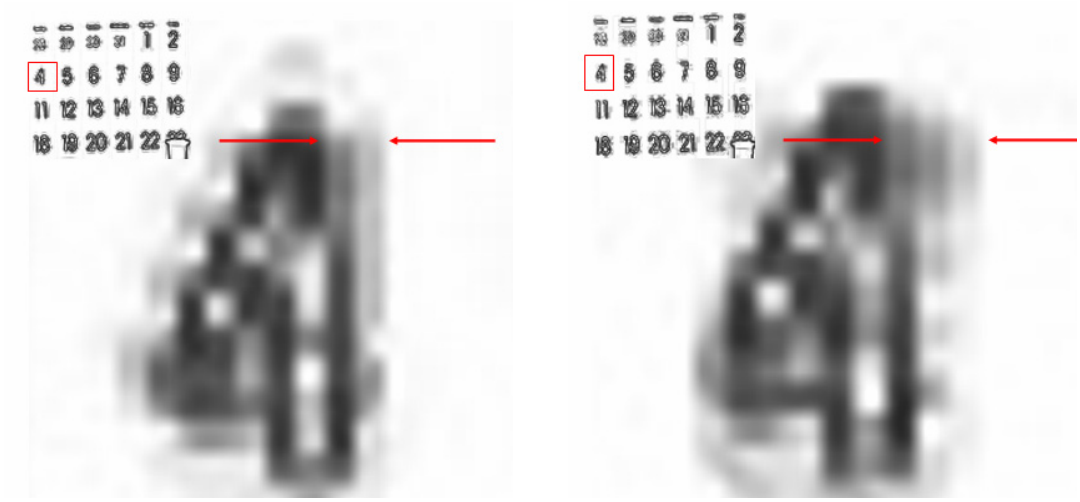
2.3 Blocking Effect Measurements

Finding out a quality assessment model closer to the human visual system (HVS) is always an interesting but difficult research topic in the area of image processing. A reliable model can be used as a metric for evaluating the visual quality of multimedia content which benefits applications in many areas such as Internet streaming, mobile communications and data compression. In the literature, all quality metrics can be classified into two categories: full-reference [20-22] and no-reference [23-26] metrics. In the former case, the inputs are the original content and the reproduced one and the output is the numeric result calculated by the quality evaluation system. As for the latter case, original content is not available or does not exist. Therefore, the reproduced multimedia content is analyzed directly by the quality metric and the numeric measurement representing the distortion level is given. Full-reference metric is often applicable in the image and video coding. For example, in the application of video coding, peak signal-to-noise ratio (PSNR) is widely used as performance evaluation of the coding

system. The object of full-reference metric is to distinguish the perceptual differences of the original and reproduced contents. On the other hand, no-reference metrics try to model the feeling of humans as closely as possible. Human beings do not need to possess a reference to evaluate the visual quality of certain contents. So, the no-reference metric is to give a quantitative measurement of the visual quality for given multimedia data directly. In general, the modeling of no-reference metric is harder than that of full-reference one.

In the process of data compression, the most annoying artifact is the blocking effect. Most video coding standards choose post deblocking filters to remove this artifact. To enhance the performance of deblocking filters, the property of the blocking effect should be investigated in detail, and then it can be detected precisely. With the assistance of the blocking artifact metric, this goal can be reached by developing the enhanced deblocking filters, and therefore, the overall coding performance will also be improved.

Blurring artifact often comes along with blocking effect in the case that the deblocking algorithms are adopted for removing blocking effect, especially when it applies too strong filtering on areas covering sharp object edges. Besides, in the wavelet-based coding such as JPEG2000, the blurring and ringing effects are main distortions introduced in the reconstruction image. Pina Marziliano *et al.* [26] propose the full-reference and no-reference quality metrics for measuring the blurring effect as well as a full-reference metric for the ringing artifact. The basic concept of designing the blurring effect metric is to evaluate the average spread of object edges.



(a) (b)

Fig. 2.4: (a) The original image and (b) the original image after encoding, decoding and strong deblocking.

It is obvious that the blurring artifact becomes more noticeable when the object edge disperses. From the illustration of Fig. 2.4, this idea can be easily approved. The right image is the blurred version of the left one. The width pointed by two red arrows stands for the width of the object edge. It is apparent that Fig. 2.4(b) has longer edge width comparing with that of Fig. 2.4(a) in corresponding positions. Consequently, the first step of the blurring metric is applying the edge detection algorithm, e.g. Sobel filter. The full-reference and no-reference metrics only differ in the input image to be detected. In the prior case, the edge detector is operated on the original image while edge detection is applied to the decoded image in the posterior case. To measure the spread of object edges, the edge width is calculated whose definition is the distance between the local luminance extreme values (i.e. local maximum and local minimum) closest to the edge. To make it more clear, Fig. 2.5 is taken as an example. P_1 is the location of detected edge and its corresponding edge width is the distance between P_2 and P_2'.

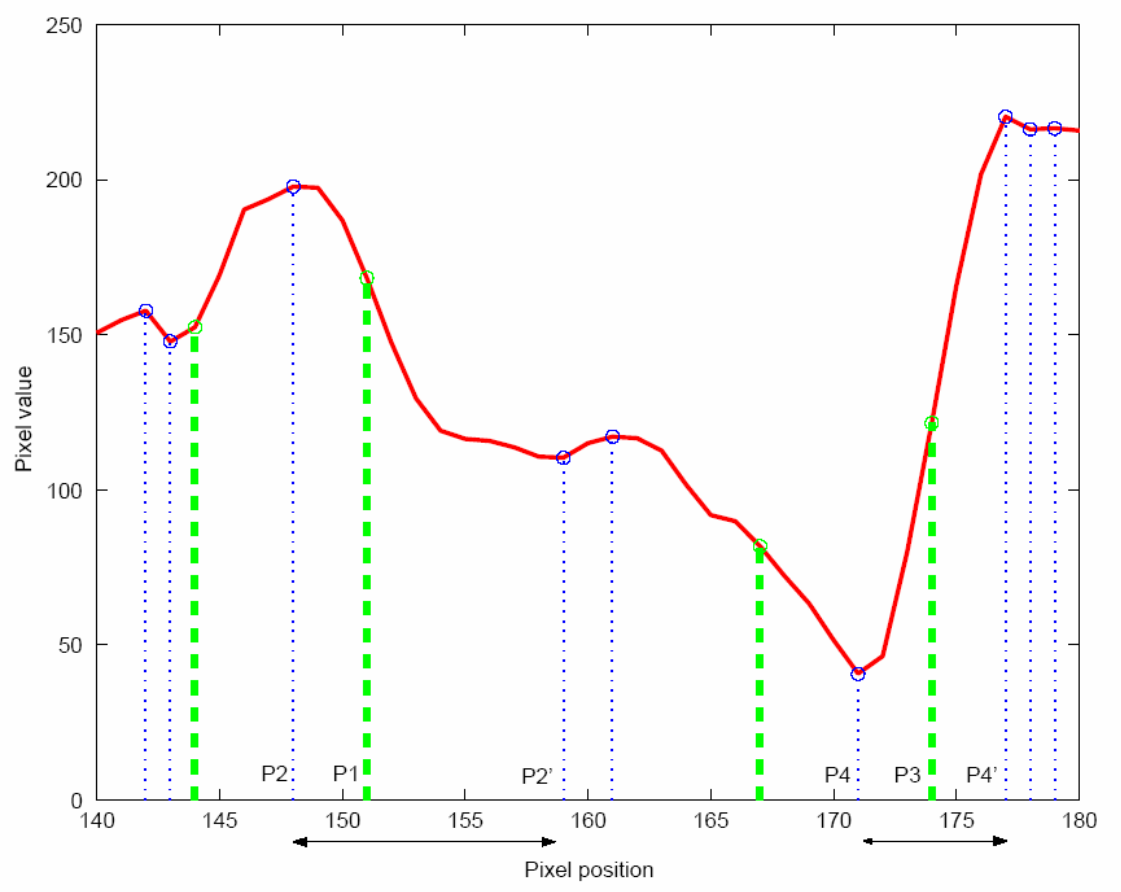


Fig. 2.5: The pixel distribution of one row of the blurred image. The detected edges are marked by dash lines and local extreme values are indicated by dot lines. This figure is cited from [26].

Eventually, the blurring measurement is obtained by calculating the average edge width of all object edges. The full-reference ringing metric also adopts the first procedure used in the blurring metric by inputting original image. Next, the ringing width is acquired by subtracting the edge width from the ring width, in which the ring width is determined given a prior knowledge of the wavelet decomposition. To analyze the ringing effect more precisely, a difference image is generated by subtracting the original image from the decoded image. The ringing distortion for each edge, called ringing measure, is obtained by multiplying the difference between the maximum and minimum of the difference image inside its corresponding ringing region and the ringing width as formulated in Eqn. (2.6). L1 and L2 are the original and reconstruction images as illustrated in Fig. 2.6. The

distance between P_3 and P_3' is the ringing width. As a result, the overall ringing measurement of an image is the average of all ringing measures.

$$ringing\ measure = | max(L_1 - L_2) - min(L_1 - L_2) | \times | P_3^{'} - P_3 | \quad (2.6)$$

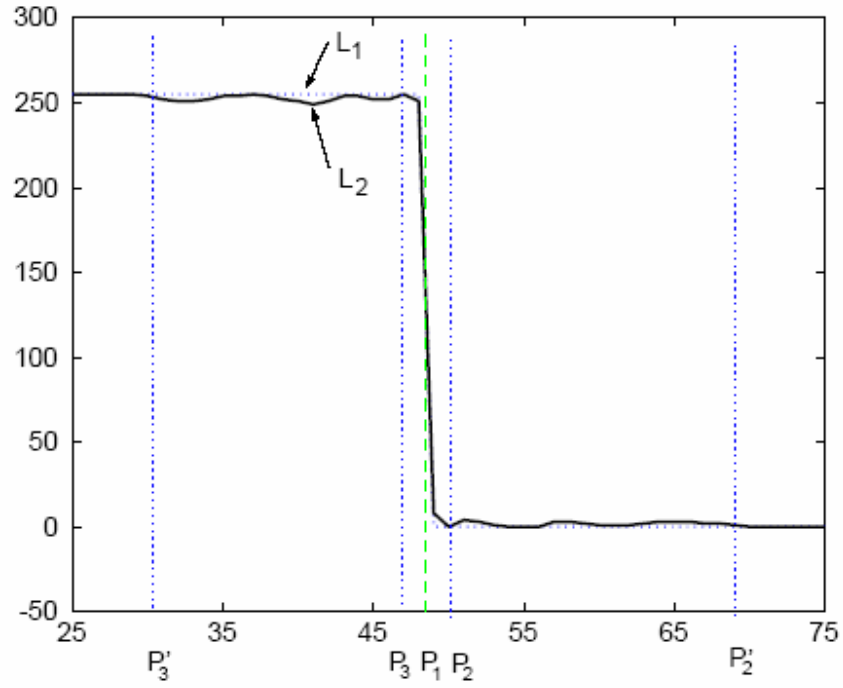


Fig. 2.6: One row of the original and reconstruction images. This figure is cited from [26].

The work in [20] is a full-reference metric for evaluating the blocking artifacts based on human visual sensitivity. The measurement of the blocking distortion includes the following procedures: edge detection, masking effect consideration and nonlinear transform. Edge detection is to find out regions affected by the blocking effect. Masking effect is one of the main phenomena of HVS, which reflects the different visual sensitivity for the same distortion energy. In this step, texture (activity) masking and luminance masking are both employed in the proposed blocking metric. The final step, nonlinear transform, is to match the nonlinear processing of the HVS. In every step, the required parameter thresholds are obtained from various subjective experiments with the

variation of the edge amplitude, edge length, background luminance and background activity.

Perceptual blocking distortion metric (PBDM) [21] also belongs to the class of full-reference quality assessment metrics. The original and reproduced sequences are independently processed by a series of vision-model-based operations such as temporal filtering, steerable pyramid decomposition, contrast sensitivity function filtering, and contrast gain control. After these procedures, quality evaluations named as sensor outputs are given to the original and reproduced sequences respectively. Finally, the quality distortion measurement is a squared error norm of the difference of between the sensor outputs of the original sequence and the reproduced sequence. The main contribution of this proposal [21] is addressing the concept that different types of distortions are predominant in different regions. For example, blocking effect is more obvious in smooth regions while ringing effect often take places in object edges. To obtain the measurement of the blocking effect more aligned to the HVS, identifying regions mainly distorted by the blocking effect is also important. Consequently, the blocking impairment metric presented in [21] first detects the occurrence of vertical and horizontal blocking artifacts, respectively, via the one-dimensional waveform and the characteristics of six consecutive points across block boundaries. Next, the *blocking region map* is generated by removing real edges. The real edges are determined by finding out the common blocking existing both in the original and the reproduced one. Short isolated edges in the reproduced sequence, those located at regions dominated by the ringing distortion are also removed from the blocking region map. At the end, the measurement of the distortion level of the blocking effect is represented by the mean of squared error (MSE) between two *blocking region maps*.

Some quality assessment metrics globally analyze the blocking artifact, whereas others [25] explore the local characteristics around block boundaries. In [25], it investigates the local characteristics not only for blocking effect but also for the blurring artifact. Besides,

the output range of these two sorts of distortions is well-defined, *i.e.* ranging from zero to ten. The larger the output value, the more serious distortion will be observed in that region. For every 8×8 block boundary, it is given two numerical results. One describes the blocking effect and the other stands for the blurring effect. After considering the just noticeable distortion (JND) [27], the quality evaluation result for each 8×8 block boundary is determined by choosing the larger one between two distortion values. This behavior conforms to the idea in [21] that different distortions are predominant in different regions. Next, the quality measurement for each 8×8 block is the average of quality assessment outputs of its horizontal and vertical boundaries. Finally, the overall image quality is obtained by averaging quality assessment outputs of all 8×8 blocks. The blocking effect is measured by a ratio where the numerator is the pixel value difference cross the block boundary and denominator is the texture variation in the nearby region close to the block boundary. To evaluate the distortion resulting from the blurring effect, a pre-defined zero crossing function is employed. The goal of the zero crossing function is to judge whether the consecutive data points having the same pixel value or not. Thus, the measurement of the blurring effect is the ratio of the zero crossing for reference and reproduction images in the region around the block boundary.

The generalized block-edge impairment metric (GBIM) [23] takes advantage of the property of luminance masking effect when modeling the distortion incurred by the blocking effect. In the study of [28], it points out that distortions are most noticeable when the luminance value is between 70 and 90. Therefore, in the work of [23], a weighting function is proposed to approximate the phenomenon of luminance masking effect. It gives larger weights on areas whose luminance values range from 70 to 90. In [20], the luminance masking effect has also been integrated into the blocking artifact metric but it needs complicated subjective experiments to obtain the parameter values used in the luminance masking model as compared to GBIM.

Chapter 3
System Framework

3.1 Adaptive Deblocking of H.264/AVC

In order to reduce the blocking effect and enhance the compression efficiency, H.264/AVC puts the deblocking filter inside the main coding loop. With adopting the in-loop deblocking filter, the encoder always takes deblocked frames as references in the motion estimation process. Comparing with the frame before applying the deblocking filter, the frame after the filtering is not only closer to the original frame but also contains less blocking artifacts. This quality improvement for inter-frame prediction can make the estimated motion vector more accurate, hereafter. Further, different from the previous coding standards the 4×4 block is the minimal processing unit of block prediction and block transform in H.264/AVC. The deblocking process should scan all edges of 4×4 blocks within each macroblock so as to smooth out all occurrences of artificial blocking boundaries. Additionally, according to the design, the adopted deblocking filter is adaptive to the coding conditions and content properties, either in terms of filtering types or filtering strength. To understand the whole deblocking process, we need to view it from three hierarchical levels: the slice level, the block-edge level and the pixel (sample) level, as described below.

3.1.1 The Slice Level

The parameters in the slice level delineate the global characteristics of the current video slice. In H.264/AVC, a frame may be split into one or several slices and each slice has its own characteristics. The adaptability of H.264/AVC deblocking filter is achieved by adjusting the encoder-selectable offsets, referred to as *OffsetA* and *OffsetB*. These offsets are both even numbers within the range (-12 to 12), inclusive, and are used to adjust the filter strength. Since a fixed filtering operation may under- or over-smooth the

reconstructed frame, finding a good parameter set, i.e. a pair of (*OffsetA*, *OffsetB*), can help to regulate the filtering thresholds, and therefore, optimizes the subjective quality. As a result, how to classify similar macroblocks into the same slice and find out the best parameter set is the major work for bringing the deblocking to its maximal effect. Moreover, by using the tool "flexible macroblock ordering" (FMO), it is possible to segment different objects and then code them in each independent slice, which provides the chance for further subjective improvements.

3.1.2 The Block-edge Level

At this level, for each edge between two adjacent 4×4 blocks, the Boundary-Strength (*BS*) parameter is assigned an integer value from 0 to 4, according to some pre-defined criteria at the block or macroblock level. For example, the following criteria are applied: Is the neighboring block intra- or inter-coded? , Is the edge a macroblock edge? , Does the block have coded residuals? , Are their motion vectors and reference frames the same? The detail description for deciding the *BS* parameter is described in Table 3.1. In our implementation, *BS* determines the filter strength performed on the edge: '*BS* = 0' implies no filtering is applied, while '*BS* = 4' allows the application of the longest and strongest filtering. Otherwise, the short filtering is applied. In addition, the deblocking filter depends also on the average quantization parameter (QP) of the two blocks adjacent to the edge. With a larger QP, the filtering strength is likely to be stronger so as to smooth out the large block discontinuity incurred in the low-quality reconstruction; while with a smaller QP the visual quality of the decoded frame is guaranteed to be above a certain extent, thus, the deblocking filter is applied less probably and with less strength.

Table 3.1. The coding mode based decision for the parameter Boundary-Strength (*BS*)

Block modes and conditions	*BS*
At least one of the blocks is Intra coded and the edge is a macroblock edge	4
At least one of the blocks is Intra coded	3
One of the blocks has coded residuals	2
Difference of luma block motion in one or both directions $\geqq 1$	1
Motion compensation from different reference frames	1
Otherwise	0

3.1.3 The Sample Level

After deciding the strength of each edge, the next step is to dynamically enable the chosen filters, which are applied to all sample pairs across every 4×4 block boundary. For each pair of boundary pixels, we first analyze the discontinuity across the block boundary, and then judge that whether the discontinuity corresponds to the blocking artifact or is a real object/texture edge. The accurate judgment helps to reduce the visibility of the artificial edges and, at the same time, preserve the sharpness of the object edges. The judgment is accomplished by checking the absolute differences between several pairs of samples across the block boundary, as formulated in Eqns. (3.1) to (3.7) where p_2, p_1, p_0, q_0, q_1 and q_2 are sample values inside the two neighboring 4×4 blocks as labeled in Fig. 3.1(a). The determination of threshold values, α and β, will be addressed in the following paragraph. If the differences of all pixel pairs are smaller than the corresponding threshold, as shown in Table 3.2, this boundary is marked as an artificial

edge and filtered based on the parameter *BS*, determined at the block-edge level. Otherwise, it is marked as an object edge and no smoothing operations will be applied. As shown in Fig. 3.1, there are 9 possible deblocking situations, with 0 to 6 pixel changes, determined for every edge between two neighboring 4×4 blocks.

$$Flag1 = Abs(p_0 - q_0) < \alpha(Index_A) \quad (3.1)$$

$$Flag2 = Abs(q_1 - q_0) < \beta(Index_B) \quad (3.2)$$

$$Flag3 = Abs(p_1 - p_0) < \beta(Index_B) \quad (3.3)$$

$$Flag4 = (Abs(p_2 - p_0) < \beta(Index_B)) \quad (3.4)$$

$$Flag5 = (Abs(q_2 - q_0) < \beta(Index_B)) \quad (3.5)$$

$$Flag6 = (Abs(p_0 - q_0) < ((\alpha(Index_A) >> 2) + 2) \quad (3.6)$$

cFlag = Is it a chroma edge? (3.7)

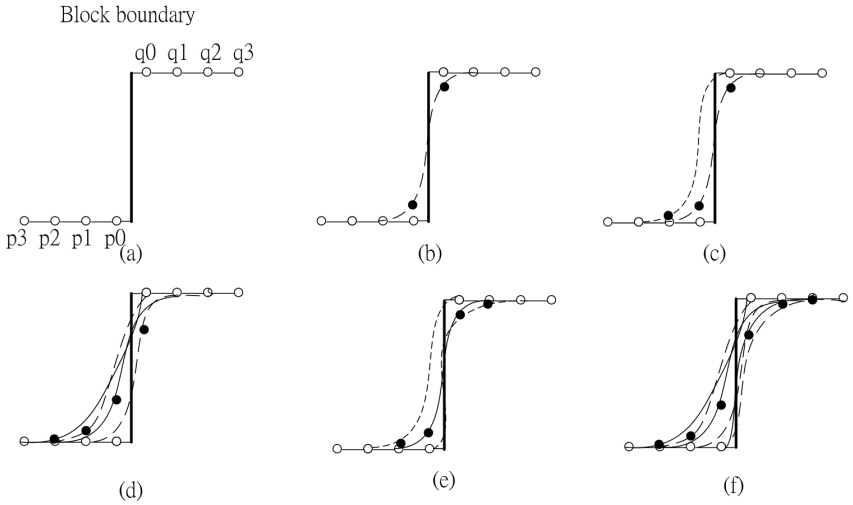


Fig. 3.1: **(a)** When *BS* equals to 0, no pixel at both sides of the edge will be changed. **(b)** The nearest pixels next to the edge at both sides may change, while *BS* ranges from 1 to 4. **(c~f)** The second (and the third) nearest pixel(s) may change at one side or both sides while *BS* equals 1 to 3 (4). There are at most 4 and 6 pixels changed if *BS* varying from 1 to 3 and 4, respectively.

The determination of thresholds is crucial since it directly affects the enabling flag. The thresholds, α and β, specified in H.264/AVC are functions of *IndexA* and *IndexB*. Their

values are calculated approximately according to Eqns. (3.8) and (3.9). In fact, *IndexA* and *IndexB* are decided by the values of quantization parameters (*QP*) and slice level parameters, *OffsetA* and *OffsetB*, as given by Eqns. (3.10) and (3.11).

$$\alpha(Index_A) = 0.8(2^{Index_A/6} - 1) \quad (3.8)$$

$$\beta(Index_B) = 0.5 \times Index_B - 7 \quad (3.9)$$

$$Index_A = Min(Max(0, QP + Offset_A), 51) \quad (3.10)$$

$$Index_B = Min(Max(0, QP + Offset_B), 51) \quad (3.11)$$

In Table 3.2(a), the value in the last column referred to y should be clipped in the pre-defined range to avoid too much low-pass filtering. x denotes the original pixel value and x' is the value after clipping. The clipped range is x-c_0 to x+c_0 for p_0 and q_0 as well as x-c_1 to x+c_1 for p_1 and q_1.

$$x' = ICLIP(y, x - c_0, x + c_0) \quad (3.12)$$

$$x' = ICLIP(y, x - c_1, x + c_1) \quad (3.13)$$

The value of c_1 is determined by a 2-D look-up table with $Index_A$ value used as one dimension and BS value as the other. c_0 is equal to c_1 plus 1.

Table 3.2: The relationships among enabling flags and the pixels to be filtered: (a) BS = 1 to 3 and (b) BS = 4, where "**T**", "**F**" and "**-**" stand for that the specified flag must be "true", "false" and "don't care", respectively. The filter coefficients are shown in the last column. The filtering is operated when all flags conform to any of row specified.

(a)

Pixel	Flag1	Flag2	Flag3	Flag4	Flag5	CFlag	Filter coefficient $[p_2, p_1, p_0, q_0, q_1, q_2]$
p_0	T	T	T	-	-	-	$[0, \frac{1}{8}, \frac{1}{2}, \frac{1}{2}, -\frac{1}{8}, 0]$
q_0	T	T	T	-	-	-	$[0, -\frac{1}{8}, \frac{1}{2}, \frac{1}{2}, \frac{1}{8}, 0]$
p_1	T	T	T	T	-	F	$[\frac{1}{2}, 0, \frac{1}{4}, \frac{1}{4}, 0, 0]$
q_1	T	T	T	-	T	F	$[0, 0, \frac{1}{4}, \frac{1}{4}, 0, \frac{1}{2}]$

(b)

Pixel	Flag4	Flag5	Flag6	CFlag	Filter coefficient $[p_3, p_2, p_1, p_0, q_0, q_1, q_2, q_3]$
p_0	T	-	T	F	$[0, \frac{1}{8}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{8}, 0, 0]$
p_1	T	-	T	F	$[0, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, 0, 0, 0]$
p_2	T	-	T	F	$[\frac{1}{4}, \frac{3}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, 0, 0, 0]$
q_0	-	T	T	F	$[0, 0, \frac{1}{8}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{8}, 0]$
q_1	-	T	T	F	$[0, 0, 0, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, 0]$
q_2	-	T	T	F	$[0, 0, 0, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{3}{8}, \frac{1}{4}]$
p_0	-	-	F	-	$[0, 0, \frac{1}{2}, \frac{1}{4}, 0, \frac{1}{4}, 0, 0]$
	F	-	-	-	
	-	-	-	T	
q_0	-	-	F	-	$[0, 0, \frac{1}{4}, 0, \frac{1}{4}, \frac{1}{2}, 0, 0]$
	-	F	-	-	
	-	-	-	T	

3.2 Proposed System Framework

As prescribed, H.264/AVC deblocking filter provides two encoder-selectable parameters, *OffsetA* and *OffsetB*, to adaptively adjust the strength of filtering. To further optimize

coding performance, pre-process and post-process should be jointly considered. Here, an enhanced encoding architecture for H.264/AVC is proposed as depicted in Fig. 3.2. Every input frame is encoded by using a two-pass algorithm. In the first pass, post-processing stage, the appropriate *OffsetA* and *OffsetB* are determined for each slice/frame based on the measurements of blocking artifacts. Filtering modes for all edges under the chosen (*OffsetA*, *OffsetB*) pair are stored, because they are necessary in the subsequent procedures. In the second pass, pre-processing stage, every pixel vector lying over all 4×4 block boundaries will be pre-processed. If a pre-processed macroblock results in a lower bitrate and also a lower mean square error (MSE) value, after deblocking, than the non-pre-processed one, the pre-process macroblock replaces the original macroblock.

Due to small block size in H.264/AVC, a pixel may be influenced by 4 times of filtering, each for one of its four surrounding edges. Consequently, if a pixel is modified during the current pre-process, its most updated value will be used as the input to the next pre-process. Finally, after the second pass, the reconstruction is stored into the frame buffer for next frame encoding. The kernel of the proposed encoding framework is composed of two components: post-processing and pre-processing. The detail descriptions of them will be presented in the next two chapters.

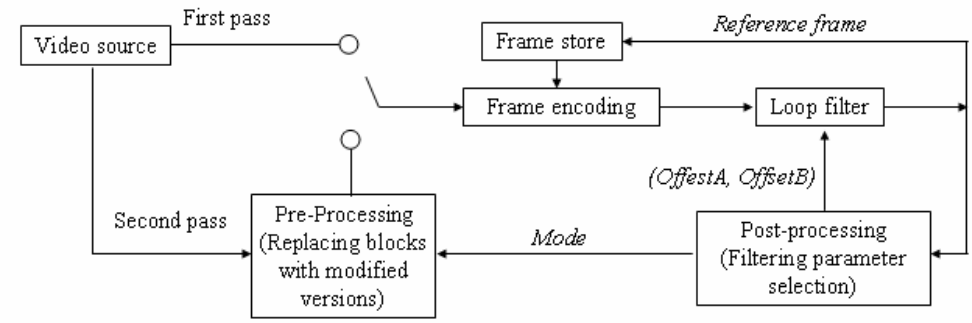


Fig. 3.2: The proposed architecture for H.264/AVC encoder.

Chapter 4
The Post-processing

4.1 Boundary-Energy Sensitive Deblocking Algorithm

Based on the design of H.264/AVC deblocking filter, the encoder can filter individual frames or individual parts of a frame (coded in a slice) with different settings. The encoder can adjust the values of *OffsetA* and *OffsetB* at the slice level to enable the filter and to vary filter strengths. Since each frame has different texture/color characteristics and will be coded under different bitrate budgets, applying the same filter setting throughout the whole video sequence as did in the reference software [11] will not bring the in-loop deblocking process into its full play. Consequently, an algorithm to explore the most adequate threshold offset is desired, which can refine the decoded frame, slice or region to the one having the best visual quality. The proposed algorithm provides an efficient way for finding out the most suitable *OffsetA* and *OffsetB* for each individual frame (or slice) that optimizes the resultant objective quality or minimizes the perceived blocking energy, or both.

Because the blocking effect results from the discontinuity across the block boundaries, investigating the mismatch energy around block boundary will help to determine the parameters (*OffsetA*, *OffsetB*). Moreover, applying too strong filtering results in blurring. Thus, it is beneficial to also have a criterion of blurring measurement. In H.264/AVC, it allows these two offsets to be any even numbers ranging from -12 to 12 (inclusive), and as a result, there are totally 169 (13×13) combinations. It is a time consuming work if the encoder tries all possibilities for obtaining the best pair of (*OffsetA*, *OffsetB*). To avoid it, the proposed algorithm translates the original problem into a search problem in a 2-dimensional space, with *OffsetA* as one axis and *OffsetB* as another, and then finds the best pair of (*OffsetA*, *OffsetB*) based on a specific cost function. The exploration of the

cost function, which reflects the perceived visual quality, and the detail of the proposed fast search algorithm are described in Sections 4.2 and 4.3, respectively.

4.2 Blocking Effect and Blur Degree Analysis

In order to determine the best deblocking parameters, a criterion function for representing the frame's fineness is required. Although the overall distortion energy, at a certain extent, represents a frame's quality, it is not effective for the case that a certain type of error patterns is known in advance. For example, two reconstructions of the same image may have similar PSNR values but one may suffer from obvious blocking artifact or excessive blurring while the other does not. Thus, in our implementation, instead of using PSNR alone we define two new measurements for representing how serious the blocking effect or blur is perceived. One is signal-based impairment measurement for block discontinuity energy, and the other is perceptual-based measurement of the blocking and blurring artifacts.

4.2.1 Block Discontinuity Energy

From the signal-based viewpoint of distortions, a measurement of the blocking discontinuity, *blocking_degree (BD),* is defined to determine the most appropriate parameters for deblocking filters. By analyzing the distortion frame, the block discontinuity energy is calculated by summing differences of pixel pairs in the distortion frame, which are categorized to boundary pixels and have been filtered by H.264/AVC deblocking filter, as formulated below.

$$BD = \frac{\sum_{i=0}^{height-1}\sum_{j=0,j\%4==0}^{j=width-4}\{[D(i,j)-D(i,j-1)]^2\times(1-\delta(Bs))\}+\sum_{j=0}^{j=width-1}\sum_{i=0,i\%4==0}^{i=height-4}\{[D(i,j)-D(i-1,j)]^2\times(1-\delta(Bs))\}}{\sum_{i=0}^{height-1}\sum_{j=0,j\%4==0}^{j=width-4}(1-\delta(Bs))+\sum_{j=0}^{j=width-1}\sum_{i=0,i\%4==0}^{i=height-4}(1-\delta(Bs))}$$

(4.1)

where D is obtained by subtracting the reconstruction (U) from its original frame (I) for every column and row (indexed as i and j), and δ is the delta function, as shown in Eqns. (4.2) and (4.3), respectively.

$$D(i, j) = I(i, j) - U(i, j) \tag{4.2}$$

$$\delta(Bs) = \begin{cases} 1, & Bs = 0 \\ 0, & others \end{cases} \tag{4.3}$$

Note that $D(i, j)$-$D(i, j\text{-}1)$ denotes an approximation of the neighboring distortion gradient, and a large value of this term implies an apparent blocking effect. Hence, *BD* is obtained by computing the gradient difference energy across block boundaries. The larger the *BD* is, the more perceptible the blocking effect does.

As prescribed in Chapter 3, the visual deblocking by H.264/AVC, at most, smoothes out 6 pixels across each boundary location, correctly detecting the occurrence of blocking boundaries and removes undesired discontinuity, and therefore, reduces mismatch energy. Therefore, the mean square error (MSE), which represents the average error signal energy, is also taken into account in our criterion function. The final criterion used throughout our work is the *weighted_blocking_degree* (*WBD*), which is a linearly weighted result of *BD* and *MSE*, as defined in Eqn. (4.5), where the value of γ is set to 0.7 empirically. That is,

$$MSE = \sum_{i=0, j=0}^{i=height-1, j=width-1} D^2(i, j) \Big/ height \times width \tag{4.4}$$

and

$$WBD = \gamma \times BD + (1.0 - \gamma) \times MSE. \tag{4.5}$$

4.2.2 Perceptual-based Measurement for Blocking and Blurring Artifacts

In addition to employing *WBD* to evaluate the degree of blocking effect, another measuring function is proposed to compute the visual impaction caused by blocking or blurring effect instead of evaluating them by signal comparison. From the formulation of (4.6), the main concept is to measure the perceptual difference between the original frame and the one processed by deblocking filters. To this end, we first detect the occurrences of blocking and blurring artifacts. For every block boundary, it is given two measurement values, *B* and *Z*, for these two types of artifacts and multiplied by a weighting factor, *w*. The value of *w* is derived from the phenomenon of luminance masking effect. Moreover, these procedures are both done in original frames and filtered ones to discover the difference in degree of distortion between them. The values of *w*, *B* and *Z* are obtained from the original frame while those of *w'*, *B'* and *Z'* are calculated from the filtered ones. The numbers of the total horizontal and vertical boundaries are *m* and *n*, respectively. *Blocking_Diff* computes the sum of absolute difference of the corresponding *wB* and *w'B'*. The definition of *Blurring_Diff* is similar with that of *Blcoking_Diff* with inputs being *Z* and *Z'* instead of *B* and *B'*. Finally, the adopted vector of (*OffsetA, OffsetB*) controlling the behavior of in-loop filter is the one having minimum value of *Distortion* among all 169 possibilities. γ is set to 0.5 in our implementation provided that the blocking and blurring artifacts give the same level of distortions for human eyes. That is,

$$Distortion = \gamma \times Blocking\_Diff + (1-\gamma) \times Blurring\_Diff \tag{4.6}$$

$$Blocking\_Diff = \sum_{i=1}^{m} \left| w_{h_i} B_{h_i} - w'_{h_i} B'_{h_i} \right| + \sum_{j=1}^{n} \left| w_{v_j} B_{v_j} - w'_{v_j} B'_{v_j} \right| \tag{4.7}$$

$$Blurring\_Diff = \sum_{i=1}^{m} \left| w_{h_i} Z_{h_i} - w'_{h_i} Z'_{h_i} \right| + \sum_{j=1}^{n} \left| w_{v_j} Z_{v_j} - w'_{v_j} Z'_{v_j} \right|. \tag{4.8}$$

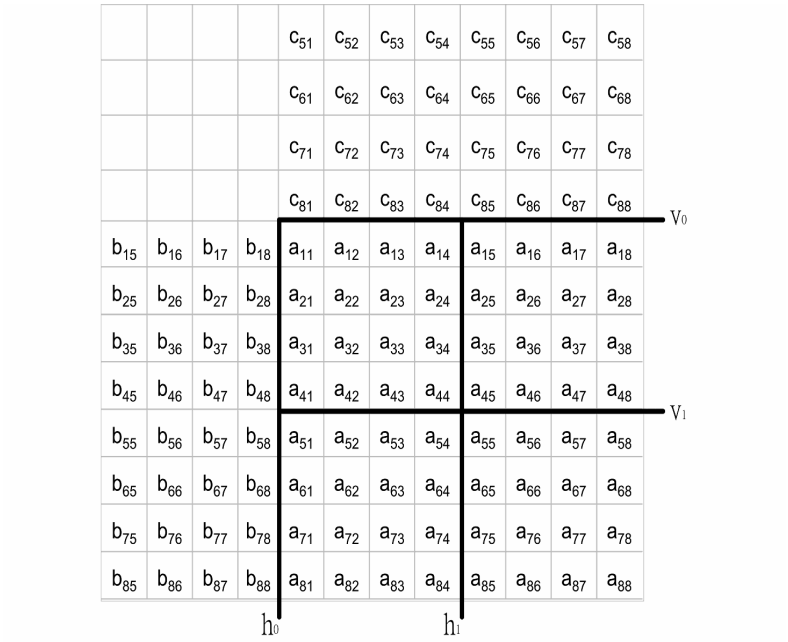


Fig. 4.1: v_0 and h_0 are the 8×8 block boundaries while v_1 and h_1 are the internal boundaries of four 4×4 blocks.

To analyze the blocking and blurring effect, local activities around each block boundary are explored. The idea comes from [25]. In their work, two values are computed for all 8×8 boundaries. One represents the severity of the blocking effect and the other is the measurement of blurring artifact. The artifact measurement for vertical boundaries is similar with that of horizontal ones. Therefore, the derivation of the criteria function is taking one vertical boundary of the 8×8 block, h_0, as example hereafter. The strength of the blocking effect represented as B_{h0} is calculated by following formulas.

$$B_{h_0} = \begin{cases} \dfrac{N_{h_0}}{D_{h_0}}, D_{h_0} \neq 0 \\ 0, D_{h_0} = 0 \end{cases} \tag{4.9}$$

The value of blocking effect measurement is composed of a fraction with N_{h0} as the numerator and D_{h0} being the denominator. N_{h0} calculates the discontinuity across block boundaries while D_{h0} detects the texture changes near block boundaries. If the block

boundary is located at smooth regions, small discontinuity across the boundary will be more noticeable than that at the high textured ones. Therefore, it is reasonable that B_{h0} is a relative relationship between local activities and the discontinuity across boundaries. γ_1 and γ_2 are set to 10 and 1.5 according to [25] which enforces B_{h0} ranging from 0 to 10.

$$N_{h_0} = \gamma_1 \times \sum_{i=1}^{8} \left| a_{i1} - b_{i8} \right|, \tag{4.10}$$

$$D_{h_0} = \gamma_2 \times \sum_{i=1}^{8} \left(\sum_{j=5}^{7} \left| b_{i(j+1)} - b_{ij} \right| + \sum_{j=1}^{3} | a_{i(j+1)} - a_{ij} | \right) + \sum_{i=1}^{8} | a_{i1} - b_{i8} | . \tag{4.11}$$

When the content is encoded in low bitrate and/or the strength of the deblocking filer is too strong, the blurring effect becomes more obvious. To model this kind of distortion, the percentage of zero crossing in the neighborhood of block boundaries is computed. Z_{h0} stands for the level of blurring of the boundary h_0 and $z(x, y)$ defines the zero crossing function. There are 56 different pixel pairs to be evaluated and the scaling factor 10 is used to force the value of Z_{h0} to fall into the range of [0, 10]. That is,

$$Z_{h_0} = \frac{10}{56} \left[\sum_{i=1}^{8} \left(\sum_{j=5}^{7} z(b_{ij}, b_{ij+1}) + \sum_{j=1}^{3} z(a_{ij}, a_{ij+1}) \right) + \sum_{i=1}^{8} z(a_{i1}, b_{i8}) \right] \tag{4.12}$$

$$z(x, y) = \begin{cases} 1, & | x - y | = 0 \\ 0, & otherwise. \end{cases} \tag{4.13}$$

The original algorithm in [25] only examines the 8×8 block boundaries. Some modifications are needed to meet the behavior of H.264/AVC in-loop deblocking filer. In H.264/AVC, every 4×4 block boundary is filtered so the two internal block boundaries inside an 8×8 block, *v1* and *h1* displayed in Fig. 4.1, should be also taken into account. Furthermore, masking effect is one of the important phenomena affecting the human visual system (HVS). Thus, the proposed criteria function incorporates the luminance masking function proposed in [23]. From the study in [28], it points out that when the

luminance value is between 70 and 90, distortions are most noticeable. Accordingly, distortions located at regions where luminance value in the range of [70, 90] would have a greater visual impact and should be assigned higher weighting values. In [23], each pixel on the block boundary has its own weighting factor. However, a weighting factor is given to an 8-pixel boundary in our proposed criteria function due to the distortion being measured per 8-pixel boundary. w_{h0} is the weighting factor of the boundary h_0. The definition of w is to approximate the property of luminance masking effect as depicted in Fig. 4.2.

$$w_{h_0} = \begin{cases} \lambda \ln(1 + \dfrac{\sqrt{\mu_{h_0}}}{1+\sigma_{h_0}}), if\ \mu_{h_0} \le \zeta \\ \ln(1 + \dfrac{\sqrt{255-\mu_{h_0}}}{1+\sigma_{h_0}}), otherwise \end{cases} \tag{4.14}$$

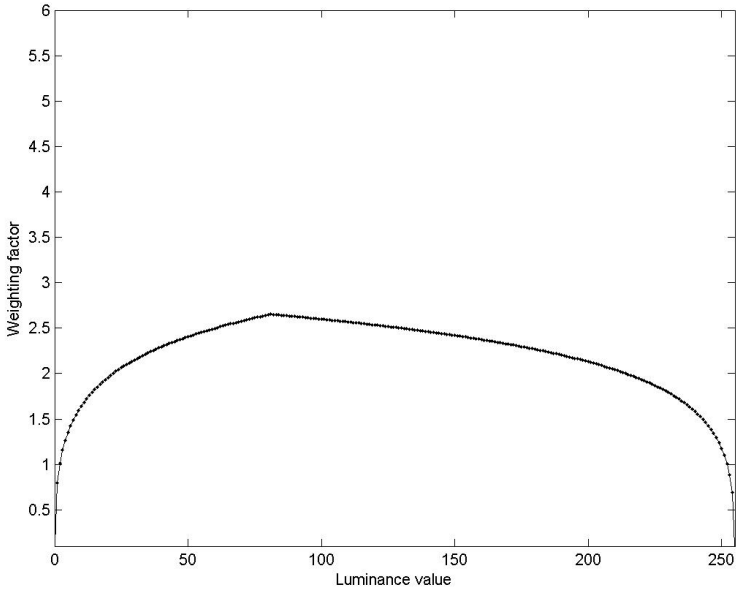


Fig. 4.2: The behavior of luminance masking function with $\zeta = 81$ and $\sigma_{h0} = 0$.

ζ is the selected luminance value given the highest weighting value. In the following experiments, ζ is set to 81 and λ is defined as

$$\lambda = \frac{\ln(1+\sqrt{255-\zeta})}{\ln(1+\sqrt{\zeta})} \tag{4.15}$$

Weighting function is dominated by the background luminance value. Hence, μ_{h0} is obtained by calculating the average luminance value while σ_{h0} is the standard deviation on both sides of the boundary h_0.

$$\mu_{h_0} = \frac{\mu_a + \mu_b}{2} \tag{4.16}$$

$$\mu_a = \frac{1}{32}\sum_{i=1}^{8}\sum_{j=1}^{4} a_{i,j} \tag{4.17}$$

$$\mu_b = \frac{1}{32}\sum_{i=1}^{8}\sum_{j=5}^{8} b_{i,j} \tag{4.18}$$

σ_{h0} is defined as the average value of σ_a and σ_b which are the standard deviations of pixels located at the right and left sides of $h_{0,}$ respectively.

$$\sigma_{h_0} = \frac{\sigma_a + \sigma_b}{2} \tag{4.19}$$

$$\sigma_a = \sqrt{\frac{1}{32}\sum_{i=1}^{8}\sum_{j=1}^{4}(a_{i,j} - \mu_a)^2} \tag{4.20}$$

$$\sigma_b = \sqrt{\frac{1}{32}\sum_{i=1}^{8}\sum_{j=5}^{8}(b_{i,j} - \mu_b)^2}\ . \tag{4.21}$$

4.3 Effective Search Algorithms

As above-mentioned, the original problem of choosing the best (*OffsetA*, *OffsetB*) is translated into a search problem in a 2-D space with *OffsetA* as one axis and *OffsetB* as another. After this alteration, search algorithms are needed to discover the appropriate (*OffsetA*, *OffsetB*). Although exhaustive search can ensure the optimal solution, the demanding computational complexity is very high. To overcome this problem, two new

efficient search algorithms, the predicted diamond search and the predicted local square search, are introduced in the following paragraphs.

4.3.1 Predicted Diamond Search (PDS)

PDS is a predictor-centric coarse-to-fine search method. The search procedure is composed of four steps as shown in Fig. 4.3. They are start-point initialization, large diamond search, small diamond search and linear search along *OffsetA*-axis in order.

In the first step, start-point initialization, the initial value of (*OffsetA*, *OffsetB*) is set to that of the slice at the same location of the previous frame. For the case of the first frame or the scene-change frame the central value (0, 0) is used instead. The initial value is treated as the center point of the large diamond search, and criteria evaluation is conducted on all points of the search pattern. The searching procedure iterates one to several times, which moves its center to the point having the optimal cost until the center of the searching pattern is the optimal among all neighbors. And in the follow-up, the small diamond search is applied for finer adjustment. The patterns of the large and the small diamond searches are illustrated in Fig. 4.4.

Next, after applying all diamond searches, linear search along *OffsetA*-axis is conducted. Since H.264/AVC deblocking filter is more sensitive to the variation of *OffsetA* than that of *OffsetB*, the proposed algorithm linearly searches all possible values for *OffsetA* along the fixed *OffsetB*, which prevents the search from falling into local optimal. Finally, the most appropriate (*OffsetA*, *OffsetB*) for the current frame is found. The result is also taken as the initial start point for the next frame.

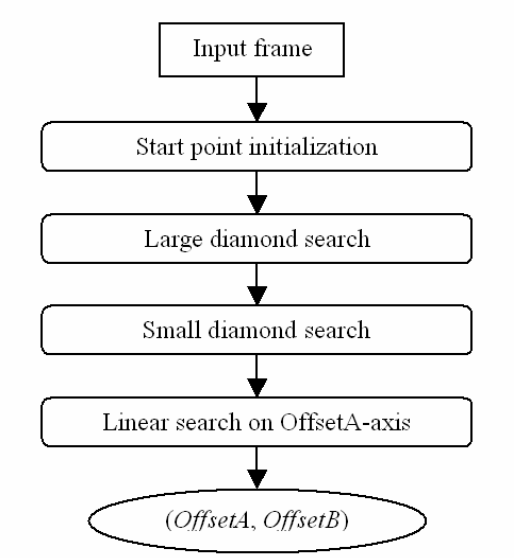


Fig. 4.3: The steps of the proposed predicted diamond search (PDS).

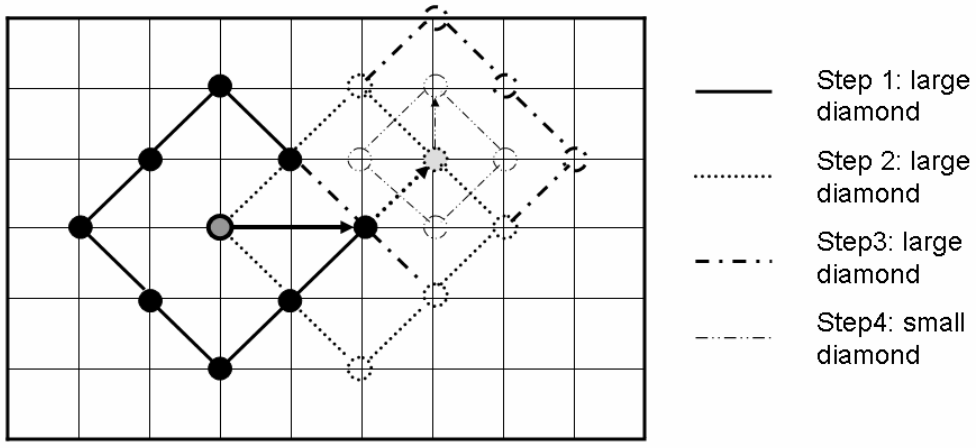


Fig. 4.4: An example to show search patterns during the large and the small diamond searches employed by the PDS. The dark grey and light grey points are the center points of the large and the small diamond searches, respectively.

4.3.2 Predicted Local Square Search (PLSS)

PLSS is a predictor-constrained local search method. This search is simpler and faster than PDS, and its procedure is illustrated in Fig. 4.5. In this method, if the input frame is the leading frame of a sequence or the first frame of a new scene, we apply PDS to get a

precise initiation of (*OffsetA*, *OffsetB*). Otherwise, we initiate the value of (*OffsetA*, *OffsetB*) to the one copied from the previous frame. After the initiating step, we take the so-obtained (*OffsetA*, *OffsetB*) as the center point to apply one to two times of square search. Only when the best value in the first run is not at the centered point, the second square search will apply. By this way, at most 12 times of evaluation are made. One example of PLSS is illustrated in Fig. 4.6.

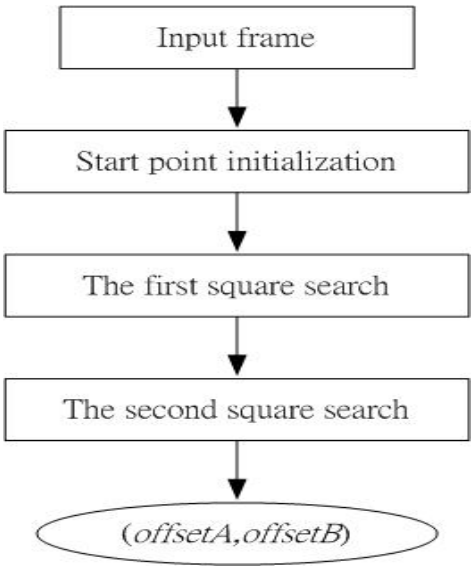


Fig. 4.5: The steps of the proposed predicted local square search (PLSS).

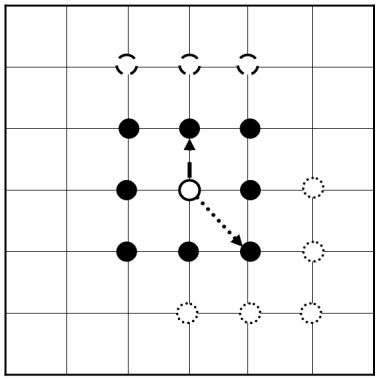


Fig. 4.6: Search patterns by applying 1 to 2 iterations of the square search employed by the PLSS.

Chapter 5
The Pre-processing

5.1 The Advantages of Deblocking Filter

The conventional thought about the usage of deblocking filter is to automatically eliminate the blocking effect after an image is coded. In [3], it explores another advantage of the deblocking filter to improve the coding performance. Since deblocking filter changes pixel values adaptively, different input images may result in the same output. By utilizing pre-process, the input image can be altered to the one with less bitrate cost but having the same output. Therefore, the coding gain is obtained. Rate-distortion experiments validate the effectiveness of this idea. To comprehend this new idea, Fig. 5.1 depicts an example. S is the original signal to be coded and S_p is one of its pre-processed versions. Next, both signals are compressed, and the quality-degraded versions, S' and S_p', are obtained. It follows that S' is still more similar to the input S as compared with S_p'. However, after applying the mandatory in-loop deblocking filter, both the final reconstructions S'' and S_p'' become similar to S. That is to say, both S and S_p produce the same final reconstructed result, but S_p needs fewer bits to be represented due to the complication of S. Based on these observations, some interesting ideas are arisen. Finding out a simple but equivalent (in terms of reconstructed quality) signal (i.e., the pre-processed signal), we can still get a faithful signal back after applying the deblocking process. Further, it is also possible to have a better final reconstruction and less bitrate cost at the same time.

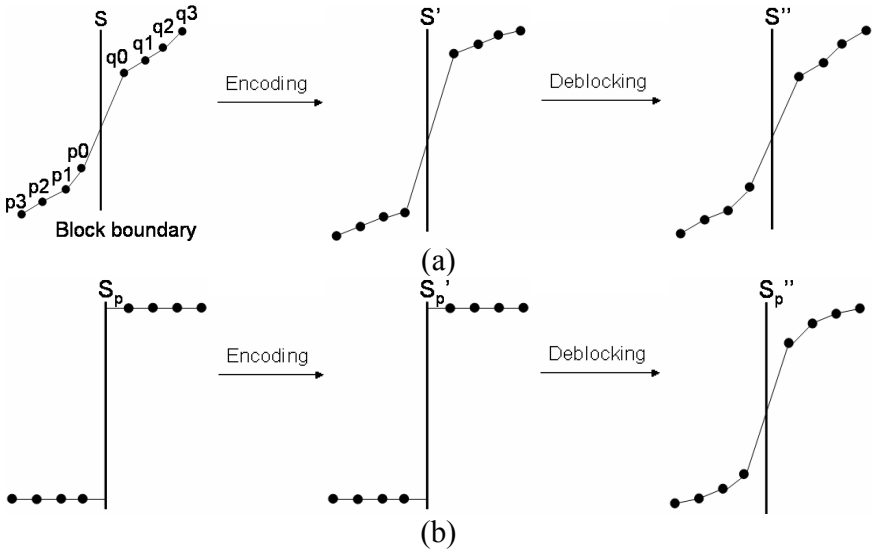


Fig. 5.1: Different signals distributions around the block boundary. (a) *S*: Original signal, *S'*: Signal after encoding, and *S''*: Signal after encoding and deblocking. (b) S_p: Signal after pre-process, S_p': Signal after pre-process and encoding, and S_p'': Signal after pre-process, encoding and deblocking.

5.2 Maximizing the Effect of Deblocking Filter

As abovementioned, the compression performance can be enhanced further when considering both the deblocking filter and the pre-processing. To reach this goal, the behavior of different deblocking filters should be explored. Actually, H.264/AVC allows the application of 5 different filters adaptively, and results in 8 different modes, each changes different pixel pairs. For every different filtering mode, there exists a different optimal pre-processing solution. Table 5.1 summarizes all the 8 different modes. The term 'Boundary Strength' (*BS*) is defined in H.264/AVC standard. The larger the value of *BS*, the stronger the filter will be applied. In Fig. 5.2, taking mode 4 as an example, it allows two pixels neighboring to the block boundary on each side to be filtered. *I* is the pixel vector across block boundaries in the original image. *R* is the vector after pre-processing stage and *V* is the corresponding pixel vector that has been filtered. The relation of *R* and *V* can be found in H.264/AVC. The input signal for H.264/AVC encoding is *R* rather than *I*. Further, there is an additional assumption that the signal before encoding is very similar to that after decoding. Thus, *V* is also used as the

reconstruction signal in deriving the pre-processing formulations. To prevent the interference between neighboring blocks, the pre-process is only conducted in the interior of the block. In this example, only pixels located inside the interior (Q, a sub-vector of R) of the block are modified. From the viewpoint of optimization, the goal is to minimize the distortion between the original signal and the filtered one. That is to minimize ε (sum of the square error). So, solving the equations $\partial\varepsilon/\partial Q = 0$ can derive the optimal value of Q. As for the optimal solutions of other modes, similar derivation is applicable. The only difference is the relation between R and V. The detail derivation for the pre-process formula of the mode 4 filtering is shown in Fig. 5.2. Other modes are revealed in Appendix B.

Table 5.1: H.264/AVC deblocking modes. The realization of tap filter is defined in H.264/AVC specification.

Filtering Mode	Boundary Strength (BS)	Filtered Points
1	1~3	p_0,q_0
2	1~3	p_0,q_0,q_1
3	1~3	p_1,p_0,q_0
4	1~3	p_1,p_0,q_0,q_1
5	4	p_0,q_0
6	4	p_0,q_0,q_1,q_2
7	4	p_2,p_1,p_0,q_0
8	4	p_2,p_1,p_0,q_0,q_1,q_2

$$\begin{cases} Original(I) : (p_3, p_2, p_1, p_0, q_0, q_1, q_2, q_3) \\ Pre-processed(R) : (p_3{}', p_2{}', p_1{}', p_0{}', q_0{}', q_1{}', q_2{}', q_3{}') \\ Filtered(V) : (p_3{}', p_2{}', p_1{}'', p_0{}'', q_0{}'', q_1{}'', q_2{}', q_3{}') \end{cases} \quad \begin{cases} \triangle_1 = \frac{1}{8}[4(q_0{}' - p_0{}') + (p_1{}' - q_1{}') + 4] \\ \triangle_2 = \frac{1}{2}[q_2{}' + \frac{1}{2}(p_0{}' + q_0{}' + 1) - 2q_1{}'] \\ \triangle_3 = \frac{1}{2}[p_2{}' + \frac{1}{2}(p_0{}' + q_0{}' + 1) - 2p_1{}'] \end{cases} \quad \begin{cases} D_1 = (0,0,0,1,-1,0,0,0) \\ D_2 = (0,0,0,0,0,1,0,0) \\ D_3 = (0,0,1,0,0,0,0,0) \end{cases}$$

$$V = R + \triangle_1 D_1 + \triangle_2 D_2 + \triangle_3 D_3$$

$$\varepsilon = \|V - I\|^2, Q = (q_0{}', q_1{}', q_2{}', q_3{}') \qquad \frac{\partial \varepsilon}{\partial Q} = \frac{\partial \|V - I\|^2}{\partial Q} = 2(V - I)\left[\frac{\partial V}{\partial Q}\right]^T = 0$$

$$V - I = (p_3{}' - p_3, p_2{}' - p_2, p_1{}' + \triangle_3 - p_1, p_0{}' + \triangle_1 - p_0, q_0{}' - \triangle_1 - q_0, q_1{}' + \triangle_2 - q_1, q_2{}' - q_2, q_3{}' - q_3)$$

$$\frac{\partial V}{\partial Q} = \frac{\partial (R + \triangle_1 D_1 + \triangle_2 D_2 + \triangle_3 D_3)}{\partial Q} = \frac{\partial R}{\partial Q} + \frac{\partial (\triangle_1 D_1)}{\partial Q} + \frac{\partial (\triangle_2 D_2)}{\partial Q} + \frac{\partial (\triangle_3 D_3)}{\partial Q} = \begin{bmatrix} 0 & 0 & \frac{1}{4} & \frac{1}{2} & \frac{1}{2} & \frac{1}{4} & 0 & 0 \\ 0 & 0 & 0 & \frac{-1}{8} & \frac{1}{8} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\Rightarrow \begin{cases} q_0{}' = \frac{1}{49}(-9 + 40p_0 + 20p_1 + 40q_0 + 16q_1 - 8q_2 - 49p_0' - 10p_2') \\ q_1{}' = 4 - 4p_0 + 4q_0 + p1{}' \\ q_2{}' = \frac{1}{49}(-4 - 4p_0 - 2p_1 - 4q_0 + 18q_1 + 40q_2 + p_2') \\ q_3{}' = q_3 \end{cases}$$

Fig. 5.2: The detail derivation for the pre-process formula of the mode 4 filtering, which is applied when BS is from 1 to 3. The goal is set to minimize ε by replacing p_1, p_0, q_0 and q_1.

5.3 Pre-processing Flowchart

The realization of the pre-processing is illustrated in Fig. 5.3. All 4×4 block boundaries will be pre-processed in incompliance with the H.264/AVC deblocking behavior. In each pre-processing, pixel values are modified according to filtering modes that it most probably uses and the formulations derived in Section 5.2. Each 4×4 block boundary will be encoded with the pre-processed version if the needed bitrate and mean square error both decrease simultaneously. Otherwise, the non-modified version of the 4×4 boundary is used as the encoding input. In the H.264/AVC deblocking process, at most four pixels, in each side, neighbor to the block boundary may be altered. Thus, these pixels should be considered in computing the mean square error. Fig. 5.4 illustrates the whole region used in pre-processing and edge processing order for one macroblock. The additional 4×16 and 16×4 pixels in the upper and left areas also involve in the calculation of the mean square

error for the current macroblock. The pre-processing procedure does not apply to the next macroblock until every 4×4 block boundary belonging to the current macroblock are scanned. As for the implementation issues, two MB-level operation orders, raster scan and checker scan, are employed in the process of optimization as described in Section 5.5.

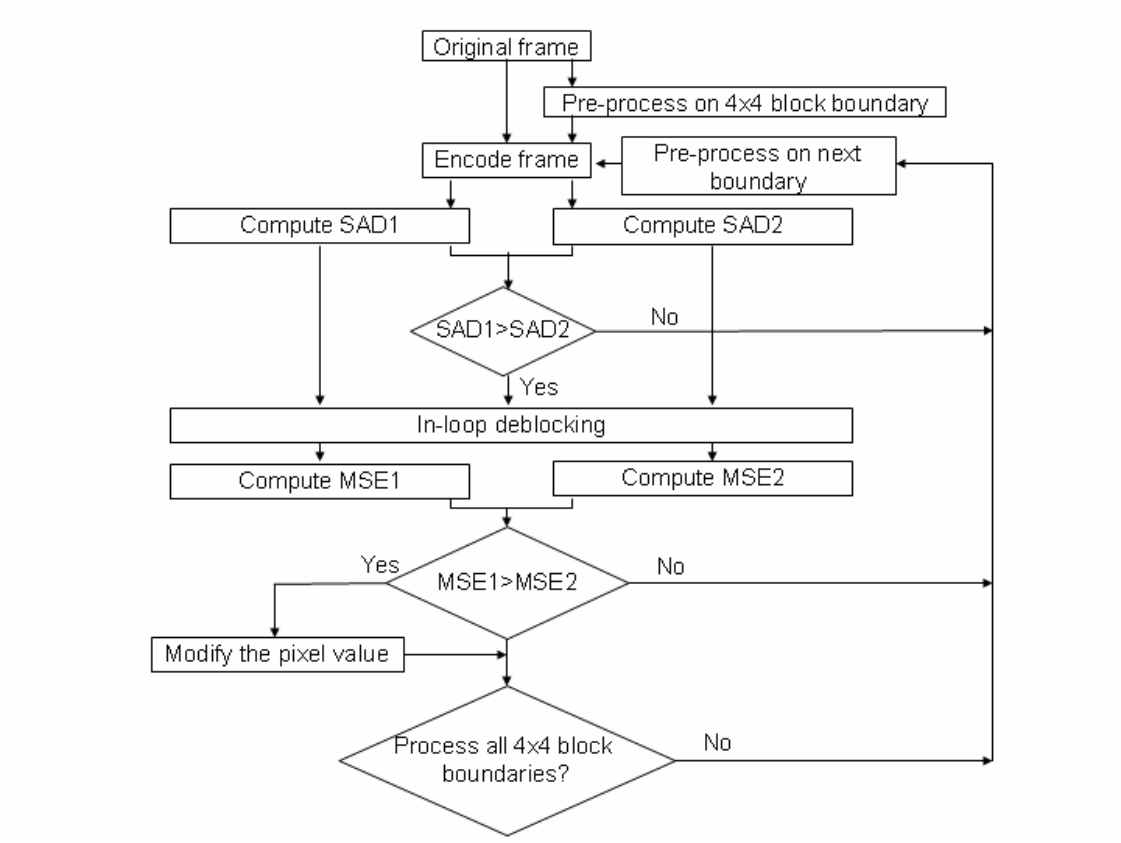


Fig. 5.3: The flowchart of the pre-process procedure.

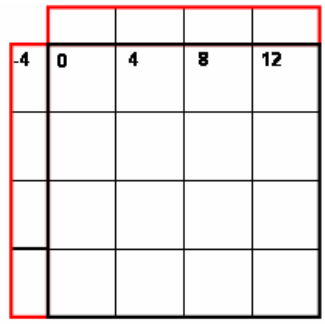


Fig. 5.4: The covered region used for computing MSE.

5.4 Rate-Distortion Optimization

Coding performance highly depends on the coding strategies of encoders. In Fig. 5.5, three different strategies are illustrated. If only distortion is considered in the selection of coding modes, the obtained decoded signal will achieve minimum distortion but requires a higher rate. Similarly, if only rate is considered, the required bitrate is very low but the quality of decoded signal will be damaged severely. Generally speaking, these two extreme strategies are not adopted in encoders. The most often used one is to minimize these two factors together. That is to minimize the distortion under the given bitrate or vice versa. To realize this goal, Lagrangian optimization techniques are often employed. The optimal solution found by the Lagrangian optimization process is the one having lowest cost value which is the weighted result of distortion and rate, as shown in Eqn. (5.1). The weighting factor named as Lagrangian multiplier, λ, is to control the balance between the distortion and rate. When this kind of strategy is conducted in coding modes decision, the encoded signal will have better coding performance both on the quality distortion and rate consumption.

$$cost\_value = Distortion + \lambda \times Rate \tag{5.1}$$

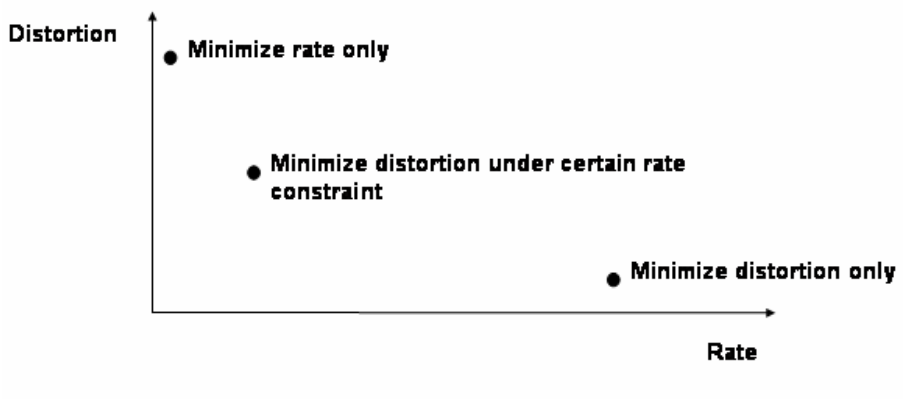


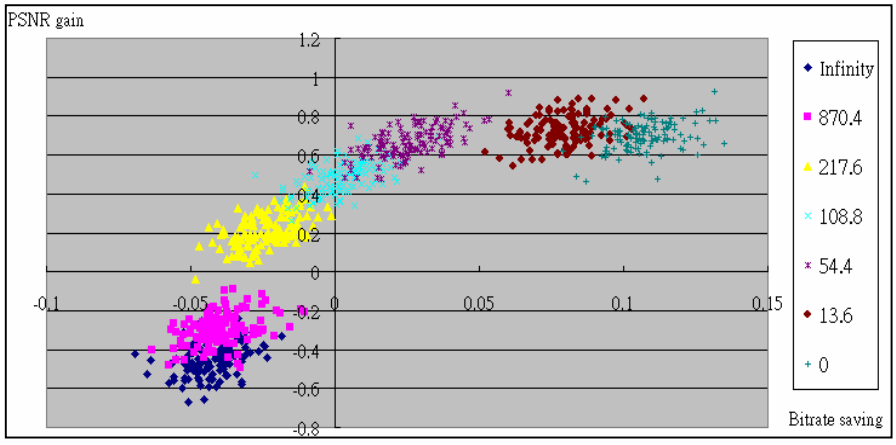
Fig. 5.5: Working points of different encoder strategies.

To further improve the coding performance of the proposed encoding system, the concept of rate-distortion optimization needs to be included in the pre-processing procedure. In the prescribed pre-processing, the modified signal replaces the original one when both the SAD and MSE constraints are hold. This scheme is to make the pre-processing enabled only in the extreme conditions where the estimation of coded bits, SAD, and the quality distortion, MSE, are both reduced. To enhance the original pre-processing, the Lagrangian optimization technique is integrated into the pre-processing. Moreover, the actual coded bits rather than the estimation of coded bits (SAD) are also required. After these considerations, the modified pre-processing procedure is as follows: the replacement of the pre-processed signal with the original one is determined according to the rate-distortion performance rather than the prescribed SAD and MSE constraints. About the value of λ, the Lagrangian multiplier adopted by H.264/AVC is used in our refined pre-processing scheme. The formulation of λ is shown in Eqn. (5.2). It has a close relationship with the value of quantization parameters, since the visual quality and rate consumption significantly affected by QP.

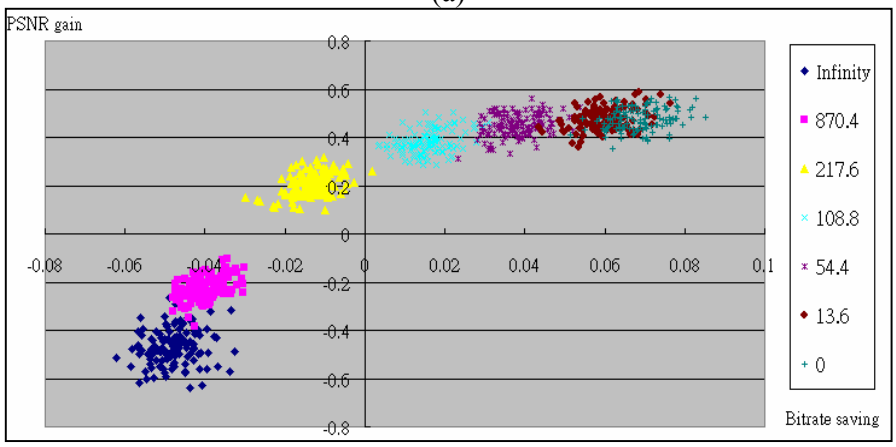
$$\lambda = 0.85 \times 2^{(QP-12)/3} \tag{5.2}$$

In the procedure of Lagrangian optimization, the multiplier, λ, plays an important role on obtaining a good trade-off between affecting factors. To explore how it influences the rate-distortion performance, different values of λ are used in the modified pre-processing scheme and the corresponding R-D curves are plotted in Fig. 5.6 These experimental

results are obtained by conducting different λ in the refined pre-processing approach which adopts the rate-distortion as its criterion function. All experiments take the non-pre-processed coding results as the baseline. The x-axis of Fig. 5.6 represents the PSNR difference and the y-axis stands for the bitrate difference of corresponding frames. We can find that various λ result in much different rate-distortion performance. When λ approaches infinity, only bitrate is taken into the cost and, on the other hand, when it nears zero, only distortion is considered in the optimization process. The λ value of 217.6 is get from Eqn. (5.2) whose corresponding experimental results demonstrate good balance between distortion and bitrate in both testing sequences.



(a)

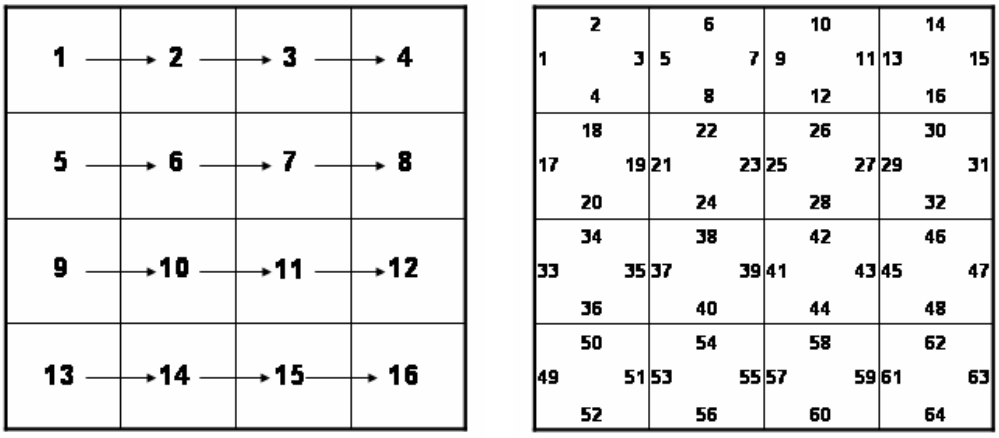


(b)

Fig. 5.6: The rate-distortion performances for different values of lambda. All frames are encoded as I frames. (a) The R-D curve of the Akiyo sequence with QCIF resolution. The quantization parameter is set to 36. (b) The R-D curve of the Foreman sequence with QCIF resolution. The quantization parameter is set to 28.

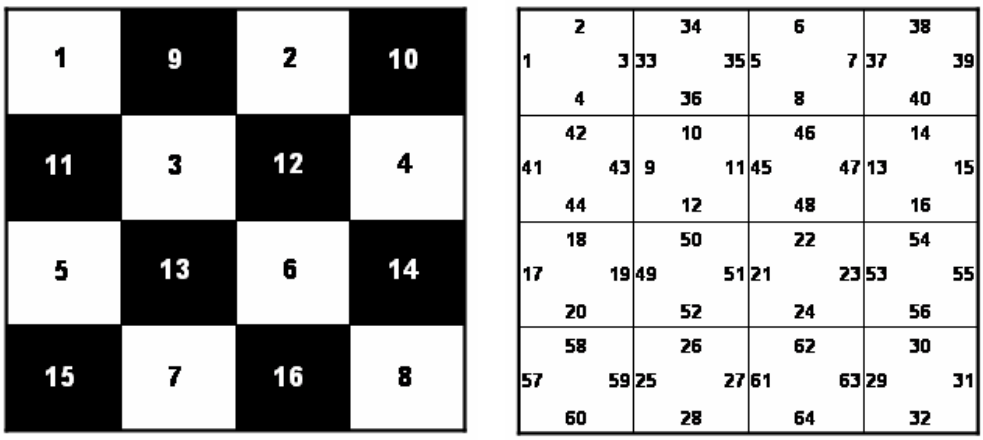
5.5 Operation Orders

Three operation orders of pre-processing for visiting all edges are implemented. The first is the raster scan order, the second is the checker scan and the third is z scan. In macroblock level, these three operation orders have the same behavior. Each macroblock is accessed in the raster order. When the pre-processing is conducted inside one macroblock, different operation orders are employed for these three orders as displayed in Fig. 5.7, Fig. 5.8 and Fig. 5.9 respectively. The left images illustrate the pre-processing orders for each 4×4 block and the right ones demonstrate the processing orders of each edge as indicated by the Arabic numerals. For the raster scan, every 4×4 block is scanned from left to right and from top to bottom. In the checker scan, 4×4 blocks are accessed in check board-like ordering while the z scan is to simulate the encoding ordering of 4x4 blocks. At last, for each 4x4 block, it is pre-processed from the left-bottom corner pixel and then moves clockwise until all four boundaries are scanned.



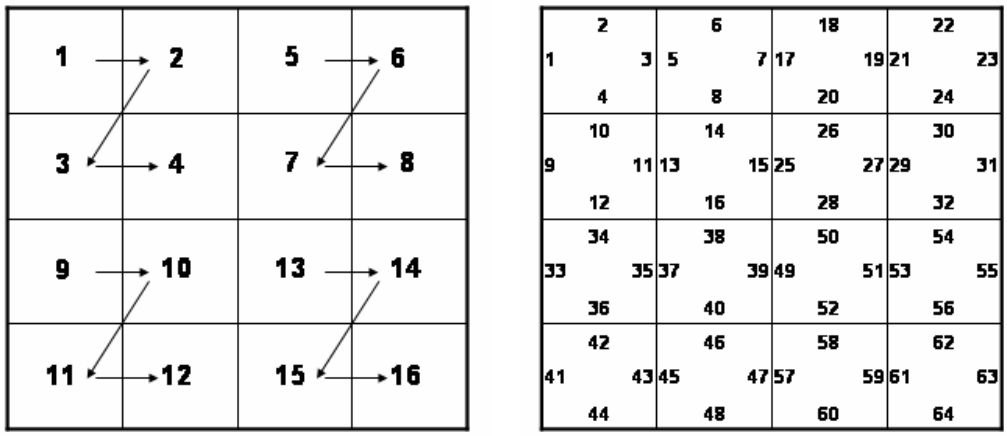
(a) (b)

Fig. 5.7: Raster scan order and its corresponding processing order on (a) the macroblock level and (b) the edge level.



(a) (b)

Fig. 5.8: Checker scan order and its corresponding processing order on (a) the macroblock level and (b) the edge level.



(a) (b)

Fig. 5.9 Z scan order and its corresponding processing order (a) the macroblock level and (b) the edge level.

Chapter 6
Experimental Results

6.1 Post-processing Experiments

Computer simulations are conducted using the proposed approach and the reference software of H.264/AVC, JM 8.0 [29]. In the experiments of post-processing, three different criterion functions, PSNR, Weighted Blocking Degree (WBD) and Perceptual-based Blocking and Blurring Measurements (PBBM) are used to compare the performance with that of fixed deblocking approach both in objective and subjective issues. As shown in Fig. 6.1, the fixed-offset deblocking approach has the most serious blocking effects, especially on the regions near collars, chins and eyes of the foreground. When (*OffsetA, OffsetB*) is decided according to PSNR, we can find that blocking artifacts are still obvious. Fig. 6.1(c) adopts WBD as the criterion function to determine the best (*OffsetA, OffsetB*). It is easy to see that most blocking effects are removed but near the areas of eyes, some blocking distortions still exist. Fig. 6.1 (d) demonstrates the best visual quality among these four pictures. Its deblocking parameters are chosen based on PBBM which measures the distortions caused by the blocking and blurring artifacts. From these observations, the idea of different slices having their own appropriate deblocking strengths is proved. The fixed-offset deblocking method ignores the content properties on different slices so it suffers from the most severe blocking effect. Furthermore, this experiment also shows that PSNR is not a good quality assessment for the blocking effect. PSNR measures the overall energy distortion but blocking artifacts often appear near block boundaries. Therefore, most blocking effects are not eliminated after the deblocking processes (whose deblocking parameters are determined based on PSNR), as illustrated in Fig. 6.1(b). To compensate this drawback, one of our proposed cost functions, WBD, considers both the phenomenon of the discontinuity around block boundaries and the overall energy distortion. From the depiction of Fig. 6.1(c), it is apparent that most block discontinuities are removed in the foreground. Hence, WBD

reveals its moderate measurement for the blocking discontinuity energy. Finally, the second proposed cost function, PBBM, presents its qualification for evaluating the blocking effect. It is because PBBM is a perceptual-based assessment function and the blurring artifact is also measured in its assessing process. Blurring effect often comes with the blocking artifact when deblocking filters are applied with too strong strength. As a result, jointly evaluating the distortions incurred by the blurring effect will help the decision of deblocking parameters.

(a) Fixed (*OffsetA, OffsetB*) (b) PSNR

(c) WBD (d) PBBM

Fig. 6.1: The 17th frame of Foreman sequence in QCIF resolution under different deblocking schemes, where the quantization parameter is set to 36. The choice of (*OffsetA, OffsetB*) is based on different criterion functions: (a) (*OffsetA, OffsetB*) is fixed at (0, 0), while in (b), (c), and (d) the determination of (*OffsetA, OffsetB*) is based on PSNR, WBD and PBBM, respectively.

As described in Chapter 4, the problem of discovering the optimal pair of (*OffsetA*, *OffsetB*) is translated into a search problem in the 2-D space with *OffsetA* and *OffsetB* as two axes. When different cost functions are employed in the searching process, the corresponding search map and the so-obtained optimal offset pair will also be different as depicted in Fig. 6.2. The goal of the two proposed search algorithms is to find the optimal point on the prescribed 2-D space. To evaluate the performance of the proposed search algorithms, PDS and PLSS, exhaustive search is used as a benchmark. The experimental results are summarized in Table 6.1 and Table 6.2

As illustrated in Table 6.1, the reduction of the number of search points is up to 85%, as a comparison, 169 trials as required in the exhaustive search. Despite fewer search points, the (*OffsetA, OffsetB*) pair obtained from the proposed algorithms is almost identical to that obtained from the exhaustive search, except the Forman sequence, based on the PLSS algorithm, produce less than 60% of accuracy. Here, the accuracy stands for the percentage that the obtained pair of (*OffsetA, OffsetB*) is the same as the exhaustive search. Among different obtained pairs between the exhaustive and fast search algorithm, we further measure the average relative distance by the following formula:

$$\sqrt{(OffsetA_{full}-OffsetA_{fast})^2+(OffsetB_{full}-OffsetB_{fast})^2}\Big/diff\_num\ . \qquad (6.1)$$

where *diff_num* is the number of frames which are of different parameter sets in comparison with those acquired from the exhaustive search. According to Table 6.1, the value of *diff_num* is considerably small, which means that even some pairs of (*OffsetA, OffsetB*) are not identical to those obtained from exhaustive search, they are only of little difference. Moreover, the increase in WBD is small enough to be ignored. Again, these observations prove that our proposed algorithm can find *OffsetA* and *OffsetB* very close to those obtained from the exhaustive search, while the time spent is much less than it. As a result, our proposed algorithm achieves comparable performance with the exhaustive search. Generally speaking, PLSS provides worse performance than that of PDS, but the time saving is higher. Thus, we may apply two search algorithms to different application scenarios, or integrate them for getting a good balance, which is one of our future works. In addition, we also compare our results with the one adopting the fixed parameter set in Table 6.2. There is little reduction in the objective quality assessment, PSNR. Based on the mean of WBD, it is undoubted that the approaches, which dynamically change the deblocking parameters can eliminate the blocking effects better than the fixed-parameter

approach. As for the resultant bitrate, there is no obvious variation among all approaches. With regarding to the encoding time measured in seconds, per each frame, it increases respectively less than 14% and 10% in PDS and PLSS, while it almost doubles in the exhaustive search, as compared with the fixed-parameter approach.

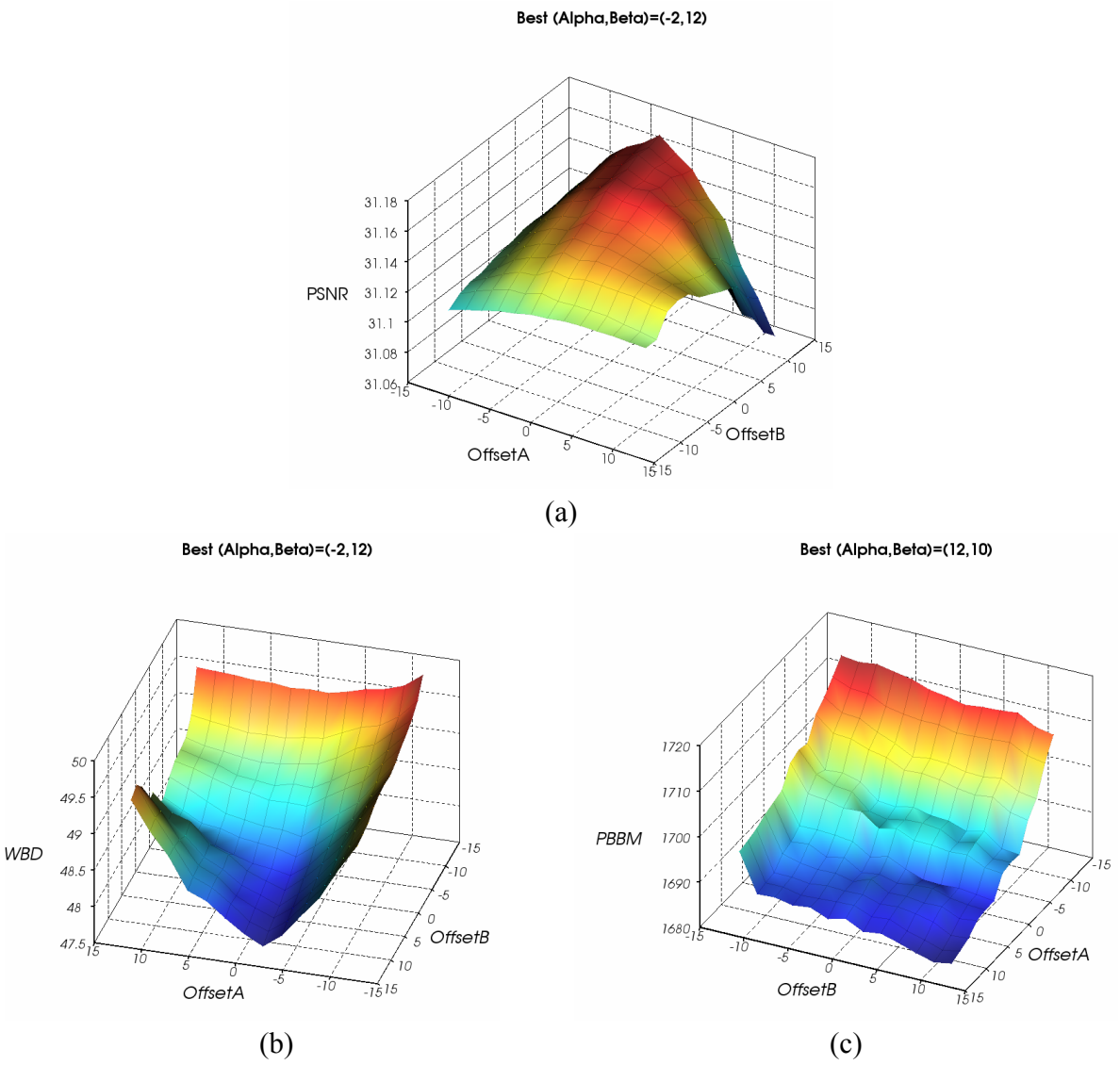


Fig. 6.2: The searching maps and the optimal *(OffsetA, OffsetB)* of using three different criterion functions: (a) PSNR, (b) WBD, and (c) PBBM.

Table 6.1: The results of the proposed search algorithms compared with the exhaustive search. The source consists of 100 frames at 30 fps. All frames are coded as P frames except for the leading I frame.

Sequence	Search algorithm	Reduction in number of search points	Increase in *WBD*	Ratio of accuracy	Average relative distance
Foreman, QCIF, QP=28	PDS	85.32%	1.2×10^{-2} %	86%	7.93
	PLSS	93.48%	7.8×10^{-2} %	57%	5.05
Foreman, QCIF, QP=36	PDS	87.04%	6.1×10^{-3} %	89%	9.36
	PLSS	94.11%	3.9×10^{-2} %	72%	6.93
Football, CIF, QP=28	PDS	85.94%	1.4×10^{-4} %	99%	6.00
	PLSS	93.33%	7.6×10^{-3} %	86%	3.00
Football, CIF, QP=36	PDS	89.71%	0%	100%	0.00
	PLSS	95.56%	3.8×10^{-4} %	95%	4.80

Table 6.2: The time spent of different search algorithms. Among these approaches, "Fixed" means the set OffestA and OffestB equal to zero. The first 100 frames of each sequence are encoded.

	Criterion	Search algorithm	Average PSNR	Average *WBD*	Files size (Bytes)	Encoding time (sec/frame)
Foreman, QCIF, QP=28	*N.A.*	*Fixed*	36.487	17.725	52160	1.15567
	PSNR	*Full Search*	36.515	16.931	52132	2.09532
		PDS	36.504	16.953	52045	1.29723
		PLSS	36.500	16.968	51996	1.22832
	WBD	*Full Search*	36.482	16.852	52254	2.10534
		PDS	36.445	16.854	52117	1.29954
		PLSS	36.448	16.866	52220	1.22985
Foreman, QCIF, QP=36	*N.A.*	*Fixed*	31.007	47.634	19760	1.11988
	PSNR	*Full Search*	31.097	46.31257	19586	3.14877
		PDS	31.121	46.20294	19745	1.24498
		PLSS	31.063	46.6995	19548	1.17592
	WBD	*Full Search*	30.981	47.168	19750	1.96956
		PDS	30.985	47.171	19680	1.24814
		PLSS	30.986	47.186	19817	1.18782
Football, CIF, QP=28	*N.A.*	*Fixed*	34.232	37.228	728517	4.76772
	PSNR	*Full Search*	34.265	36.763	728269	8.90518
		PDS	34.266	36.763	728153	5.44488
		PLSS	34.273	36.807	728459	5.09203
	WBD	*Full Search*	34.240	36.914	729505	9.32912
		PDS	34.240	36.914	729128	5.65504
		PLSS	34.234	36.917	728867	5.26369
Football, CIF, QP=36	*N.A.*	*Fixed*	28.456	119.364	244537	4.55422
	PSNR	*Full Search*	28.474	117.27363	244213	8.27995
		PDS	28.473	117.19417	244826	5.14681
		PLSS	28.463	117.4597	244305	4.86298
	WBD	*Full Search*	28.428	117.584	243932	8.65531
		PDS	28.428	117.584	243932	5.15972
		PLSS	28.441	117.588	244492	4.91099

6.2 Pre-processing Experiments

To verify the performance of the proposed pre-process algorithm, various video sequences of different motion characteristics are used in the experiments. The proposed optimized H.264/AVC encoding architecture is modified from that of the H.264/AVC reference software, JM8.0. The raster and checker scan orders are implemented to evaluate the performance. Fig. 6.3 (a) and (b) depict PSNR traces of the pre-process applying to different amount of deblocking modes. It is reasonable that applying more modes results in a higher PSNR. We, next, apply all modes to different amount of edges. As shown in Fig. 6.3 (c) and (d), when additional edges are considered by the pre-process, PSNR will increase apparently. Maximally, the average PSNR improvements go up to 0.35 dB, comparing with that of the non-pre-process case.

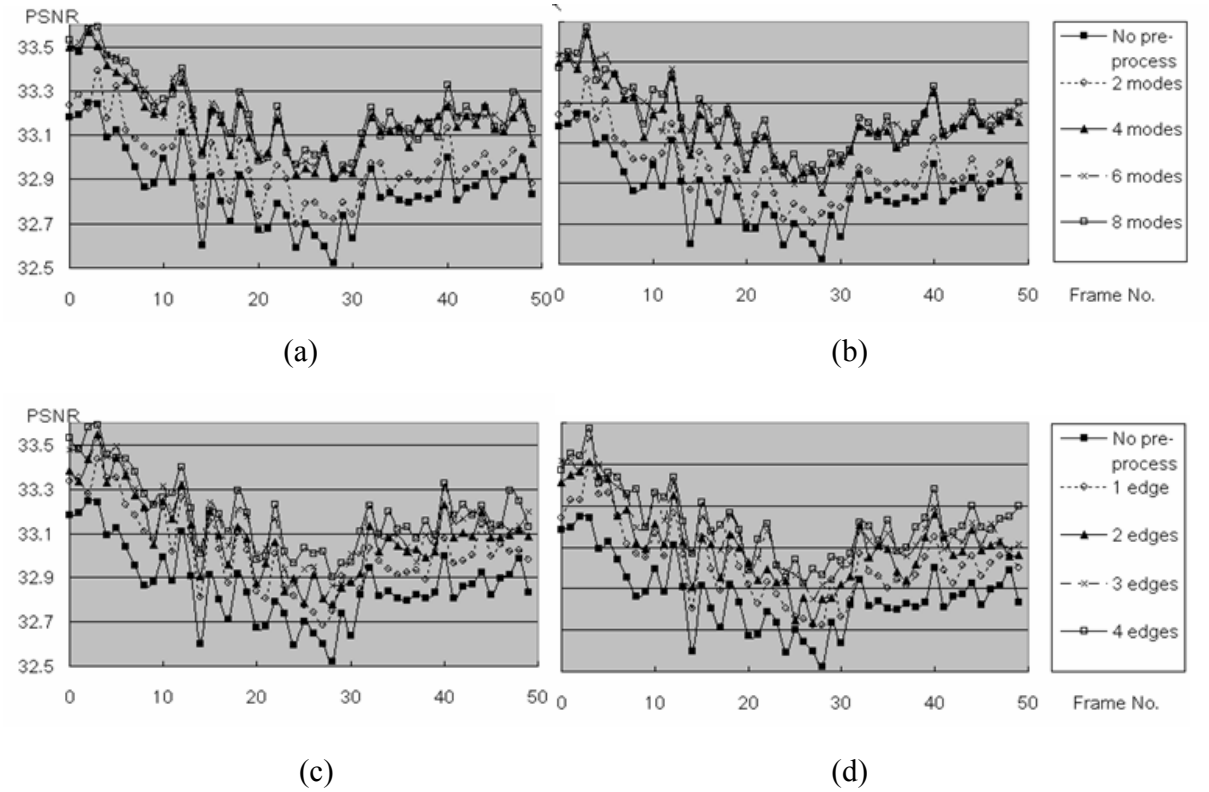


Fig. 6.3: The PSNR traces of the pre-process applying to different amount of deblocking modes and edges for I-frame only. (a) and (c) are using checker scan, while (b) and (d) adopt the raster scan.

As mentioned above, when rate-distortion model is employed in the pre-processing scheme, the overall coding performance will be enhanced. Thus, encoding schemes with various criterion functions are implemented to evaluate their coding performance as illustrated in In . In Table 6.3, the average PSNR and bits per frame under different encoding schemes are shown. In which I frames are coded in every 30 frames while others are coded as P frames. All sequences are with QCIF resolution. For the ease of understanding, the following notations are defined first:

No_pre_fixed_post: no pre-processing is used and deblocking filers are enabled with fixed parameters.

Pre_fixed_post (SAD): the pre-processing scheme adopting SAD and MSE as criterion functions is enabled and deblocking filers are also enabled with fixed parameters.

Pre_fixed_post (R-D): the pre-processing scheme adopting rate-distortion cost as criterion functions is enabled and deblocking filers are also enabled with fixed parameters.

No_pre_opz_PSNR: no pre-processing is used and deblocking filers are enabled with adaptive parameters.

Pre_opz_PSNR (SAD): the pre-processing scheme adopting SAD and MSE as criterion functions is enabled and deblocking filers are also enabled with adaptive parameters.

Pre_opz_PSNR (R-D): the pre-processing scheme adopting rate-distortion cost as criterion function is enabled and deblocking filers are also enabled with adaptive parameters.

Table 6.3: The coding performance.

Sequence	Model	PSNR	Bits/frame	PSNR gain	Bitrate gain
Akiyo, QP28	No_pre_fixed_post	39.27	1108	0	0
	Pre_fixed_post(SAD)	39.48	1119	0.21	0.99%
	Pre_fixed_post(R-D)	39.46	1085	0.19	-2.08%
	Pre_fixed_post(R-D2)	39.68	1119	0.41	0.99%
	No_pre_opz_PSNR	39.31	1117	0.04	0.81%
	Pre_opz_PSNR(SAD)	39.49	1118	0.22	0.90%
	Pre_opz_PSNR(R-D)	39.50	1090	0.23	-1.62%

	Pre_fixed_post(R-D2)	39.70	1117	0.43	0.81%
Akiyo, QP33	No_pre_fixed_post	35.59	665	0	0
	Pre_fixed_post(SAD)	35.72	660	0.13	-0.75%
	Pre_fixed_post(R-D)	35.78	621	0.19	-6.62%
	Pre_fixed_post(R-D2)	36.00	640	0.41	-3.76%
	No_pre_opz_PSNR	35.73	655	0.14	-1.50%
	Pre_opz_PSNR(SAD)	35.87	657	0.28	-1.20%
	Pre_opz_PSNR(R-D)	35.77	629	0.18	-5.41%
	Pre_opz_PSNR(R-D2)	36.17	651	0.58	-2.11%
Akiyo, QP36	No_pre_fixed_post	33.65	476	0	0
	Pre_fixed_post(SAD)	33.98	478	0.33	0.42%
	Pre_fixed_post(R-D)	33.91	458	0.26	-3.78%
	Pre_fixed_post(R-D2)	34.17	473	0.52	-0.63%
	No_pre_opz_PSNR	33.72	484	0.07	1.68%
	Pre_opz_PSNR(SAD)	33.97	486	0.32	2.10%
	Pre_opz_PSNR(R-D)	33.75	465	0.10	-2.31%
	Pre_opz_PSNR(R-D2)	34.08	478	0.43	0.42%
Foreman, QP28	No_pre_fixed_post	36.41	4979	0	0
	Pre_fixed_post(SAD)	36.54	5081	0.13	2.05%
	Pre_fixed_post(R-D)	36.51	4898	0.10	-1.63%
	Pre_fixed_post(R-D2)	36.72	5139	0.31	3.21%
	No_pre_opz_PSNR	36.45	4986	0.04	0.14%
	Pre_opz_PSNR(SAD)	36.55	5118	0.14	2.79%
	Pre_opz_PSNR(R-D)	36.54	4876	0.13	-2.07%
	Pre_fixed_post(R-D2)	36.79	5169	0.38	3.82%
Foreman, QP33	No_pre_fixed_post	33.10	2690	0	0
	Pre_fixed_post(SAD)	33.21	2696	0.11	0.22%
	Pre_fixed_post(R-D)	33.05	2548	-0.05	-5.28%
	Pre_fixed_post(R-D2)	33.40	2714	0.30	0.89%
	No_pre_opz_PSNR	33.12	2679	0.02	-0.41%
	Pre_opz_PSNR(SAD)	33.26	2729	0.16	1.45%
	Pre_opz_PSNR(R-D)	33.05	2532	-0.05	-5.87%
	Pre_opz_PSNR(R-D2)	33.44	2739	0.34	1.82%
Foreman, QP36	No_pre_fixed_post	31.07	1850	0	0
	Pre_fixed_post(SAD)	31.24	1859	0.17	0.49%

	Pre_fixed_post(R-D)	31.08	1761	0.01	-4.81%
	Pre_fixed_post(R-D2)	31.40	1882	0.33	1.73%
	No_pre_opz_PSNR	31.11	1851	0.04	0.05%
	Pre_opz_PSNR(SAD)	31.26	1879	0.19	1.57%
	Pre_opz_PSNR(R-D)	31.06	1749	-0.01	-5.46%
	Pre_opz_PSNR(R-D2)	31.42	1880	0.35	1.62%
Mother, QP28	No_pre_fixed_post	37.56	2090	0	0
	Pre_fixed_post(SAD)	37.68	2115	0.12	1.20%
	Pre_fixed_post(R-D)	37.65	2026	0.09	-3.06%
	Pre_fixed_post(R-D2)	37.95	2168	0.39	3.73%
	No_pre_opz_PSNR	37.64	2091	0.08	0.05%
	Pre_opz_PSNR(SAD)	37.75	2126	0.19	1.72%
	Pre_opz_PSNR(R-D)	37.70	2041	0.14	-2.34%
	Pre_opz_PSNR(R-D2)	37.92	2183	0.36	4.45%
Mother, QP33	No_pre_fixed_post	34.28	1020	0	0
	Pre_fixed_post(SAD)	34.46	1032	0.18	1.18%
	Pre_fixed_post(R-D)	34.35	960	0.07	-5.88%
	Pre_fixed_post(R-D2)	34.67	1037	0.39	1.67%
	No_pre_opz_PSNR	34.31	1024	0.03	0.39%
	Pre_opz_PSNR(SAD)	34.50	1040	0.22	1.96%
	Pre_opz_PSNR(R-D)	34.35	972	0.07	-4.71%
	Pre_opz_PSNR(R-D2)	34.70	1056	0.42	3.53%
Mother, QP36	No_pre_fixed_post	32.35	668	0	0
	Pre_fixed_post(SAD)	32.52	670	0.17	0.30%
	Pre_fixed_post(R-D)	32.45	625	0.10	-6.44%
	Pre_fixed_post(R-D2)	32.77	682	0.42	2.10%
	No_pre_opz_PSNR	32.39	678	0.04	1.50%
	Pre_opz_PSNR(SAD)	32.55	686	0.20	2.69%
	Pre_opz_PSNR(R-D)	32.41	637	0.06	-4.64%
	Pre_opz_PSNR(R-D2)	32.75	694	0.40	3.89%
Silent, QP28	No_pre_fixed_post	35.87	3662	0	0
	Pre_fixed_post(SAD)	36.00	3736	0.13	2.02%
	Pre_fixed_post(R-D)	36.04	3620	0.17	-1.15%
	Pre_fixed_post(R-D2)	36.24	3792	0.37	3.55%
	No_pre_opz_PSNR	35.93	3663	0.06	0.03%

	Pre_opz_PSNR(SAD)	36.04	3757	0.17	2.59%
	Pre_opz_PSNR(R-D)	36.05	3620	0.18	-1.15%
	Pre_opz_PSNR(R-D2)	36.27	3818	0.40	4.26%
Silent, QP33	No_pre_fixed_post	32.43	1962	0	0
	Pre_fixed_post(SAD)	32.61	1996	0.18	1.73%
	Pre_fixed_post(R-D)	32.48	1876	0.05	-4.38%
	Pre_fixed_post(R-D2)	32.81	2009	0.38	2.40%
	No_pre_opz_PSNR	32.45	1955	0.02	-0.36%
	Pre_opz_PSNR(SAD)	32.64	2015	0.21	2.70%
	Pre_opz_PSNR(R-D)	32.47	1873	0.04	-4.54%
	Pre_opz_PSNR(R-D2)	32.81	2030	0.38	3.47%
Silent, QP36	No_pre_fixed_post	30.52	1313	0	0
	Pre_fixed_post(SAD)	30.71	1339	0.19	1.98%
	Pre_fixed_post(R-D)	30.63	1249	0.11	-4.87%
	Pre_fixed_post(R-D2)	30.97	1365	0.45	3.96%
	No_pre_opz_PSNR	30.56	1315	0.04	0.15%
	Pre_opz_PSNR(SAD)	30.75	1347	0.23	2.59%
	Pre_opz_PSNR(R-D)	30.59	1250	0.07	-4.80%
	Pre_opz_PSNR(R-D2)	30.97	1381	0.45	5.18%

Chapter 7
Conclusions and Future Work

7.1 Conclusions

In this thesis, jointly considering the functionality of pre-process and post-process, the effects of deblocking filters are maximized to improve the video coding performance. By integrating these two components, pre-process and post-process, into H.264/AVC, an enhanced coding system for H.264/AVC is realized. In the post-processing procedure, effective approaches to decide the deblocking parameter set, *OffsetA* and *OffsetB*, to adaptively control the filtering strength of H.264/AVC are proposed. We address two new criterion functions to measure the blocking effect from the analyses on the energy distortion of signals and the visual impaction resulted from the blocking artifacts. After translating the original problem of determining the optimal pair of *OffsetA* and *OffsetB* into a 2-D space search problem, and then, fast search techniques are applied. Combining the multiple search approaches such as the large and the small diamond search as to find out the best parameters, two efficient search algorithms are implemented and the experimental results show their practical usage as compared with the exhaustive search. The results demonstrate that the proposed post-processing mechanism outperforms the fixed-offset approach adopted by the reference software of H.264/AVC, JM 8.0 [29], especially in the improvement of the subjective quality. The abilities of deblocking filters are fully utilized when the pre-processing procedure joins the overall encoding flow. The pre-processing is composed of an interesting concept that by changing some pixels next to the block boundary, the same reconstruction can be built up with spending fewer bits when deblocking filers take part in the coding loop. Moreover, the formulations of replacing the original pixels with modified ones for H.264/AVC in-loop deblocking filers are also derived in our works on the pre-processing component. Under the evaluation of rate-distortion performance, the proposed encoding architecture for H.264/AVC performs better than the reference software of H.264.AVC, JM8.0 [29]. According to the results of

encoding different sequences under different coding conditions, the proposed coding system shows its reliability and superiority.

7.2 Future Work

Due to our flexible implantations of the post-processing component, if there comes up another better criterion function, the original one used to determine the best parameter set of deblocking filters can be effectively replaced. In this thesis, two proposed criterion functions target only on the blocking and blurring artifacts. In general, the overall visual degradation inside images consist of multiple kinds of distortions, to closely measure these distortions to the feelings of human beings, more accurate and perceptual-based quality assessment models are required in the calculation of the quality distortions. After taking the sensitivity of the Human Visual System (HVS) into account, a better quality assessment model can be obtained and the selected deblocking parameter set will be more appropriate. In spite of the criterion function, search algorithms for discovering the optimal *OffsetA* and *OffsetB* can also be refined to avoid being trapped into the local optimal. As for the pre-processing mechanism, employing the rate-distortion model to make a judgment between the original signals or the pre-processed ones is a good way to improve the overall coding performance. The key factor influencing the rate-distortion analysis is the value of the Lagrangian multiplier, λ. The rate-distortion performance will downgrade much when using an inappropriate value of λ. Therefore, finding out the appropriate value of λ for different video sequences is one of our future works. On the other hand, the current strategy of pre-processing is optimizing the coding performance only on the macroblock level, which is viewed as a greedy algorithm in the frame level. To further enhance the performance of the proposed encoding system, how to obtain the optimal pre-processed frame is important. That is also one of our subsequent research directions.

Bibliography

[1] “Information Technology – Coding of Audio-Visual Objects, Part 2: Visual,” ISO/IEC 14496-2:2001, Second Edition, International Organization for Standardization, 2001.

[2] “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.

[3] W. F. Cheung and Y. H. Chan, “Improving MPEG-4 coding performance by jointly optimizing compression and blocking effect elimination,” *IEE Proceedings-Vision, Image and Signal Processing*, vol. 148, pp.194-201, Jun. 2001.

[4] W. C. Fong, S.C. Chan, A. Nallanathan, K.L. Ho, "Integer lapped transforms and their applications to image coding," *IEEE Trans. on Image Processing*, vol. 11, pp.1152-1159, Oct. 2002.

[5] B. Tao and M.T. Orchard, "A parametric solution for optimal overlapped block motion compensation," IEEE Trans. on Image Processing, vol.10, pp.341 -350, Mar. 2001.

[6] S. Nogaki and M. Ohta, “An overlapped block motion compensation for high quality motion picture coding,” *Proc. Int. Sump. Circuits Systems*, 1992.

[7] D. C. Youla and H. Webb, “Image restoration by the method of convex projections: Part I--- Theory,” *IEEE Trans. Med. Imag.* vol 1, pp. 81-94, Oct. 1982.

[8] ZAKHOR, “Iterative procedures for reduction of blocking effects in transform image coding,” *IEEE Trans. on Circuits Syst. Video Technology*, vol. 2, pp.91-95, Mar. 1992.

[9] Weerasinghe, A. W. C. Liew, H. Yan, “Artifacts reduction in compressed images based on region homogeneity constraints using the projection onto convex sets algorithm,” *IEEE Trans. on Circuits Syst. Video Technology*, vol. 12, pp.891-897, Oct. 2002.

[10] T. P. O’Rourke and R. L. Stevenson, “Improved image decompression for reduced transform coding artifacts,” *IEEE Trans. on Circuits Syst. Video Technology*, vol. 5, pp.490-498, Dec. 1995.

[11] S. D. Kim, J. Yi, H. M. Kim, J. B. Ra, “A deblocking filter with two separate modes in block-based video coding,” *IEEE Trans. on Circuits Syst. Video Technology*, vol. 9, pp.156-160, Feb. 1999.

[12] P. List, A. Joch, J. Lainema, G. Bjntegaard , M. Karczewicz, “Adaptive deblocking filter,” *IEEE Trans. on Circuits Syst. Video Technology*, vol. 13, pp.614-619, Jul. 2003.

[13] M. Zhao, R. E. J. Kneepkens, P. M. Hofman, G. de Haan, “Content adaptive image de-blocking,” *Proc. of ISCE*, Sept. 2004.

[14] Y. C. Zhu and M. J. Chen, “Improvement of adaptive deblocking filter in H.264,” Workshop on Consumer Electronics and Signal Processing, 2004.

[15] Y.H. CHAN, S.W.HONG, and W.C SIU., “A practical post-processing technique for real-time block-based coding system,” *IEEE Trans. on Circuits Syst. Video Technology*, vol.8, pp.4-8, Feb. 1998.

[16] Z. XIONG, M.T. ORCHARD, and Y.Q. ZHANG, “A deblocking algorithm for jpeg compressed images using overcomplete wavelet representations,” *IEEE Trans. on Circuits Syst. Video Technology*, vol. 7, pp.433-437, Apr. 1997.

[17] K. U. Barthel and H. L. Cycon, “Image denoising using fractal and wavelet-based methods,” SPIE's International Symposium on Photonics Technologies for Robotics, Automation, and Manufacturing Wavelet Applications in Industrial Processing, Oct. 2003.

[18] A . W. C. Liew and H. Yan, “Blocking artifacts suppression in block-coded images using overcomplete wavelet representation,” *IEEE Trans. on Circuits Syst. Video Technology*, vol. 14, pp.450-460, Apr. 2004.

[19] 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.

[20] S. A. Karunasekera and N. G. Kingsbury, “A distortion measure for blocking artifacts in images based on human visual sensitivity,” *IEEE Trans on Image Processing*, pp. 713-724, June 1995.

[21] Z. YU, H.R. Wu, S. Winkler, T. Chen, “Vision-model-based impairment metric to evaluate blocking artifacts in digital video,” *Proc. IEEE*, pp. 154-169, Jan. 2002.

[22] Z. Wang, L. Lu, and A. C. Bovik, “Video quality assessment based on structural distortion measurement”, *Signal Processing: Image Communication*, pp. 121-132, 2004.

[23] H. R. Wu, M. Yuen, “A generalized block-edge impairment metric for video coding,” *IEEE Signal Process. Lett*., vol. 4, pp. 317–320, Nov. 1997.

[24] Z. Wang, H. R. Sheikh, and A. C. Bovik, “No-Reference perceptual quality assessment of JPEG compressed images”, *IEEE International Conference on Image Processing*, pp. I-477–480, Sept. 2002.

[25] F. Pan, X. Lion, S. Rahardja, W. Lin, E. Ong, S. Yao, Z. Lu, X. Yang, “A locally adaptive algorithm for measuring blocking artifacts in images and videos”, *Signal Processing: Image Communication*, pp. 499-506, 2004.

[26] P. Marziliano, F. Dufaux, S. Winkler, T. Ebrahimi, “Perceptual blur and ringing metrics: application to JPEG2000”, *Signal Processing: Image Communication*, pp. 163-172, 2004.

[27] C. H. Chou and Y. C. Li, “A perceptually tuned subband image coder based on the measure of just-noticeable-distortion profile,” *IEEE Trans. on Circuits Syst. Video Technology*, vol. 5, pp.467-476, Dec. 1995.

[28] B. Girod, “The information theoretical significance of spatial and temporal masking in video signals”, in *Proc. SPIE Conf. Human Vision, Visual Processing, and Digital Display*, vol. 1077, pp. 178-187, 1989.

[29] “Information technology: Coding of audio-visual objects, Part 5: Reference software Amendment 6: Advanced video coding and high efficiency advanced audio coding reference software,” ISO/IEC JTC 1/SC 29/WG 11 Docs. No. 6248.

Appendix A

Publication List

• International Conference

1. I-Hsuan Yang, Sheng-Ho Wang, Yang-Ho Chen, Pao-Hsian Huang, Liang Ye, Xiaoqiu Huang, Kun-Mao Chao, "Efficient methods for generating optimal single and multiple spaced seeds," IEEE Fourth Symposium on Bioinformatics and Bioengineering (BIBE2004), pp. 411－416, May 2004.
2. Sheng-Ho Wang, Sung-Wen Wang, Yi-Chin Huang, Yi-Shin Tung, Ja-Ling Wu, "Boundary-energy sensitive visual de-blocking for H.264/AVC coder," SPIE Proc. Applications of Digital Image Processing XXVII, Denver, CO, August 2004.
3. 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.

Appendix B

The derivation of optimal pre-processing solutions for each deblocking mode

Filtering Mode: **1**

BS: **1~3**

Filtering pixels: **p_0, q_0**

$$\begin{cases} Original(I) : (p_3, p_2, p_1, p_0, q_0, q_1, q_2, q_3) \\ Pre-processed(R) : (p_3{}', p_2{}', p_1{}', p_0{}', q_0{}', q_1{}', q_2{}', q_3{}') \\ Filtered(V) : (p_3{}', p_2{}', p_1{}', p_0{}'', q_0{}'', q_1{}', q_2{}', q_3{}') \end{cases}$$

$$V = R + \Delta_1 D_1$$

$$\Delta_1 = \frac{1}{8}[4(q_0{}' - p_0{}') + (p_1{}' - q_1{}') + 4]$$

$$D_1 = (0, 0, 0, 1, -1, 0, 0, 0)$$

$$\varepsilon = ||V - I||^2, Q = (q_0{}', q_1{}', q_2{}', q_3{}')$$

The goal is to minimize ε.

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

where

$$V - I = (R + \Delta_1 D_1 - I) = (p_3{}' - p_3, p_2{}' - p_2, p_1{}' - p_1, p_0{}' + \Delta_1 - p_0, q_0{}' - \Delta_1 - q_0, q_1{}' - q_1, q_2{}' - q_2, q_3{}' - q_3)$$

$$\frac{\partial V}{\partial Q} = \frac{\partial (R + \Delta_1 D_1)}{\partial Q} = \frac{\partial R}{\partial Q} + \frac{\partial (\Delta_1 D_1)}{\partial Q}$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & \frac{1}{2} & \frac{-1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{-1}{8} & \frac{1}{8} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{-1}{8} & \frac{1}{8} & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\Rightarrow \begin{cases} q_0{}' = p_0 + q_0 - p_0{}' \\ q_1{}' = \frac{1}{33}[4 - 4p_0 + 4q_0 + 32q_1 + p_1{}'] \\ q_2{}' = q_2 \\ q_3{}' = q_3 \end{cases}$$

Filtering Mode: **2**

BS: **1~3**

Filtering pixels: **p_0, q_0, q_1**

$$\begin{cases} Original(I) : (p_3, p_2, p_1, p_0, q_0, q_1, q_2, q_3) \\ Pre-processed(R) : (p_3{}', p_2{}', p_1{}', p_0{}', q_0{}', q_1{}', q_2{}', q_3{}') \\ Filtered(V) : (p_3{}', p_2{}', p_1{}', p_0{}'', q_0{}'', q_1{}'', q_2{}', q_3{}') \end{cases}$$

$$V = R + \Delta_1 D_1 + \Delta_2 D_2$$

$$\begin{cases} \Delta_1 = \frac{1}{8}[4(q_0{}' - p_0{}') + (p_1{}' - q_1{}') + 4] \\ \Delta_2 = \frac{1}{2}[q_2{}' + \frac{1}{2}(p_0{}' + q_0{}' + 1) - 2q_1{}'] \end{cases}$$

$$\begin{cases} D_1 = (0,0,0,1,-1,0,0,0) \\ D_2 = (0,0,0,0,0,1,0,0) \end{cases}$$

$$\varepsilon = ||V - I||^2, Q = (q_0{}', q_1{}', q_2{}', q_3{}')$$

The goal is to minimize ε.

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

where

$$\begin{aligned} V - I &= (R + \Delta_1 D_1 + \Delta_2 D_2 - I) \\ &= (p_3{}' - p_3, p_2{}' - p_2, p_1{}' - p_1, p_0{}' + \Delta_1 - p_0, q_0{}' - \Delta_1 - q_0, q_1{}' + \Delta_2 - q_1, q_2{}' - q_2, q_3{}' - q_3) \end{aligned}$$

$$\frac{\partial V}{\partial Q} = \frac{\partial(R + \Delta_1 D_1 + \Delta_2 D_2)}{\partial Q} = \frac{\partial R}{\partial Q} + \frac{\partial(\Delta_1 D_1)}{\partial Q} + \frac{\partial(\Delta_2 D_2)}{\partial Q}$$

$$= \begin{bmatrix} 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1 \end{bmatrix} + \begin{bmatrix} 0&0&0&\frac{1}{2}&\frac{-1}{2}&0&0&0 \\ 0&0&0&\frac{-1}{8}&\frac{1}{8}&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \end{bmatrix} + \begin{bmatrix} 0&0&0&0&0&\frac{1}{4}&0&0 \\ 0&0&0&0&0&-1&0&0 \\ 0&0&0&0&0&\frac{1}{2}&0&0 \\ 0&0&0&0&0&0&0&0 \end{bmatrix}$$

$$= \begin{bmatrix} 0&0&0&\frac{1}{2}&\frac{1}{2}&\frac{1}{4}&0&0 \\ 0&0&0&\frac{-1}{8}&\frac{1}{8}&0&0&0 \\ 0&0&0&0&0&\frac{1}{2}&1&0 \\ 0&0&0&0&0&0&0&1 \end{bmatrix}$$

$$\Rightarrow \begin{cases} q_0{}' = \dfrac{1}{11}(-1 + 10p_0 + 10q_0 + 4q_1 - 2q_2 - 11p_0') \\ q_1{}' = 4 - 4p_0 + 4q_0 + p_1' \\ q_2{}' = \dfrac{1}{11}(-1 - p_0 - q_0 + 4q_1 + 9q_2) \\ q_3{}' = q_3 \end{cases}$$

Filtering Mode: **3**

BS: **1~3**

Filtering pixels: **p_1, p_0, q_0**

$$\begin{cases} Original(I) : (p_3, p_2, p_1, p_0, q_0, q_1, q_2, q_3) \\ Pre-processed(R) : (p_3{}', p_2{}', p_1{}', p_0{}', q_0{}', q_1{}', q_2{}', q_3{}') \\ Filtered(V) : (p_3{}', p_2{}', p_1{}'', p_0{}'', q_0{}'', q_1{}', q_2{}', q_3{}') \end{cases}$$

$$V = R + \Delta_1 D_1 + \Delta_3 D_3$$

$$\begin{cases} \Delta_1 = \frac{1}{8}[4(q_0{}' - p_0{}') + (p_1{}' - q_1{}') + 4] \\ \Delta_3 = \frac{1}{2}[p_2{}' + \frac{1}{2}(p_0{}' + q_0{}' + 1) - 2p_1{}'] \end{cases}$$

$$\begin{cases} D_1 = (0,0,0,1,-1,0,0,0) \\ D_3 = (0,0,1,0,0,0,0,0) \end{cases}$$

$$\varepsilon = ||V - I||^2, Q = (q_0{}', q_1{}', q_2{}', q_3{}')$$

The goal is to minimize ε.

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

where

$$\begin{aligned} V - I &= (R + \Delta_1 D_1 + \Delta_3 D_3 - I) \\ &= (p_3{}' - p_3, p_2{}' - p_2, p_1{}' + \Delta_3 - p_1, p_0{}' + \Delta_1 - p_0, q_0{}' - \Delta_1 - q_0, q_1{}' - q_1, q_2{}' - q_2, q_3{}' - q_3) \end{aligned}$$

$$\frac{\partial V}{\partial Q} = \frac{\partial(R + \Delta_1 D_1 + \Delta_3 D_3)}{\partial Q} = \frac{\partial R}{\partial Q} + \frac{\partial(\Delta_1 D_1)}{\partial Q} + \frac{\partial(\Delta_3 D_3)}{\partial Q}$$

$$= \begin{bmatrix} 0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1 \end{bmatrix} + \begin{bmatrix} 0&0&0&\frac{1}{2}&\frac{-1}{2}&0&0&0 \\ 0&0&0&\frac{-1}{8}&\frac{1}{8}&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \end{bmatrix} + \begin{bmatrix} 0&0&\frac{1}{4}&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \end{bmatrix}$$

$$= \begin{bmatrix} 0&0&\frac{1}{4}&\frac{1}{2}&\frac{1}{2}&0&0&0 \\ 0&0&0&\frac{-1}{8}&\frac{1}{8}&1&0&0 \\ 0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&1 \end{bmatrix}$$

$$\Rightarrow \begin{cases} q_0{}' = \frac{1}{9}(-1 + 8p_0 + 4p_1 + 8q_0 - 9p_0' - 2p_2') \\ q_1{}' = \frac{1}{33}(4 - 4p_0 + 4q_0 + 32q_1 + p_1') \\ q_2{}' = q_2 \\ q_3{}' = q_3 \end{cases}$$

Filtering Mode: **4**

BS: **1~3**

Filtering pixels: **p_0, q_0, q_1**

$$\begin{cases} Original(I) : (p_3, p_2, p_1, p_0, q_0, q_1, q_2, q_3) \\ Pre-processed(R) : (p_3{}', p_2{}', p_1{}', p_0{}', q_0{}', q_1{}', q_2{}', q_3{}') \\ Filtered(V) : (p_3{}', p_2{}', p_1{}'', p_0{}'', q_0{}'', q_1{}'', q_2{}', q_3{}') \end{cases}$$

$$V = R + \Delta_1 D_1 + \Delta_2 D_2 + \Delta_3 D_3$$

$$\begin{cases} \Delta_1 = \frac{1}{8}[4(q_0{}' - p_0{}') + (p_1{}' - q_1{}') + 4] \\ \Delta_2 = \frac{1}{2}[q_2{}' + \frac{1}{2}(p_0{}' + q_0{}' + 1) - 2q_1{}'] \\ \Delta_3 = \frac{1}{2}[p_2{}' + \frac{1}{2}(p_0{}' + q_0{}' + 1) - 2p_1{}'] \end{cases}$$

$$\begin{cases} D_1 = (0, 0, 0, 1, -1, 0, 0, 0) \\ D_2 = (0, 0, 0, 0, 0, 1, 0, 0) \\ D_3 = (0, 0, 1, 0, 0, 0, 0, 0) \end{cases}$$

$$\varepsilon = ||V - I||^2, Q = (q_0{}', q_1{}', q_2{}', q_3{}')$$

The goal is to minimize ε.

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

where

$$V - I = (p_3{}' - p_3, p_2{}' - p_2, p_1{}' + \Delta_3 - p_1, p_0{}' + \Delta_1 - p_0, q_0{}' - \Delta_1 - q_0, q_1{}' + \Delta_2 - q_1, q_2{}' - q_2, q_3{}' - q_3)$$

$$\frac{\partial V}{\partial Q} = \frac{\partial(R + \triangle_1 D_1 + \triangle_2 D_2 + \triangle_3 D_3)}{\partial Q} = \frac{\partial R}{\partial Q} + \frac{\partial(\triangle_1 D_1)}{\partial Q} + \frac{\partial(\triangle_2 D_2)}{\partial Q} + \frac{\partial(\triangle_3 D_3)}{\partial Q}$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & \frac{1}{2} & \frac{-1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{-1}{8} & \frac{1}{8} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$+ \begin{bmatrix} 0 & 0 & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & \frac{1}{4} & \frac{1}{2} & \frac{1}{2} & \frac{1}{4} & 0 & 0 \\ 0 & 0 & 0 & \frac{-1}{8} & \frac{1}{8} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\Rightarrow \begin{cases} q_0{}' = \frac{1}{49}(-9 + 40p_0 + 20p_1 + 40q_0 + 16q_1 - 8q_2 - 49p_0' - 10p_2') \\ q_1{}' = 4 - 4p_0 + 4q_0 + p1' \\ q_2{}' = \frac{1}{49}(-4 - 4p_0 - 2p_1 - 4q_0 + 18q_1 + 40q_2 + p_2') \\ q_3{}' = q_3 \end{cases}$$

Filtering Mode: **5**

BS: **4**

Filtering pixels: **p_0, q_0**

$$\begin{cases} Original(I) : (p_3, p_2, p_1, p_0, q_0, q_1, q_2, q_3) \\ Pre-processed(R) : (p_3{}', p_2{}', p_1{}', p_0{}', q_0{}', q_1{}', q_2{}', q_3{}') \\ Filtered(V) : (p_3{}', p_2{}', p_1{}', p_0{}'', q_0{}'', q_1{}', q_2{}', q_3{}') \end{cases}$$

$$p_0{}'' = \frac{1}{4}(2p_1{}' + p_0{}' + q_1{}' + 2)$$

$$q_0{}'' = \frac{1}{4}(2q_1{}' + q_0{}' + p_1{}' + 2)$$

$$\varepsilon = ||V - I||^2, Q = (q_0{}', q_1{}', q_2{}', q_3{}')$$

The goal is to minimize ε.

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

where

$$V - I = (p_3{}' - p_3, p_2{}' - p_2, p_1{}' - p_1, p_0{}'' - p_0, q_0{}'' - q_0, q_1{}' - q_1, q_2{}' - q_2, q_3{}' - q_3)$$

$$\frac{\partial V}{\partial Q} = \begin{bmatrix} 0 & 0 & 0 & 0 & \frac{1}{4} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{4} & \frac{1}{2} & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\Rightarrow \begin{cases} q_0{}' = \frac{1}{17}(-30 - 8p_0 + 68q_0 - 32q_1 + 2p_0{}' - 13p_1{}') \\ q_1{}' = \frac{1}{17}(-2 + 4p_0 + 16q_1 - p_0{}' - 2p_1{}') \\ q_2{}' = q_2 \\ q_3{}' = q_3 \end{cases}$$

Filtering Mode: **6**

BS: **4**

Filtering pixels: **p_0, q_0, q_1, q_2**

$$\begin{cases} Original(I) : (p_3, p_2, p_1, p_0, q_0, q_1, q_2, q_3) \\ Pre-processed(R) : (p_3{}', p_2{}', p_1{}', p_0{}', q_0{}', q_1{}', q_2{}', q_3{}') \\ Filtered(V) : (p_3{}', p_2{}', p_1{}', p_0{}'', q_0{}'', q_1{}'', q_2{}'', q_3{}') \end{cases}$$

$$p_0{}'' = \frac{1}{4}(2p_1{}' + p_0{}' + q_1{}' + 2)$$

$$q_0{}'' = \frac{1}{8}(q_2{}' + 2q_1{}' + 2q_0{}' + 2p_0{}' + p_1{}' + 4)$$

$$q_1{}'' = \frac{1}{4}(q_2{}' + q_1{}' + q_0{}' + p_0{}' + 2)$$

$$q_2{}'' = \frac{1}{8}(2q_3{}' + 3q_2{}' + q_1{}' + q_0{}' + p_0{}' + 4)$$

$$\varepsilon = ||V - I||^2, Q = (q_0{}', q_1{}', q_2{}', q_3{}')$$

The goal is to minimize ε.

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

where

$$V - I = (p_3{}' - p_3, p_2{}' - p_2, p_1{}' - p_1, p_0{}'' - p_0, q_0{}'' - q_0, q_1{}'' - q_1, q_2{}'' - q_2, q_3{}' - q_3)$$

$$\frac{\partial V}{\partial Q} = \begin{bmatrix} 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{8} & 0 \\ 0 & 0 & 0 & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{8} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{8} & \frac{1}{4} & \frac{3}{8} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & 1 \end{bmatrix}$$

$$\Rightarrow \begin{cases} q_0{}' = \frac{1}{181}(104 - 724p_0 + 616q_0 + 316q_1 - 416q_2 + 104q_3 + 285p_1{}') \\ q_1{}' = -2 + 4p_0 - p_0{}' - 2p_1{}' \\ q_2{}' = \frac{1}{181}(-144 - 296q_0 + 8q_1 + 576q_2 - 144q_3 + 37p_1{}') \\ q_3{}' = \frac{1}{181}(-1 + 8q_0 - 10q_1 + 4q_2 + 180q_3 - p_1{}') \end{cases}$$

Filtering Mode: **7**

BS: **4**

Filtering pixels: **p_2, p_1, p_0, q_0**

$$\begin{cases} Original(I) : (p_3, p_2, p_1, p_0, q_0, q_1, q_2, q_3) \\ Pre-processed(R) : (p_3{}', p_2{}', p_1{}', p_0{}', q_0{}', q_1{}', q_2{}', q_3{}') \\ Filtered(V) : (p_3{}', p_2{}'', p_1{}'', p_0{}'', q_0{}'', q_1{}', q_2{}', q_3{}') \end{cases}$$

$$p_0{}'' = \frac{1}{8}(p_2{}' + 2p_1{}' + 2p_0{}' + 2q_0{}' + q_1{}' + 4)$$

$$p_1{}'' = \frac{1}{4}(p_2{}' + p_1{}' + p_0{}' + q_0{}' + 2)$$

$$p_2{}'' = \frac{1}{8}(2p_3{}' + 3p_2{}' + p_1{}' + p_0{}' + q_0{}' + 4)$$

$$q_0{}'' = \frac{1}{8}(q_2{}' + 2q_1{}' + 2q_0{}' + 2p_0{}' + p_1{}' + 4)$$

$$\varepsilon = ||V - I||^2, Q = (q_0{}', q_1{}', q_2{}', q_3{}')$$

The goal is to minimize ε.

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

where

$$V - I = (p_3{}' - p_3, p_2{}'' - p_2, p_1{}'' - p_1, p_0{}'' - p_0, q_0{}'' - q_0, q_1{}' - q_1, q_2{}' - q_2, q_3{}' - q_3)$$

$$\frac{\partial V}{\partial Q} = \begin{bmatrix} 0 & \frac{1}{8} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{8} & \frac{1}{2} & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\Rightarrow \begin{cases} q_0{}' = \frac{1}{953}(-2068 + 1216p_0 + 1296p_1 + 648p_2 + 976q_0 - 640q_1 - 709p_0' - 953p_1' - 719p_2' - 162p_3') \\ q_1{}' = \frac{1}{953}(20 - 56p_0 - 160p_1 - 80p_2 + 256q_0 + 832q_1 + 64p_0' + 77p_2' + 20p_3') \\ q_2{}' = q_2 \\ q_3{}' = q_3 \end{cases}$$

Filtering Mode: **8**

BS: **4**

Filtering pixels: **p_2, p_1, p_0, q_0, q_1, q_2**

$$\begin{cases} Original(I) : (p_3, p_2, p_1, p_0, q_0, q_1, q_2, q_3) \\ Pre-processed(R) : (p_3{}', p_2{}', p_1{}', p_0{}', q_0{}', q_1{}', q_2{}', q_3{}') \\ Filtered(V) : (p_3{}', p_2{}'', p_1{}'', p_0{}'', q_0{}'', q_1{}'', q_2{}'', q_3{}') \end{cases}$$

$$p_0{}'' = \frac{1}{8}(p_2{}' + 2p_1{}' + 2p_0{}' + 2q_0{}' + q_1{}' + 4)$$

$$p_1{}'' = \frac{1}{4}(p_2{}' + p_1{}' + p_0{}' + q_0{}' + 2)$$

$$p_2{}'' = \frac{1}{8}(2p_3{}' + 3p_2{}' + p_1{}' + p_0{}' + q_0{}' + 4)$$

$$q_0{}'' = \frac{1}{8}(q_2{}' + 2q_1{}' + 2q_0{}' + 2p_0{}' + p_1{}' + 4)$$

$$q_1{}'' = \frac{1}{4}(q_2{}' + q_1{}' + q_0{}' + p_0{}' + 2)$$

$$q_2{}'' = \frac{1}{8}(2q_3{}' + 3q_2{}' + q_1{}' + q_0{}' + p_0{}' + 4)$$

$$\varepsilon = ||V - I||^2, Q = (q_0{}', q_1{}', q_2{}', q_3{}')$$

The goal is to minimize ε.

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

where

$$V - I = (p_3{}' - p_3, p_2{}'' - p_2, p_1{}'' - p_1, p_0{}'' - p_0, q_0{}'' - q_0, q_1{}'' - q_1, q_2{}'' - q_2, q_3{}' - q_3)$$

$$\frac{\partial V}{\partial Q} = \begin{bmatrix} 0 & \frac{1}{8} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{8} & 0 \\ 0 & 0 & 0 & \frac{1}{8} & \frac{1}{4} & \frac{1}{4} & \frac{1}{8} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{8} & \frac{1}{4} & \frac{3}{8} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{4} & 1 \end{bmatrix}$$

$$
\Rightarrow \begin{cases}
{q_0}' = \frac{1}{5489}(-13300 + 5792p_0 + 15248p_1 + 7624p_2 - 2464q_0 - 1264q_1 + 1664q_2 - 416q_3 \\
\quad - 5489p_0' - 5905p_1' - 7395p_2' - 1906p_3') \\
{q_1}' = \frac{1}{5489}(5276 + 3368p_0 - 18912p_1 - 9456p_2 + 17248q_0 + 8848q_1 - 11648q_2 + 2912q_3 \\
\quad + 2912p_1' + 7853p_2' + 2364p_3') \\
{q_2}' = \frac{1}{5489}(-4236 - 6000p_0 + 2400p_1 + 1200p_2 - 6424q_0 + 1552q_1 + 15744q_2 - 3936q_3 \\
\quad + 1553p_1' - 300p_2' - 300p_3') \\
{q_3}' = \frac{1}{5489}(-36 + 260p_0 - 104p_1 - 52p_2 + 132q_0 - 360q_1 + 196q_2 + 5440q_3 - 49{p_1}' + 13p_2' + 13p_3')
\end{cases}
$$