A Learning-by-Example Method for Reducing BDCT Compression Artifacts in High-Contrast Images

WANG, Guangyu

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Philosophy
in
Computer Science & Engineering

December 2003

Abstract

The quantization procedure of block-based discrete cosine transform (BDCT) compression (such as JPEG) introduces annoying visual artifacts. In this thesis, we propose a novel learning-by-example method to reduce BDCT compression artifacts in high-contrast images (images with large smooth color areas and strong edges/outlines), for example cartoon images. Our main focus is on the removal of ringing artifacts that is seldom addressed by existing methods. In the proposed method, the contaminated image is modeled as a Markov random field (MRF). We 'learn' the behavior of contamination by extracting massive number of artifact patterns from a training set, and organizing them using tree-structured vector quantization (TSVQ). Instead of post-filtering the input contaminated image, we synthesize an artifact-reduced image. Utilizing the proposed method, we show that substantial improvement (both statistical and visual) is achieved. Our method is non-iterative and hence it can remove artifacts within a very short period of time.

摘 要

量化過程是基於塊狀離散餘絃變換（Block-Based Discrete Cosine Transform，簡稱 BDCT）的圖像壓縮方法（例如 JPEG）的一個重要組成部分。該過程會產生噪聲，進而降低圖像的視覺質量。此現象在高對比度的圖像中尤其明顯。高對比度圖像通常包含一些均勻色塊和明顯的邊緣或輪廓綫。本論文中，我們提出一種嶄新的方法，用於減少高對比度圖像中分佈在邊緣附近的環狀噪聲。現存的方法基本不處理此類環狀噪聲。此方法使用馬爾哥夫隨機域（Markov Random Field）模擬被壓縮的圖像。通過在訓練集合中提取出標準的噪聲模式，我們可以學習 BDCT 引入噪聲的機制。為實現快速索引大量的噪聲模式，它們被組織成樹形向量量化表（Tree-Structured Vector Quantization）。通過本方法得到的圖像，即噪聲被減弱的圖像，是重新生成的，而不是像大部分現有的方法那樣，通過利用濾波的方法來處理被壓縮的圖像。使用我們的方法，圖像質量在統計和視覺角度上都有大幅度的提高。同時，由於本方法只需掃描整幅圖像一遍，而不是迭代的方法，所以在較短時間內可以有效地提高圖像質量。

Acknowledgments

First, I wish to express my great gratitude to my supervisors Prof. Pheng-Ann Heng and Prof. Tien-Tsin Wong for their guidance and support throughout these two years. This thesis cannot be done without their valuable ideas and advices.

I would like to thank Philip Fu, Xie Yongming, Jiang Zhongding, Chen Xinyu, Yim Pan Chui and Ping-Fu Fung for giving me helpful advices. I would also like to thank my colleagues, Wang Jianqing, Lu Yang, Wei Dan, Wu Wen, Yang Rong, Huang Kaizhu, Chen Hui, He Jifu, and the others, with whom I shared my happy time in the beautiful Chinese University of Hong Kong.

Most of all, I would like to express the deepest gratitude to my parents. Without their love, support and encouragement, I cannot finish my studies.

Contents

List of Tables

List of Figures

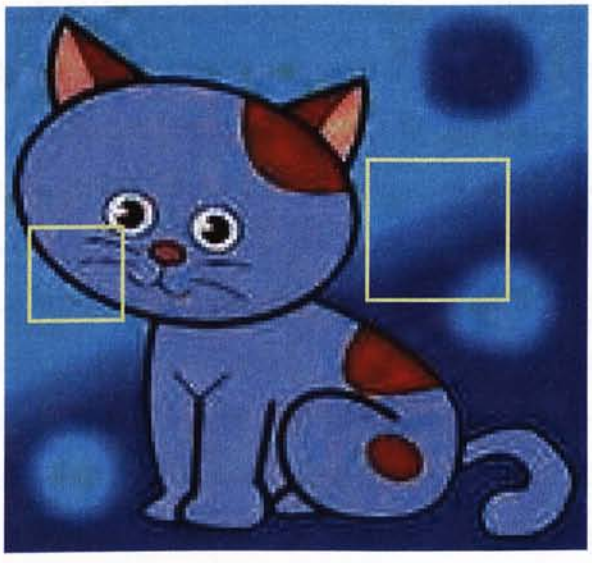
Chapter 1

Introduction

The block-based discrete cosine transform (BDCT) is the core of many current image and video compression standards, such as JPEG [27] and MPEG [31]. In particular, JPEG has been a popular image standard since early 90's. Even though we now have wavelet-based JPEG2000, there are still many existing images already encoded using JPEG. Unfortunately, BDCT-based JPEG inherits a drawback, which are annoying visual artifacts. Such artifacts are especially apparent in *high-contrast images* compressed at low bit rates. We refer high-contrast images as the ones with large smooth color regions and strong edges/outlines. A typical example is the cartoon image (Figure 1.1(a)).

1.1 BDCT Compression Artifacts

Since 1992, JPEG was established as the international standard for still image compression. An image is first divided into 8×8 pixel blocks and each block is transformed from spatial domain to frequency domain using discrete cosine transform (DCT) [1]. DCT can be regarded as a discrete version of the Fourier-Cosine series. The DCT coefficients are then quantized, resulting in many of them becoming zero, especially for high spatial frequencies. To take the maximum advantage of this, the coefficients are organized in a zigzag order to produce long runs of zero, before run-length and Huffman encoding achieves



(a) a typical BDCT-encoded high-contrast image

(b) blocking artifacts

(c) ringing artifacts

Figure 1.1: BDCT compression artifacts. (a) A typical BDCT-encoded (contaminated) high-contrast image, with (b) blocking artifacts and (c) ringing artifacts. Sub-figures (b) and (c) are the blowups of two boxes in (a).

the actual compression.

In principle, DCT does not introduce artifacts. It merely transforms the original image to a domain in which it can be more effectively encoded. However, the following quantization of DCT coefficients is lossy and introduces visual artifacts.

There are mainly two types of BDCT compression artifacts, namely *blocking artifacts* and *ringing artifacts*. Excessive quantization of low-frequency coefficients introduces blocking artifacts, which exhibit color discontinuity at the block boundary, as shown in Figure 1.1(b). Excessive quantization of high-frequency coefficients causes ringing artifacts around strong edges [38], such as outlines of cartoon character (Figure 1.1(c)). Ringing artifacts may spread within the whole 8×8 block. As shown in Figure 1.1, ringing artifacts are visually more apparent.

1.2 Previous Artifact Removal Methods

Both subjective and objective qualities of BDCT-encoded images can be significantly improved by reducing BDCT compression artifacts. Roughly, there are three categories techniques for reducing BDCT compression artifacts, which are post-filtering, projection onto convex sets (POCS), and learning-by-example methods.

As the visual artifacts in the contaminated image are usually high-frequency artifacts, a straight-forward solution for reducing such artifacts is to apply lowpass filtering. Reeve and Lim [32] first proposed a space-invariant lowpass filtering method. However such filtering often causes the loss of high-frequency details, such as edges, in the original image. Therefore, a number of adaptive spatial filtering techniques [4, 19, 20, 25] have been proposed to overcome this problem. Projection onto convex sets (POCS) [6] is a theory widely used for artifact removal [17, 23, 26, 38, 42, 33]. This method updates the BDCT coefficients by iteratively projecting onto several constraints, which are formulated by a *priori* knowledge of contaminated images. In recent years, learning-by-example methods [3, 29] have been proposed. These methods only process pixels at the block boundary and does not handle ringing artifacts within the block.

Yang *et al.* [41] presented a maximum-likelihood approach to the ringing artifact removal problem. Their approach employs a parameter-estimation method based on the k-means algorithm with the number of clusters determined by a cluster-separation measure. This algorithm is applied to remove ringing artifacts in images compressed by JPEG2000. There are also some techniques based on wavelet [5, 11, 15, 18, 34, 40].

1.3 Our Method

Existing methods mainly handle blocking artifacts. We believe that such techniques reducing blocking artifacts are not effective for ringing artifacts. Because the nature of ringing artifacts is very different from that of blocking artifacts. In this thesis, we aim at reducing ringing artifacts in BDCT-encoded high-contrast images. We propose a learning-by-example method which *synthesizes* the image with fewer artifacts, rather than post-filters the BDCT-encoded image. A training set is first prepared to provide *priori* knowledge that assists us to reduce BDCT compression artifacts. The training set consists of pairs of original images (images before compression and without any artifacts) and BDCT-encoded images (contaminated images). In the proposed method, we model compression artifacts as Markov random field. To do so, we extract a set of *artifact patterns* from the training set. These massive artifact patterns are looked up during synthesis. To manage and query such mass of patterns efficiently, we organize them with the tree-structured vector quantization (TSVQ).

As ringing artifacts usually appears near the strong edges (Figure 1.1), we divide the input contaminated image in two regions: edge region and non-edge region. For the edge region, we synthesize each pixel value by querying the TSVQ with its neighborhood as the search key; for the non-edge region where ringing artifacts are not apparent, we simply copy the pixel values from the input image.

1.4 Structure of the Thesis

This thesis is organized as follows. In Chapter 2, previous work in the related areas is reviewed. Chapter 3 explains the rationale and the basic idea of our method. The following 4 chapters describe the details of the proposed

learning-by-example method. Preparation of the training set is described in Chapter 4. Chapter 5 introduces artifact vectors. In Chapter 6, tree-structured vector quantization is explained. Chapter 7 describes the synthesis process. Chapter 8 evaluates the proposed method. Finally, conclusions are drawn in Chapter 9.

Chapter 2

Related Work

2.1 Image Compression

Image compression is a very important problem for many applications, including facsimile transmission (FAX), televideo conference, remote sensing (the use of satellite imagery for weather and other earth-resource applications), and digital image documentation. In short, an ever-expanding number of applications depend on the efficient transmission and storage of binary, gray-scale, and color images.

Image compression addresses the problem of reducing the amount of data required to represent a digital image. The underlying basis of the reduction process is the removal of redundant data. A common characteristic of most images is that the neighboring pixels are correlated and therefore contain redundant information. **Redundancy reduction** and **irrelevancy reduction** are two fundamental components of compression. Redundancy reduction tends to remove duplication from the signal source (image/video). Irrelevancy reduction omits parts of the signal that will not be noticed by the signal receiver, namely the Human Visual System (HVS). In general, there are three types of redundancy:

- **Spatial Redundancy** or correlation between neighboring pixel values;

- **Spectral Redundancy** or correlation between different color planes or spectral bands;
- **Temporal Redundancy** or correlation between adjacent frames in a sequence of images (in video applications).

In general, there are two kinds of compression techniques to reduce redundancy mentioned above: **Lossless and Lossy compression**. By lossless compression technique, the encoded image, after compression, is numerically identical to the original image. However, lossless compression can only achieve a modest amount of compression. An image encoded by lossy compression contains degradation relative to the original. Often this is because the compression technique completely discards redundant information. However, lossy techniques are capable of achieving much higher compression. Under normal viewing conditions, no visible loss is perceived (visually lossless). There are several famous image compression techniques: GIF (by reducing color depth), JPEG (based on BDCT), JPEG2000 (based on wavelet), TIFF and so on. In this thesis, we mainly explore BDCT compression.

2.2 A Typical BDCT Compression: Baseline JPEG

Since the mid-1980's, members from both the International Telecommunication Union (ITU) and the International Standard Organization (ISO) have been working together to establish a joint international standard for the compression of multilevel still images. This effort has been known as the Joint Photographic Experts Group (JPEG). The goal of JPEG is to develop a general method for image compression that meets a number of diverse requirements, as following [2]:

- Be as close as possible to the state of the art in image compression.
- Allow applications (or a user) to tradeoff easily between desired compression and image quality.
- Work independently of the image type. That is, the method should not be restricted by the type of image source, image content, color spaces, dimensions, pixel resolution, etc.
- Have modest computational complexity that would allow software-only implementations even on low-end computers. Low-complexity hardware implementations should also be feasible.
- Allow both sequential coding (single scan) and progressive coding (multiple scans).
- Offer the option for hierarchical encoding, in which a low-resolution version of the image can be accessed without a need to decompress the image at full resolution.

JPEG is not a single algorithm; instead, it may be thought of as a toolkit of image compression methods that may be altered to fit different needs of the user. The JPEG standard specifies four modes of operation: sequential DCT-based, progressive DCT-based, lossless, and hierarchical. The JPEG standard also defines a minimal subset of the standard called **Baseline JPEG**, which are required to supported by all JPEG-aware applications. This baseline employs the block-based discrete cosine transform (BDCT) to achieve compression. The baseline JPEG compression scheme can be divided into the following five stages:

1. Transform the image into an optimal color space;
2. Downsample chrominance components by averaging groups of pixels together;

3. Apply a Discrete Cosine Transform (DCT) to blocks of pixels;

4. Quantize each block of DCT coefficients using weighting functions optimized for the human visual system;

5. Encode the resulting coefficients (image data) using a Huffman variable word-length algorithm to remove redundancies in the coefficients.

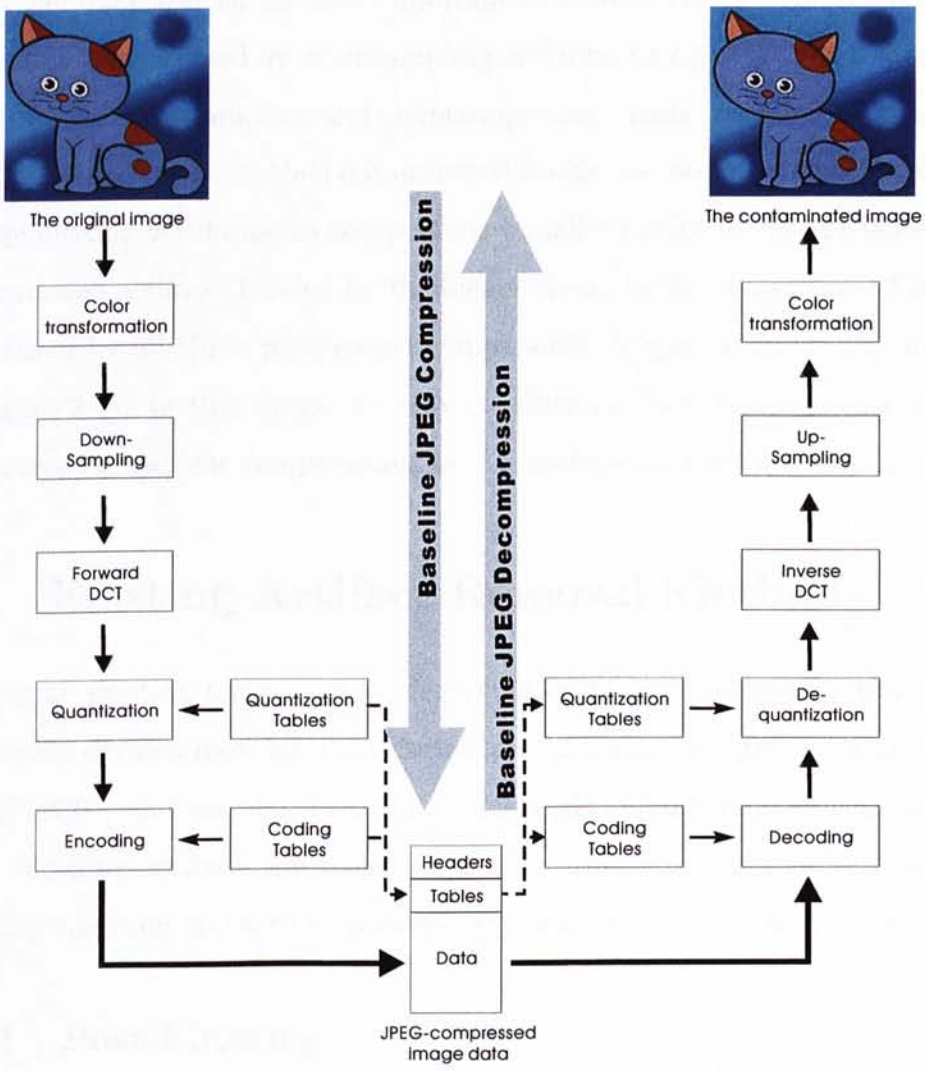


Figure 2.1: Baseline JPEG compression and decompression.

In Figure 2.1, these steps are summarized. The baseline JPEG decompression performs the reverse of these steps. Among the compression steps, downsampling chrominance components and quantizing DCT coefficients cause image information loss. Because of the eye's less sensitivity to chrominance information, JPEG uses fewer pixels for the chrominance components. Also in order to discard an appropriate amount of information, the compressor divides each DCT coefficient by a "quantizer" and rounds the result to an integer. The larger the quantizer is, the more information is lost. Hence compression artifacts are mainly caused by downsampling and quantizing. Figure 2.2 shows artifacts due to downsampling and quantizing components. Strange color labeled by the smaller circle, in the contaminated image, is caused by downsampling and quantizing chrominance components (smaller circles in (c) of Figure 2.2). Compression artifacts labeled by the bigger circle, in the contaminated image, are caused by all three compressed components (bigger circles in (b) and (c) of Figure 2.2). In this thesis, we only consider artifacts which are caused by luminance component compression, as it is more sensitive to human eyes.

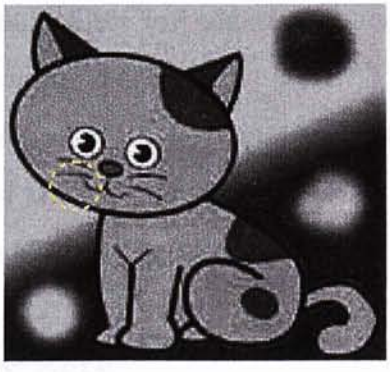
2.3 Existing Artifact Removal Methods

In general, existing techniques for removing BDCT compression artifacts can be roughly divided into three categories: post-filtering, projection onto convex sets (POCS), and learning-by-example methods. Comparing to ringing artifacts, blocking artifacts are easier to reduce. Most existing methods aims at reducing blocking artifacts by processing pixels at the block boundary.

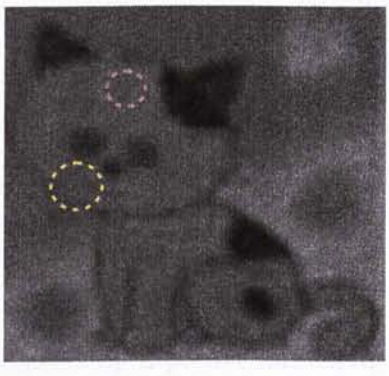
2.3.1 Post-Filtering

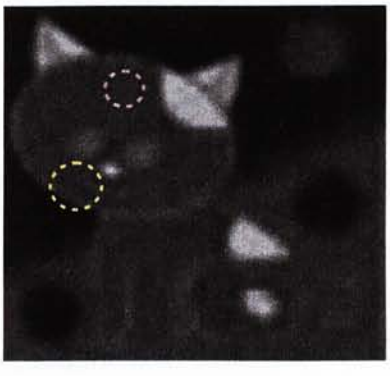
As visual artifacts in the contaminated image are usually high-frequency artifacts, a straight-forward solution for reducing such artifacts is to apply low-pass filtering. Reeve and Lim [32] first proposed a space-invariant low-pass filtering

(a) the contaminated image



(b) the quantized luminance component





(c) the downsampled and quantized chrominance components

Figure 2.2: (a) is a contaminated image. (b) is BDCT-encoded luminance component. (c) is BDCT-encoded chrominance components. Strange color labeled by the smaller circle, in (a), is caused by downsampling and quantizing chrominance components (smaller circles in (c)). Compression artifacts labeled by the bigger circle, in (a), are caused by all three compressed components (bigger circles in (b) and (c)).

method. However such filtering often causes the loss of high-frequency details, such as edges, in the original image. Therefore, a number of adaptive spatial filtering techniques [4, 14, 16, 19, 20, 22, 25, 30] have been proposed to overcome this problem. Kuo *et al.* [19] proposed an adaptive filter, which is a space-variant low-pass filter. It smoothes pixel values at the block boundary, while pixels close to edges are adaptively applied with a regional or directional low-pass filter. The filter coefficients change according to the local characteristics of images and visual artifacts. Lee *et al.* [20] proposed a method based

on signal adaptive filter. The signal is edge information. The filter consists of an one-dimensional directional smoothing filter for edge areas and a two-dimensional adaptive average filter for monotone areas.

The main difficulty of post-filtering techniques is how to distinguish high-frequency details from BDCT compression artifacts. Incorrect filtering usually introduces excessive blurring of details and/or exaggeration of visual artifacts.

2.3.2 Projection onto Convex Sets

Projection onto convex sets (POCS) [6] is a theory widely used for artifact removal [17, 23, 26, 33, 38, 42, 43]. The key idea is to represent every known property of the original image by a closed convex set. The solution is an image that is an element in all sets and can be found by alternating projections onto each set, starting from the contaminated image itself. These methods update the BDCT coefficients by iteratively projecting onto several constraints which are formulated by a *priori* knowledge of contaminated images.

Zakhor *et al.* [33] proposed an iterative technique based on this theory. The basic idea is to impose a number of constraints on the contaminated image in order to restore its original artifact-free form. In their method, there are two constraint sets: band-limitation constraint and quantization constraint. The band-limitation constraint is derived from the fact that the encoded image with blocks contains high-frequency horizontal and vertical artifacts corresponding to the discontinuities at the edges of blocks. This constraint can be referred as a low-pass filtering constraint. The quantization constraint is derived from the quantizer. Because the quantization intervals for each BDCT coefficient is assumed to be known in decoding a BDCT-encoded image, the quantization constraint ensures that in restoring images with compression artifacts, BDCT coefficients remain in their original quantization interval. Their iterative procedure can be summarized as follows. In the first part of each iteration, they

low-pass filter the image that has high-frequency artifacts. In the second part, they apply the quantization constraint. First, they divide the image into blocks and take the DCT for each block. Then they project any coefficient outside its quantization range onto its appropriate value. Under these conditions, the POCS theory guarantees that iterative projection between the two constraint sets results in convergence to an element in the intersection of the two sets.

Weerasinghe *et al.* [38] introduced a new family of convex smoothness constraint sets, which allows adaptive smoothing on different regions of the image. The algorithm formulates region smoothness constraint by exploiting the property of images having regions of uniform color value.

Since methods based on POCS are usually iterative, they are computationally expensive and seldom used in interactive applications.

2.3.3 Learning by Examples

In recent years, some learning-by-example methods have been proposed. Chan *et al.* [3] proposed to classify small local boundary regions according to their pixel value distribution. Then appropriate linear predictor is selected to estimate the corresponding pixels at the block boundary. One difficulty is that linear predictor is not accurate enough to deblock the contaminated image. Another problem is that one-dimensional line segment neighborhood provides insufficient neighborhood information.

Qiu [29] proposed a learning-by-example method to postprocess block-coded images. The technique is based on the multilayer perceptron (MLP) neural network. Just like [3], this method only processes pixels at the block boundary and does not handle artifacts within the block. Since ringing artifacts may appear anywhere in the image, these methods may not be able to handle ringing artifacts comprehensively.

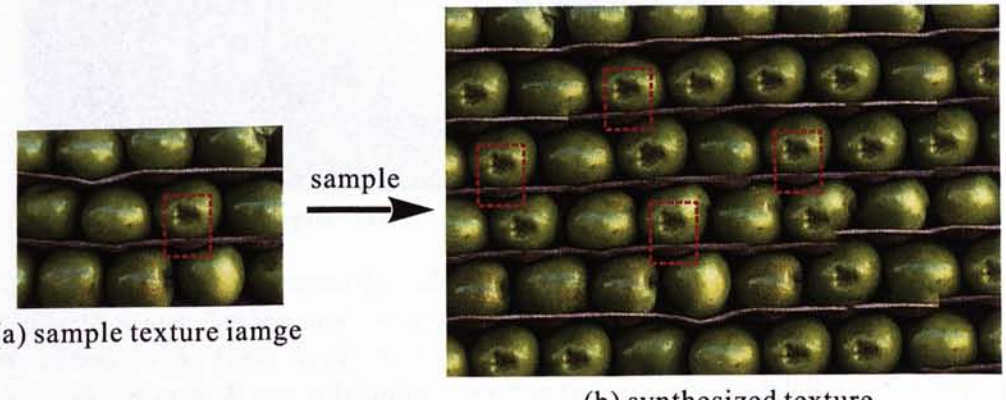
Our method is also a learning-by-example approach. We extract a set

of artifact patterns from training examples. Artifact pattern is a block of pixels. These patterns are organized by TSVQ for efficient query (explained in Section 6). This tree is the *priori* knowledge of BDCT compression artifacts behavior. With this knowledge, we *synthesize* an artifact-reduced image, from the input contaminated high-contrast image.

2.4 Other Related Work

Our learning-by-example method is inspired by the previous work in texture synthesis [7, 8, 28, 35, 39]. When synthesizing a texture, we want the generated texture to be perceptually similar to the example texture, as shown in Figure 2.3. This concept of perceptual similarity has been formalized as a Markov random field (MRF). MRF has been studied extensively in the context of computer vision [21].

Efros *et al.* [8] first modeled the sample input texture as MRF and synthesized larger texture seamlessly. The key idea is to copy pixels from the input texture by matching neighborhood. An example of texture synthesis is given in Figure 2.3. Sampled neighborhoods are labeled by boxes. Texture synthesis result is made by Efros *et al.* [7].

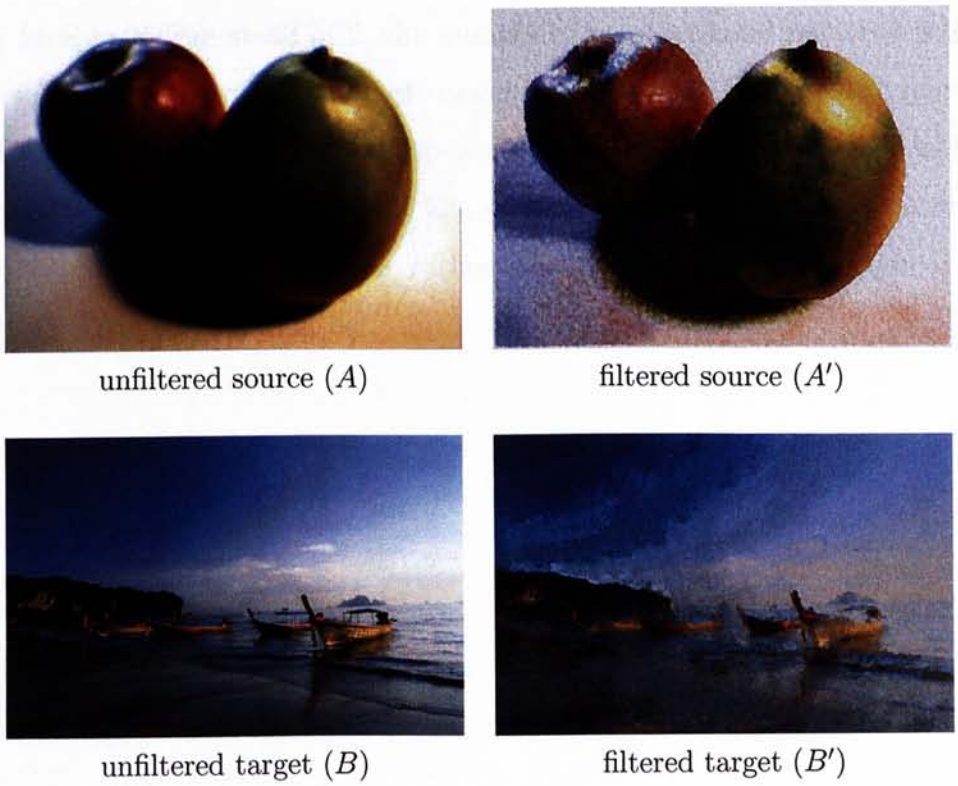


(a) sample texture iamge

(b) synthesized texture

Figure 2.3: This is an example of texture synthesis. Sampled neighborhoods are labeled by boxes. Texture synthesis result is made by Efros *et al.* [7].

Hertzmann *et al.* [13] extended the idea to a learning-by-example technique for non-photorealistic rendering (NPR). They gave a new framework for processing images by example, which generalizes texture synthesis for the case of two corresponding image pairs. Their method is to compute a new "analogous" image B' that relates to B in "the same way" as A' relates to A. Here A, A', and B are inputs, and B' is the output. An example is given in Figure 2.4. In this example, A and A' are a training pair for the watercolor NPR filter. B is a test image. The output B' is in the style of watercolor, learned from A and A'. This watercolor filter result is made by Hertzmann *et al.* [13].



unfiltered source (A) filtered source (A')

unfiltered target (B) filtered target (B')

Figure 2.4: An image analogy. Hertzmann *et al.* [13] proposed a method to compute a new "analogous" image B' that relates to B in "the same way" as A' relates to A. Here A, A', and B are inputs, and B' is the output. In this example, A and A' are a training pair for the watercolor NPR filter. B is a test image. The output B' is in the style of watercolor, learned from A and A'. This watercolor filter result is made by Hertzmann *et al.* [13].

At the same time, Mese *et al.* [24] proposed a Look Up Table (LUT) based method for inverse halftoning of images. The LUT for inverse halftoning is obtained from the histogram gathered from a few sample halftone images and corresponding original images. To determine the inverse halftone value at a point, their algorithm looks at the pixel's neighborhood, and depending upon the distribution of pixels in the neighborhood, it assigns a contone value from the precomputed LUT. Because halfton images are binary images, it is much easier to build LUT. The number of neighborhood patterns is not so large, which is 2^n, where n is the number of neighborhood pixels. As to BDCT artifact removal, the number of different patterns is very large. Because the gray level is 256, in stead of 2, the number of neighborhood patterns is 256^n. It is impossible to build a LUT of reasonable size for BDCT artifact removal.

Recently, Freeman *et al.* [9] proposed another learning-by-example method for super-resolution. This method is a supervised approach, learning a low to high resolution patch model (or rather storing examples of such maps) and utilizing a Markov random field.

Chapter 3

Contamination as Markov Random Field

3.1 Markov Random Field

In the proposed method, we model the contaminated image as a Markov random field (MRF) [21]. First, we explain the concept of MRF. The *Markov random field* theory, a branch of probability theory, provides a foundation for the characterization of contextual constraints and the derivation of the probability distribution of the interacting features.

Let $\mathcal{S}$ index a discrete set of m sites

$$\mathcal{S} = \{1, \ldots, m\}$$

in which $1, \ldots, m$ are indices. A site often represents a point or a region in the Euclidean space such as an image pixel. The inter-relationship between sites is maintained by a so-called *neighborhood system.* A neighborhood system for $\mathcal{S}$ is defined as

$$\mathcal{N} = \{\mathcal{N}_i \mid \forall i \in \mathcal{S}\}$$

where $\mathcal{N}_i$ is the set of sites neighboring i. The neighboring relationship has the following properties:

(1) a site is not neighboring to itself: $i \notin \mathcal{N}_i$;

(2) the neighboring relationship is mutual: $i \in \mathcal{N}_{i'} \Longleftrightarrow i' \in \mathcal{N}_i$.

Let $F = \{F_1, \ldots, F_m\}$ be a family of random variables defined on the set of $\mathcal{S}$, in which each random variable F_i takes a value of f_i. The family F is called a random field. The probability that random variable F_i takes the value f_i is denoted $P(F_i = f_i)$, abbreviated $P(f_i)$. F is said to be a Markov random field on $\mathcal{S}$ with respect to a neighborhood system $\mathcal{N}$ if and only if the following two conditions are satisfied:

$$\begin{aligned} P(f) &> 0, \ \forall f \in \mathbb{F} && \text{(positivity)} \\ P(f_i \,|f_{\mathcal{S}-\{i\}}) &= P(f_i \mid f_{\mathcal{N}_i}) && \text{(Markovianity)} \end{aligned}$$

where $\mathcal{S} - \{i\}$ is the set difference. The positivity is assumed for some technical reasons and can usually be satisfied in practice. The Markovianity depicts the local characteristics of F. In MRFs, only neighboring labels have direct interactions with each other. We will explore the Markovianity of BDCT contamination in the following section.

3.2 Contamination as MRF

To reduce visual artifacts due to BDCT-encoding, we need to understand its source. During BDCT-encoding, a 8×8 block of pixel values, $f(u, v)$, is transformed to frequency domain by the following discrete cosine transform:

$$F(u,v) = \frac{1}{4} C(u) C(v) \left[\sum_{i=0}^{7} \sum_{j=0}^{7} f(i,j) \cos \frac{(2i+1)u\pi}{16} \cos \frac{(2j+1)v\pi}{16} \right], \quad (3.1)$$

where

$$C(u), \ C(v) = \begin{cases} \dfrac{1}{\sqrt{2}} & \text{if} \quad u, v = 0 \\ 1 & \text{otherwise,} \end{cases}$$

and $F(u, v)$'s are 64 DCT coefficients to be stored. Each DCT coefficient is then quantized according to a fine-tuned quantization table in JPEG standard. DCT itself does not introduce error, but the following quantization does.

During DCT, the block of pixel values is transformed in a deterministic manner. Moreover, since DCT coefficients are quantized by predefined quantization table in JPEG standard, the introduced error is also deterministic. In other words, given the same input image and compression ratio, the same visual artifacts will be obtained.

After inverse discrete cosine transform (IDCT), a quantization error in one DCT coefficient may affect every pixel in the 8×8 block. In other words, once a pixel is error-contaminated, its local neighborhood is very likely to be contaminated as well. This observation suggests us to model the *contaminated image* as Markov Random Field (MRF) [21]. MRF is the 2D expansion of Markov Random Chain. It has been widely used in image processing and analysis. In MRF, the probability that a pixel takes certain value is determined by the values of its local neighbors. This local property is known as Markovianity. As contaminated image is subdivided into 8×8 pixel block and each block is encoded independently, visual artifacts must also be localized. Hence it conforms to the spirit of MRF.

Our method is different from the traditional MRF techniques which are usually iterative. Instead, it is inspired by the previous work in texture synthesis [7, 8, 28, 35, 39]. Efros *et al.* [8] first modeled the sample input texture as MRF and synthesized larger texture seamlessly. The key idea is to synthesize a pixel value $\hat{q}$ by looking up the pixel value $\hat{p}$ in the sample texture with $\hat{p}$'s neighborhood matches $\hat{q}$'s neighborhood. Hertzmann *et al.* [13] extended the idea to a learning-by-example technique for non-photorealistic rendering. At the same time, Mese [24] proposed a Look Up Table (LUT) based method for inverse halftoning of images. Recently, Freeman [9] proposed another learning-by-example method for super-resolution. All these methods are based on MRF.

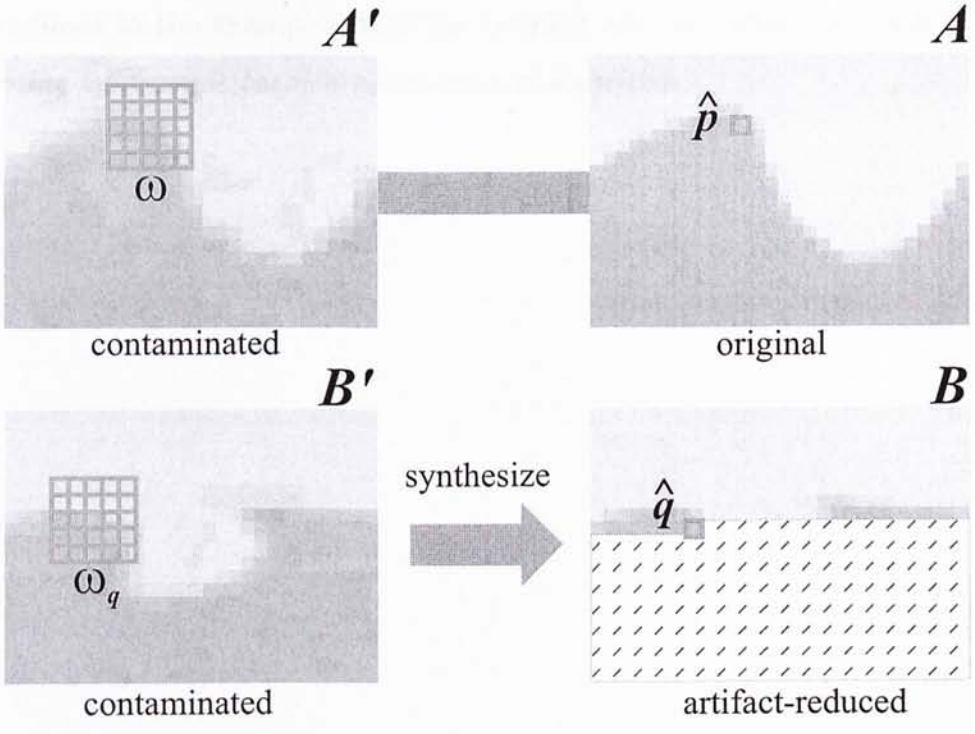


Figure 3.1: The basic idea. Instead of post-filtering the contaminated image B', we try to synthesize an artifact-reduced image B based on the knowledge learned from the training pairs of the original error-free (A) and the contaminated images (A').

Intuitively speaking, the basic idea of our method is to learn the relationship between the original error-free (A) and the error-contaminated images (A') from a training set (Figure 3.1). The training set contains pairs of the original and contaminated images. With the learned knowledge, we try to *synthesize* an artifact-reduced image (B) instead of post-filtering artifacts in the contaminated image (B'). The synthesis is performed in a pixel-by-pixel manner. When synthesizing a pixel $\hat{q}$ in image B based on the contaminated image B', the corresponding local neighborhood ω_q in B' is used to search within the training set. When the best matching block ω is found in the contaminated image A', its corresponding center pixel $\hat{p}$ in the original image A is copied to the synthesized image B. This process is applied to every pixel in the edge region where ringing artifacts usually appear. Since we learn how artifacts are

introduced in the examples from the training set, we called our method the *learning-by-example based artifact-removal algorithm.*

Chapter 4

Training Set Preparation

The first step of the proposed method is to setup a training set. We employ several pairs of original images and their corresponding BDCT-encoded images (contaminated images) to form the training set.

4.1 Training Images Selection

There are two ways to obtain the training set. The first one is to randomly select images from any image library. Since our application is on high-contrast images, the image pairs we choose are mainly cartoon images. One training set generated by this way is shown in Figure 4.1. For simplicity, we only show one image for each pair.

Randomly selected images cannot provide sufficient representability, and cannot cover all cases. It is impracticable that the number of training images is too large. So we try to generate a typical training set to tailor for our purpose. As ringing artifacts appear near the sharp edges, we prepare the training set by generating images containing two basic types, *circle* and *corner* (Figure 4.2). Although circle looks simple, it captures most edge information, and almost any curve can be formed by connecting pieces of circular arcs. The corner type images are mainly used to tackle sharp changing outlines like the star shape. Multiple instances of these two basic types are generated, each

with different foreground intensity, background intensity, and line width.

Figure 4.1: One training set consists of image pairs randomly selected from one cartoon image library. For simplicity, we only show one image for each pair.

4.2 Bit Rate

We need to consider the problem of bit rate, since it is one of the most important parameters of BDCT compression. In image compression study, bit rate ("bits per pixel") is the number of bits of information stored per pixel of an image. BDCT compression versus different bit rates will introduce different ringing artifacts, as shown Figure 4.3. Even though given the same training images, when compression bit rates are distinct, different training sets may be generated. In the proposed method, training images are all compressed with a specified bit rate. If we generate a training set combining many bit rates, the size of this training set will be greatly increased, which further more degrades the efficiency of our method. In the Chapter 8, we will vary bit

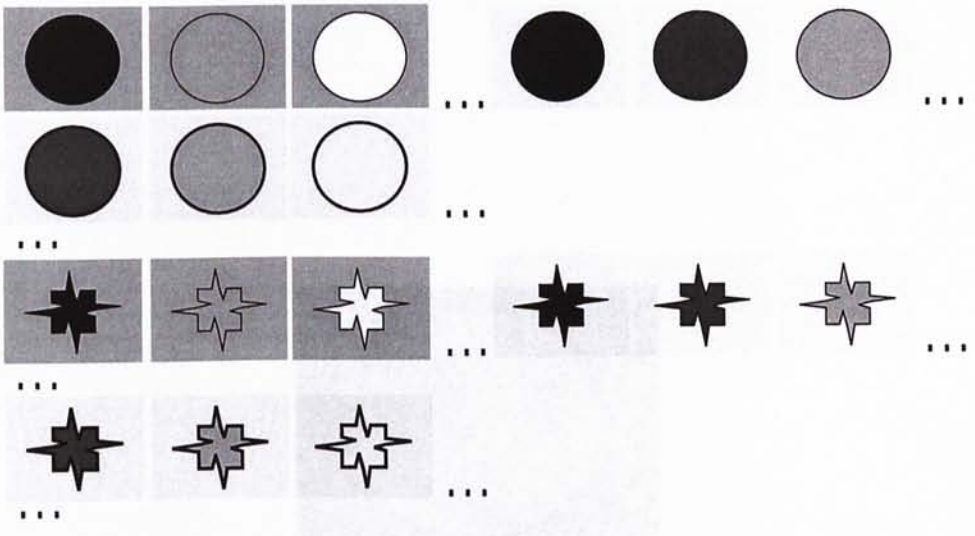
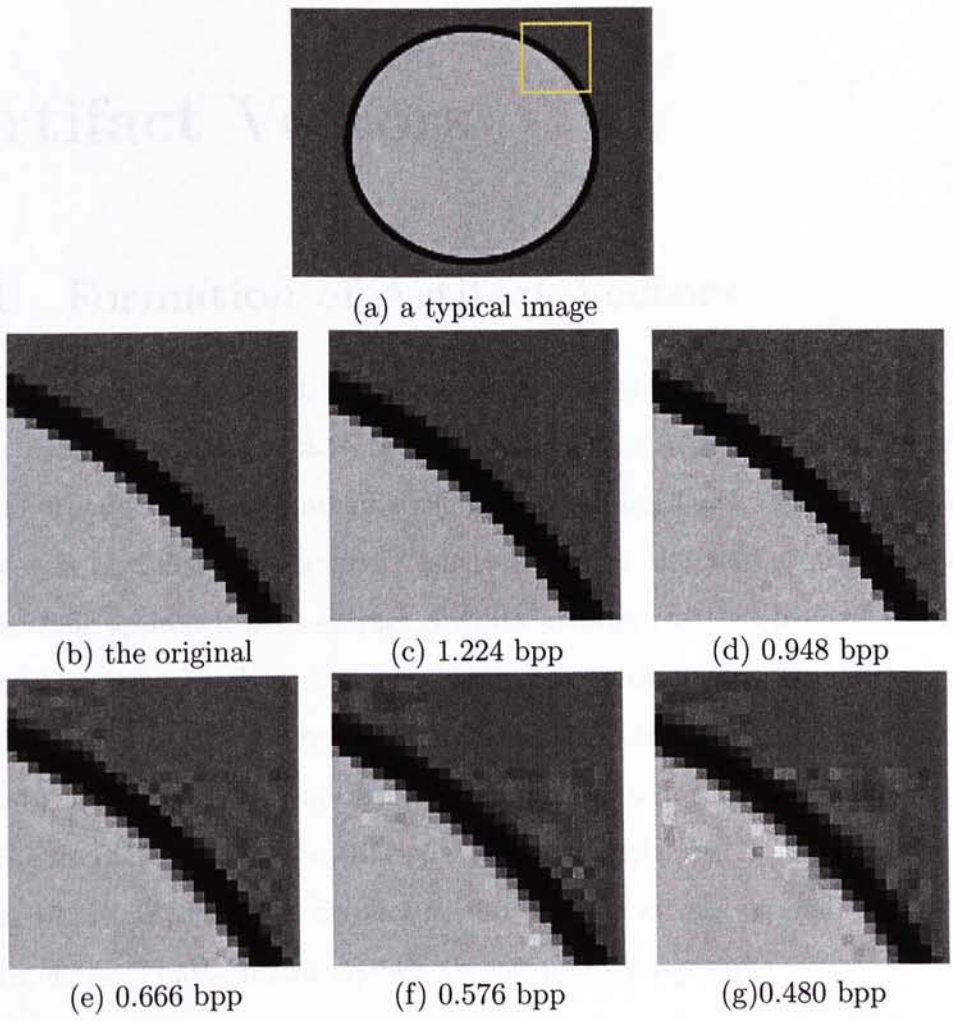


Figure 4.2: We generate this typical training set tailored for our purpose.

rates for both our training set and the test set, to verify the feasibility of our learning-by-example method when bit rates are different.



(a) a typical image

(b) the original (c) 1.224 bpp (d) 0.948 bpp

(e) 0.666 bpp (f) 0.576 bpp (g)0.480 bpp

Figure 4.3: The original image (a) is compressed to different bit rates. (b) is the blowup of the original image. (c)-(g) are the blowups of the compressed images with different bit rates shown below.

Chapter 5

Artifact Vectors

5.1 Formation of Artifact Vectors

Given training image pairs, we extract distinct *artifact vectors* from them. The most computational expensive part of our method is the query of matching neighborhood. Naively searching the best matching block in the training set images is obviously impractical, especially when the size of training set is large. Our approach is to extract distinct *artifact patterns* from the training set and we only search within these artifact patterns. An artifact pattern is a block of pixels (neighborhood) ω in the contaminated image A'. Each artifact pattern *implies* a corresponding center pixel value $\hat{p}$ in the original counterpart A (Figure 3.1). Since the artifact pattern is linearized and stored in a vector form, we usually called it *artifact vector*. From now on, we shall use the two terms, artifact pattern and artifact vector, interchangeably.

Selecting an appropriate size of artifact vector (the size of neighborhood) is a trade-off between the computational cost and the accuracy. Larger neighborhood in general provides more local information and hence returns better result, but the computation is more expensive. Since 8×8 blocks are used in BDCT, the size of neighborhood should not exceed 8×8. From our experience, a 5×5 neighborhood provides adequate local information and tractable computation.

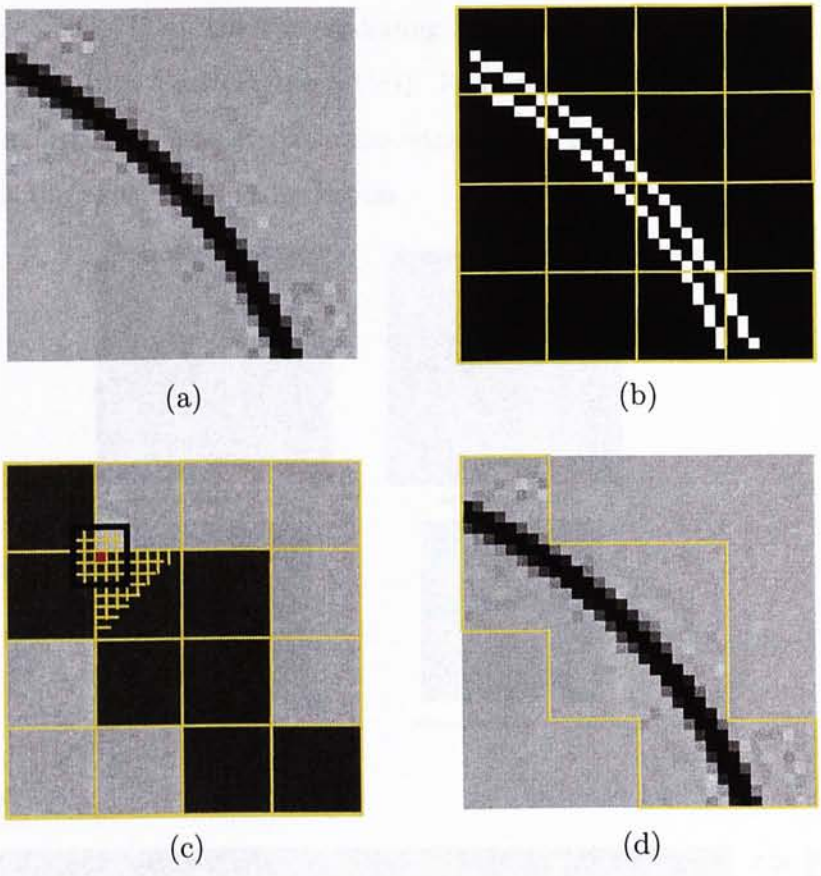


Figure 5.1: Extraction of artifact vectors from edge blocks. (a) The contaminated image. (b) Edge detection. (c) The tagged edge blocks. The 5×5 neighborhood of each pixel in the edge block is used to form an artifact pattern in the training set. (d) Edge region in the contaminated image.

Since ringing artifacts mainly appear at the region with strong edges, we only need to extract distinct artifact patterns from edge region. This can drastically reduce the number of distinct artifact patterns. Therefore, the first step to identify the edge region. The identification process is shown in Figure 5.1. We first apply "Sobel" edge detection to the *contaminated* image (Figure 5.1(a)) in the training set and edge pixels are marked (Figure 5.1(b)). The reason we did not apply the edge detection to the original image is because the original image is not available during artifact removal. Next, we check each non-overlapping 8×8 block in the BDCT-encoded image to see if it contains

any edge pixel. If so, the corresponding 8×8 block in the *original* image is tagged as an *edge block* (Figure 5.1(c)). From Figure 5.1(d), we can see that almost all artifacts appear within the edge region. So we only synthesize new value for the pixels in the edge region.

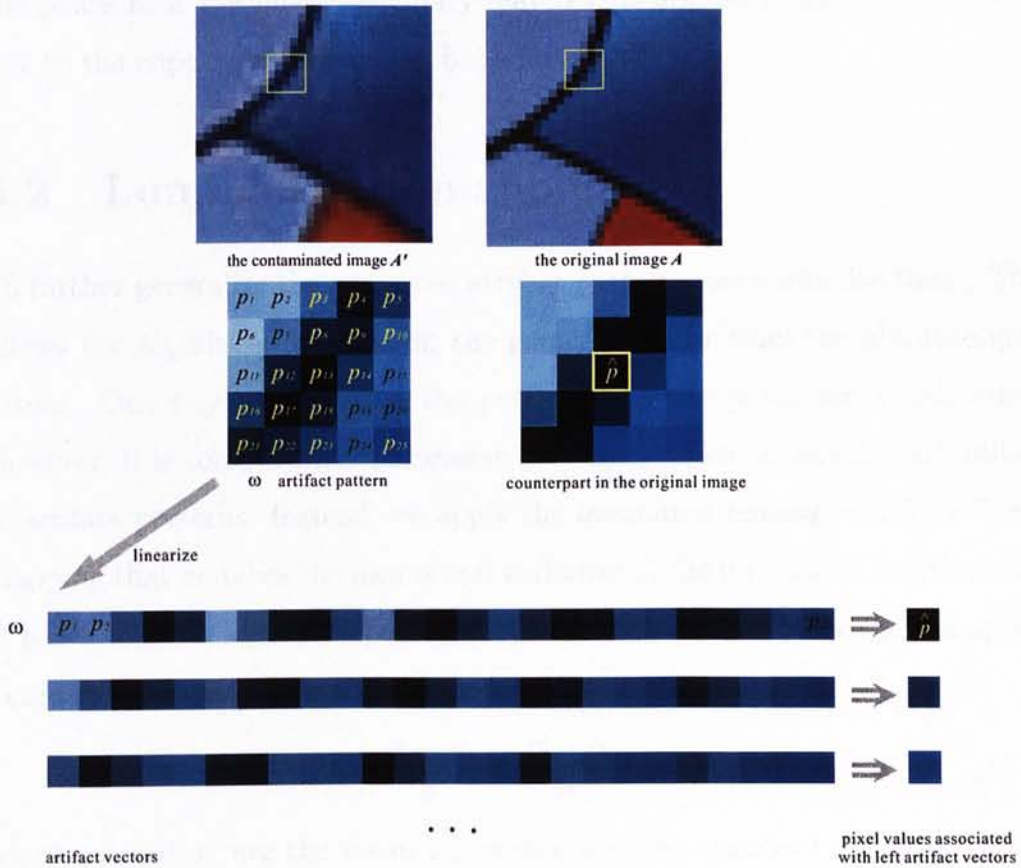


Figure 5.2: Formation of artifact vectors. The original image A (top left) and its contaminated counterpart A' (top right) are shown on the top row. The current pixel $\hat{p}$ and its corresponding neighborhood ω in A' are blown up in the second row. The 25 pixels in ω are linearized and stored together with $\hat{p}$.

For each pixel $\hat{p}$ in the edge block of the *original* image, we look up its corresponding 5×5 neighborhood ω in the *contaminated* counterpart (Figure 5.2). This neighborhood is then linearized to form the artifact vector. If the artifact vector is distinct, it is stored together with $\hat{p}$. We say ω implies $\hat{p}$, and denoted as,

$$\omega \Rightarrow \hat{p}.$$

Note that only ω is searched and matched during synthesis. The implied pixel value $\hat{p}$ is the data to retrieve. During the extraction of artifact vectors, the pixels near the image boundary may not be able to form artifact vectors due to the clipping by the image boundary.

5.2 Luminance Remapping

To further generalize the extracted artifact patterns, we normalize them. This allows the algorithm to focus on the patterns rather than the absolute gray values. One way to normalize the pattern is to use histogram equalization. However, it is computational intensive because we have to handle half million of artifact patterns. Instead, we apply the *luminance remapping* [13], a linear mapping that matches the means and variances of the luminance distributions. If p is the luminance of a pixel in artifact pattern, and p' is the remapped luminance, luminance remapping is described by:

$$\frac{p - \mu_p}{\sigma_p} = \frac{p' - \mu_n}{\sigma_n}, \tag{5.1}$$

where μ_p and σ_p are the mean luminance and the standard deviation of the original pattern respectively; μ_n and σ_n are the mean luminance and standard deviation of the normalized pattern and predefined as 0.5 and 0.3 respectively. These normalized patterns are then stored. Note that this luminance remapping is also needed during the artifact removal.

5.3 Dominant Implication

Theoretically, one artifact pattern ω may not imply (be associated with) a unique pixel value $\hat{p}$. An artifact pattern may in fact imply a set of k possible

pixel values, each with different probability.

$$\omega \Rightarrow \{\hat{p}_1, \hat{p}_2, \cdots, \hat{p}_k\}$$

and

$$\sum_{i=1}^{k} \text{Prob}(\hat{p}_i) = 1$$

where $\text{Prob}(\hat{p}_i)$ is the probability of $\hat{p}_i$.

The probability of each $\hat{p}_i$ can be obtained from the training set. During the extraction of artifact patterns, we can record the occurrences of each possible $\hat{p}_i$. The normalized count of each $\hat{p}_i$ will then be its probability. This probability distribution should be stored together with the artifact pattern. During synthesis, the pixel value should be synthesized according to this probability distribution.

Interestingly, we observed a phenomenon that there is always a dominant pixel value $\hat{p}^*$ associated with each artifact pattern. One reason is related to the deterministic properties of DCT and quantization. Another possible reason is because we focus on high-contrast images which usually have less variation in pixel values. High-contrast images usually have large smooth color regions and strong edges/outlines.

To verify our finding, we test this problem on the training set generated by ourselves, which contains two basic types, *circle* and *corner*, as shown in Figure 4.2. We plot the probabilities of the dominant pixel values of all artifact patterns from our training set in Figure 5.3, with 0.735 bpp. In this training set of 112 image pairs, there are 377,520 distinct artifact patterns. The average probability of dominant pixel value, associated with these patterns, is 99.98%. Among these artifact patterns, 377,420 patterns (99.97%) are associated with a unique pixel value. Only 100 patterns associate with two or more pixel values. In Figure 5.3, we sort the artifact patterns according to the probabilities of their dominant pixel values, so that the smallest is on the left. The non-unique dominant pixels occupy a tiny portion only.

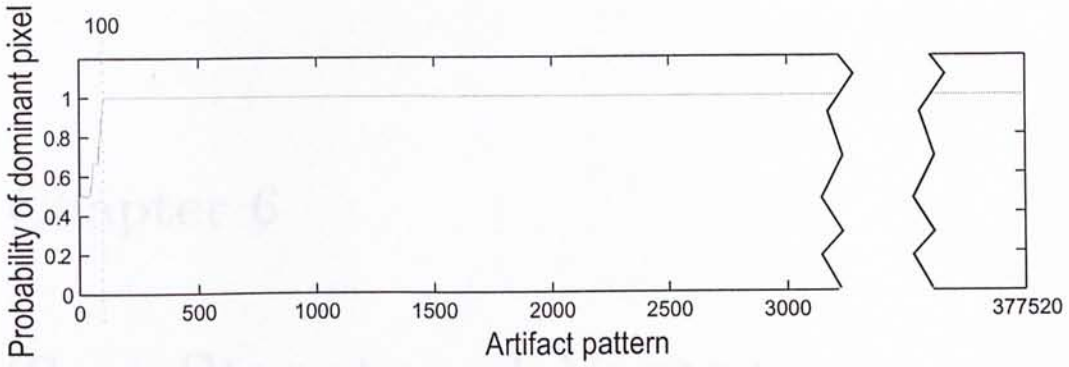


Figure 5.3: Probabilities of dominant pixels in our training set with the bit rate of 0.735 bpp. For illustration purpose, artifact patterns are sorted increasingly according to the probabilities of dominant pixels.

This observation of dominant implication suggests us a way to simplify and speed up the process. We can simply keep the dominant pixel value $\hat{p}^*$ and discard all other pixel values $\{\hat{p}_1, \hat{p}_2, \cdots, \hat{p}_k\} - \{\hat{p}^*\}$. Hence, there is no need to store the probability distribution.

Chapter 6

Tree-Structured Vector Quantization

Even though we only extract artifact patterns from the Y component of color images, there are still enormous number of patterns from the training set. Management of such huge number of artifact patterns is crucial to fast query, hence fast synthesis. Naive searching among the sea of artifact patterns is obviously impractical. Therefore, an indexing scheme is needed to achieve fast query. Furthermore, even though we have a lot of extracted artifact patterns, we may not find an exact match in some cases. This is especially true when the training set is small or it contains images with less diversity. In case no exact matching exists, the indexing scheme should allow us to locate the closest artifact pattern. To solve this problem, we employ the indexing technique called tree-structured vector quantization (TSVQ) [10].

6.1 Background

6.1.1 Vector Quantization

Vector quantization (VQ) [10] is a generalization of scalar quantization to the quantization of a vector, an ordered set of real numbers. VQ is usually , but not exclusively, used for the purpose of data compression.

A vector quantizer Q of dimension k and the size N is a mapping from a vector (or a "point") in k-dimensional Euclidean space, $\mathcal{R}^k$, into a finite set $\mathcal{C}$ containing N outputs or reproduction points, called *code vectors* or *codewords.* Thus,

$$Q : \mathcal{R}^k \rightarrow \mathcal{C} ,$$

where $\mathcal{C} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N)$ and $\mathbf{y}_i \in \mathcal{R}^k$ for each $i \in \{1, 2, \ldots, N\}$. The set $\mathcal{C}$ is called the *codebook* or the *code* and has size N, meaning it has N distinct elements, each a vector in $\mathcal{R}^k$.

6.1.2 Tree-Structured Vector Quantization

Tree-structured vector quantization (TSVQ), one of the most effective and widely-used techniques for reducing the search complexity in VQ is to use a tree-structured codebook search. The search complexity of normal VQ is N. If the codebook tree is an m-ary tree, the search complexity is reduced to $\log_m N$. In an m-ary tree, each non-leaf node has m children. In TSVQ, the search is performed in stages. In each stage a substantial subset of candidate code vectors is eliminated from consideration by a relatively small number of operations. In an m-ary tree search with a balanced tree, the input code vector is compared with m predesigned test vectors at each stage or node of the tree. The nearest (minimum distortion) test vector determines which of m paths through the tree to select in order to reach the next stage of searching. At each stage the number of candidate code vectors is reduced to $1/m$ of the previous set of candidates.

The algorithm of a standard TSVQ design is as follows [10]:

Step 1. Use the training set $\mathcal{T}$ to generate a code book $\mathcal{C}^*$ of size m test vectors for the root node (level 0) of the tree. Partition the training set into m new subsets $\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_{m-1}$;

Step 2. For each i, design a test codebook $\mathcal{C}_i$ of size m using $\mathcal{T}_i$; (We thus obtain the test codebooks for the m nodes at level 1 of the tree.)

Step 3. Partition each training set $\mathcal{T}_i$ into m training subsets $\mathcal{T}_{ij}$ and use these new training sets to design the m^2 test codebooks $\mathcal{C}_{ij}$ for level 2;

Step 4. Continue **Step 3** until the level $d-1$ is reached, where d is the maximum depth of this tree structure. (The test vectors in the test codebooks obtained for the nodes at this level are actually code vectors for the terminal nodes of level d. The collection of all these test vectors at this level constitute the codebook.)

In the following subsection, we will introduce how to generate a codebook $\mathcal{C}$, given the training set $\mathcal{T}$. In fact, we use the K-Means clustering algorithm to generate the codebook.

6.1.3 K-Means Clustering

In the proposed method, a codebook $\mathcal{C}$ is generated using the K-Means clustering [12] applied to the given training set $\mathcal{T}$. The K-Means method is known as a clustering method since the user must first define the number of clusters, after which the algorithm partitions the data iteratively until a solution is found. The algorithm of the standard K-Means clustering is as follows:

Step 1. Randomly initialize m cluster centers $\mathcal{C}^{(0)} = (\mathbf{y}_1^{(0)}, \mathbf{y}_2^{(0)}, \ldots, \mathbf{y}_m^{(0)})$, for iteration $j = 0$;

Step 2. Assign each data vector in $\mathcal{T}$ to the cluster $\mathcal{T}_i$ with the nearest center $\mathbf{y}_i^{(j)}$;

Step 3. Set new cluster center (The center $\mathbf{y}_{i+1}^{(j)}$ is the average of all vectors in $\mathcal{T}_i$, at iteration j.) $\mathbf{y}_i^{(j+1)}$ of each $\mathcal{T}_i$;

Step 4. Goto **Step 2** until convergence.

By K-Means clustering, the codebook is updated iteratively, until convergence. Convergence is defined by the change of codebook. If after one iteration, the codebook does not change at all, the convergence is reached.

After introducing these basic technologies, we will explain how to employ TSVQ in the proposed artifact removal method.

6.2 TSVQ in Artifact Removal

TSVQ is usually used in data compression. It also has a nice feature that allows efficient searching of the nearest pattern when the exact match does not exist. Because of its hierarchical tree structure, an artifact pattern can be rapidly looked up.

In the proposed method, the construction of TSVQ tree is as follows. Firstly, the centroid of all artifact vectors is computed, and this centroid represents all artifact vectors. A node holding this centroid is formed and is assigned as the root of TSVQ tree. Then we divide the vectors into τ_m groups, where τ_m is user-defined constant. For each group, we compute the centroid and let it be the representative of that group. To do so, we create a node for each centroid and connect it as a child node of the root node. Each child node is actually the root of the subtree representing the corresponding group. Each group (subtree) is then recursively subdivided (branched) until one of the stopping criteria is reached.

To subdivide the set of artifact vectors into groups, we need a metric to measure the similarity among vectors. We employed the Euclidean distance

function.

$$D_e(\omega_a, \omega_b) = \| \omega_a - \omega_b \|^2, \tag{6.1}$$

where ω_a and ω_b are artifact vectors being compared.

There are also two user-controllable stopping criteria: the number of subdivided groups τ_m, and the maximum tree depth τ_d. If any one of them is satisfied, the subdivision should be stopped. Constant τ_m controls the number of groups after subdivision. That is, it controls the number of children of each interior node. If the number of vectors in the current group is less than τ_m, the subdivision should be stopped. Constant τ_d controls the maximum depth of tree, excluding the root level. If the depth of current group is equal to τ_d, further subdivision of the current branch should be prohibited.

Figure 6.1 illustrates an example. For visualization purpose, the artifact vectors are not linearized. In this example, $\tau_m = 3$ and $\tau_d = 2$. There are 10 artifact vectors, $\{\omega_1, \cdots, \omega_{10}\}$. The centroid of these patterns $\{\omega_1, \cdots, \omega_{10}\}$ is computed as ω_{11} and assigned as the root of tree. During the first iteration, these vectors are subdivided into 3 (τ_m) groups $\{\omega_1, \omega_3, \omega_6\}$, $\{\omega_2, \omega_4, \omega_5, \omega_7, \omega_{10}\}$ and $\{\omega_8, \omega_9\}$, according the Euclidean distance function D_e. The centroid of $\{\omega_1, \omega_3, \omega_6\}$ is denoted as ω_{12} and connected as the child node of ω_{11}. Similarly, the centroids of $\{\omega_2, \omega_4, \omega_5, \omega_7, \omega_{10}\}$ and $\{\omega_8, \omega_9\}$ are ω_{13} and ω_{14} respectively. They are then connected as the child nodes of ω_{11}. Nodes ω_{12}, ω_{13} and ω_{14} are then the roots of 3 corresponding subtrees.

During the second iteration, the group represented by ω_{12} contains 3 vectors, that is equal to τ_m. Further subdivision is done by subdividing the group into 3 subgroups, each contains one vector. These 3 leaf nodes are denoted as the ω_1, ω_3 and ω_6, same as the notation of 3 vectors. The group represented by ω_{14} contains only 2 vectors which is less than τ_m. Subdivision is terminated and ω_{14} becomes a leaf node holding vectors $\{\omega_8,\omega_9\}$. In the middle, the group represented by ω_{13} is further subdivided into 3 subgroups. Each of the first

2 subgroups contains only 1 vector while the last subgroup ω_{15} holds 3 vectors. As ω_{15} contains τ_m vectors, further subdivision can be done. However, ω_{15} is already at the maximum allowed depth level of the tree, $\tau_d = 2$. The subdivision should be stopped at this level.

To query a pattern in the tree, we start the comparison with the child nodes of the tree root. The closest child node is then located and its branch is traversed. The process is continued until a leaf node is encountered. The closest vector in the leaf node is matched and returned as the query result. The time complexity of querying a vector is $O(\log_{\tau_m} n)$, where n is the total number of artifact patterns. In Section 7.2, we go through an example of query.

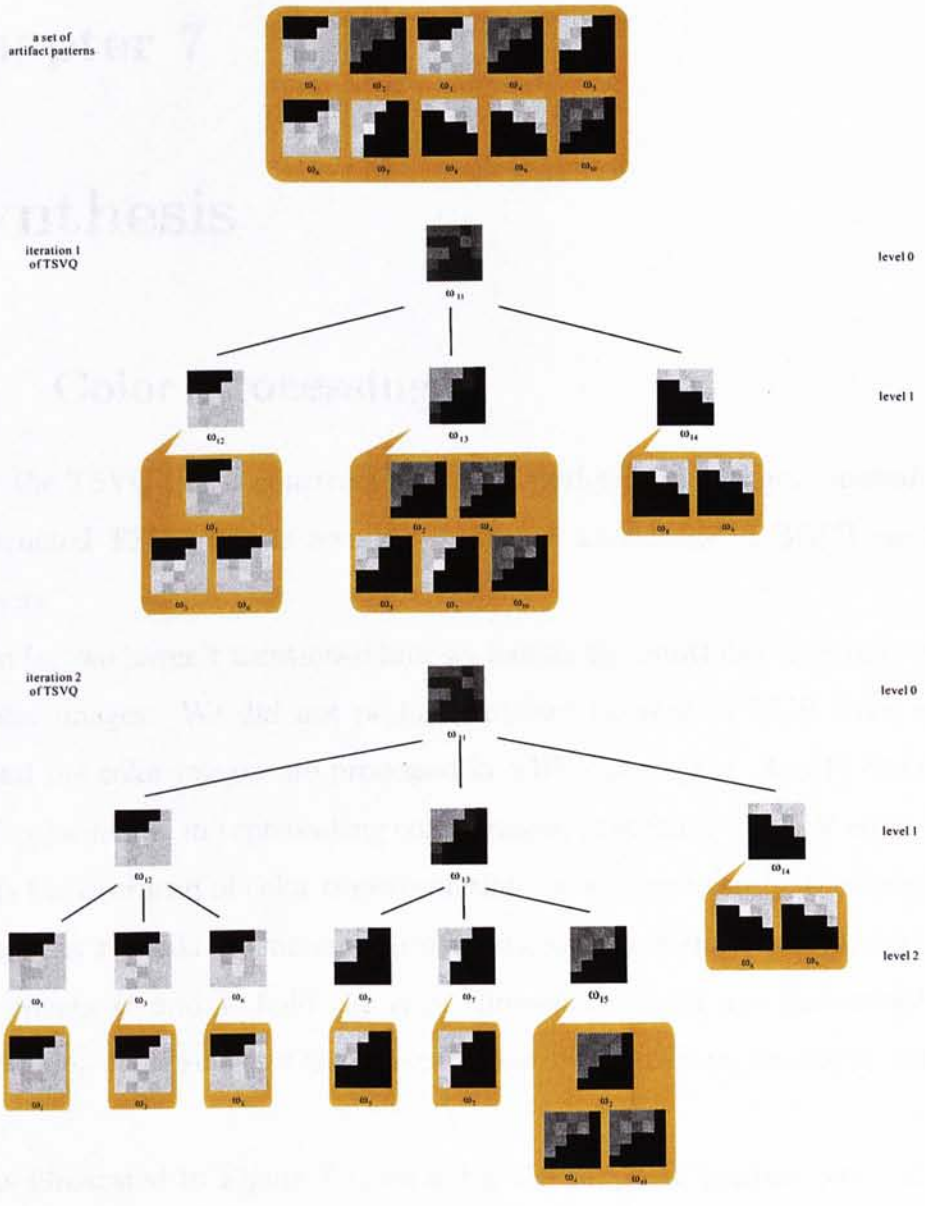


Figure 6.1: The construction of an example TSVQ tree. ω_1-ω_{10} are artifact patterns. The artifact patterns are recursively subdivided into groups during the construction.

Chapter 7

Synthesis

7.1 Color Processing

Once the TSVQ tree is constructed, we can perform the artifact removal. The constructed TSVQ tree is actually the *priori* knowledge of BDCT-encoding artifacts.

So far, we haven't mentioned how we handle the multiple color components in color images. We did not perform artifact removal in RGB color space. Instead the color images are processed in YUV color space. As JPEG utilizes YUV color model in representing color images, processing in YUV color space avoids the overhead of color transformation. More importantly, the luminance component Y holds the human-sensitive visual information while chrominance components U and V hold the color differences which are less sensitive to human [36]. This suggests that we can treat different components in different ways.

As illustrated in Figure 7.1, we apply the proposed artifact-removal algorithm on the Y component only, while components U and V are retained. Original U, V components and processed Y component are then combined. We decide not to touch U and V components because the observable visual artifacts are mainly due to the quantization error in Y component. This heterogeneous strategy reduces both memory consumption and computational

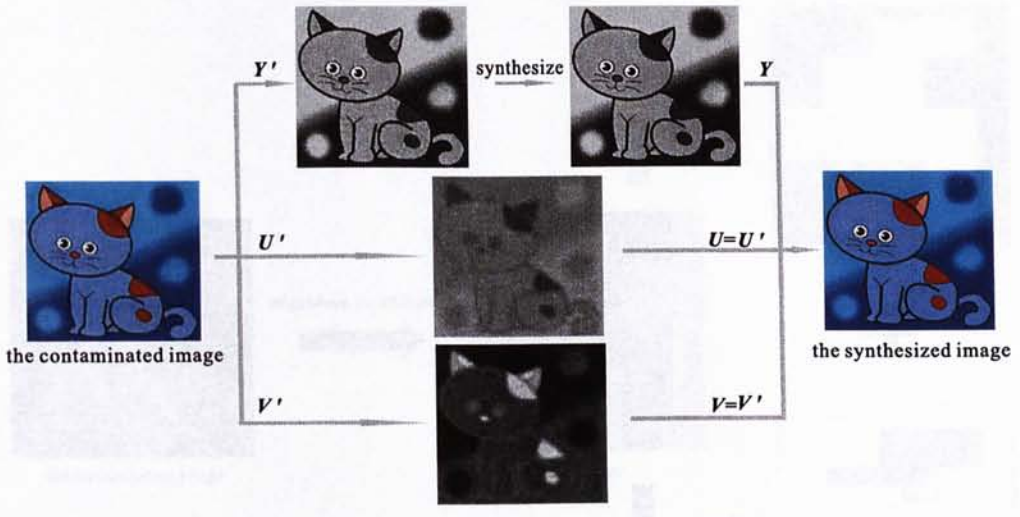


Figure 7.1: Heterogeneous strategy for different color components. For visualization purpose, U and V components are shifted to $[0, 1]$. We apply the proposed artifact-removal algorithm on the Y component only, while components U and V are retained.

cost due to searching, hence it shortens the time for artifact removal while retaining the visual quality of the recovered images.

7.2 Artifact Removal

Just like the formation of artifact vectors, given an input contaminated image, we need to identify the edge and non-edge blocks. The same identification method as in Section 5.1 is used to identify edge blocks (Figure 7.2). All pixels in the non-edge blocks of the contaminated image are simply copied to the output as illustrated on the upper path of Figure 7.2. The non-edge blocks contain no recognizable strong edge and hence they rarely contain ringing artifacts. On the other hand, all pixels in the edge blocks have to be synthesized by querying the TSVQ tree.

The query process is illustrated in Figure 7.3. Given a contaminated image B', we try to synthesize an artifact-reduced image B in a pixel-by-pixel manner for each pixel in edge blocks. For each pixel being synthesized, we form a query

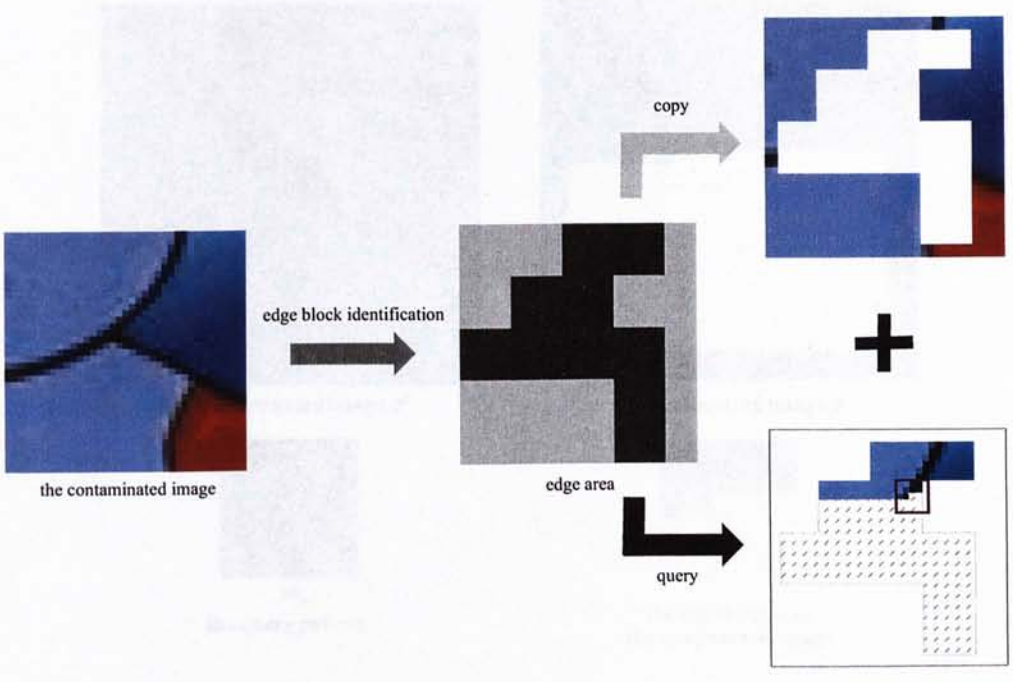


Figure 7.2: Pixels in the contaminated image are classified into two types. Pixels in the non-edge blocks are copied directly as ringing artifacts are not apparent. Pixels in the edge blocks are synthesized by querying the TSVQ.

pattern ω_q by linearizing its corresponding 5×5 neighborhood in image B'. The example contaminated image B' in Figure 7.3 contains only edge blocks. Image B is initialized with a frame of width of 2 pixels. The reason is that pixels on this frame cannot form a 5×5 neighborhood for query. Therefore, these pixels are simply copied from the contaminated image B'. This is one restriction of our current approach.

We then query the artifact pattern ω_q, by traversing the TSVQ tree, as illustrated in Fig 7.4. Firstly, we compare the query vector ω_q with every child node of the root. The one with the smallest Euclidean distance (Equation 6.1) is selected. The search continues to the next level of the branch until a leaf node is encountered. Since the closest pattern should be inside this leaf node, we sequentially compare with all patterns ($\{\omega_1, \omega_2, \omega_3\}$ in this example). This sequential matching process does not take much time as the number of patterns

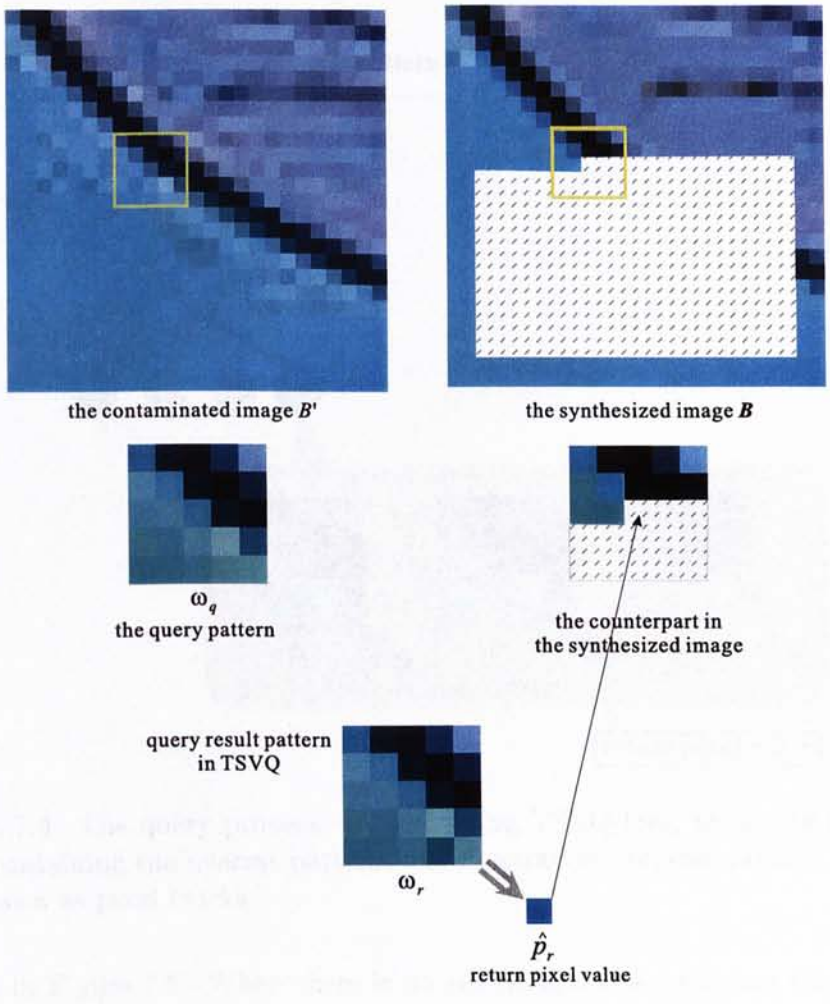


Figure 7.3: Artifact removal. Given the edge region in contaminated image B', we synthesize the artifact-reduced image B in a pixel-by-pixel manner. For illustration purpose, artifact vectors are drawn as a block of pixels.

inside a leaf node is very likely to be bounded by constant τ_m. In this specific example, ω_2 is the closest pattern. Its implied pixel value $\hat{p}_2$ is returned and copied to image B. The synthesis continues until the whole image B is filled.

7.3 Selective Rejection of Synthesized Values

During experiment, we find that if we accept every returned value, the proposed method will introduce error to the synthesized image. An example is

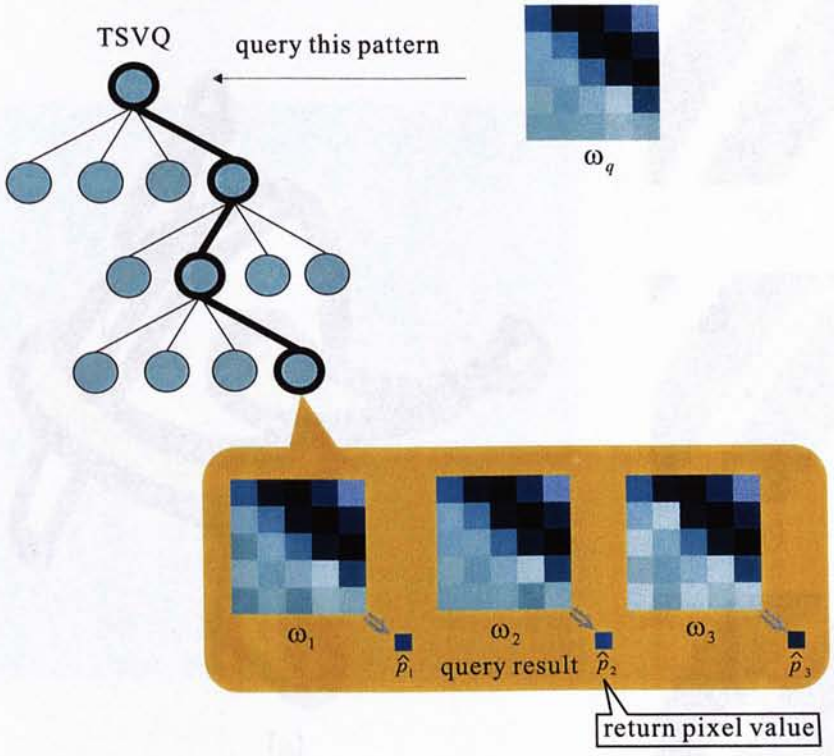


Figure 7.4: The query process. By traversing TSVQ tree, we locate the leaf node containing the nearest pattern. For illustration purpose, artifact vectors are drawn as pixel blocks.

shown in Figure 7.5. When there is no rejection, some errors are introduced in the synthesized image. If a proper threshold is applied, a better image is synthesized. To get a proper threshold, we study the change of the original image and the contaminated image to find the nature of BDCT compression. Especially, we want to know how much BDCT compression changes the pixel value. Our synthesis procedure modifies the pixel value and tries to recover the information destroyed by BDCT compression. This procedure should not change the pixel value too much. From the training set, we want to find the maximum accepted change that our method can introduce. Then we can decide a threshold for pixel value change.

We employ our training set to study the pixel value change introduced by BDCT compression. Figure 7.6 shows the absolute change of pixel value after

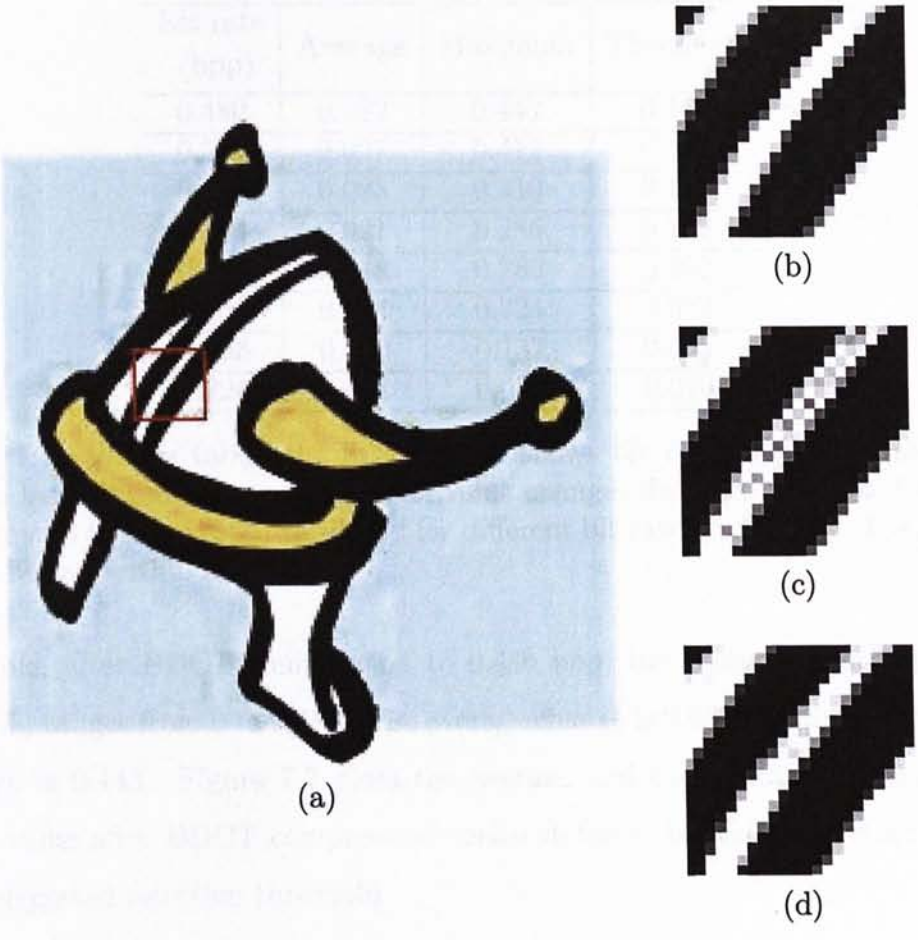


Figure 7.5: (a) A contaminated image. (b) The blowup of the boxed area in (a). (c) The synthesized result without rejection. (d) The synthesized result when applying a proper rejection threshold.

BDCT compression versus different bit rates, ranging from 0.480 bpp to 1.224 bpp. As we can see, while increasing the bit rate, we get the histogram with a higher peak and a larger gradient. That is, when we increase the compression bit rate, more and more pixels get less absolute change and we get the image with better quality. This phenomena tells us that for different bit rates we should apply different thresholds for the pixel value change. This threshold should bound the change of most pixels. By experiment, we find that, if this threshold bounds the change of a large portion (99.5%) pixels, a better result can be gained. Table 7.1 shows the average pixel value change, the maximum pixel value change and the suggested threshold for different bit rates. For

Bit rate (bpp)	Average	Maximum	Threshold
0.480	0.032	0.443	0.188
0.533	0.027	0.412	0.157
0.576	0.023	0.310	0.137
0.614	0.021	0.286	0.122
0.666	0.018	0.286	0.102
0.735	0.014	0.224	0.082
0.863	0.010	0.137	0.055
1.224	0.004	0.039	0.020

Table 7.1: In this table, the first column shows bit rate. From the second to the last column, the average pixel value change, the maximum pixel value change and the suggested threshold for different bit rates, are listed. The pixel value ranges within $[0, 1]$.

example, after BDCT compression to 0.480 bpp, the change of most pixels (99.5%) ranges from 0 to 0.188. The average change is 0.032 and the maximum change is 0.443. Figure 7.7 plots the average and the maximum change of pixel value after BDCT compression versus different bit rates, together with the suggested rejection threshold.

Because of above reasons and experiment data, during artifact removal, we will test the absolute change between the return value and the contaminated value (pixel value in BDCT-encoded image). If the change is larger than a threshold, we will retain the current contaminated pixel. This method solves the problem of introducing error very well.

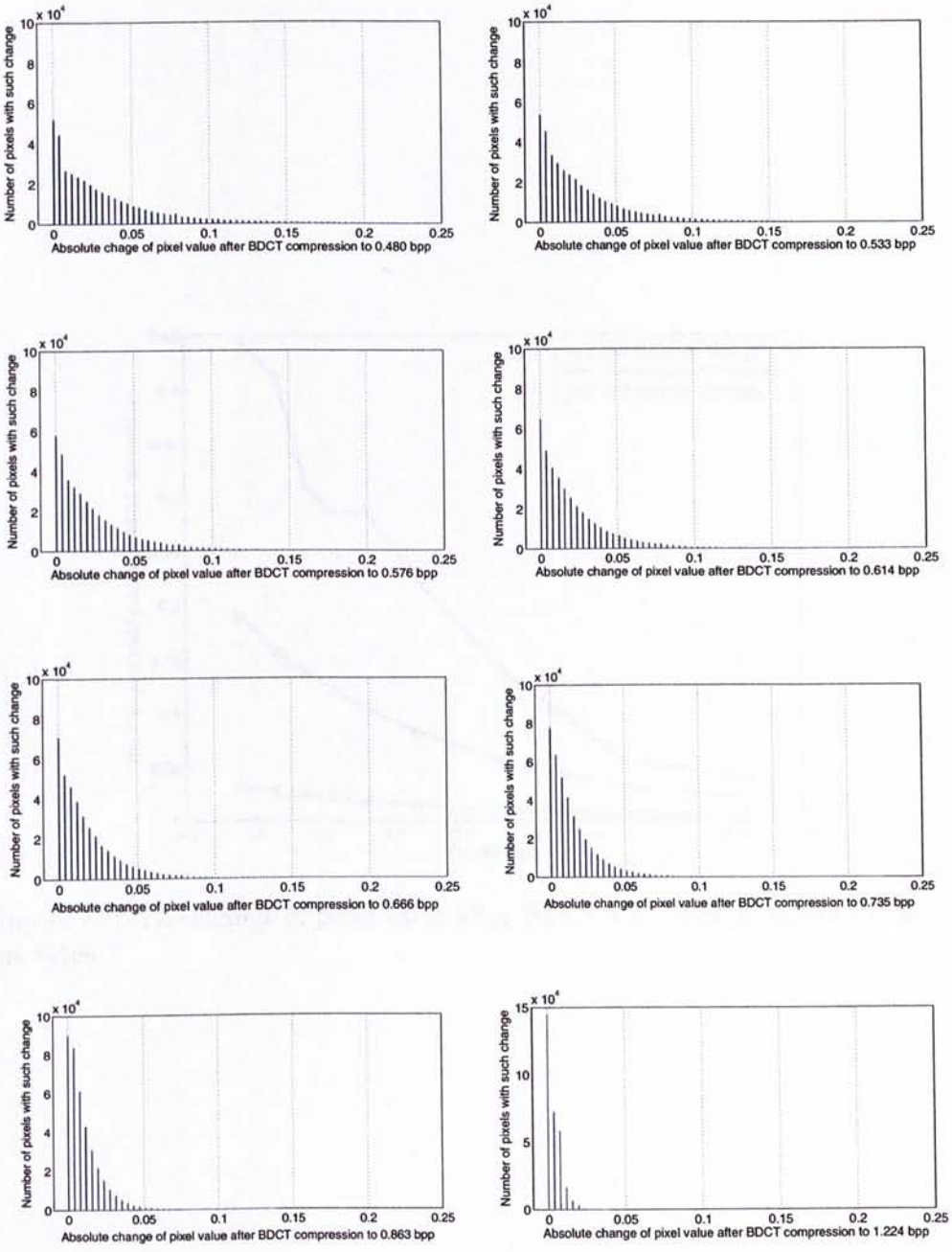


Figure 7.6: The absolute change of pixel value after BDCT compression versus different bit rates, ranging from 0.480 bpp to 1.224 bpp.

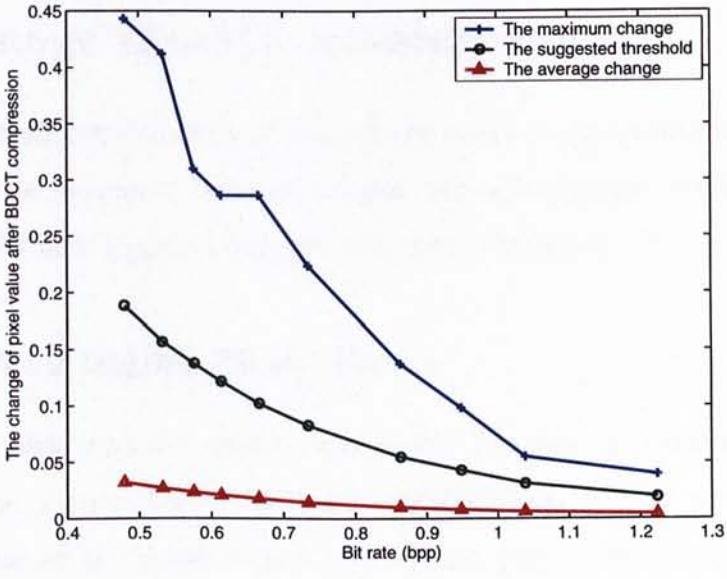


Figure 7.7: The change of pixel value after BDCT compression versus different bit rates.

Chapter 8

Experimental Results

8.1 Image Quality Assessments

In the following experiments, to objectively measure the quality of synthesized images by the proposed method, we use two assessments: peak signal-noise ratio (PSNR) and Mean Structural SIMilarity (MSSIM) [37].

8.1.1 Peak Signal-Noise Ratio

PSNR is a very common quality assessment in image processing. A sample use is in the comparison between an original image and an encoded image. PSNR is one of the most widely used image quality assessments. Assume that we have an original image $\mathbf{X}(i,j)$ that contains $M \times N$ pixels and an assessed (contaminated) image $\mathbf{Y}(i,j)$. PSNR is computed on the luminance component and the pixel values range between black (0) and white (255). First, we compute the mean squared error (MSE) of the contaminated image as follows:

$$MSE = \frac{\sum [\mathbf{X}(i,j) - \mathbf{Y}(i,j)]^2}{M \times N},$$

where, the summation is over all pixels. The root mean squared error (RMSE) is the square root of MSE, as follows:

$$RMSE = \sqrt{MSE}.$$

Then PSNR in decibels (dB) is computed by

$$PSNR = 20\log_{10}(\frac{255}{RMSE}) \ . \tag{8.1}$$

8.1.2 Mean Structural SIMilarity

However, PSNR does not fully capture the perceptual quality. Hence, besides PSNR, we will utilize another assessment called MSSIM [37] to measure our training set experiment results. MSSIM is a particular implementation of the philosophy of structural similarity. MSSIM follows a strategy of modifying the MSE measure so that errors are penalized in accordance with their visibility.

Suppose $\mathbf{X}$ and $\mathbf{Y}$ are the original image and the assessed image, respectively. Then the mean SSIM (MSSIM) will be employed to evaluate the overall image quality:

$$MSSIM(\mathbf{X}, \mathbf{Y}) = \frac{1}{M}\sum_{j=1}^{M} SSIM(\mathbf{x}_j, \mathbf{y}_j) \ ,$$

where $\mathbf{x}_j$ and $\mathbf{y}_j$ are the image contents at the j-th local window; and M is the number of samples in the quality map. The local statistics are computed within a local square window which moves pixel-by-pixel over the entire image. For simplicity, $\mathbf{x}_j$ and $\mathbf{y}_j$ are linearized to vectors. The local statistics include mean μ_x, contrast σ_x, and correlation σ_{xy}, which are defined as

$$\mu_x = \sum_{i=1}^{N} w_i x_i \ ,$$

$$\sigma_x = \left(\sum_{i=1}^{N} w_i (x_i - \mu_x)^2\right)^{1/2} \ ,$$

$$\sigma_{xy} = \sum_{i=1}^{N} w_i (x_i - \mu_x)(y_i - \mu_y) \ ,$$

where $\mathbf{w} = \{w_i | i = 1, 2, \ldots, N\}$ is a Gaussian weighting function.

Given local μ_x, σ_x and σ_{xy}, the local SSIM is defined as

$$SSIM(\mathbf{x}, \mathbf{y}) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} . \tag{8.2}$$

Here the constants C_1 and C_2 are included to avoid instability when $\mu_x^2 + \mu_y^2$ and $\sigma_x^2 + \sigma_y^2$ are very close to zero, respectively. From this definition, we can see that a larger SSIM value implies better image quality. The maximum SSIM value is 1.

To illustrate the advantages of MSSIM over PSNR, an example (from [37]) is shown in Figure 8.1. The original image "Boat" is altered with different distortions, each of which is adjusted to yield nearly identical PSNR relative to the original image. Despite this, the images can be seen to have drastically different perceptual qualities. By the philosophy of MSSIM, it is easy to explain why contrast-stretched image ((b) in Figure 8.1) and mean-shifted image ((c) in Figure 8.1) have higher MSSIM than the JPEG compressed image ((d) in Figure 8.1) and the blurred image ((e) in Figure 8.1). Because in contrast-stretch and mean-shift procedures, nearly all the structure information of the original image is preserved. On the other hand, some structural information from the original image is permanently lost in the JPEG compressed image and the blurred image. In the salt-pepper impulsive noise contaminated image, the structural information is destroyed moderately, so this image gets a medium MSSIM value.

8.2 Performance

As mentioned in Section 4, we setup a training set by ourselves. As shown in Figure 4.2, we create 112 images to setup such training set. These images are originally not contaminated with any BDCT compression artifacts. They are then compressed by the BDCT compression. Then 112 original images and 112 contaminated images are generated. The TSVQ tree is constructed with

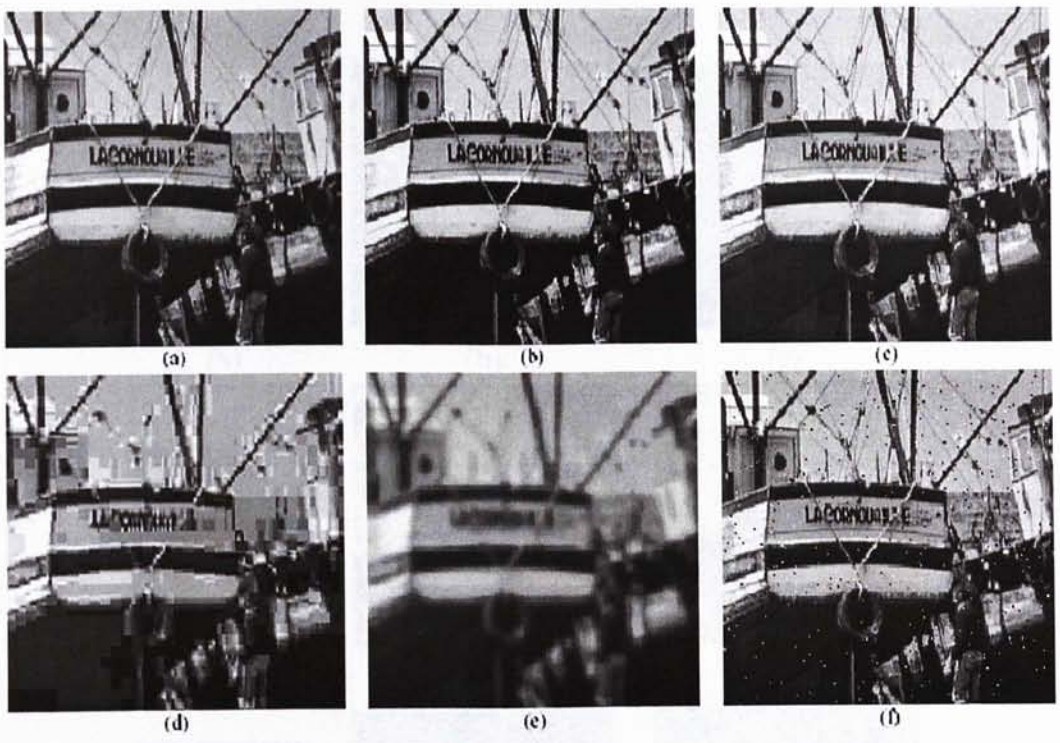


Figure 8.1: Images from [37]. Comparison of "Boat' images with different types of distortions, all with PSNR = 24.9 dB. (a) Original image; (b) Contrast stretched image, MSSIM = 0.9168; (c) Mean-shifted image, MSSIM = 0.9900; (d) JPEG compressed image, MSSIM = 0.6949; (e) Blurred image, MSSIM = 0.7052; (f) Salt-pepper impulsive noise contaminated image, MSSIM = 0.7748.

$\tau_m = 9$ and $\tau_d = 7$, as we found that this configuration maintains a balance between the query efficiency and the complexity of tree.

We tested our method with 20 typical cartoon images. The control images are the original non-contaminated images. Our method obtains an average PSNR improvement of 2.07dB and an average MSSIM improvement of 0.019. The average time of synthesizing each image is 0.86s.

Figure 8.2 visually compares three input contaminated images with the corresponding synthesized images. Parts of the images are blown up for comparison. The images in (a), (d) and (g) of Figure 8.2 are in the resolution of 200×232, 200×256, and 200×159, respectively. According to Table 8.1, our method improves the image quality in terms of both PSNR and MSSIM.

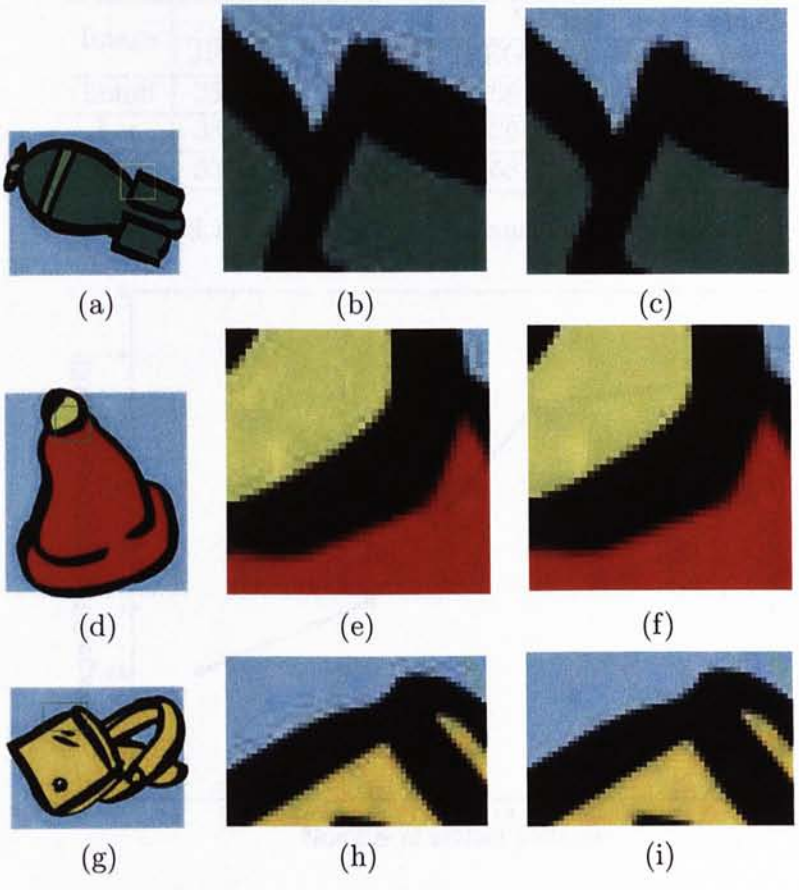


Figure 8.2: Examples of artifact removal: bomb, hat, and bag. (a), (d) and (g) are the contaminated images. (b), (e) and (h) show the blow-up of the boxes in the corresponding contaminated images. (c), (f) and (i) blow up the corresponding parts in the synthesized images.

8.3 How Size of Training Set Affects the Performance

It is nature to expect that the performance is directly affected by the size of the training set. To observe the effect of the size of training set, we conduct another experiment. During this experiment, we enlarge the size of training set gradually, from 50,000 to 200,000 (the number of distinct artifact patterns). We then measure the average PSNR improvement between contaminated input images and the synthesized images. The result is shown in Figure 8.3. As the

Image	PSNR (dB)		MSSIM		Time
	JPEG	Proposed	JPEG	Proposed	(s)
bomb	35.76	38.08	0.968	0.992	0.87
hat	38.19	40.73	0.976	0.993	0.87
bag	33.74	36.22	0.958	0.991	0.81

Table 8.1: Statistical performance of our method.

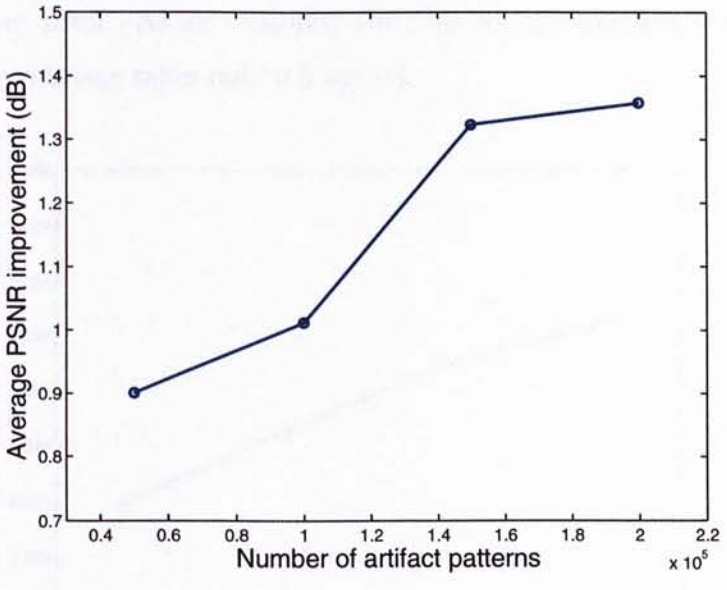


Figure 8.3: The average PSNR improvement as the number of artifact patterns increases from 50,000 to 200,000.

number of artifact patterns increases, the PSNR improvement increases.

The size of training set affects not only PSNR improvement but also the synthesis time. Obviously, a larger training set produces a larger TSVQ tree. The time of synthesis depends on the size of TSVQ tree, the resolution of input image, and the image content. Copying of non-edge block takes very little amount of time. The major computation is spent on the synthesis of pixels in the edge blocks.

To measure the time of synthesis, we compute the average query time for a pixel, which equals to the total time of synthesizing edge pixels divided by the total number of edge pixels. We plot a graph of this average query time against

the number of distinct artifact vectors in TSVQ tree in Figure 8.4. Obviously, the time slightly increases as the number of artifact patterns increases. It confirms with the $O(\log_{\tau_m} n)$ running time property of TSVQ. Since only a portion of the image requires query, the total time for artifact removal is usually smaller than the multiplication of the total number of pixel and the average query time per pixel. As an example, the time for synthesizing a 256×256 artifact-reduced image takes only 0.8 second.

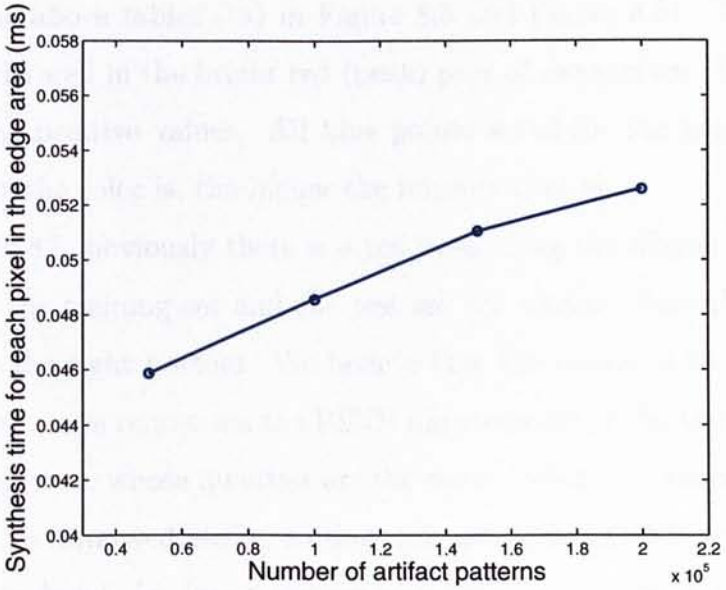


Figure 8.4: The average query time per pixel. In general, the average query time slowly increases as the number of artifact patterns in TSVQ increases.

8.4 How Bit Rates Affect the Performance

As mentioned in Section 4.2, different bit rates will produce different ringing artifact patterns. If the bit rate of our training set does not match with that of the input image, our method still works well. In this experiment, we vary bit rates for both our training set and the test set to verify the feasibility of our learning-by-example method. The objective is to find the well working

area of our method. In this experiment, our training set and the test set are compressed to different bit rates ranging from 0.480 bpp to 1.224 bpp. We employ our training set (Figure 4.2). The test images are 39 typical cartoon images.

In Figure 8.5 and Figure 8.6, the average PSNR and MSSIM improvements are shown, respectively. The feasibility surface is built by 10×10 key points. The two horizontal axes are the bit rates of the training set and the test set, respectively. The vertical axis is the average improvement. These data are shown in the above tables ((a) in Figure 8.5 and Figure 8.6). The proposed method works well in the bright red (peak) part of the surface. All red points represent the positive values. All blue points stand for the negative values. The brighter the color is, the higher the improvement is.

In Figure 8.5, obviously there is a red peak along the diagonal, where the bit rates of the training set and the test set are similar. Also the left top is higher than the right bottom. We believe that the reason is as follows. The top (farthest) curve represents the PSNR improvement of the test images with the lowest bit rate, whose qualities are the worst. With this lowest base, these images can be improved easily, so they can get higher PSNR improvements. On the other hand, for input test images with the highest bit rate, whose qualities are already very nice, it is difficult to improve them further. By the training sets with similar bit rats, these high quality images can be improved a little. And by the training sets with widely different bit rates, image qualities are even degraded. From PSNR improvement table ((a) in Figure 8.5), we can find some details. In fact the red peak shifts left a little. That means, a certain training set works best on the test set with a bit rate, which is a little smaller than the bit rate of this training set. We believe that images with lower bit rates are relatively easier to improve, can be considered as an answer.

In Figure 8.6, the average MSSIM improvement is given. The main difference between MSSIM and PSNR improvement is that, the diagonal peak

Bit rate of test image (bpp)	Bit rate of training set (bpp)	
	with the most PSNR improvement	with the most MSSIM improvement
0.480	0.576	0.533
0.533	0.614	0.614
0.576	0.614	0.614
0.614	0.666	0.614
0.666	0.735	0.666
0.735	0.735	0.735
0.863	0.948	0.863
0.948	1.038	0.863
1.038	1.224	0.948
1.224	1.224	1.224

Table 8.2: The suggested training set for the test images with different bit rates.

descends when the bit rate grows. We believe that it is because of the definitions of MSSIM and PSNR. The definition of MSSIM (Equation 8.2) determines that MSSIM converges to 1 when the image quality is improved. Hence, when the bit rate of the test set is large, even synthesized by the training set with the same bit rate, the MSSIM improvement is limited. The definition of PSNR (Equation 8.1) is different, because the PSNR can go to infinity when the image quality is improved. Another difference is that most MSSIM improvements are positive, while almost 1/3 of PSNR improvements are negative, which means that the proposed method can usually visually improve the quality of the contaminated image, even though the PSNR descends.

From the above statistic data, we can suggested a specified training set with a certain bit rate to apply to a given test image with a bit rate. Table 8.2 gives the suggested training sets.

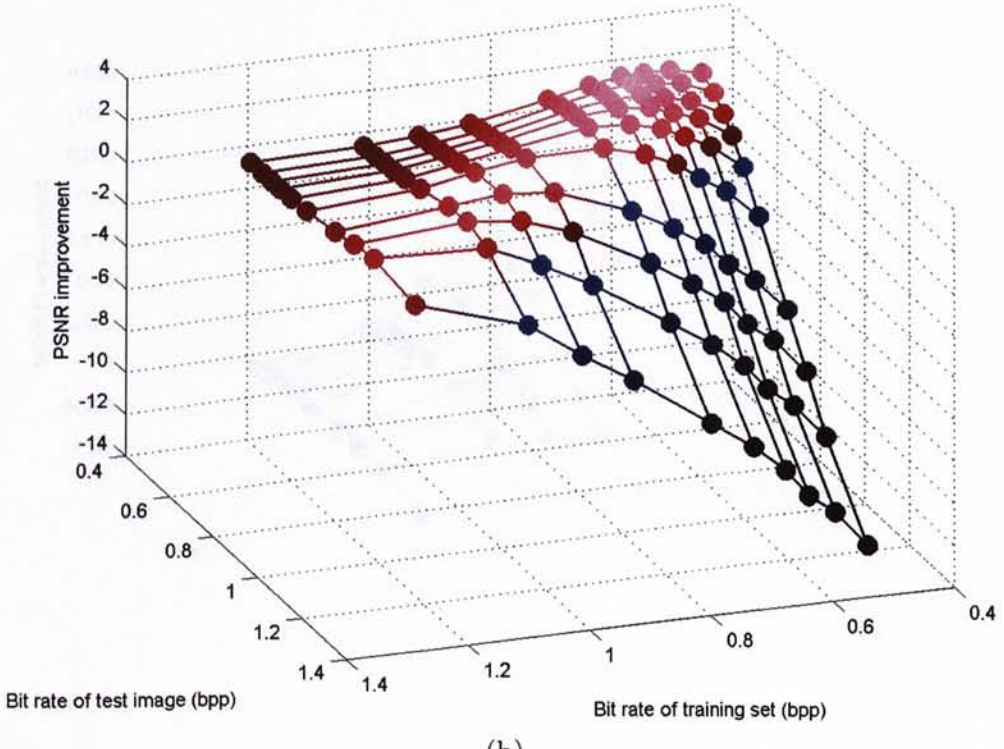
8.5 Comparisons

In order to evaluate the performance of the proposed method, we compare it with the method proposed in [33], which is one the most famous POCS

methods. This method applies a low-pass filter to the contaminated image, and constrains the change of each pixel by the given quantization table. Details are described in Section 2.3.2. The test set is compressed to different bit rates. For each bit rate, we take an average of improvements. Figure 8.7 shows the comparison of PSNR improvement. By PSNR assessment, the method [33] even degrade the image quality high-contrast images, while our method can improve the quality very well. The MSSIM comparison results are shown in Figure 8.8. The proposed method achieves much better MSSIM improvements, especially at lower bit rates.

PSNR improvement (dB)		Bit rate of training set (bpp)									
		0.480	0.533	0.576	0.614	0.666	0.735	0.863	0.948	1.038	1.224
Bit rate of test set (bpp)	0.480	1.755	2.069	2.083	2.052	1.770	1.345	0.636	0.364	0.158	0.039
	0.533	1.320	1.952	2.062	2.178	2.009	1.593	0.798	0.476	0.213	0.057
	0.576	0.709	1.507	1.809	2.050	2.014	1.729	0.942	0.592	0.268	0.073
	0.614	0.028	1.027	1.441	1.822	1.952	1.814	1.077	0.691	0.336	0.095
	0.666	-0.976	0.183	0.805	1.322	1.701	1.791	1.245	0.856	0.444	0.135
	0.735	-2.642	-1.278	-0.499	0.280	0.942	1.470	1.455	1.104	0.645	0.221
	0.863	-5.811	-4.225	-3.347	-2.259	-1.325	-0.266	1.077	1.313	1.062	0.482
	0.948	-7.922	-6.263	-5.348	-4.150	-3.124	-1.906	0.017	0.849	1.156	0.718
	1.038	-10.175	-8.484	-7.557	-6.309	-5.145	-3.860	-1.614	-0.361	0.757	0.932
	1.224	-13.489	-11.735	-10.817	-9.473	-8.194	-6.844	-4.346	-2.869	-1.075	0.566

(a)

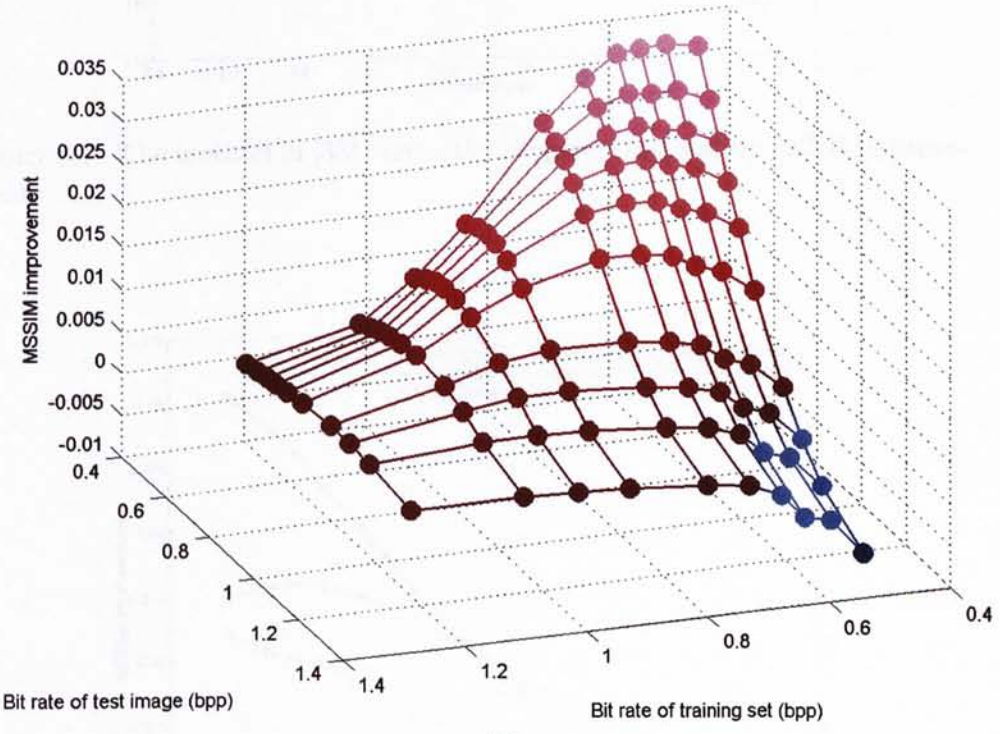


(b)

Figure 8.5: We vary bit rates for both our training set and the test set. (a) is the table of the average PSNR improvement. (b) is the surface plotted by data in (a). The redder (peak) part of this surface is the well working area of the proposed method. That means, training set with a certain bit rate can work well on the contaminated images with corresponding bit rate.

MSSIM improvement		Bit rate of training set (bpp)									
		0.480	0.533	0.576	0.614	0.666	0.735	0.863	0.948	1.038	1.224
Bit rate of test set (bpp)	0.480	0.0328	0.0335	0.0333	0.0330	0.0305	0.0258	0.0148	0.0092	0.0045	0.0013
	0.533	0.0277	0.0292	0.0292	0.0296	0.0283	0.0248	0.0157	0.0103	0.0053	0.0017
	0.576	0.0236	0.0254	0.0260	0.0267	0.0260	0.0237	0.0160	0.0111	0.0060	0.0020
	0.614	0.0198	0.0220	0.0226	0.0235	0.0233	0.0218	0.0157	0.0112	0.0063	0.0022
	0.666	0.0155	0.0178	0.0186	0.0198	0.0201	0.0193	0.0149	0.0112	0.0067	0.0025
	0.735	0.0097	0.0125	0.0135	0.0150	0.0156	0.0157	0.0134	0.0107	0.0070	0.0028
	0.863	0.0014	0.0045	0.0056	0.0076	0.0084	0.0090	0.0092	0.0084	0.0065	0.0033
	0.948	-0.0028	0.0008	0.0018	0.0040	0.0050	0.0057	0.0063	0.0063	0.0054	0.0033
	1.038	-0.0061	-0.0023	-0.0013	0.0012	0.0024	0.0031	0.0038	0.0040	0.0040	0.0030
	1.224	-0.0097	-0.0053	-0.0046	-0.0016	0.0001	0.0007	0.0014	0.0017	0.0020	0.0020

(a)



(b)

Figure 8.6: We vary bit rates for both our training set and the test set. (a) is the table of the average MSSIM improvement. (b) is the surface plotted by data in (a). The redder (peak) part of this surface is the well working area of the proposed method.

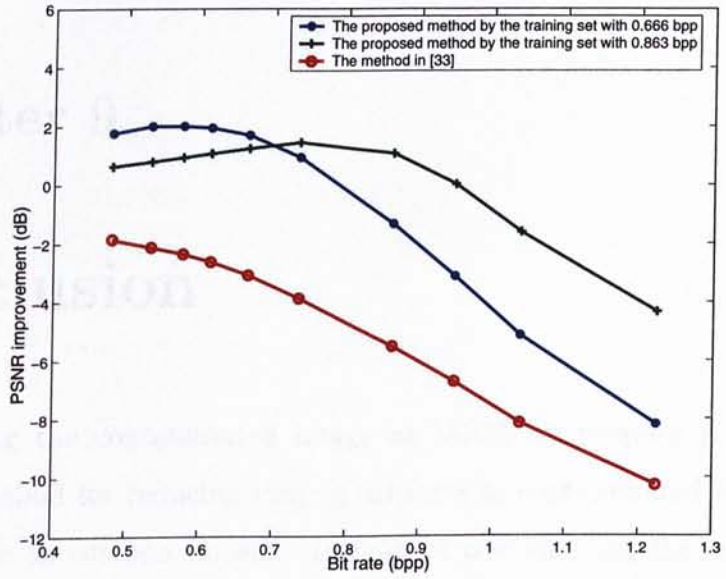


Figure 8.7: The method in [33] versus the proposed method by PSNR improvement.

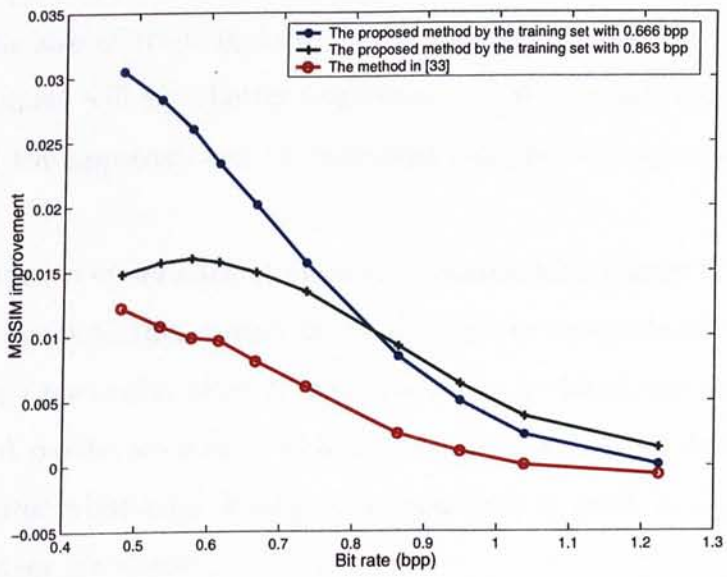


Figure 8.8: The method in [33] versus the proposed method by MSSIM improvement.

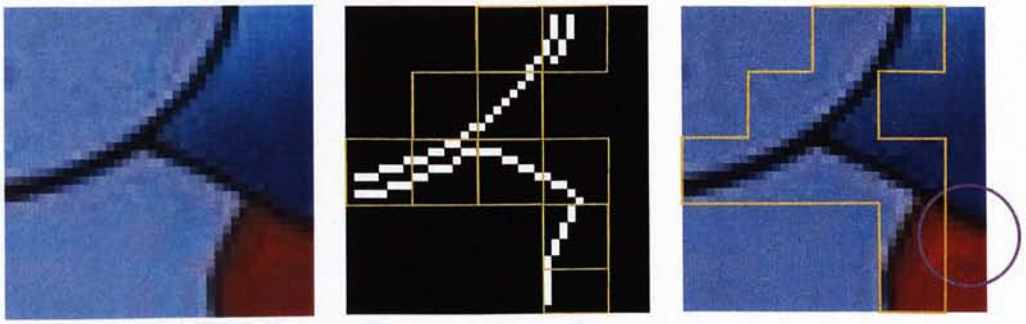
Chapter 9

Conclusion

By modeling the contaminated image as MRF, we propose a learning-by-example method for reducing ringing artifacts in contaminated high-contrast images, such as cartoon images. Instead of post-filtering the contaminated image, we synthesize an artifact-reduced image. Using our method, the image quality of all test cases is substantially improved, not only subjectively but also objectively. Our method can effectively handle ringing artifacts which cannot be effectively solved by previous methods. Like other learning-by-example methods, the size of training data affects the performance. In general, more training samples will give better improvement. We should find out how the accuracy of the approach can be improved with better edge detection algorithms.

One limitation of our current method is illustrated in Figure 9.1. In this example, the edge detection cannot detect the edge in the circle of Figure 9.1(c). Therefore that particular block is tagged as non-edge block and no synthesis is performed. A partial solution is to make use of multiple edge detectors. However, no matter what kind of edge detection filter is used, it is still possible that some edges are missed.

Another limitation of the proposed method is that artifacts caused by



(a) (b) (c)

Figure 9.1: Some edges (in the circle of (c)) may not be identified and hence cannot be synthesized.

chrominance components cannot be reduced. Because we only apply the proposed method on the luminance component, while the chrominance components are retained.

Currently, in the proposed method, training sets are classified by different bit rates. An obvious direction for future work is to mix training sets with different bit rates, in proper proportion. Such composite training set can improve the quality of images with different bit rates. From Table 8.2, we can see that the training set with 0.614 bpp works well on the input image with lower bit rates, and for the input image with higher bit rates, the training set with 0.735 bpp can give a better result. We may mix these two training sets to get an optimal training set, which works well in most cases. Also we should find out if the training set can be reduced for increasing efficiency of the approach without sacrificing its accuracy.

Another direction is to explore a patch-based version of the proposed method. Our method synthesizes images in a pixel-by-pixel manner. We believe the speed can be substantially increased if a patch-based approach is used. However whether the quality of synthesized images is still preserved requires further investigation.

Appendix A

Color Transformation

Images and videos are usually displayed in the RGB color space. While compressed and stored, the original data are typically in some type of a luminance-chrominance color space, such as YUV. Y is the luminance component and U and V are the chrominance components.

Given RGB inputs (R, G and B in $[0, 1]$),

$$\begin{pmatrix} Y \\ U \\ V \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.148 & -0.289 & 0.437 \\ 0.615 & -0.515 & -0.100 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix} .$$

Given YUV inputs (Y in $[0, 1]$ and U,V in $[-0.5, 0.5]$),

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1.140 \\ 1 & -0.395 & -0.581 \\ 1 & 2.032 & 0 \end{pmatrix} \begin{pmatrix} Y \\ U \\ V \end{pmatrix} .$$

Appendix B

Image Quality

Almost all JPEG-related applications allow an end user to "tune" the quality of a JPEG encoder using a parameter sometimes called "image quality". The "image quality" ranges from 1 to 100 typically. The value of 1 produces the smallest, worst quality image and the value of 100 produces the largest, best quality image. The optimal "image quality" factor depends on the image content and is therefore different for every image. The art of JPEG compression is to find the lowest factor that produces an image which is visibly acceptable, and preferably as close to the original image as possible.

B.1 Image Quality vs. Quantization Table

In JPEG, typical quantization tables are given, and the user input factor, "image quality", scales these quantization tables. Typical luminance quantization

table is:

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

and typical chrominance quantization table is:

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

These typical tables are scaled by "image quality". The process is as follows:

Clamp *quality* to $[1, 100]$;

Linearize *quality*:

$$\textbf{if(quality} < \textbf{50)}$$
$$\textbf{quality} = \textbf{5000/quality};$$
$$\textbf{else}$$
$$\textbf{quality} = \textbf{200} - \textbf{quality} * \textbf{2};$$

Scale the typical table by *quality*:

$$\textbf{myQ(i,j)} = (\textbf{typQ(i,j)} * \textbf{quality} + \textbf{50})/\textbf{100}\ .$$

When "image quality" equals to 1, elements of the result table are all 255; when "image quality" equals to 100, elements of the result table are all 1; and when "image quality" equals to 50, the result table is same as the typical one.

B.2 Image Quality vs. Bit Rate

We employ a typical image software ACDSee to do BDCT compression. In image compression study, the bit rate ("bits per pixel") is the number of bits of information stored per pixel of an image. It is more understandable and technological. So we find the relation between "image quality" and bit rate (bpp). Since the original image is a graylevel image, its bit rate is 8 bpp. We apply BDCT compression to the training original images, by "image qualities" of $25, 35, \ldots, 95$. Then we calculate bit rates for all compressed images, and get an average. The bit rates after compression range from 0.480 bpp to 1.224 bpp. The relation between "image quality" and bit rate is shown in Figure B.1.

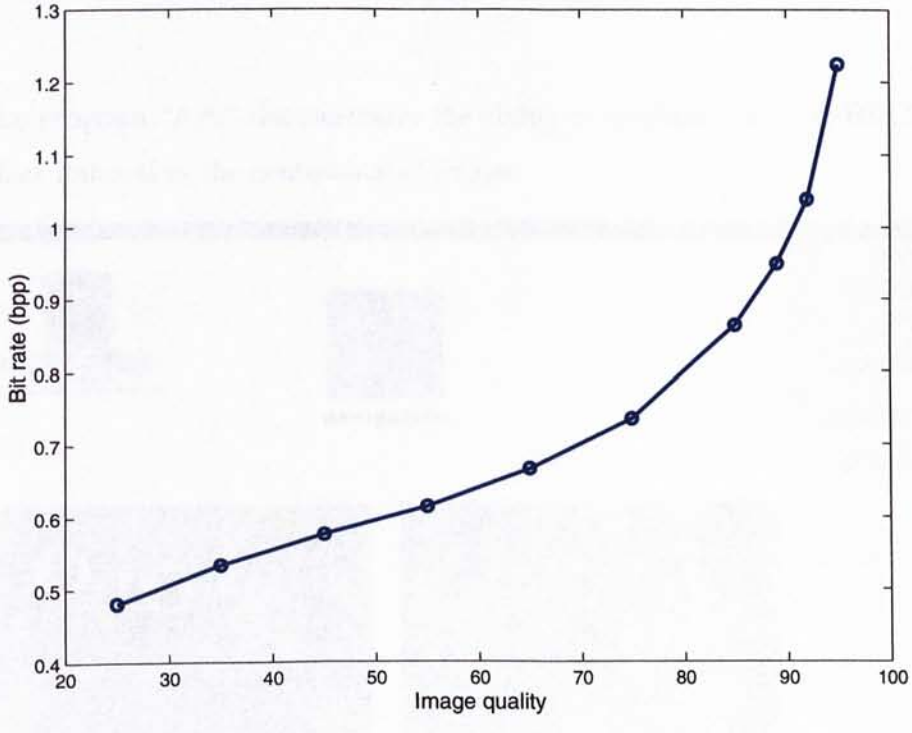


Figure B.1: The relation between "image quality" and bit rate.

Appendix C

Arti User's Manual

The program "Arti" demonstrates the ability to perform real-time BDCT artifact removal in the contaminated image.

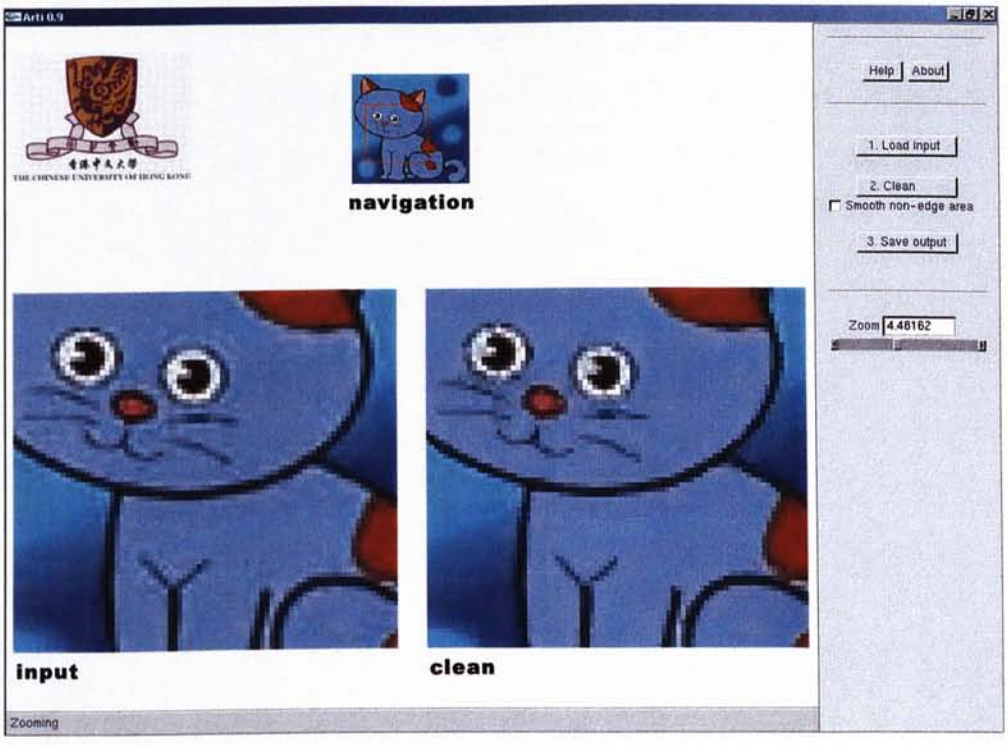


Figure C.1: User interface of Arti.

Figure C.1 shows the user interface of "Arti". The input (contaminated) image and the clean (synthesized) image are displayed side by side. Navigation

image shows the current position of the displayed part. By pressing **PageUp** ('a'), you can zoom in the displayed part, and **PageDown** ('d') for zooming out. **Left**, **Up**, **Down** and **Right** ('z', 's', 'x' and 'c') are used for translation.

Synthesis procedure is as follows:

(a) Click the button of "Load" to load the contaminated image, and then it is initialized and displayed in the left window;

(b) Click the button of "Clean" and after a few seconds, the synthesized image is displayed in the right window;

(d) Click the button of "Save". The synthesized image can be saved by BMP format.

Bibliography

[1] N. Ahmed, T. Natatajan, and K. R. Rao. Discrete cosine transform. *IEEE Trans. Computers*, C-23:90–93, January 1974.

[2] V. Bhaskaran and K. Konstantinides. *Image and Video Compression Standards.* Kluwer Academic Publishers, Boston, 1997.

[3] Y.-H. Chan, S.-W. Hong, and W.-C. Siu. A practical postprocessing technique for real-time block-based coding system. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(1):4–8, February 1998.

[4] T. Chen, H. R. Wu, and B. Qiu. Adaptive postfiltering of transform coefficients for the reduction of blocking artifacts. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(5):594–602, May 2001.

[5] H. Choi and T. Kim. Blocking-artifact reduction in block-coded images using wavelet-based subband decomposition. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(5):801–805, August 2000.

[6] P. L. Combettes. The foundation of set theoretic estimation. *Proc. IEEE*, 81:182–208, February 1993.

[7] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *SIGGRAPH 2001 Conference Proceedings*, pages 341–346, 2001.

[8] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *IEEE International Conference on Computer Vision*, pages 1033–1038, Corfu, Greece, September 1999.

[9] W. T. Freeman, T. R. Jones, and E. C. Pasztor. Example-based super-resolution. *IEEE Computer Graphics and Applications*, 22(2):56–65, Mar./Apr. 2002.

[10] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression.* Kluwer Academic Publishers, Boston, 1992.

[11] R. A. Gopinath, H. Guo, M. Lang, and J. E. Odegard. Wavelet-based post-processing of low bit rate transform coded. In *Proc. 1994 IEEE Int. Conf. Image Processing*, pages 913–917, 1994.

[12] J. A. Hartigan. *Clustering algorithms.* Wiley, New York, 1975.

[13] A. Hertzmann, C. Jacobs, N. Oliver, B. Curless, and D. Salesin. Image analogis. In *SIGGRAPH 2001 Conference Proceedings*, pages 327–340, 2001.

[14] Y.-F. Hsu and Y.-C. Chen. A new adaptive separable median filter for removing blocking effects. *IEEE Trans. Consumer Electron.*, 39(3):510–513, June 1993.

[15] T. C. Hsung, D. P. Lun, and W. Siu. A deblocking technique for block-transform compressed image using wavelet transform modulus maxima. *IEEE Transactions on Image Processing*, 7(10):1488–1496, October 1998.

[16] T. Jarske, P. Haavisto, and I. Defe'e. Post-filtering methods for reducing blocking effects from coded images. *IEEE Trans. Consumer Electron.*, 40(3):521–526, June 1994.

[17] Y. Jeong, I. Kim, and H. Kang. A practical projection-based postprocessing of block-coded images with fast convergence rate. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(4):617–623, June 2000.

[18] N. C. Kim, I. H. Jang, D. H. Kim, and W. H. Hong. Reduction of blocking artifact in block-coded images using wavelet transform. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(3):253–257, June 1998.

[19] C. J. Kuo and R. J. Hsieh. Adaptive postprocessor for block encoded images. *IEEE Transactions on Circuits and Systems for Video Technology*, 5(4):298–304, August 1995.

[20] Y. L. Lee, H. C. Kim, and H. W. Park. Blocking effect reduction of JPEG images by signal adaptive filtering. *IEEE Transactions on Image Processing*, 7(2):229–234, February 1998.

[21] S. Z. Li. *Markov Random Field Modeling in Image Analysis.* Springer-Verlag, Tokyo, 2001.

[22] C. Liu, C. Wang, and J. Lin. A new postprocessing method for the block-based DCT coding based on the convex-projection theory. *IEEE Trans. Consumer Electron.*, 44(3):1054 –10613, August 1998.

[23] J. Luo, C. W. Chen, K. J. Parker, and T. S. Huang. Artifact reduction in low bit rate DCT-based image compression. *IEEE Transactions on Image Processing*, 5(9):1363–1368, September 1996.

[24] M. Mese and P. P. Vaidyanathan. Look-up table (LUT) method for inverse halftoning. *IEEE Transactions on Image Processing*, 10(10):1566–1578, October 2001.

[25] S. Minami and A. Zakhor. An optimization approach for removing blocking effects in transform coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 5(2):74–82, April 1995.

[26] H. Paek, R.-C. Kim, and S.-U. Lee. On the pocs-based postprocessing techinque to reduce the blocking artifacts in transform coded images. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(3):358–367, June 1998.

[27] W. B. Pennebaker and J. L. Mitchell. *JPEG Still Image Data Compression Standard*. Van Nostrand, New York, 1993.

[28] E. Praun, A. Finkelstein, and H. Hoppe. Lapped textures. In *SIGGRAPH 2000 Conference Proceedings*, pages 465–470, 2000.

[29] G. Qiu. MLP for adaptive postprocessing block-coded images. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(8):1450–1454, December 2000.

[30] B. Ramamurthi and A. Gersho. Nonlinear space-variant post-processing of block coded images. *IEEE Trans. Acoust., Speech, Signal Processing*, 34(5):1258–1268, October 1986.

[31] K. R. Rao and J. J. Hwang. *Techniques and Standards for Image, Video, and Audio Coding*. Prentice Hall, NJ, 1996.

[32] H. C. Reeve and J. S. Lim. Reduction of blocking effect in image coding. *Opt. Eng.*, 23(1):34–37, Jan./Feb. 1984.

[33] R. Rosenholtz and A. Zakhor. Iterative procedures for reduction of blocking effects in transform image coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 2(1):91–95, March 1992.

[34] M. Shen and C.-C. J. Kuo. Artifact reduction in low bit rate wavelet coding with robust nonlinear filtering. In *1998 IEEE Second Workshop on Multimedia Signal Processing*, pages 480–485, 1998.

[35] G. Turk. Texture synthesis on surfaces. In *SIGGRAPH 2001 Conference Proceedings*, pages 347–354, 2001.

[36] B. Wandell. *Foundations of Vision.* Sinauer Associates Inc., 1995.

[37] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error measurement to structural similarity. *IEEE Transactions on Image Processing*, 13(1), January 2004.

[38] C. Weerasinghe, A. W. Liew, and H. Yan. Artifact reduction in compressed images based on region homogeneity constraints using the projection onto convex sets algorithm. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(10):891–897, October 2002.

[39] L. Y. Wei and M. Levoy. Fast texture synthesis using tree-structured vector quantization. In *SIGGRAPH 2000 Conference Proceedings*, pages 479–488, 2000.

[40] Z. Xiong, M. T. Orchard, and Y.-Q. Zhang. A deblocking algorithm for JPEG compressed images using overcomplete wavelet representations. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(2):433–437, April 1997.

[41] S. Yang, Y.-H. Hu, T. Q. Nguyen, and D. L. Tull. Maximum-likelihood parameter estimation for image ringing-artifact removal. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(8):963–973, August 2001.

[42] Y. Yang and N. P. Galatsanos. Removal of compression artifacts using projections onto convex sets and line process modeling. *IEEE Transactions on Image Processing*, 6(10):1345–1357, October 1997.

[43] Y. Yang, N. P. Galatsanos, and A. K. Katsaggelos. Projection-based spatially adaptive reconstruction of block-transform compressed images. *IEEE Transactions on Image Processing*, 4(7):896–908, July 1995.