USC-SIPI REPORT #395

Texture Processing for Image/ Video Coding and Super-Resolution Applications

by

Byung Tae Oh

August 2009

Signal and Image Processing Institute
UNIVERSITY OF SOUTHERN CALIFORNIA
Viterbi School of Engineering
Department of Electrical Engineering-Systems
3740 McClintock Avenue, Suite 400
Los Angeles, CA 90089-2564 U.S.A.

TEXTURE PROCESSING FOR IMAGE/VIDEO CODING AND SUPER-RESOLUTION APPLICATIONS

by

Byung Tae Oh

A Dissertation Presented to the
FACULTY OF THE GRADUATE SCHOOL
UNIVERSITY OF SOUTHERN CALIFORNIA
In Partial Fulfillment of the
Requirements for the Degree
DOCTOR OF PHILOSOPHY
(Electrical Engineering)

August 2009

Dedication

This dissertation is dedicated to my parents, wife and daughter for their endless love.

Table of Contents

List Of Tables

List Of Figures

Abstract

Textured image/video processing for coding and super-resolution is investigated in this thesis to improve coding efficiency and prediction accuracy. The research consists of three main parts.

First, a synthesis-based texture coding technique that uses low-quality video as the side information to control the output texture for video coding is proposed. As compared with the current pure synthesis algorithm, the proposed algorithm is generic, in the sense that the behavior and quality of the output texture can be adjusted by the amount of the side information and determined by the user. We develop an area-adaptive side information assignment technique to improve coding efficiency by given bit-budget. Additionally, we also provide the texture decomposition algorithm to maximize the synthesis performance by decomposing the non-synthesizable illumination component from the input video. Simulations demonstrate the performance of the proposed technique.

Second, the addition of a new component to the traditional synthesis-based texture video coding algorithm is investigated in this chapter. That is, we add the side information in form of low-quality video to enhance the texture video synthesis performance with reducing the unpleasant mismatch between analyzed and synthesized regions. As compared with the conventional synthesis algorithm, our algorithm is more flexible since

the behavior and quality of the output texture can be adjusted by the amount of the side information, which is determined by the user. To this end, we develop an area-adaptive side information selection scheme that chooses the proper amount of the side information for a given bit budget. Furthermore, we propose a texture decomposition scheme that extracts the non-synthesizable illumination component from the source video for separate coding so as to maximize the synthesis functionality. The superior performance of the proposed texture video synthesis technique is demonstrated by several coding examples.

Third, a texture interpolation technique based on the locally piecewise auto-regressive (PAR) model and the non-local (NL) training procedure is investigated. The proposed PAR/NL scheme selects model parameters adaptively based on local image properties with an objective to improve the interpolation performance of non-adaptive models, *e.g.,* the bicubic algorithm. To determine model parameters for stochastic texture, we use the non-local (NL) learning algorithm to update and refine these local model parameters under the assumption that the PAR model parameters are self-regular. As compared to previous interpolation algorithms, the proposed PAR/NL scheme boosts texture details, and eliminates blurring artifacts perceptually. Experimental results are given to demonstrate the performance of the proposed technique.

Chapter 1

Introduction

1.1 Significance of the Research

Many real world objects have textured surface appearance such as terrain, wools, fur, skin, cloud, grass, and so on. Texture classification and segmentation is one of the oldest research topics in image/video processing. Although texture has been carefully studied for its specific property over four decades, it is still difficult to model and characterize mathematically. Loosely speaking, texture has spatially homogeneous, quasi-stationary statistical properties with repeated patterns.

The performance of texture analysis algorithms is directly dependent on the extracted salient feature sets. From early 60's, people tried the pixel-domain or the transform-domain approaches with its statistical information [59, 61]. Among them, the application of a set of FIR filters [36], wavelet filters [15], Gabor filters [32] offers better performance. In addition, texture decomposition [2], modeling [58], coding [72] have been widely investigated. More recently, texture synthesis becomes popular [34, 35]. Some of these results

have been applied other research problems, such as inpainting [4], compression [46] or rendering.

Adaptation of texture synthesis technique for textured video coding can be assumed as a content-adaptive algorithm. In current video coding standards, the encoder does not distinguish different video contents and adopts the same coding techniques for all input contents, for example, motion-compensated prediction (MCP) and quantization in the block transform domain, where MCP and the block transform are used to remove temporal and spatial redundancies, respectively. However, it may be possible to develop special coding techniques for specific video contents by synthesis. There are two reasons that synthesis can replace conventional encoding/decoding for texture. First, texture is often perceived differently than other objects, and does not require reproduction with pixel accuracy. Second, the repeated pattern enables us to generate visually-similar texture by exploiting its statistical and stationary properties. Previous work on synthesis-based texture coding will be reviewed in Sec. 1.2.

However, there are several challenging research problems in synthesis-based video coding as described below. Some of them will be addressed in Chapter 3.

- Synthesized texture should be perceptually similar to the target one. In order for synthesized texture to replace conventional texture encoding/decoding, perceptual similarity of synthesized texture is critical. In fact, this requirement is valid not only for synthesis-based coding but also for all texture synthesis algorithms.

- Synthesized texture should be well-matched with its decoded neighborhood. Being different from the texture synthesis problem in computer graphics, the video output

contains both decoded video background and synthesized texture. In this case, it would be important to hide boundary artifacts between decoded video blocks and synthesized texture blocks. In addition, the general structure and motion of synthesized texture should be consistent its neighborhood. Otherwise, undesirable artifacts will be noticeable.

- The synthesis algorithm should only be used for synthesizable texture. Since most video content is not synthesizable, we need to classify or decompose them for the proposed synthesis algorithm to be applicable.

As an application of synthesis-based texture coding, we propose a novel framework to increase coding efficiency by handling film grain noise of high definition video in Chapter 4. Film grain noise can be viewed as a type of texture due to its perceptual unimportance, large high-frequency energy and statistical property. Thus, synthesis-based texture coding can be applied to film grain noise compression. Some research problems for film grain noise analysis and synthesis algorithm are described below. They will be addressed in detail in Chapter 4.

- Film grain noise should be well-suppressed to maximize the coding gain. Since the ultimate objective for film grain denoising is to improve coding efficiency, film grain noise should be extracted as much as possible in the encoder.

- The denoising filter used in film grain extraction should not distort the original data. Note that the denoising filter tends to blur original video contents by its low-pass filter property, which should be avoided.

- Since film grain noise enhances the naturalness of video contents, we need a post-processing step to re-synthesize the grain-like noise at the decoder. Extracted film grain noise should be analyzed and parameterized for the rendering purpose. Besides, synthesized grain noise should be perceptually similar to the original one.

- Finding objective quality measurement for synthesized film grain noise is important. Objective quality measurement is needed to evaluate the performance of the algorithm in addition to subjective quality measurement. For this reason, salient features of film grain noise have to be studied.

Finally, we propose a method to increase the resolution of random texture by properly interpolating textured image. Since random texture is spatially complex and sophisticated, the traditional edge-oriented interpolation schemes do not work well. One of the image restoration problem, *i.e.* texture super-resolution, has several challenges as described below. This subject will be treated in detail in Chapter 5.

- The resolution-increased texture should properly represent its detail. Actually, the simple frequency-domain extension with zero-padding would not produce satisfactory results.

- The interpolation filter should be large enough to estimate the complex behavior of random texture. Although a textured pattern may only have short-term correlation, the prediction in the 2D image domain often involves a large number of pixels and the corresponding filter coefficients are not easy to estimate.

1.2 Review of Previous Work

Texture synthesis is an active research topic in many research fields. Besides of synthesis-based texture coding approach, texture synthesis technologies have been broadly applied to several research areas, such as image/video texture rendering [88], completion [70], and inpainting [4]. Broadly speaking, there are two major approaches for texture synthesis. The first one is based on the parametric approach [54, 57, 58, 87], where an image sequence is modeled by a number of parameters such as the histogram or correlation of pixel values. Given a sufficient number of model parameters, it is then possible to recreate the "look and feel" of any texture that meets the parameterized constraints. The second approach is based on a non-parametric approach [21, 22, 34, 35, 39], where synthesized texture is derived from an example texture as the seed that is known *a priori*. The texture synthesis process creates additional texture data by inspecting the seed texture and copying intensity values in the seed to the new texture region. Most texture synthesis work considers the image domain, with only a few are applied to the video space.

There have been efforts in developing the texture synthesis approach to improve the coding efficiency. In [46], the texture region of the target frame is first analyzed by TA, and then it is reconstructed by TS using frame-to-frame displacement and image warping techniques. This method is however not effective for non-rigid texture objects, and the frame-by-frame synthesis method can yield temporal inconsistence. Similar texture synthesis technique using frame-to-frame mapping was proposed in [6], but it was more focused on texture analysis and segmentation algorithms. A cube-based texture growing method was proposed in [47, 48], which can be viewed as an extension of [35] from

image to video. Temporal consistency is guaranteed in [48] due to cube-based synthesis. However, the global texture structure can be easily broken by texture growing with the raster-scanning order. Furthermore, the output texture tends to have repeated patterns by the fixed-size cube-based growing procedure. An algorithm with an alternative approach was proposed in [20], in which it replaces the input texture with a perceptually similar and well-encodable synthesized texture. In comparison to other methods, the approach only manipulates the input video at the encoder side, so that existing coding modules can be used without any modification. However, its constraint-based texture synthesis method based on [58] is generally not efficient for structured texture.

Image restoration including noise detection and removal has been one of active research topics in image processing during last several decades. The main objective of these algorithms is suppressing the noise as much as possible without distorting the original image, and various approaches have been proposed so far [9]. When extending the denoising problem from image to video, temporal correlation of noise should be considered. Ozkan *et al.* [53] applied temporal filtering to noise suppression yet preserving image edges at the same time. The integrated spatial and temporal filtering approach was examined in [18, 56]. Temporal filtering methods in these works were built upon block motion estimation. Boo *et al.* [5] applied the Karhunen-Loeve (KL) transform along the temporal direction to decorrelate dependency between successive frames and then used adaptive Wiener filtering to smooth frames. Most methods using temporal filtering work well for still or slow-motion video since temporal filtering can be done nicely with accurate motion information. However, large motion estimation errors occurring in fast motion video tend

to result in erroneous noise estimation. Furthermore, it demands a large memory buffer to implement temporal filtering.

Among hundreds of papers published on this topic, some of them target at film grain noise processing, *e.g.* [1, 5, 43, 84, 85]. It was assumed by Yan *et al.* [84, 85] and Al-Shaykh *et al.* [1] that film grain noise is proportional to a certain power, p, of the background signal intensity, which imposes strong limitation on the extraction and modeling of film grain noise. More recently, Moldovan *et al.* [43] proposed new algorithms using Bayesian model to detect film grain noise, where film grain noise is assumed to be of the beta distribution and spatially independent. Under this assumption, an image can be estimated by an iterative gradient descent method with a pre-determined model and proper parameters. This approach is claimed to be robust against different image contents. However, according to our observation, the distribution of film grain noise is close to the Gaussian distribution and it is not spatially independent.

In the denoising process, it is important to find out an edge region of the image, since most of denoising algorithm tends to blur the image, especially around the edge. There has been a large amount of work for edge detection. In particular, Canny [13] proposed an edge detection algorithm for noisy images. More recently, a multi-layer approach is used to reduce false detection due to noise. For example, Mallet *et al.* [41] used local maxima of overcomplete wavelet transform coefficients for edge detection. They proved that finding local maxima is equivalent to multi-scale Canny's edge detector. Instead of finding local maxima of wavelet coefficients, Xiong *at el.* [83] used the cross-scale correlation for edge detection by multiplying cross-scale wavelet coefficients under the assumption that the multi-scale noise intensity is related in a logarithmic ratio. However,

this method would not work well for film grain noise whose energy values deviate from the logarithmic variation assumption. As more specific application for the film grained image, Schallauer *et al.* [64] proposed new algorithms to detect homogeneous image region, in which the block-based 2D DFT was adopted to detect the directionality of edges and/or fine textures and, then, properties of film grain noise are extracted only from homogenous regions. Since film grain noise is isotropic while edges and fine textures have strong directionality, 2D DFT provides a good tool for their distinction. However, the decision made based on several points with four-different angles { 0°, 45°, 90°, 135° } and a couple of radial bands is not accurate enough to determine the directionality, since the spectrum of image edges tends to be across a wide range of frequency bands and sampled values at fixed points in the DFT domain may lead to false conclusion.

Two different methods were proposed for film grain synthesis. One is to use the film grain database for the sake of low complexity. The film grain pattern is first identified, and the decoder generates a larger size of film grain from a smaller size of film grain stock. However, the block-based copy-and-paste method might yield artificial boundary artifacts. Besides, the method is workable only when the film stock information is known a priori. The other is to use some models for blind film grain synthesis. Several methods have been proposed, *e.g.* high-order-statistics-based [84, 85], parametric-modeling-based [11, 12, 58, 63] or patch-based noise synthesis methods [21, 22, 39, 35]. Since film grain noise can be viewed as one type of random texture, the conventional texture synthesis method can also be adopted. However, there is one challenge. That is, since film grain noise has special properties as will be mentioned in Chapter 4, these criteria must be

considered and satisfied when synthesizing the film grain. So far, there has been no effort to generate the film grain noise based on the specific film grain properties.

As another class of image restoration problem, image interpolation can be done with two major approaches: the model-based approach and the learning-based approach. In the model-based approach, pixels are predicted using a pre-defined model. Some algorithms may change the model and/or model parameters adaptively according to local image contents. For example, Li and Orchard [38] proposed a piecewise auto-regressive (PAR) model that adjusts local model parameters using a covariance duality assumption. In the learning-based approach, the relationship between known and unknown data is extracted from a training set [25, 71]. The learning-based approach has more flexibility in handling a broader range of image contents, but the training-set size should be restricted due to limited memory space and computational complexity. Instead of relying on a pre-selected training-set, a learning-based algorithm with the self training-set was proposed by Buades *et al.* [8], which has received a lot attention due to its simplicity and superior performance. This scheme, called the non-local (NL) algorithm, is initially developed for image denoising. However, it is proven to be efficient for the image restoration problem as well, including deblurring, inpainting and interpolation [37].

1.3 Contributions of the Research

The main contributions of this research are summarized below.

- We propose a novel framework to control the output synthesized texture by sending additional side information, which is used to make the synthesized texture be

consistent with the behavior of its neighborhood texture. As a result, the overall output texture looks more natural. The side information can be obtained by the standard encoder at a lower bit-rate so that we do not need any additional module to generate the side information. Besides, the amount of the side information is adjustable at the encoder by varying the quantization parameter (QP).

- We develop an optimal method to select the side information to maximize the texture similarity and to minimize the texture synthesis distortion for a given bit budget. The side information can affect its spatial or temporal neighbor in the synthesis process, and its importance could vary in different local regions. Consequently, the amount of the side information should be determined by its local importance. We use the R-D based optimization to determine the local importance of the side information.

- We present a texture decomposition method to maximize the synthesis performance by removing the non-synthesizable yet easily-coded background illumination component from the input video. This allows the proposed algorithm to encode illumination-varying texture more effectively. As a result, the resultant algorithm becomes more flexible and robust.

For research on film grain noise modeling and applications, we develop a denoising algorithm for film grain noise extraction, and a novel rendering algorithm for film grain noise synthesis in Chapter 4. The following contributions are made along this line.

- The unique properties of film grain noise are studied and exploited fully. Film grain noise is generated by the physical process and it is more visible than white noise.

We analyze these properties and build an effective model for its extraction and synthesis.

- In the denoising process, a variational approach is adopted due its superior behavior in edge preservation. That is, we formulate a problem that minimizes the total variational (TV) of the underlying image. It results in better estimation of the noise behavior and helps extract film grain noise without distorting edges of the original image.

- We develop a causal AR model for noise synthesis. Since the AR model only needs a small number of parameters in texture synthesis, the number of additional bits required to represent the side information is negligible. Furthermore, the synthesis process at the decoder is fast by exploiting the causality of the AR model. We also propose a quantitative metric to evaluate the performance of the proposed texture synthesis method.

Finally, we develop a novel super-resolution algorithm for random texture in Chapter 5. The following contributions are made along this line.

- We propose a hybrid approach that consists of model-based and learning-based schemes to solve the super-resolution problem. Consequently, it can reduce visual artifacts caused by the learning-based algorithm while efficiently estimating the adaptive model parameters.

- We adopt the self-learning based algorithm for texture restoration. Since random texture is homogeneous itself, a self-learning training set can yield enough information.

- We propose to adopt the non-local information to break the 'curse of dimensionality' problem in parameter estimation. The order of a texture model should be large to capture its sophisticated behavior. However, it is often limited by the available local information in traditional image processing algorithms. Here, we use the non-local information with a proper similarity measure, which helps find out more accurate parameter sets.

- We analyze the proposed scheme and compare it with the conventional non-local algorithm. We also conduct theoretical analysis and provide an explanation to the superior performance of the proposed algorithm.

1.4 Organization of the Dissertation

The rest of the dissertation is organized as follows. The background of knowledge about standard video coding, texture analysis/synthesis, and textured region identification algorithms are reviewed in Chapter 2. The synthesis-based texture coding using the side information is studied in Chapter 3. Film grain noise analysis and synthesis is presented in Chapter 4. Textured image super-resolution algorithm is investigated in Chapter 5. Finally, concluding remarks are given and future research directions are pointed out in Chapter 6.

Chapter 2

Research Background

In this chapter, we provide a brief review to the background required for our current research. An overview on H.264/AVC and the rate-distortion (R-D) optimization is first described in Sec. 2.1. Then, previous work on texture synthesis, including parametric and non-parametric texture synthesis methods, is reviewed in Sec. 2.2. Finally,

2.1 H.264 Video Coding Standard and Rate-Distortion Optimization

2.1.1 H.264/AVC Video Coding Standard

The block-diagram of an H.264/AVC encoder is shown in Fig. 2.1. Being similar to previous video coding standards such as MPEG1, MPEG2 [45] and H.263 [31], the input sequence is encoded as one of three frame types, *i.e.*, the intra-coded (I), the predicted (P), or the bi-directionally predicted (B) frames. An input frame is further segmented into macroblocks (MBs), where each MB is coded by the inter- or intra-prediction modes. One 16x16 MB can further be partitioned into one 16x16 block, two 16x8 or 8x16 blocks

or four 8x8 blocks, where each 8x8 blocks can be further partitioned into two 8x4 or 4x8 blocks or four 4x4 blocks, depending on the inter-prediction mode. On the other hand, one MB can be partitioned into one 16x16 block, four 8x8 blocks or 16 4x4 blocks if the intra prediction mode is selected.

For intra-coded blocks, a pre-determined block transform is directly applied according to its block size. It is followed by scalar quantization, which is controlled by the quantization parameter (QP). For inter-coded block, inter prediction is first performed using the motion estimation (ME) process to search its best-matched block from the set of reference frames. After the search, motion vectors and the residual signals are generated as the output data. Then, this residual signal is encoded by the pre-determined block transform and quantization, which is similar to that in the coding of intra blocks.

Finally, an entropy encoder encodes quantized coefficients of intra and inter blocks to reduce redundancy furthermore. There are two kinds of entropy encoders in H.264/AVC: the context-adaptive binary arithmetic coder (CABAC) and the variable length coder (VLC). CABAC is used to encode all syntax elements. There are two VLC coders: the context-based adaptive variable length coding (CAVLC) and the universal variable length coding (UVLC), which are used to encode source data and header data, respectively.

2.1.2 Rate-Distortion Optimization

The rate-distortion optimization (RDO) process is often used to decide the best coding mode among various coding modes in H.264/AVC so as to minimize the distortion under a certain rate constraint [52, 69]. One of the example for RDO in H.264/AVC is intra/inter/skip mode decision selection. Since the coded MB is partitioned into a different

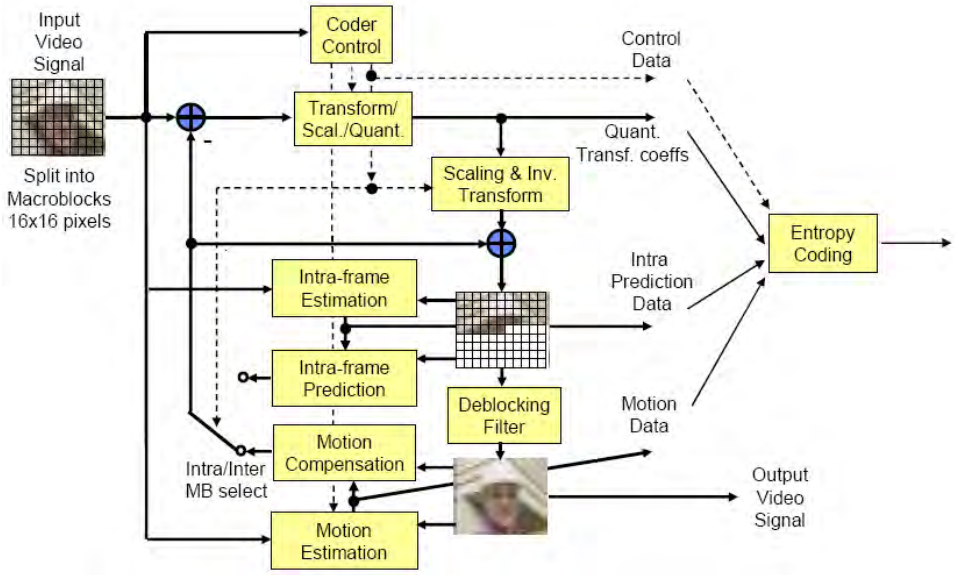


Figure 2.1: The structure of the H.264/AVC video encoder.

number of blocks according to the inter or intra prediction mode, the RDO process first finds the best MV or the best intra prediction direction for a specific inter prediction mode or intra type, respectively. After that, the RDO process either actually performs the encoding task, which includes the spatial domain transform, quantization and entropy encoding or uses a model to determine the associated coding rate and distortion. Finally, the RDO process finds the best inter or intra prediction mode that yields the minimal rate-distortion (RD) cost for the MB. Typically, the RDO process performs several times of the encoding process to decide the best coding mode that minimizes the distortion and meets the rate constraint simultaneously.

2.2 Texture Synthesis

As mentioned in Sec. 1.2, there are two texture synthesis approaches: parametric and non-parametric approaches. Their common objective is to generate naturally-looking texture that is visually similar to given seed texture. However, they adopt different ways to achieve this goal. We describe these two approaches and comment on their advantages and disadvantages in the following two subsections.

2.2.1 Parametric Texture Synthesis

The parametric texture synthesis approach has a longer history. It first infers an appropriate texture model from the seed texture and identifies its model parameters. Then, new texture is synthesized from the derived model. Since the parametric synthesis approach attempts to model the basic physical behavior of the seed texture with a model and its parameters, it is also called physics-based synthesis.

In the early development of the parametric synthesis approach, the simple auto-regressive (AR) or moving-average (MA) model was derived in the pixel [10, 17, 73] or the transform domain [23, 62]. From late 90's, a stochastic model was shown to be more efficient in specifying the texture behavior, and several algorithms based on this principle were proposed [54, 57, 58, 87]. They determine a set of statistical features and new texture that matches these statistics with its seed is synthesized. Portilla *et al.* [57, 58] considered a multi-resolution statistical model, and showed that matching statistics between subbands plays a critical role in visual output experimentally.

For the purpose of texture coding, the parametric texture synthesis method is depicted in Fig. 2.2, where the texture analyzer (TA) analyzes the seed texture and finds an appropriate model and its parameters. Then, the texture synthesizer (TS) synthesizes new texture from the model and its parameters.

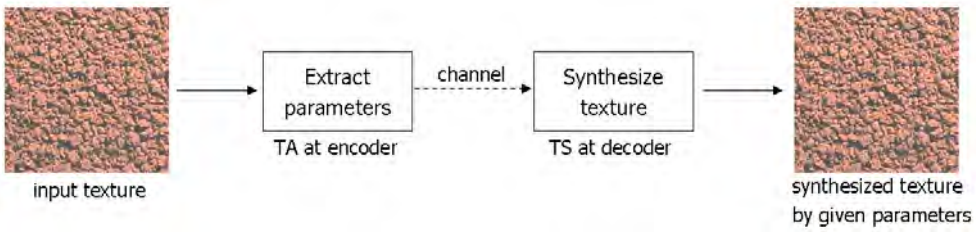


Figure 2.2: Texture coding with parametric texture synthesis.

The main advantage of parametric synthesis is its broad range of possible texture to be synthesized. Since it is based on the features of seed texture, it can generate an arbitrary shape of texture as long as the texture model is able to catch all texture behavior (which implies a reasonable size of seed texture). However, it is in general difficult to find a good model for given texture due to the limited size and the inherent physical complexity of the seed texture. Besides, it is well known that the parameter approach cannot represent structural texture well.

2.2.2 Non-parametric Texture Synthesis

In non-parametric texture synthesis, we do not analyze and build a model for the seed texture. Instead, it uses the information from the seed texture directly through a copy-and-paste process. It is sometimes called image-based texture synthesis.

The main difficulty of the non-parametric approach is to hide the boundary between adjacent patched blocks so that synthesized texture is perceptually natural. Various algorithms have been introduced to hide patch mismatch, such as feathering [39], dynamic programming [21] or graph-cut [35]. The graph-cut algorithm proposed by Kwatra provides a promising solution since it often gives a good result with low computational complexity. Besides, it can be easily extended to the 3D case.

In the context of texture coding, we depict the non-parametric texture synthesis approach in Fig. 2.3, where the TA determines the seed and the target textures and the TS synthesizes new texture with translated seed texture.

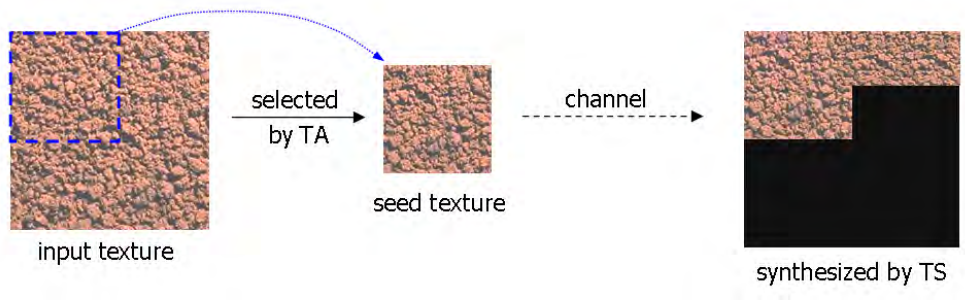


Figure 2.3: Texture coding with non-parametric texture synthesis.

The main advantage of the non-parametric synthesis approach is that it generally generates more natural texture than the parametric synthesis approach. Especially, it can preserve the texture structure well while the parametric synthesis approach often fails to render the strong structure of underlying texture. Besides, it does not need to build a different model for each seed texture as the parametric synthesis approach does. That is, the same mechanism such as raster-order patching or texture growing can be

applied to the synthesis of all types of texture. However, the huge complexity is its main disadvantage. This is especially true in the video coding application.

2.3 Texture Region Identification and Segmentation

Extracting the textured regions from the input image/video is important pre-processing step in our proposed framework, since the proposed synthesis-based coding or texture super-resolution algorithm is only applied to the identified textured region. Typically, we can classify the previous texture feature extraction algorithm into four classes : structural methods, statistical methods, model-based methods and transform or spatial-frequency methods [6].

Structural method assumes that texture is composed of small texture element, which is often called 'texton' or 'textel' [16]. Statistical method represents texture by its statistical properties such as second order statistics [29]. It is generally true that second order statistic is not enough to perfectly represent texture, so that high-order texture analysis schemes are also popular [79]. Model-order texture analysis schemes rely on the specific models, such as linear auto-regressive (AR) or moving-average (MA) model [42], fractal model [33], etc. Finally, transform based method utilize the sparsity of texture by compacting the texture signal into the specific bases. Gabor filter [32] , tree-structured packet-wavelet filter [15] are its popular choice. Though we classify the texture analysis and feature extraction methods as above, many algorithms adopt more than two schemes at the same time to improve its performance and compensate the drawbacks.

Chapter 3

Synthesis-based Texture Coding with Side Information

3.1 Introduction

Over the last two decades, video compression technologies have continued to improve in coding efficiency. This is evidenced by the development of several successful video coding standards, such as MPEG1, MPEG2 [45], H.261 [30], H.263 [31] and H.264 [82]. Although the storage capacity and the transmission bandwidth continue to increase, we assert that further improvement in video coding efficiency is desirable for two reasons. First, the size of video data is getting larger due to the arrival of high definition video contents such as high-definition TV (HDTV), digital cinema, and ultra-high-definition TV (UHDTV). The resolution of HDTV is 1920x1080, which is about six times of the single-definition (SD) video, and UHDTV has a resolution of at least 24 times that of the SD video. Second, new consumer electronics demands better visual quality than before due to the state-of-the-art display technology. Historically, video coding systems have been targeted at the cathode-ray-tube (CRT) based display. However, the CRT has been largely replaced by alternative display technologies nowadays such as the liquid crystal,

plasma and digital micro-mirrors. These technologies have a brighter and higher dynamic range that allows consumers to perceive more visual information. As a result, efforts in developing the next generation video technology are still in progress, and various creative ways have been attempted to improve coding efficiency furthermore.

One possible approach to video coding efficiency improvement is to develop a content-adaptive compression algorithm. In current video coding standards, the same high-level coding structure is used for all video contents; namely, the motion-compensated prediction (MCP) and quantization in the block transform domain, where MCP and the block transform are used to remove temporal and spatial redundancies, respectively. It is however desirable to develop special coding techniques for specific video contents. Along this line, dynamic motion texture is one class of contents receiving special attention. Note that block-based MCP does not work well due to its dynamic motion while a block transform does not work well due to fine details and high-frequency components of texture. One special texture coding technique is synthesis-based coding. There are two reasons that synthesis can replace conventional encoding/decoding for texture. First, texture is often perceived differently in such a manner that it does not demand reproduction with pixel accuracy. Second, its quasi-stationary pattern enables the generation of visually-similar texture by exploiting its statistical properties.

The overall system of a typical synthesis-based texture coding algorithm is depicted in Fig. 3.1. Instead of decoding the texture data directly, a module called the texture synthesizer (TS) is used at the decoder to generate visually similar texture. For this scheme to work properly, one module called the texture analyzer (TA) is included in the encoder to examine the input video, separate texture and non-texture regions, extract

parameters from texture regions and encode non-texture regions. When the coded bit-stream is sent to the decoder, the decoder then decodes the bit-stream of non-texture regions as usual and uses the TS to fill out texture regions.

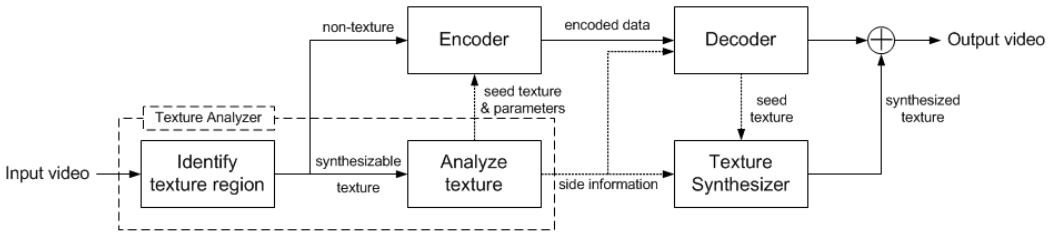


Figure 3.1: A video coding system with the texture analyzer (TA) module and the texture synthesizer (TS) module.

Texture synthesis is an active research topic in several fields. Besides synthesis-based texture coding, texture synthesis technologies have been applied to image/video texture rendering [88], completion [70], and inpainting [4]. Generally speaking, there are two major approaches for texture synthesis. The first one is the parametric approach [54, 57, 58, 87], where an image sequence is modeled by a number of parameters such as the histogram or correlation of pixel values. Given a sufficient number of model parameters, one may recreate the "look and feel" of any texture that meets the parameterized constraints. The second one is the non-parametric approach [21, 22, 34, 35, 39], where synthesized texture is derived from an example texture as the seed that is known *a priori*. The texture synthesis process creates new texture data by inspecting the seed texture and copying intensity values from the seed for the extrapolated area. Most texture synthesis work is conducted in the image domain with only a few done in the video space.

Most previous synthesis-based texture video coding algorithms follow the basic diagram as shown in Fig. 3.1. In [46], the texture region of the target frame is analyzed by

the TA and reconstructed by the TS using the frame-to-frame displacement and image warping techniques. This method is however not effective for non-rigid texture objects and the frame-by-frame synthesis can yield temporal inconsistence. A similar texture synthesis technique using frame-to-frame mapping was proposed in [6], which focused more on texture analysis and segmentation. A cube-based texture growing method was proposed in [47, 48], which can be viewed as an extension of [35] from image to video. Temporal consistency is guaranteed in [48] due to cube-based synthesis. However, the global texture structure can be easily broken by texture growing with the raster-scanning order. Furthermore, the output texture tends to have repeated patterns by the fixed-size cube-based growing procedure. An alternative approach was proposed in [20], where it replaces the input texture with a perceptually similar and well-encodable synthesized texture. As compared with other methods, it manipulates the input video at the encoder only so that existing decoding modules can be used without any modification. However, the constraint-based texture synthesis method derived from [58] is generally not efficient for structured texture.

We extend the result in [50] and provide thorough theoretical analysis in this work. Specifically, we propose a new framework in which texture video is synthesized with the help of a low-quality version of the source video obtained by the TA, which is called the *side information*. Under the assumption that the texture region of input video is given, we focus on the side information preparation at the encoder and texture synthesis at the decoder in this research. Although the side information demands bits for its coding, the proposed algorithm is more flexible since the behavior and quality of the output texture can be adjusted by the amount of the side information, which is determined by the

user. To this end, we develop an area-adaptive side information selection scheme that chooses the proper amount of the side information for a given bit budget. Furthermore, we propose a texture decomposition scheme that extracts the non-synthesizable illumination component from the source video for separate coding to enhance the overall coding performance. As a result, the unpleasant mismatch between analyzed and synthesized regions can be removed and the output texture can be more visually similar to the target texture. The superior performance of the proposed texture video synthesis technique is demonstrated by several coding examples.

The rest of this paper is organized as follows. First, a brief review of example-based texture synthesis is given in Sec. 3.2. Then, the overall structure of the proposed scheme is detailed in Sec. 3.3. Experimental results are given in Sec. 3.4 to demonstrate the effectiveness of the proposed scheme. Finally, concluding remarks are given in Sec. 3.5.

3.2 Example-based Texture Optimization

The example-based texture optimization technique proposed in [34] is adopted for texture synthesis in our work. The core of the algorithm is defining the texture distortion between the seed and the synthesized textures, and minimizing it using an iterative optimization method. For the latter, various energy minimizing methods can be used.

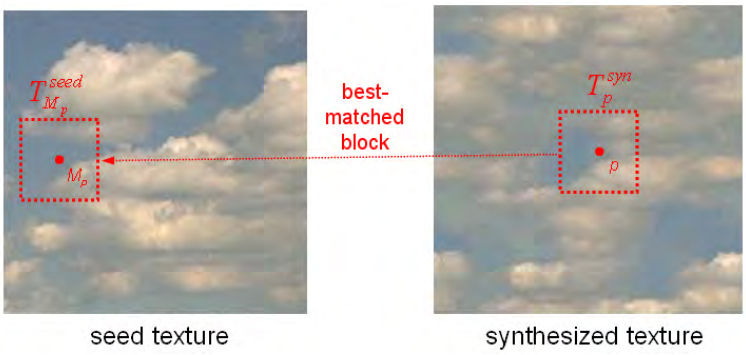


Figure 3.2: Illustration of the texture distortion.

Texture distortion is defined as the distance between a block of the synthesized image and its best-matched block in the seed texture. The total texture energy is obtained by summing distortion values in all blocks as

$$\begin{aligned} E_{syn} &= \sum_{p} D_T(T_p^{syn}, T^{seed}) \\ &= \sum_{p} ||T_p^{syn} - T_{M_p}^{seed}||^r, \end{aligned} \tag{3.1}$$

where T_p^{syn} denotes a column-vectorized form of a block of size $N \times N$ in synthesized texture around grid p, and $T_{M_p}^{seed}$ indicates its best-matched vectorized block around grid pixel M_p in seed texture. The texture distortion function D_T between a block of the synthesized texture (T_p^{syn}) and the seed texture (T^{seed}) as defined in Eqn. (3.1) is shown in Fig. 3.2. Typically, blocks are extracted from seed texture in an overlapping manner. Although this equation is defined in the image domain, it can be extended to a 3D video cube in a straight-forward manner.

The optimized output will be the solution that yields the smallest distortion energy in Eqn. (3.1). Unfortunately, it is not trivial to find the optimized solution. An Expectation-Maximization (EM) algorithm was adopted in [34], where an initial texture estimate is iteratively refined to decrease the distortion energy of the synthesized texture. One of the difficulties of this cube-wise processing is finding an appropriate value for the cube size. Generally, a larger cube is suitable in keeping the global structure while a smaller cube is advantageous in representing local detail [49]. The difficulty was solved by the multi-resolution and multi-scale texture-synthesis method in [34]. By multi-resolution, the texture in the coarse-level image is first synthesized and, then, the up-sampled texture in the finer-level image is refined. By multi-scale, the cube size can be varied from a larger one to a smaller one. Both techniques can keep the global structure and synthesize fine details at the same time. This algorithm is summarized in the following pseudo-code.

Algorithm 1 Texture synthesis by energy minimization

1: For each resolution and cube size
2: For each cube centered at pixel p,
3: $T_{M_p}^{seed} \leftarrow \arg\min_{T_q^{seed}} ||T_p^{syn} - T_q^{seed}||^r$ for all grid space q
4: $T^{syn} \leftarrow \arg\min_{T^{syn}} \sum_p ||T_p^{syn} - T_{M_p}^{seed}||^r$
5: Repeat steps 2-4 until the change of T^{syn} is small enough

3.3 Proposed Algorithm: Texture Synthesis with Side Information

The main novelty of our algorithm is to add a new component to the traditional texture video synthesis process, which is a low-quality video representation of the source video, called the side information. The basic framework of our algorithm is described in Sec. 3.3.1. The expectation-maximization optimization process is explained in Sec. 3.3.2.

Then, discussion on proper side information selection is given in Sec. 3.3.3. Finally, the texture decomposition scheme is presented in Sec. 3.3.4.

3.3.1 Basic Framework

The idea of texture video synthesis with the side information can be intuitively formulated as

$$\begin{aligned} T^{syn} &= \arg\min_{T^{syn}} \sum_{p} D_{T\_SI}(T_p^{syn}, T_p^{SI}, T^{seed}) \\ &= \arg\min_{T^{syn}} \sum_{p} \left[||T_p^{syn} - T_{M_p}^{seed}||^r + \alpha \cdot D_{SI}(T_p^{syn},\ T_p^{SI}) \right], \end{aligned} \tag{3.2}$$

where T_p^{SI} is a column-vectorized form of a cube of low-quality video around grid pixel p, and D_{SI} is a distortion measure between synthesized and low-quality video. This equation demands that the synthesized texture should be closed to the given seed texture as well as the side information. The weight between these two constraints is controlled by parameter α. For example, if we set $\alpha = 0$, this framework is reduced to the traditional texture video synthesis algorithm as given in [48]. On the other hand, the contribution of the side information to the final synthesized texture video increases as α becomes larger.

In this work, we adopt the standard video coding tool with a larger quantization parameter to obtain low-quality video and use the reconstructed video as the side information. This side information generation approach is advantageous in several aspects. First, it can be readily incorporated in any video coding system since it does not need any additional module to produce the side information. Second, the quality of the side information can be easily controlled by adjusting the quantization parameter (QP) value. Third, it allows the usage of any video coding technique. In this work, we use the

H.264/AVC standard for the coding of the side information as well as the non-texture part of the source video.

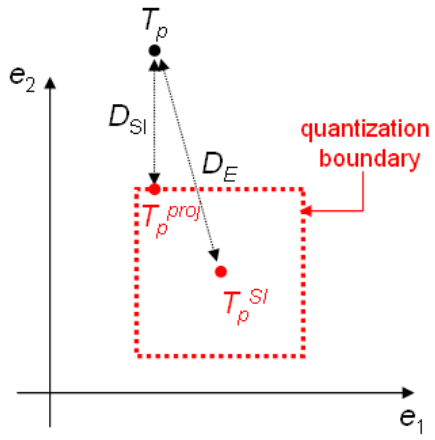


Figure 3.3: Comparison of two distance measures: the distance in the transform domain denoted by D_{SI} and the Euclidian distance denoted by D_E.

Next, we examine an efficient way to compute the distortion term, $D_{SI}(T, T^{SI})$, in Eqn. (3.2). A simple Euclidean distance in the pixel domain may not be effective since it does not take the quantization process of the side information into account. This is especially true when the QP value is large. Thus, we adopt another approach. That is, we first determine the closest point by projection in the transform domain as proposed in [86], and then compute the distance between the projected point and the synthesis point as shown in Fig. 3.3. As a result, the proposed distortion measure only penalizes the solution when the signal falls out of the quantization bin.

Mathematically, this can be expressed as

$$D_{SI}(T, T^{SI}) = || \, T - [T]^{+} ||^{r},$$
$$\text{where } [\overline{T_i}]^{+} = \begin{cases} \overline{T_i} & \text{if } |\overline{T_i} - \overline{T_i^{SI}}| \leq \Delta, \\ \overline{T_i^{SI}} + \Delta & \text{if } \overline{T_i} > \overline{T_i^{SI}} + \Delta, \\ \overline{T_i^{SI}} - \Delta & \text{if } \overline{T_i} < \overline{T_i^{SI}} - \Delta, \end{cases} \tag{3.3}$$

where $[\cdot]^{+}$ represents a projection function, the overbar indicates the vector form of the cube in the transform domain, subscript i indicates the i^{th} vector element and Δ is the quantization step size.

3.3.2 Optimization via Expectation-Maximization (EM)

Since the objective function in Eqn. (3.2) has two unknown data (*i.e.,* T^{syn} as the latent variable and $\{T_{M_p}^{seed}\}_{p=1}^{P}$ as its unknown parameter set), we apply the expectation-maximization (EM) algorithm for its minimization. In our implementation, we modify the objective function as

$$T^{syn} \;=\; \arg\min_{T^{syn}} \sum_{p} \left[\; ||T_p^{syn} - T_{M_p}^{seed}||^{r} + \alpha \cdot D_{SI}(T_{M_p}^{seed}, \; T_p^{SI}) \; \right], \tag{3.4}$$

where we replace T^{syn} by T^{seed} in the second term for the ease of computation. Although the objective function in Eqn. (3.2) is easier to understand, the two objective functions eventually target at a very similar cost function during the iterative process due to the closeness of T_p^{syn} and $T_{M_p}^{seed}$, which will be further investigated later. For the rest of this

section, we will describe the EM algorithm with respect to the objective function in Eqn. (3.4).

In the proposed EM procedure, T^{syn} is the unobserved latent image, $\{T^{seed}_{M_p}\}_{p=1}^{P}$ are its unknown parameters, and T^{seed} and T^{SI} as observations. For convenience, we let

$$\theta = \{T^{seed}_{M_p}\}_{p=1}^{P}.$$

Then, the EM alternating procedure will iteratively determine the sequence of estimates. That is, for $n = 0, 1, 2, \cdots$, we perform the following.

E-step:

Calculate the expected value of the log-likelihood function with respect to the conditional distribution of T^{syn} given observations (T^{seed}, T^{SI}) under the current estimate of parameter θ. Mathematically, we have

$$Q\Big(\theta, \theta^{(n)}\Big) = E\left[\log p\big((T^{seed}, T^{SI}), T^{syn}|\theta\big) \,\middle|\, (T^{seed}, T^{SI}), \theta^{(n)}\right], \tag{3.5}$$

where $\theta^{(n)}$ denotes the n^{th} iteration value of θ and $\theta^{(0)}$ is the initial guess.

M-step:

Find parameter θ that maximizes Q in Eqn. (3.5):

$$\theta^{(n+1)} = \arg\max_{\theta} Q\Big(\theta, \theta^{(n)}\Big). \tag{3.6}$$

To carry out the computation in the E-step, we can have the following simplification. First, it is clear that

$$\begin{aligned} p\big((T^{seed}, T^{SI}), T^{syn}\big|\theta\big) &= p\big((T^{seed}, T^{SI})\big|T^{syn}, \theta\big)\, p(T^{syn}|\theta) \\ &= p(T^{seed}|T^{syn}, \theta)\, p(T^{SI}|T^{syn}, \theta)\, p(T^{syn}|\theta) \\ &= p(T^{seed}|T^{syn})\, p(T^{SI}|\theta)\, p(T^{syn}|\theta)\,, \end{aligned} \tag{3.7}$$

since T^{seed} is independent of T^{SI} and θ, and T^{SI} is independent of T^{syn}.

By assuming that the target texture obeys the following probability distributions

$$\begin{aligned} P(T^{syn}|\{T_p^{seed}\}) &\propto \exp\left(-\frac{1}{2\sigma_1^2}\sum_p ||T_p^{syn} - T_{M_p}^{seed}||^2\right), \\ P(T^{SI}|\{T_p^{seed}\}) &\propto \exp\left(-\frac{1}{2\sigma_2^2}\sum_p D_{SI}(T_{M_p}^{seed}, T_p^{SI})\right), \end{aligned} \tag{3.8}$$

we obtain

$$\begin{aligned} &\log p\big((T^{seed}, T^{SI}), T^{syn}\big|\theta\big) \\ &\quad= -\frac{1}{2\sigma^2}\sum_p \Big[\alpha D_{SI}(T_{M_p}^{seed}, T_p^{SI}) + ||T_p^{syn} - T_{M_p}^{seed}||^2\Big] + A_1 \\ &\quad= -\frac{1}{2\sigma^2}\sum_p \Big[\alpha D_{SI}(T_{M_p}^{seed}, T_p^{SI}) + ||T_{M_p}^{seed}||^2 - 2(T_{M_p}^{seed})^T T_p^{syn}\Big] + A_2\,, \end{aligned} \tag{3.9}$$

where A_2 (or A_1) is a constant which is independent of θ, and α can be viewed as a regularization factor between variances of T^{syn} and T^{SI}. Since the log-likelihood of the

distribution is linear in latent variable T^{syn}, the E-step could be simplified by estimating the conditional expectation of T^{syn} given observation T^{seed} and current parameter estimate $\theta^{(n)}$ [24, 44]. This can be written as

$$\begin{aligned}(T^{syn})^{(n)} &= E\Big[T^{syn}\Big|T^{seed},\theta^{(n)}\Big] \\ &= E\Big[T^{syn}\Big|\{T^{seed}_{M_p}\}^{(n)}\Big]\,.\end{aligned} \tag{3.10}$$

Since the conditional probability follows the Gaussian distribution as given in Eqn. (3.8), its expected value eventually becomes the (weighted) average of the current estimate $\{T^{seed}_{M_p}\}^{(n)}$ in overlapped regions by solving a linear minimum mean square error (LMMSE) problem. It is worthwhile to point out that we expand and derive equations using the L2 norm, *i.e.*, $r = 2$ in Eqns. (3.3) and (3.4), since it is more convenient to compute the expectation value. The computation with an arbitrary r value can be done by introducing an adaptive weight $w_p = ||\cdot||^{r-2}$ at each grid p as [34]

$$||T^{syn}_p - T^{seed}_p||^r = w_p \cdot ||T^{syn}_p - T^{seed}_p||^2\ .$$

After the above simplification, we can compute the Q-function required in the E-step as

$$Q\Big(\theta,\theta^{(n)}\Big) = -\frac{1}{2\sigma^2}\sum_p \Big[||(T^{syn}_p)^{(n)} - T^{seed}_{M_p}||^r + \alpha D_{SI}(T^{seed}_{M_p}, T^{SI}_p)\Big] + A_1\,. \tag{3.11}$$

Finally, the M-step updates parameter $\theta^{(n+1)}$ by maximizing the above Q function via

$$\begin{aligned}\theta^{(n+1)} &= \arg\max_{\theta} Q\Big(\theta, \theta^{(n)}\Big) \\ &= \arg\max_{\theta} \left[||T_{M_p}^{seed} - (T_p^{syn})^{(n)}||^r + \alpha D_{SI}(T_{M_p}^{seed}, T_p^{SI}) \right] ,\end{aligned} \tag{3.12}$$

which is identical with the objective function in Eqn. (3.4).

If we attempt to solve the original objective function in Eqn. (3.2), the averaging process in the E-step will demand the exact data of T^{SI} which cannot be explicitly obtained. For example, when the current candidate $T_{M_p}^{syn}$ is outside of the quantization cube of T_p^{SI}, its optimal value in terms of D_{SI} will be always the one on the quantization boundary, which would often be far from the desired one. On the other hand, this problem is avoided with the modified objective function in Eqn. (3.4) since the non-metric distance function, D_{SI}, is only involved in the penalty function of the M-step.

To sum up, the M-step chooses a well-matched seed block between T^{syn} and T^{SI} while the E-step updates T^{syn} based on the given set of seed blocks. Since we add the additional distortion term w.r.t. T^{SI} in the M-step, the selected best-matched seed block is affected by the constraint of the side information so that it forces the synthesized texture to be close to the original target texture. On the other hand, the low-quality video as the side information is mostly composed of the low-frequency information and, consequently, it only controls the general structure of the synthesize texture with texture details to be synthesized flexibly. This is exactly what we achieve to accomplish.

The proposed texture video synthesis algorithm with the side information is summarized by pseudo-codes in the following.

Algorithm 2 Texture synthesis with side information

1: For each resolution and cube size
2: For each cube centered at pixel p,
3: $T_{M_p}^{seed} \leftarrow \arg\min_{T_q^{seed}} [||T_p^{syn} - T_q^{seed}||^r + \alpha \cdot D_{SI}(T_q^{seed}, T_p^{SI})]$ for all possible grid points q
4: $T^{syn} \leftarrow \arg\min_{T^{syn}} \sum_p ||T_p^{syn} - T_{M_p}^{seed}||^r$
5: Repeat steps 2-4 until the change of T^{syn} is small enough

3.3.3 Side Information Selection

In the proposed framework, low-quality video controls the low frequency component of synthesized output texture, and the importance of the side information will vary in different regions. That is, parts of texture can be well-synthesized with little side information while others cannot. For this reason, our algorithm assigns a different amount of side information to different regions of texture.

To determine the proper amount of the side information, we formulate the rate-distortion (R-D) optimization problem under the rate constraint as

$$\begin{aligned} J_{ad}(Q) &= D_{ad}(Q) + \lambda_{ad} \cdot R_{ad}(Q) \\ &= ||T^{syn} - T^{tar}||^r + \lambda_{ad} \cdot R(T^{SI}) \,, \end{aligned} \tag{3.13}$$

where subscript ad means the adaptive side information selection scheme, T^{tar} represents the original target texture to be synthesized. It is worthwhile to emphasize that Eqn. (3.13) is different from the traditional R-D optimization since the distortion in Eqn. (3.13) is computed using synthesized output (rather than coded output). The above Lagrangian formulation can regularize between the distortion of the output and the side information bit-rate. In our implementation, we solve the problem by pre-computing all R-D pairs for each coding unit with various QP values, and then find the QP value that minimizes

the objective function in Eqn. (3.13) for each coding unit. After that, we sum up all rates and see if the total rate exceeds the target or not, and then adjust λ_{ad} accordingly. This R-D optimization procedure will return the minimum distortion value for a given bit budget. It is summarized in the following pseudo-code. The algorithmic speed-up is possible if we are able to build the R-D model for synthesized texture, which is outside the scope of this work.

Algorithm 3 R-D optimization for adaptive side information

1: Initialize λ_l and λ_h with a small and a large numbers, respectively.
2: We set $\lambda_{ad} \leftarrow (\lambda_l + \lambda_h)/2$
3: For each non-overlapping block
4: Compute $D_{ad}(QP)$ and $R_{ad}(QP)$ for $QP = QP_n, \quad n = 1, 2, ..., N$
5: Finds k, which minimizes $D_{ad}(QP_k) + \lambda_{ad} \cdot R_{ad}(QP_k)$
6: Compute the total required bit-rate R_{total}.
7: If the total rate does not exceed the given bit budget (*i.e.,* $R_{total} \leq R_{budget}$), we set $\lambda_h \leftarrow \lambda_{ad}$. Otherwise (*i.e.,* $R_{total} > R_{budget}$), we set $\lambda_l \leftarrow \lambda_{ad}$.
8: Repeat Steps 2-8 until the change of λ_{ad} is small enough.

3.3.4 Texture Decomposition

The algorithm discussed above assumes that the input texture is stationary so that it may not perform well for non-stationary texture. In this sub-section, we address the issue of non-stationary texture synthesis using an appropriate pre-processing module. The idea is to decompose non-stationary input texture (V) into two components; namely, easily-synthesizable texture (T) and easily-encodable non-texture (NT), as

$$V = T + NT. \tag{3.14}$$

Then, NT can be encoded/decoded using a conventional method while T can be synthesized by the proposed texture synthesis algorithm. One application of this decomposition

technique is the coding of texture with spatially or temporally variant illumination, where piece-wise smooth illumination changes can be easily encoded. For simplicity, we discuss the texture decomposition problem in the context of 2D texture image. However, the same idea can be easily generalized to the 3D texture video, and we summarize the computational algorithm in the context of 3D texture video in the end of this subsection.

There has been research on the separation of texture from its smooth background image, *e.g.,* [2, 66, 80]. All of them exploit the sparsity of texture as an image prior, and introduce various methods to specify the sparsity using an appropriate transform, such as the Gabor filter. Since the statistical property of texture differs from smooth background, one can build a model to compact the texture signal while minimizing the background signal, where the model should leverage on the sparsity of input texture. However, as mentioned in Sec. 3.1, the model-based approach is limited in its applicability a large class of textures. Furthermore, to find an appropriate model and its parameters is also challenging. Here, we propose a data-driven texture decomposition approach based on the self-similar property of texture, where the sparsity transform is replaced by a data-driven scheme. This approach has received attention in the image restoration field recently [9, 19, 37, 60]. Since the data-driven approach is compatible with the proposed texture synthesis scheme, the texture decomposition method can be easily integrated with our synthesis-based texture coding framework.

Since our goal is to improve coding efficiency, the texture decomposition scheme should be applied if the total cost after the decomposition is lower than

1. the R-D cost of the conventional coding method,

2. the R-D cost of the synthesis-based coding method without decomposition.

For comparison, we have to measure the cost of decomposed NT and T components quantitatively. The cost of encoding the source video signal, V, can be written as the sum of those for T and NT as

$$J(V) = J(NT) + \omega_d \cdot J(T), \tag{3.15}$$

where ω_d is a weight parameter. A straightforward choice for $J(NT)$ is the R-D cost of NT in form of $J(NT) = D(NT) + \lambda R(NT)$. There is however an issue to consider. That is, NT should be uncorrelated with T as much as possible. If the decomposed NT still contains some texture information, the final summed result by synthesized T and decoded NT may have an unmatched pattern, which would cause large perceptual distortion. Based on this consideration, the R-D cost for NT is not appropriate, since the decomposed NT with a smaller R-D cost does not guarantee a lower correlation between NT and T. Here, we adopt a measure based on the total variation (TV) and define the cost function of NT as

$$J(NT) = TV(NT) = \int_{NT} |\nabla NT| dxdy. \tag{3.16}$$

Due to its excellent performance in controlling the gradient with the L_1 norm, the TV-based cost has been frequently used to measure the energy of the piecewise-smooth component of an image in various applications [2, 40, 80]. It is shown to be effective in

filtering out the texture/noise component. In addition, it can play a role similar to the R-D cost of a piece-wise smooth component.

To determine $J(T)$ in Eqn. (3.15), we only have to consider the synthesis cost, since the decomposed texture information will be re-synthesized in the decoder. The texture distortion energy by synthesis, denoted by D_T in Eqn. (3.1), serves as a good metric to measure the synthesis cost, since similar texture might yield smaller texture distortion as discussed in Sec. 3.3. However, pixel-wise comparison of D_T often results in large distortion between blocks of texture image (or cubes of texture video) in spite of their visual similarity. To circumvent this problem, we define function f_r to relax the seed set by allowing small pixel-wise translation and intensity deviation with respect to a given block of texture image. For example, for an $N \times N$ block $I \in T^{seed}$, I_r will be the element of a relaxed seed set $I_r \in f_r(T^{seed})$ if it meets the following equation:

$$I_r(m,n) = I(m + \delta^m_{m,n}, n + \delta^n_{m,n}) + \delta^I_{m,n}, \tag{3.17}$$

where $(\delta^m_{m,n}, \delta^n_{m,n})$ and $\delta^I_{m,n}$ are small offsets for spatial translation and intensity deviation, respectively.

This approach is based on the assumption that the limited size of the seed pool, which is one of the disadvantages of example-based method, can be overcome by small spatial texture deformation and/or intensity variation, and the assumption that relaxed T is still well-discriminated from NT. Then, the synthesis cost $J(T)$ can be defined as

$$J(T) = D_T(T, f_r(T^{seed})). \tag{3.18}$$

The advantage of seed relaxation is graphically illustrated in Fig. 3.4, where T is the target block for synthesis and C_1, C_2 and C_3 are three best matched blocks as shown in Fig. 3.4(b). Without relaxing the seed set, the residuals between the target and the best-matched blocks are still obvious as shown in Fig. 3.4(c) despite their perceptually similar outlook. In contrast, with slight texture deformation, the residuals can be reduced significantly via the distortion defined in Eqn. (3.18) as shown in Fig. 3.4(d). In other words, the correlation between decomposed NT and T is greatly minimized.

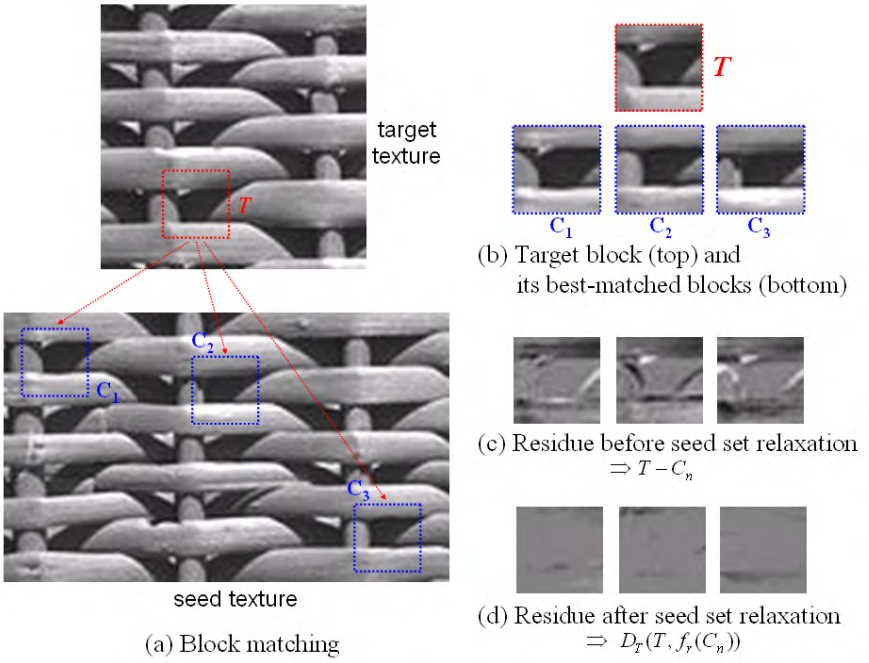


Figure 3.4: Illustration of an adjusted synthesis cost using a relaxed seed set.

To summarize, we maximize the piece-wise smoothness of the NT component as image prior with given texture candidates as the data fidelity term. This problem can be viewed

as the conventional energy minimization problem between $J(T)$ as the data fidelity term and $J(NT)$ as the regularization term. The objective function can be written as

$$\begin{aligned} &\inf_{NT} \left(J(NT) + \lambda_d \cdot J(T) \right) \\ =&\inf_{NT} \left(TV(NT) + \lambda_d \cdot D_T(T, f_r(T^{seed})) \right) . \end{aligned} \tag{3.19}$$

Our approach is analogous to certain image/video processing techniques for de-noising [80] and super-resolution [60]. For example, a de-noising process may use the observation image as its data-term while a super-resolution process may use the set of low-resolution images as its data-term. In the current context, the synthesized image is gathered using the self-similarity property. The optimization problem in Eqn. (3.19) can be solved using the variational approach [2, 80].

Based on the above discussion, one can generalize results from 2D texture image decomposition to 3D texture video decomposition. In the following, we summarize the texture decomposition scheme for 3D texture video.

Algorithm 4 Texture video decomposition from its illumination background

1: Initialize $T \leftarrow V$, $NT \leftarrow 0$
2: For each resolution and a cube size
3: Assign the seed and target regions and obtain T^{seed}, NT^{tar} and T^{tar}
4: For each cube centered at pixel p in the target texture:
5: $T^{seed}_{M_p} \leftarrow \arg\min_{T^{seed}_q} ||T^{tar}_p - T^{seed}_q||^r$ for all grid space q
6: Relax the found best-matched set $\{T^{seed}_{M_p}\}$
7: $\{NT^{tar}, T^{tar}\} \leftarrow \arg\min_{\{NT,T\}} \left(\int_{NT} |\nabla NT| dxdy + \lambda_d \cdot D_T(T, f_r(T^{seed}_{M_p})) \right)$
8: Update NT and T with obtained NT^{tar} and T^{tar}
9: Repeat steps 2-8 until the change of $\{NT, T\}$ is small enough

Although it is more accurate to first relax the entire seed set and, then, find the best-matched block from the relaxed seed set, the search complexity is too high. Thus, we

swap the computational order in the implementation as described above. That is, we find the best-matched seed cube for each overlapping grid and relax it by relaxation function f_r. Due to the overlapping grid search and seed texture relaxation, each grid position has dozens of candidate pixel values as texture prior. The intermediate best-candidate $f_r(T_{M_p}^{seed})$ at each iteration will be the candidate with the minimum distance, and its best-candidate can be altered at every iteration.

The above decomposition process targets at illumination-variant texture decomposition under the assumption that stationary seed texture is given. It is however not trivial to identify the stationary part of the seed from the source texture. This could be resolved using an iterative decomposition process (*i.e.*, randomly choosing the seed from the given input and applying the decomposition algorithm under the assumption that the seed is stationary at each iteration.) We may also apply the multi-resolution/multi-scale approach, since it tends to avoid being trapped to the local minimum in the optimization process. Generally speaking, the seed texture and the target texture become gradually stationary through iteration.

3.4 Experimental Results

In this section, we show both image and video texture synthesis results. To obtain the side information, we used the JM reference software (ver.11) maintained by the Joint Video Team (JVT) [75] operating with the IPPP GOP structure and the 4×4 transform only. For the multi-resolution/multi-scale approach, we conducted experiments using 1/4, 1/2 and full resolutions as well as with cube (or block) size = 64, 32, 16 and 8. For more

experimental results and visual performance evaluation, please visit the following website: http://www-scf.usc.edu/~byungoh/TS/TS.htm.

3.4.1 Visual Comparison of Synthesized Texture

Figs. 3.5 and 3.6 show synthesized texture image and texture video from a seed coded with $QP = 20$ and a certain amount of side information. In the toilet sequence, only the center regions of the sequence are processed as the seed or target texture. Results without the side information are given in (c) using the patch growing method in [48] and in (d) using the example-based optimization method with α=0. Results using the texture growing method often have spatial or temporal discontinuity even with the use of the seam-hiding technique due to the difficulty of finding a suitable size of blocks or cubes. The noticeable seams in the straw image and the duck-take-off sequence are its good examples. The example-based optimization could overcome this disadvantage using the multi-resolution/multi-scale approach. However, results from both methods have repeated texture patterns with a global structure quite different from that of the original texture, which is undesirable in many cases. It is obvious when comparing the target texture and synthesized texture in the toilet sequence, where the synthesized toilet hole are moving with different shape.

As our proposed approach, results with the side information are given in (e) and (g), where a larger amount of the side information makes them closer to the target as shown in (h) while a smaller amount of the side information yields more natural-looking image/video as shown in (f). In (f) and (h), we fix $\alpha = 1$ in the TS, but allow the TA to adjust the α value to control the contribution of the given side information. The

synthesized texture with the side information is perceptually better than the decoded texture which is temporally inconsistent and discontinuous and the synthesized texture without the side information. The superiority of the proposed algorithm can be easily verified by viewing video sequences given in the website described above.

3.4.2 Texture Distortion versus Bit-Rate Saving

To show the advantage of sending the side information, we study the improvement of texture similarity by comparing the original target texture and the synthesized texture. As shown in Figs. 3.7 and 3.8, we vary the side information amount by adjusting the QP value and record the bit-rate saving and two distortion types: D_T and D_D. Here, D_T (the dotted line) represents the distortion between the seed and the synthesized texture as given in Eqn. (3.1), and D_D (the solid line) represents the texture deviation by computing the pixel-wise distortion between the target and the synthesized texture as defined in Eqn. (3.13). Although the pixel-wise distortion does not offer a good way to measure the similarity of textures, it is still meaningful by evaluating the overall deviation. The bit-rate saving is computed as

$$\text{bit-rate saving} = \left(1 - \frac{\text{bits for the side information}}{\text{bits for the target with high fidelity}}\right) \times 100\ (\%)\,. \qquad (3.20)$$

For distortion analysis, we set $r = 2$ in all distortion computation, *i.e.*, we use the Euclidean-norm for distortion analysis so that it is compatible with the conventional PSNR. On the other hand, we set $r = 0.8$ in the texture synthesis process for better

Table 3.1: Bit-rate saving for two sequences.

	toilet seq.		duck-take-off seq.	
QP for SI	bits for SI	bit-rate saving	bits for SI	bit-rate saving
-	0 kbps	100 %	0 kbps	100 %
40	3.0 kbps	96.8 %	9.4 kbps	95.4 %
35	3.7 kbps	96.1 %	16.4 kbps	92.1 %
30	14.0 kbps	85.1 %	28.2 kbps	86.4 %
25	47.9 kbps	48.9 %	65.4 kbps	64.5 %
20	93.8 kbps	0 %	207.4 kbps	0 %

visual output, since it boosts the texture detail more as mentioned in [34]. The numerical bit-rate saving is provided in Table 3.1.

We see that the pure-synthesis algorithm can achieve 100% bit-rate saving while it yields perceptually obvious texture distortion. A small amount of the side information (at the cost of a small amount of bit-rate increase) helps increase the quality of synthesis texture dramatically. Since the target bit-rate saving is computed using the decoded image with $QP = 20$, we get 0% bit-rate saving when $QP = 20$. It is also observed that texture quality is preserved or even slightly improved by the side information, since the side information offers a good initial point for the optimization process in Eqn. (3.2). Furthermore, the increase of the side information does not improves texture similarity linearly, since the degree of synthesized texture is governed by the size of the seed texture pool. It is therefore important for the TA to determine the proper side information amount that balances the trade-off between texture quality and the bit-rate.

3.4.3 Fixed versus Adaptive Amount of Side Information

Although the side information helps in texture synthesis, only a small amount is needed since it is only used in providing the low-frequency trend for the target texture. This problem was addressed in Sec. 3.3.3 by the use of an area-adaptive side information technique. Its merit is illustrated in Fig. 3.9, where we compute the rate and the distortion for various candidate QP values and find the optimal adaptive side information by the Lagrangian optimization method as studied in Sec. 3.3.3. The improvement of the adaptive side information algorithm is illustrated in Fig. 3.10, where the side information amount is determined by three candidate QP values, and these results are compared with a fixed side information amount for all blocks at the same bit-rate. This figure clearly show that the adaptive side information scheme improves the performance in both texture distortion (D_T) and texture deviation (D_D).

3.4.4 Texture Decomposition

Finally, illumination-variant texture decomposition results are shown in Fig. 3.11, where the block image and the toilet sequence are spatially and temporally illumination-variant, respectively. The input texture shown in (a) is first decomposed into texture and non-texture components as shown in (b) and (c), respectively. We set regularization parameter $\lambda_d = 0.05$ in Eqn. (3.19), and all offset parameters $\delta^m_{m,n}, \delta^n_{m,n}, \delta^I_{m,n} \in \{-2, -1, 0, 1, 2\}$ in Eqn. (3.17) experimentally. Without decomposition, the synthesis algorithm cannot regenerate the input texture well as shown in (d). After decomposition, the proposed synthesis-based texture coding algorithm was applied to the texture as given in (f) with

the help of its side information as given in (e), and the final result is obtained by summing the synthesized texture and the decoded non-texture as given in (g). To compare the synthesis-based and the conventional coding methods, we encode/decode the input texture as given in (h) with the same bit-rate as the synthesis-based method. We see that texture coded by the conventional coding method has lots of spatial or temporal discontinuity, such as blocking or flickering artifacts, while the proposed method generates perceptually similar and visually pleasant results in spite of its pixel-wise difference.

3.5 Conclusion and Discussion

A new approach for synthesis-based texture coding was proposed and several important contributions were made in this work. First, a novel framework to control the synthesized texture by sending the side information was developed, and a small amount of side information can improve texture synthesis performance significantly. The side information can be easily obtained by a standard encoder operating at a larger quantization parameter. Second, we showed a way to increase the synthesis efficiency by selecting the side information adaptively. The adaptive side information selection scheme outperforms the fixed side information scheme for a given bit budget. Third, a texture decomposition algorithm was investigated to address the problem of illumination-variant texture synthesis.

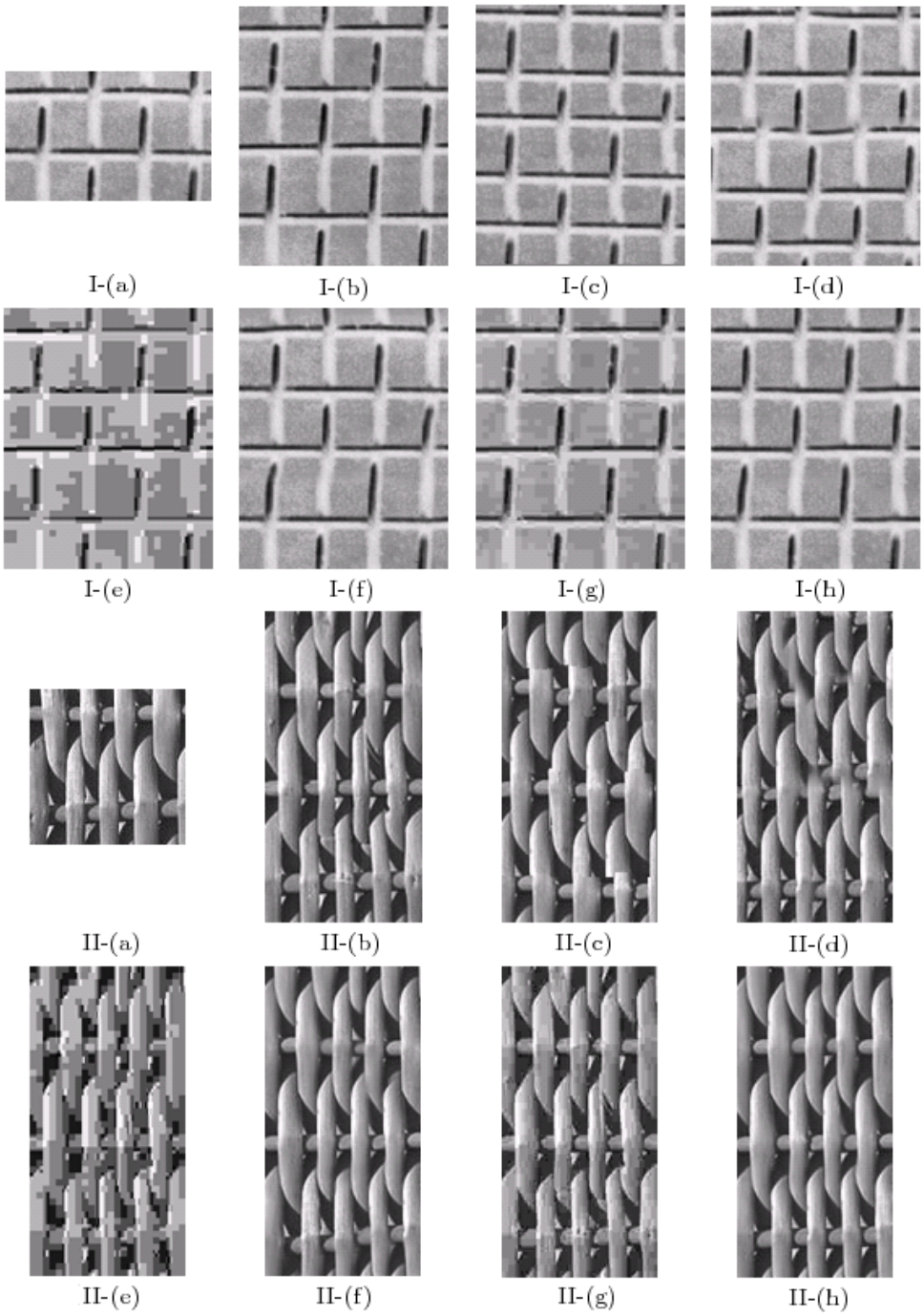


Figure 3.5: Texture image synthesis results for (I) the block image and (II) the straw image: (a) decoded seed image with $QP = 20$, (b) target image to be synthesized, (c) synthesized texture by [48], (d) synthesized texture without the side information, (f) synthesized texture with the decoded side information ($QP = 50$) as given (e), (h) synthesized texture with the decoded side information ($QP = 40$) as given (g).

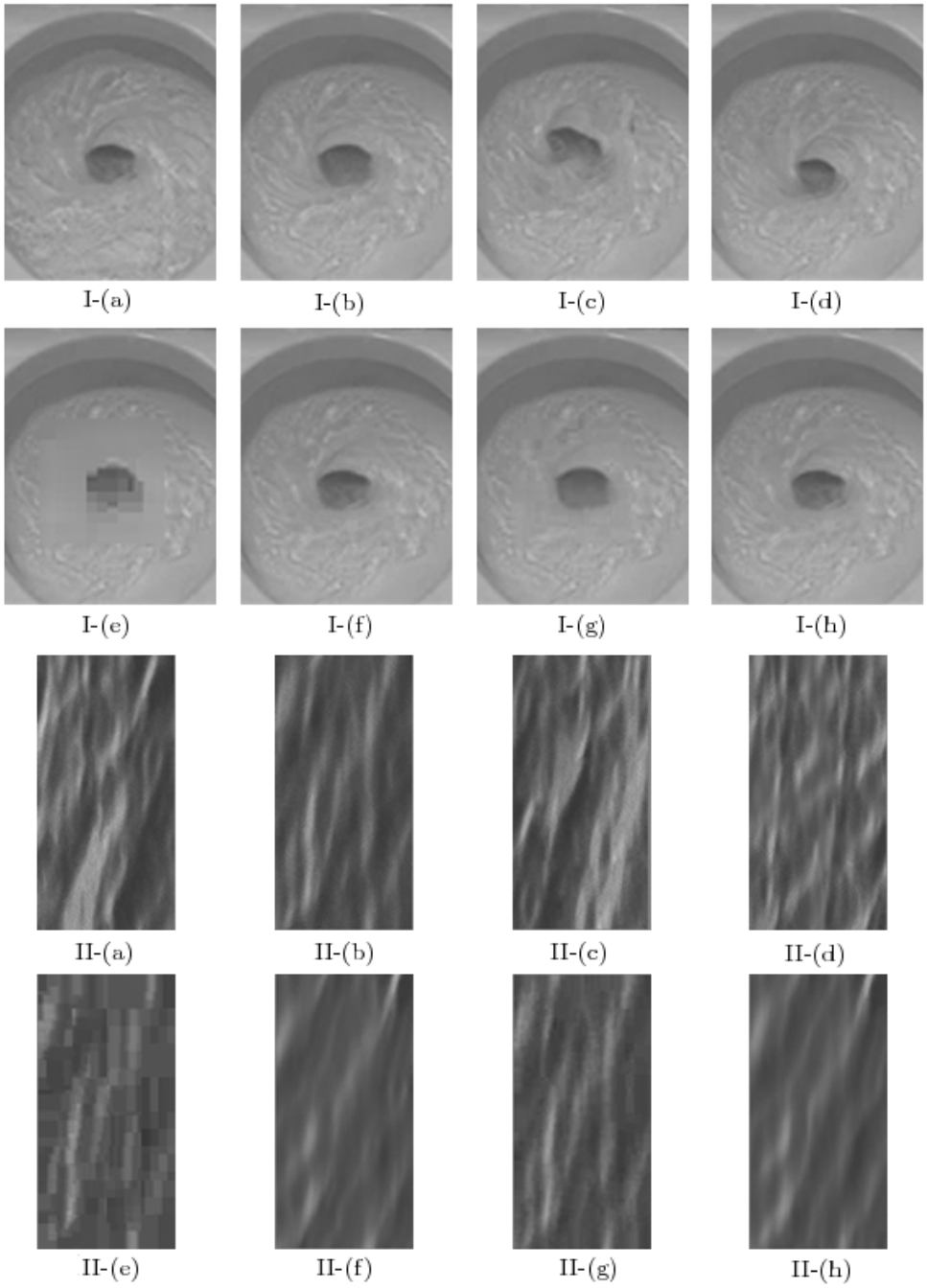


Figure 3.6: Texture video synthesis results for (I) the toilet sequence and (II) the duck-take-off sequence: (a) decoded seed sequence by $QP = 20$, (b) target sequence to be synthesized, (c) synthesized texture by [48], (d) synthesized texture without the side information, (f) synthesized texture with the decoded side information ($QP = 40$) as given (e), (h) synthesized texture with the decoded side information ($QP = 30$) as given (g).

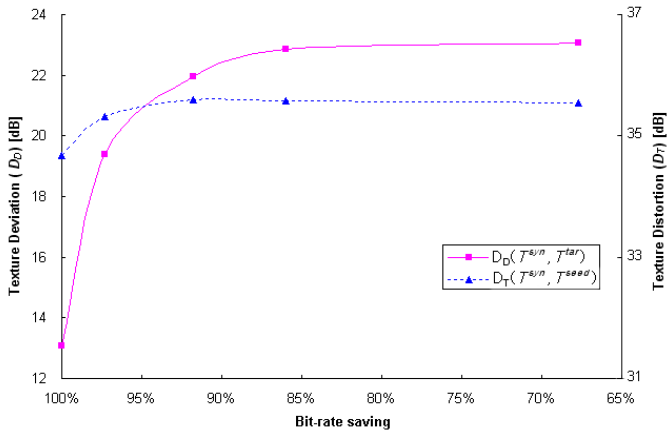


Figure 3.7: Plots of texture deviation, D_D, and texture distortion, D_T versus the bit-rate saving for the 'block' image.

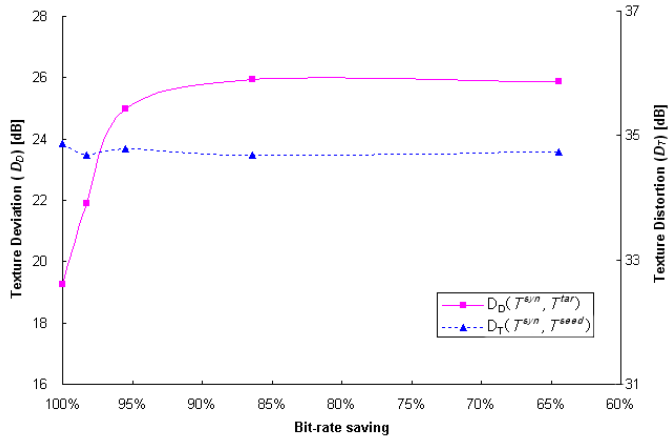


Figure 3.8: Plots of texture deviation, D_D, and texture distortion, D_T versus the bit-rate saving for the 'duck-take-off' sequence.

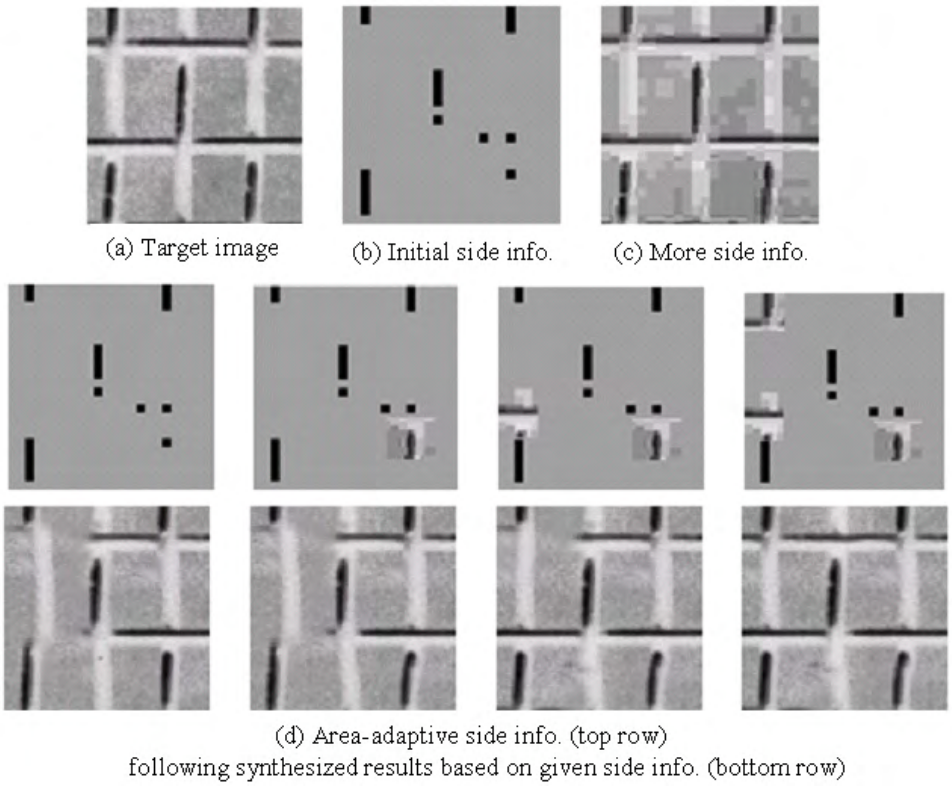


Figure 3.9: Texture synthesized results by varying the side information amount.

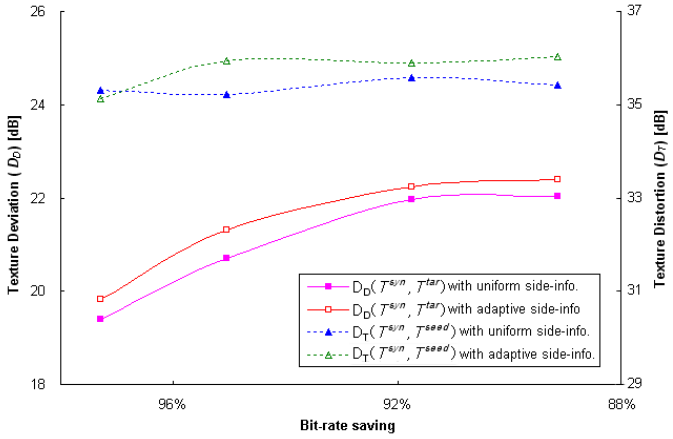


Figure 3.10: The improvement of texture similarity and distortion by area-adaptive side information algorithm for the 'block' image.

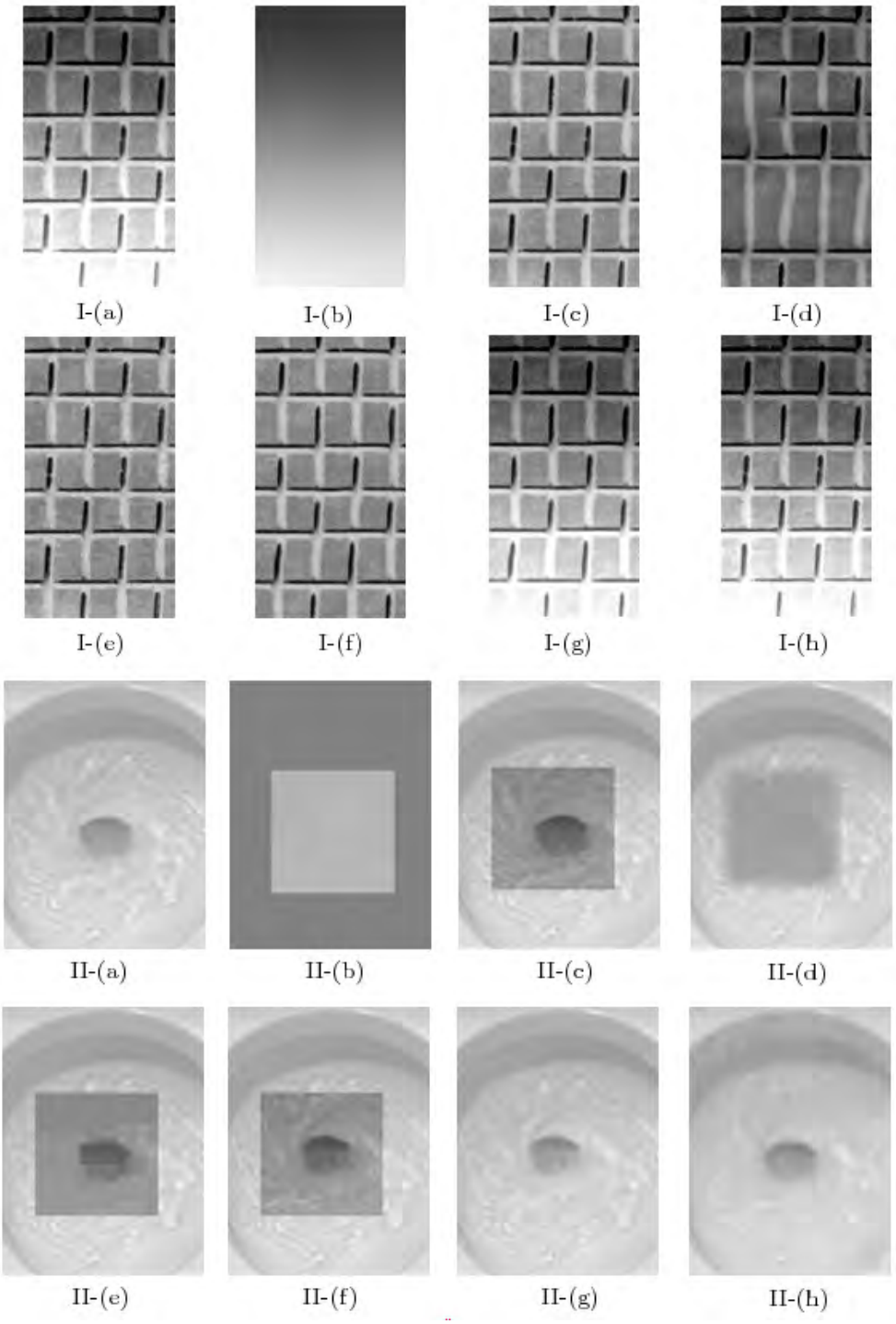


Figure 3.11: Illumination-variant texture synthesis results for (I) the block image and (II) the toilet sequence: (a) the original Illumination-variant texture, (b) the decomposed non-texture (NT) component, (c) the decomposed texture (T) component, (d) synthesized texture without decomposition, (f) synthesized texture with the decoded side information (QP=40) as given (e), (g) final results summed by decoded NT (QP=20) and synthesized T as given in (f), (h) decoded texture with same bit-rate as (g).

Chapter 4

Film Grain Noise Analysis and Synthesis for High Definition Video Coding

4.1 Introduction

Film grain noise in motion pictures is caused by the developing process of silver-halide crystals dispersed in photographic emulsion [67]. It is unavoidable in the analog film due to the physical process. When we digitize and compress high resolution movie contents obtained by scanning the analog film, such randomly distributed film grain noise is a major burden to typical video coding methods. Since film grain noise has a relatively large energy level in the high frequency region, it is more expensive to encode in the DCT domain. Besides, the underlying video suffers from inaccurate motion estimation. A natural idea to overcome this problem is to remove film grain noise as much as possible as a pre-processing step at the encoder so as to achieve a higher coding gain for denoised video [26, 27, 65, 74].

A lot of efforts have been done for Gaussian noise detection and removal. Under the assumption that film grain noise is one of the Gaussian additive or multiplicative noise,

many existing denoising methods could be used. However, this assumption does not hold since film grain noise has the following distinctive properties [26, 27, 51, 67, 84].

1. It is temporally independent.
2. Its power spectrum density is close to pink noise.
3. It is spatially dependent.
4. It has strong cross-color correlation in the RGB domain.
5. Its histogram is close to the Gaussian distribution.
6. It is dependent on the signal intensity.

Because of these properties, we need to develop a specific algorithm for film grain noise detection, modeling and removal.

To remove the film grain noise from the original video is however not enough. As high resolution devices such as HDTV are getting popular, film grain noise becomes perceptually important to human eyes since noise-removed video tends to bring an unnatural feeling to people. As a result, we should reproduce and render film grain noise at the decoder. Previous work following this line of thought will be reviewed in Sec. 4.2. In this work, we consider a novel implementation of this approach.

Specifically, we present a method to remove film grain noise from general input video without distorting the underlying video content. In the denoising process at the encoder, we adopt a method based on the principle of *total variation minimization* for film grain noise removal. It suppresses film grain noise effectively so that the video coding gain can be significantly increased. It is important to preserve the quality of the original video content as much as possible. Otherwise, it would lead to false extraction of film grain noise

and, consequently, inaccurate noise parameters. To achieve this goal, we detect edges or fine-texture regions of input video in advance and extract (or remove) noise in smooth homogenous regions only. In the meantime, we analyze film grain noise using a parametric model and determine its parameters. Based on the model and its parameters, artificial noise (which is close to the extracted one) is generated and added back to the decoded video at the decoder. Furthermore, we provide a method to measure the performance of film grain synthesis by comparing the statistical information of distinctive properties of film grain noise between the synthesized and extracted noise.

The rest of this chapter is organized as follows. The overall structure of the proposed scheme with previous work on film grain noise removal and modeling is briefly reviewed in Sec. 4.2. Then, algorithms for noise removal and synthesis are detailed in Sec. 4.3 and Sec. 4.4, respectively. Experimental results are provided in Sec. 4.5 to demonstrate the effectiveness of the proposed scheme with several performance metrics. Finally, concluding remarks are given in Sec. 4.6.

4.2 Review of Previous Work

Generally speaking, the film grain noise modeling scheme for video coding consists of two parts: 1) noise removal and extraction at the encoder, and 2) noise synthesis at the decoder as shown in Fig. 4.1. These can be viewed as the pre-processing and the post-processing steps in video coding. It is worthwhile to emphasize that it does not modify the encoding and decoding modules in any adopted video coding standard. The only additional information to be transmitted is noise parameters, with which noise can

be synthesized. Since a small set of parameters can represent the whole image, or a set of images in a GOP, the overhead of parameter transmission is negligible. One method for parameter transmission was proposed by Gomila *et al.* [26, 27] using the so-called SEI messages, with which no auxiliary transmission channel is needed. The detailed description of these two parts with the proposed algorithms will be given in Sec. 4.3 for noise removal and Sec. 4.4 for noise synthesis, respectively.

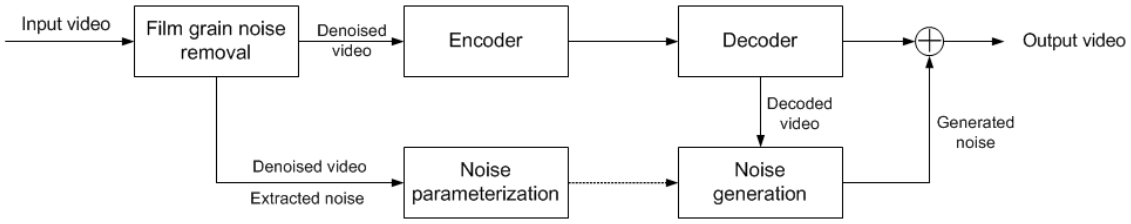


Figure 4.1: Overview of a film grain noise processing system.

The film grain noise modeling scheme for video coding was first proposed by Gomila *et al.* [26, 27], where noise removal was adopted as a pre-processing step in video encoding and noise synthesis as a post-processing step in video decoding. This idea has been standardized by AVC [74], and even deployed in commercial products for HD DVD in [77]. However, there is some room for further improvement. First, they did not provide any specific denoising method for film grain noise. As will be described later in this section, most of previous denoising schemes aiming to the conventional Gaussian additive noise is not sufficient to suppress film grain noise efficiently. One approach proposed in [27] uses the reconstructed video as its denoised version, which is attractable since it does not need an additional denoising module. However, it is observed that residual

images usually contain image edges and some structure information besides noise. Poorly extracted film grain noise would lead to false estimation of noise parameters.

Finally, it is worthwhile to mention that a similar idea was used for speech coding [3], where inactive voice signal is pre-processed before encoding, and noise is added back to the decoded signal for the comfort of human perception.

4.3 Film Grain Noise Removal

For film grain noise removal, we use the total variation minimization method to suppress film grain noise. Since a denoising process might distort areas that have sharp transition between neighboring pixels in the original image, it is important to identify regions of image edges before applying the denoising algorithm. Then, we can perform denoising selectively in smooth regions only. Moreover, the denoising process based on the total variation minimization principle could be more complete with some prior information of noise. Here, we propose to use the independence of film grain in the temporal domain to identify prior noise information. The overall pre-processing task at the encoder can be divided into three steps : 1) extract noise characteristics using the temporal information, 2) identify smooth regions of the image, and 3) denoise each image with prior noise information. The pre-processing module is shown in Fig. 4.2, and each processing step will be detailed below.

4.3.1 Extraction of Noise Characteristics

It is important to identify the accurate film grain characteristics, since the proposed denoising algorithm is conducted using the noise prior. Generally, it is non-trivial to

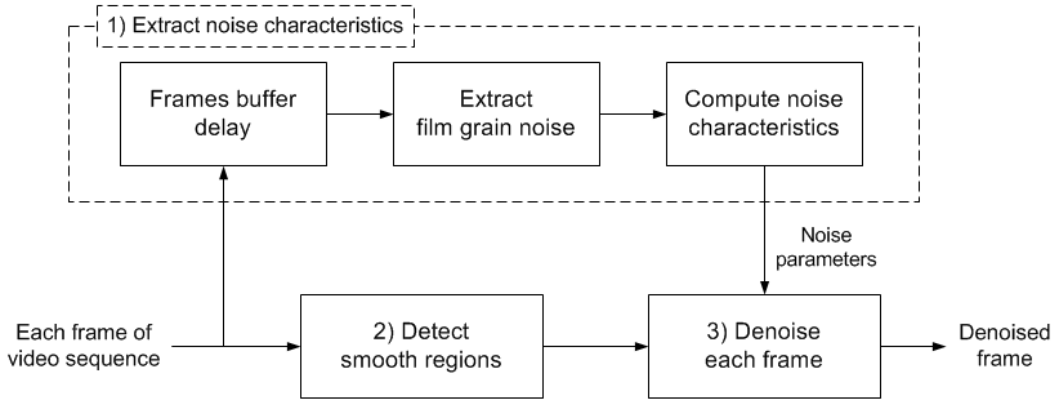


Figure 4.2: The block diagram of the pre-processing task.

identify them in that the film grain noise is not given in this stage. However, what we want to obtain is just statistical properties of film grain noise, and it can be indirectly exploited without computing film grain itself.

To obtain film grain characteristics, we assume an additive model of film grain noise of the following form:

$$U_o^n = U^n + N^n(U^n), \tag{4.1}$$

where U_o is the observed image, U is the original image and N is the film grain noise, and superscript n denotes the frame index. Since film grain noise is signal dependent, noise $N(\cdot)$ is a function of image U. If blocks of two consecutive frames are static, we can find the differential noise, N_d, as

$$\begin{aligned} N_d^n &= N^n - N^{n-1} \\ &= (U_o^n - U^n) - (U_o^{n-1} - U^{n-1}) \\ &\approx U_o^n - U_o^{n-1}, \end{aligned} \tag{4.2}$$

since $U^n \approx U^{n-1}$. The static area is identified by analyzing motion vectors of a set of non-overlapping blocks, where motion vectors are computed in noise suppressed images as done in [18].

Since film grain noise is almost temporally independent and spatially Gaussian distributed as mentioned in Sec. 4.1, differential noise N_d is another Gaussian noise with its variance twice as large. Then, all statistical values of film grain noise, such as variance, auto-correlation and cross-color correlations can be easily obtained from temporally differential noise N_d. It is also worthwhile to point out that the extracted noise characteristics remain stable for a long sequence of image frames. In practice, they can be use for the whole video sequence or, at least, for a large number of consecutive image frames. In other words, it is unnecessary to estimate the noise parameters frequently. Furthermore, as compared to other previous work that performs filtering along the temporal direction directly, the fact that only static areas of consecutive frames are considered in noise extraction makes our algorithm more robust in the presence of object motion.

4.3.2 Enhanced Edge Detection

For input image sequences that contain film grain noise, simple edge detection methods such as the Sobel or the Prewitt filter does not work well since these filters are sensitive to local noise. Note that we need to identify the edge region of noisy image, so that some additional process is necessary to suppress noise as much as possible to facilitate edge detection. In this manner, an enhanced edge detection method using multi-resolution filters is described below.

To extract edges from the input image effectively, we consider a set of filters to maximize frequency selectivity. These filters are built upon a pair of low- and high-pass filters as

$$h = \frac{1}{8}[-1 \;\; 2 \;\; 6 \;\; 2 \;\; -1], \quad g_1 = \frac{1}{2}[1 \;\; 0 \;\; -1]. \tag{4.3}$$

Then, we can construct three filters by $f_1 = h * g_1$, $f_2 = h * g_2$, $f_3 = h * g_3$ accordingly, where $*$ is the convolution operation and g_2 and g_3 are the upsampled filters of g_1 as

$$g_2 = \frac{1}{2}[1 \;\; 0 \;\; 0 \;\; 0 \;\; -1], \quad g_3 = \frac{1}{2}[1 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; -1].$$

These filters are applied along the horizontal as well as the vertical directions to detect edges of all possible orientations. This process is similar to the overcomplete wavelet decomposition. In [51], we proposed to use a multi-resolution overcomplete wavelet decomposition with simple integer-valued low- and high-pass filters. However, we found that its frequency selectivity is not good enough so that it could miss some signal with specific frequency bands. Instead, these filters are chosen to improve frequency selectivity while keeping integer-valued filter coefficients.

By following the terminology used in wavelet theory, the application of filter f_i, $i = 1, 2, 3$, horizontally and vertically can generate LH_i and HL_i output images. Then, the edge energy map is calculated by

$$EE_i = (\, |LH_i|^p + |HL_i|^p \,)^{1/p}, \quad i = 1, 2, 3, \tag{4.4}$$

and the unified edge energy map is obtained by

$$EE = \max[\ EE_1,\ EE_2,\ EE_3\ ], \tag{4.5}$$

where the maximum operation is performed pixel-by-pixel. Then, the binary edge map (EM) is obtained by thresholding, *i.e.* if the value of $EE(i,j)$ at pixel position (i,j) is larger than a pre-determined threshold, it is set to an edge point.

Since the proposed noise extraction and removal algorithm depend on the edge map, it is critical to find a reliable edge threshold value. Recall that the edge threshold depends on the signal intensity due to the unique property of film grain noise. For example, 8-bit image should need 256 different threshold values according to its signal intensity. To the threshold value, we set the initial threshold to 3 for all signal intensities, and adaptively update them according to the noise level in smooth areas (*i.e.* where the edge map value is equal to 0). The update formula is given by

$$Th_{new}[L] = (1-w)\cdot Th_{old}[L] + w\cdot c\cdot EE, \tag{4.6}$$

where weighting coefficient w is set to a small number (*e.g.* 10^{-4}) to avoid abrupt change, L is the background signal luminance, coefficient c is a scaling factor used to adjust the input value, and EE is the current edge energy value obtained by the above-mentioned method. Note that the updating process is done pixel-by-pixel manner, *i.e.* each threshold value corresponding to pixel illumination L is a scaled mean of EE by c. It is observed that threshold values converge after 4-5 frames with the updating process

and this algorithm is not sensitive for initial threshold value. Under the assumption that film grain noise is Gaussian distributed, we can show analytically that $c = 2.5$ would help detect more than 99% of film grain noise. Details of the analysis and its experimental verification are given in Sec. 4.3.3.

In the implementation, we use the luminance channel to get the edge map rather than processing each RGB color channel individually, and quantize the signal pixel value $U(i, j)$ by step size $Q_s = 8$ via

$$L = floor(U(i, j)/Q_s). \tag{4.7}$$

As a result, a total of 32 different threshold values are used to determine the edge map.

In the last stage of edge detection, an additional post-processing module is added after thresholding to minimize the false detection of edge pixels. That is, all isolated pixels are detected and eliminated under the assumption that edge pixels are connected with each other. In spite of the post-processing, there might still be misclassified pixels. One possible example is that film grain noise may have a wider spatial dependency than edge detection kernels, by which misclassified pixels are likely to be connected. As a result of false decision, the overall coding gain would be degraded slightly, and the re-generated noise will be added to the original decoded noise as edge regions. However, regions with false decision are often small, and the overall coding performance will not be significantly affected.

4.3.3 Adaptive Threshold Selection

To determine threshold values discussed in Sec. 4.3.2, we have to find the distribution of film grain noise first. It is observed that the distribution of film grain noise is close to the Gaussian distribution. Furthermore, the distribution of film grain noise in each subband is also close to the Gaussian distribution, which is experimentally verified in Fig. 4.3(a) for the LH subband. A similar observation holds for other subbands. Note also that subbands LH and HL are independent of each other, since film grain noise is isotropic. In the following analysis, we assume that both LH and HL subbands have the normal distribution with zero mean and standard deviation σ.

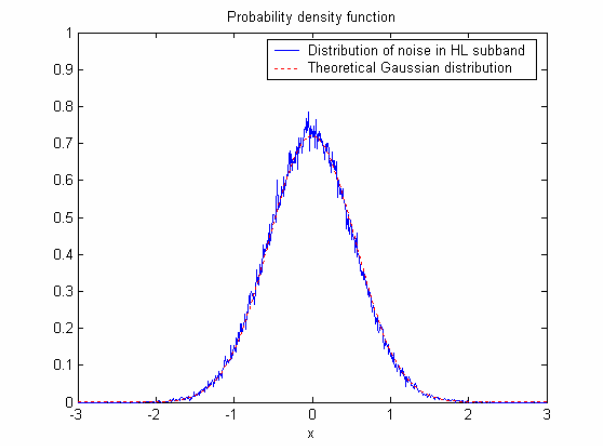
The distribution of the edge energy map of film grain noise can be derived based on the above assumptions. By choosing $p = 2$ in Eqn. (4.4), EE represents the Euclidian distance of the corresponding pixels in LH and HL subbands. It is known that EE has the following Rayleigh distribution

$$P(EE \leqslant t) = 1 - e^{-\frac{t^2}{2\sigma^2}}. \tag{4.8}$$

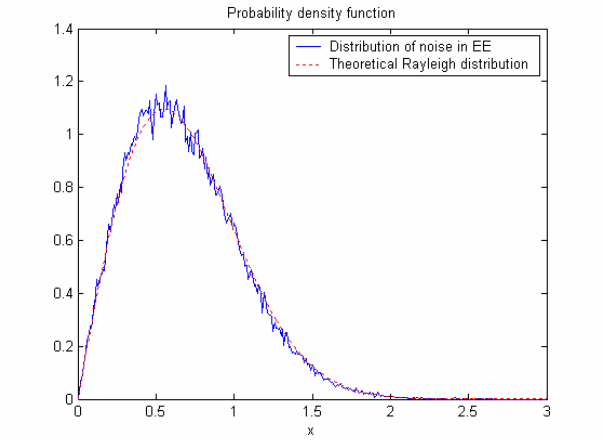
This is experimentally verified in Fig. 4.3(b). Thanks to the explicit distribution function, threshold value th can be determined by setting variable r as

$$\begin{aligned} P(EE \leqslant th) &= 1 - e^{-\frac{th^2}{2\sigma^2}} = r, \\ th &= \sqrt{-2\log(1-r)} \cdot \sigma. \end{aligned} \tag{4.9}$$

In the Rayleigh distribution, its mean is determined by σ, *i.e.* $m = \sigma\sqrt{\pi/2}$. Thus, the threshold value can be determined by Eqn. (4.10). For example, if we want to include



(a) Distribution of film grain noise in the LH subband



(b) Distribution of edge energy values in the LH subband

Figure 4.3: Distribution of film grain noise and its edge energy values in a typical subband, where the LH subband is used as an example.

exactly 99% of grain noise (*i.e.* $r = 0.99$), the scaling coefficient, c, in Eqn. (4.6) should be set to 2.42 due to the following relationship:

$$th = \sqrt{-2\log(1-r)} \cdot \sqrt{2/\pi} \cdot m. \tag{4.10}$$

As an alternative, we may use the median value instead of the mean value since the median is proven to be more robust with respect to outliers and/or noise. In the edge

map decision step, there must be false decision, which may lead to undesired perturbation of the threshold value. Under such a scenario, we may use the median to determine the threshold value. Being similar to the previous case, the median is also determined by σ, *i.e.*, $\text{med} = \sigma\sqrt{\log 4}$. Then, the threshold value can be determined as

$$th = \sqrt{-2\log(1-r)} \cdot \sqrt{1/\log 4} \cdot \text{med}. \tag{4.11}$$

It is worthwhile to point out that the median-based algorithm needs additional memory to store the signal distribution while the mean-based method can be performed without additional memory due to the dynamic updating process as specified in Eqn. (4.6).

4.3.4 Fine Texture Detection

When detecting smooth regions using the edge energy map for denoising, the main difficulty is that we often misclassify fine texture regions into smooth regions. It is desirable to treat fine texture regions the same as edge regions, since the fine texture pattern is perceptually visible when it has strong periodicity in the spatial domain in spite of its low edge energy. That is, we should not perform the denoising algorithm on them. However, since the edge energy of these texture pixels tends to be low, they may not be detectable by thresholding. To address this problem, we include the fine texture detection task as shown in Fig. 4.4, in which periodicity property of fine texture is explored for its detection. Finally, the non-smooth region is obtained by taking the union of detected edges and fine texture regions.

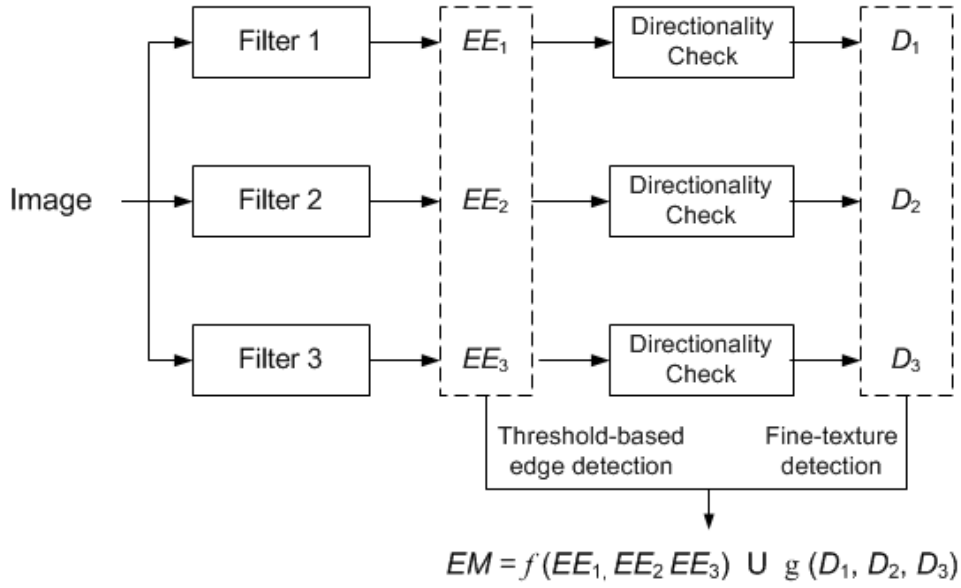


Figure 4.4: Detection of non-smooth regions in an image.

The fine texture detection process is illustrated in Fig. 4.5. First, the edge energy map is binarized by its local median to even identify low-intensity texture. Then, we check whether there is strong auto-correlation $R(m, n)$ along 8 different directions denoted by vector (m, n). In the discrete domain, we approximate the 8 directions by

$$(m, n) \in D = \{(1, 0), (2, 1), (1, 1), (1, 2), (0, 1), (-1, 2), (-1, 1), (-2, 1)\}.$$

Since film grain noise has no directionality, the maximum correlation value of film grain noise is smaller than that of the fine texture. However, there may be some false alarm if the correlation value of film grain noise becomes larger. To improve the robustness of our algorithm, we check the correlation value twice. That is, if $|R(p, q)|$ gives the maximum correlation value, we compute the correlation value for $|R(2p, 2q)|$, too. The final decision is made based on the product of $|R(p, q)|$ and $|R(2p, 2q)|$.

This procedure can be explained mathematically as follows. The correlation value of a pixel c can be written as

$$M_c = |R_c(p,q)| \cdot |R_c(2p,2q)| \quad \text{where } (p,q) = \underset{(m,n)\in D}{\arg\max} |R_c(m,n)|. \tag{4.12}$$

A pixel c will be determined as fine texture when it satisfies

$$M_c > \beta \cdot M_{N^d}, \tag{4.13}$$

where N^d is the differential noise obtained in Eqn. (4.2). This condition says that the threshold value for fine texture detection is directly dependent on the autocorrelation value of the estimated grain noise. In our implementation, we fix $\beta = 2$, which means a pixel is assumed as fine texture if its maximum correlation value is at least twice as large as the maximum correlation of film grain noise.

As discussed above, our approach relies on the periodicity of fine texture under the assumption that strong periodicity boosts the perception of fine texture. On the other hand, low-intensity non-periodic fine texture would be missed by the proposed algorithm. Spatially non-periodic but temporally continuous low-intensity texture should be handled by temporal-based filtering, which is out of our current scope.

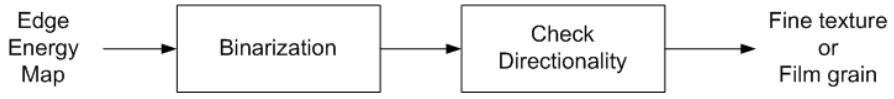


Figure 4.5: The process of fine texture detection.

4.3.5 Denoising with Total Variation (TV) Minimization

As mentioned in Sec. 4.1, film grain noise has many specific properties, and it is desirable to exploit these properties in noise removal. Traditionally, linear spatial smoothing filters such as the Gaussian filter have been widely used as the denoising filter. Their performance is however limited since a linear filter is not effective in detecting and extracting low-frequency energy of film grain noise. Here, we proposed to use a non-linear filtering approach based on the variational principle for film grain denoising, especially total variation (TV) minimization method. Note that the denoising process is only applied in smooth regions, which are detected by methods discussed in Secs. 4.3.2 and 4.3.4. For more references about the TV method, we refer to [8, 28, 40, 68].

4.3.5.1 Denoising with TV Minimization

The variational approach with TV minimization [40] is an effective non-linear denoising algorithm. It can preserve edges well while removing background noise. In addition, it can provide a complete solution if the noise prior is given, which is suitable in our current context. The algorithm is detailed below. For the additive noise model in Eqn. (4.1), we want to reconstruct the noise-free image U^n based on observed noisy image U_o^n. Then, the solution of the ill-posed inverse problem can be obtained by solving the optimization problem:

$$\min_U \int_\Omega |\nabla U| dU, \quad \text{s.t.} \quad ||U - U_o||^2 = \sigma^2, \tag{4.14}$$

where function $U : \Omega \rightarrow R$, Ω is a non-empty bounded open set in R^2, ∇ is the differential operator [28, 40], and superscript n is dropped for simplicity. In this case, noise is assumed to be white. By Lagrange's theorem, the best estimator can be written as

$$\begin{aligned} \widehat{U} &= \arg\min_{U} \left[\int_{\Omega} |\nabla U| dU \, + \, \frac{\lambda}{2} \, ||U - U_o||^2 \right] \\ &= \arg\min_{U} \left[\int_{\Omega} \sqrt{U_x^2 + U_y^2} du \, + \, \frac{\lambda}{2} \int_{\Omega} (U - U_o)^2 du \right]. \end{aligned} \tag{4.15}$$

To solve the above problem, the Euler-Lagrange differential equation is used as a necessary condition, and the update process is given by

$$\begin{aligned} U_t &= div(\frac{\nabla U}{|\nabla U|}) - \lambda(U - U_o) \\ &= \frac{\partial}{\partial x} \frac{U_x}{\sqrt{U_x^2 + U_y^2}} + \frac{\partial}{\partial y} \frac{U_y}{\sqrt{U_x^2 + U_y^2}} - \lambda(U - U_o), \end{aligned} \tag{4.16}$$

where λ is the Lagrangian multiplier, which is iteratively updated via

$$\lambda = \frac{1}{\sigma^2} \int_{\Omega} div(\frac{\nabla U}{|\nabla U|}) \, (U - U_o) \, dU. \tag{4.17}$$

4.3.5.2 Film Grain Denoising with TV Minimization

It is worthwhile to point out that the conventional TV-based denoising algorithm is not enough since it does not take the properties of film grain noise into account. Our main contribution in this work is to exploit the distinctive properties of film grain and incorporate them as a constraint in the denoising process. Specifically, Properties 2, 3 and 5 can be used in a single channel input image, *i.e.* the gray (or luminance) channel.

With assuming that film grain noise has Gaussian distribution by Property 5, we can find a coloring matrix P to estimate the spatial correlation of noise among neighborhood pixels by Property 2 and 3, such that

$$U - U_o = P\,w, \tag{4.18}$$

where P is a linear transform and w is white Gaussian noise, so that the spectrum of Pw matches with that of the extracted noise. Then, Eqn. (4.15) can be rewritten as

$$\widehat{U} = \arg\min_{U} \left[F(U) \; + \; \frac{\lambda}{2} (U - U_o)^T R^{-1} (U - U_o) \right], \tag{4.19}$$

where $R = PP^T$ is the auto-correlation matrix of (U-U_o). This whitening process helps estimate the noise behavior in the spatial domain, and it eventually leads to better noise suppression. To reduce the computational complexity, we approximate it by computing only 9×9 local blocks in the implementation.

If the input image is a color image of RGB three channels, we can use Property 4 (*i.e.* the cross-color correlation) furthermore to improve the noise estimation. We first find the G-channel noise data as the reference, and obtain R- and B-channel noise data based on the given extracted noise information of G-channel. For this case, we have two constraints at the same time so that the minimization process for B-channel can be modified as

$$\begin{aligned} \widehat{U}_B = \arg\min_{U_B} [& F(U_B) + \lambda_1 (U_B - U_{Bo})^T R_B^{-1} (U_B - U_{Bo}) \\ & + \lambda_2 (U_G - U_{Go})^T R_{GB}^{-1} (U_B - U_{Bo})]. \end{aligned} \tag{4.20}$$

where λ_1 and λ_2 are updated similarly to that in Eqn. (4.17).

4.4 Film Grain Noise Modeling and Synthesis

Film grain noise modeling and synthesis are discussed in this section. For noise modeling, a few parameters are determined to represent the extracted noise and transmitted to the noise synthesizer at the decoder.

4.4.1 AR Noise Model

There is no commonly agreed objective metric to measure the closeness of the synthesized and the real film grain noise. Thus, this is often done by visual inspection. As a result, film grain noise modeling is a challenging problem. There are several factors to be considered, including the spatial power spectrum density, the noise probability density, and the cross-color correlation as mentioned in Sec. 4.1

In this work, for film grain noise modeling, we consider the following AR model:

$$N(i,j,c) = \sum_{i'} \sum_{j'} \sum_{c'} a_{i'j'c'} \cdot N(i-i', j-j', c-c'), \tag{4.21}$$

which is a 3D AR model with the 2D spatial correlation and the 1D spectral correlation. Please note that the AR model is an IIR filter, which in general has a better frequency representation than a FIR filter. The power spectrum of synthesized noise can be controlled by the frequency response of the IIR filter with a white input signal. Furthermore,

the AR model as given in Eqn. (4.21) includes both the spatial and the cross-color correlation naturally. Generally speaking, the model can capture the desired properties of film grain noise well.

A set of AR parameters is obtained by following the Yule-Walker AR estimation method from the extracted noise at the encoder. Since film grain noise has the same characteristics over a sequence of image frames, a set of AR parameters is sufficient for the noise synthesis purpose. Besides, we only need a small number of coefficients for the AR model. Empirically, we choose values of (i', j', c') to be

$$(1,0,0),(0,1,0),(1,1,0),(-1,1,0),(2,0,0),(0,2,0),(0,0,1).$$

This choice results in a causal filter in the raster scanning order so that it is convenient for noise synthesis and the overhead of coding these parameters is very low.

4.4.2 Signal Dependent Noise Synthesis

The synthesized noise by the given AR model has no information about the background signal. Since film grain noise has the signal dependency property, we should modify the synthesized noise according to the decoded signal. Basically, scaling factors are obtained from the extracted noise like AR parameters, and both are transmitted to the decoder as the side information. Then, we can scale synthesized noise based on its background signal intensity according to the transmitted scaling factors. However, it is not easy to preserve the cross-color correlation by treating the signal-dependent noise directly. That is, if we scale the synthesized noise according to the background signal, it is likely to modify the

cross-color correlation as well. To preserve the cross-color correlation as much as possible in generating signal-dependent noise, the scaled excitation as shown Fig. 4.6 is adopted. That is, instead of scaling the synthesized film grain noise, we scale the excitation white signal before noise synthesis.

Film grain synthesis using the AR model was first proposed by Gomila *et al.* [27, 26] and standardized in AVC [74]. However, they did not fully utilize specific film grain properties in the AR model, *e.g.,* the signal dependent property as we do here. Moreover, they showed the advantage of the AR model only using a subjective approach. In contrast, we will show statistical properties of synthesized film grain noise in this work (see Sec. 4.5.2).

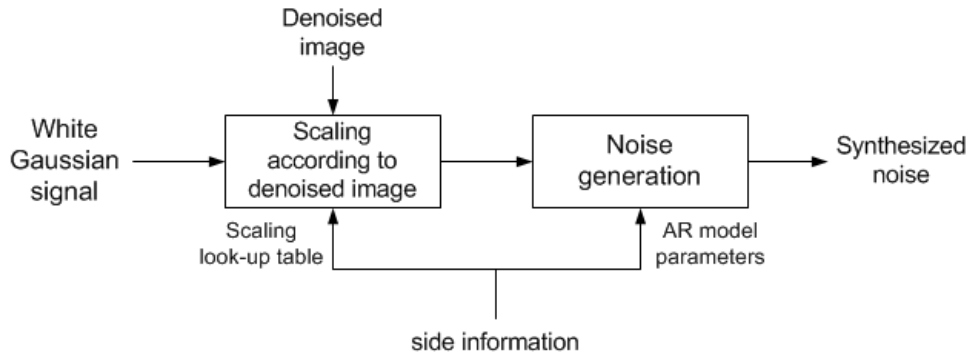


Figure 4.6: Film grain noise synthesis with scaled white noise.

4.4.3 Output Image Construction

The noise synthesizer generates film grain noise that has properties similar to the extracted one according to the procedures described above. Then, the final output image is obtained by

$$U_{out}(i,j) = U_{decod}(i,j) + N_{gen}(i,j), \tag{4.22}$$

where U_{decod} is the decoded image and N_{gen} is film grain noise generated by the AR model. Since the signal-dependent property has been considered in noise synthesis, a simple summation of the decoded image and synthesized noise is adopted in Eqn. (4.22).

It is worthwhile to mention that, since the film grain noise of edge areas is not removed at the encoder, to add synthesized noise to the edge areas of the decoded image as shown in Eqn. (4.22) could cause some problem. However, we observe that noise in edge regions is much less visible than noise in non-edge regions due to the masking effect so that this issue is negligible. Besides, the decoded image is a reconstruction of the smoothed version of the original image. Regardless of the compression method used, film grain noise in edge regions is actually suppressed during the encoding process.

4.5 Experimental Results

The film grain noise removal and synthesis processes can be integrated with any typical coding method. We use the JM(ver 11) software [75] of the H.264/AVC reference as the codec module in the experiment. We choose two RGB formatted color and one grey high definition (HD) video sequences as test sequences. Each sequence has 30 frames, and the first frames of sequences are shown in Fig. 4.7. Each sequence has different film grain noise characteristics and different type of contents. For example, 'Rolling tomatoes' sequence has lots of smooth regions, 'Playing cards' sequence mostly consists of textured region, and 'Old town cross' has both smoothed sky and textured buildings so that they provide a set of good test examples to evaluate our proposed scheme. For

more experimental results and visual performance evaluation, please visit http://www-scf.usc.edu/~byungoh/FGN/FGN.htm.

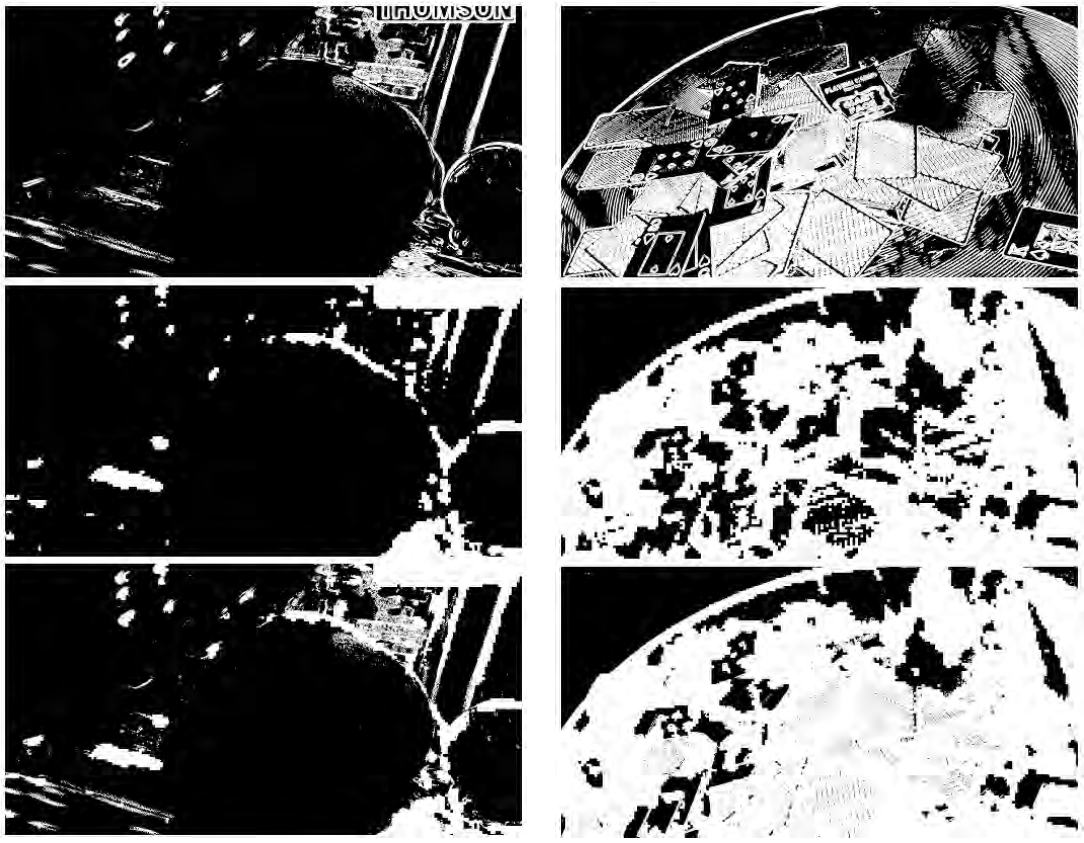
(a) Rolling tomatoes (b) Playing cards (c) Old town Cross

Figure 4.7: The first frames of HD (1920×1080) test sequences.

4.5.1 Smooth Region Detection and Denoising

Fig. 4.8 shows edge maps of the first frame of sequences, where the results of edges extraction and fine-texture extraction with the final edge map are given. They demonstrate that our algorithm can detect most of edge regions and fine texture regions successfully. Fig. 4.9 shows the close-up view of the denoised first frame of each sequence. It is worthwhile to mention that the proposed denoising algorithm is not sensitive to video contents or algorithm parameterization, since the TV minimization method automatically finds and updates the λ value, which is one of the main advantages as compared with other regularization methods.

To demonstrate the superior performance of the proposed denoising algorithm, we consider the power spectrum density (PSD) of extracted film grain noise. That is, we use the temporally extracted noise as the ground-truth and compare it with 1) Gaussian filtered noise, 2) spatio-temporal filtered noise [18], 3) noise extracted by the traditional TV algorithm [40] and 4) noise extracted by the proposed TV algorithm. The squared-roots of PSD of noise extracted by different algorithms, where the 2D power spectrum

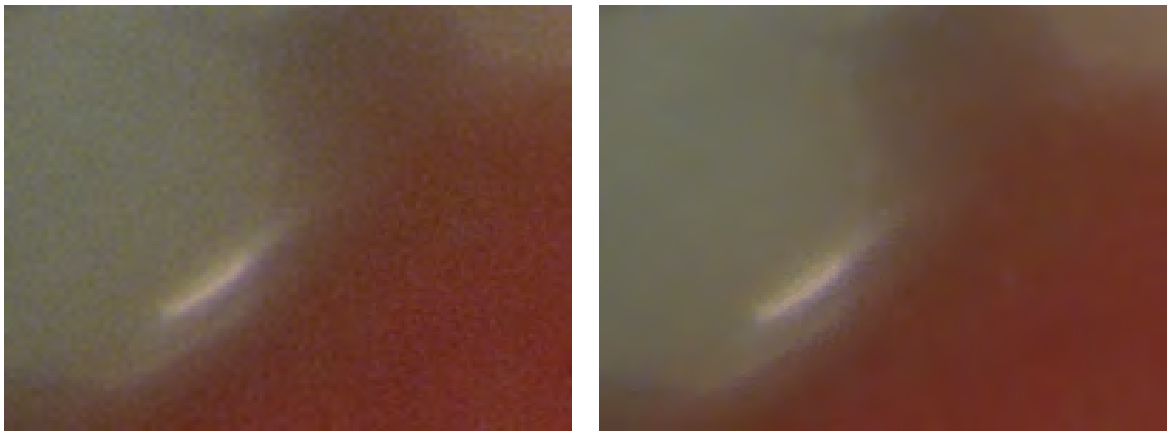


(a) Rolling tomatoes (b) Playing cards

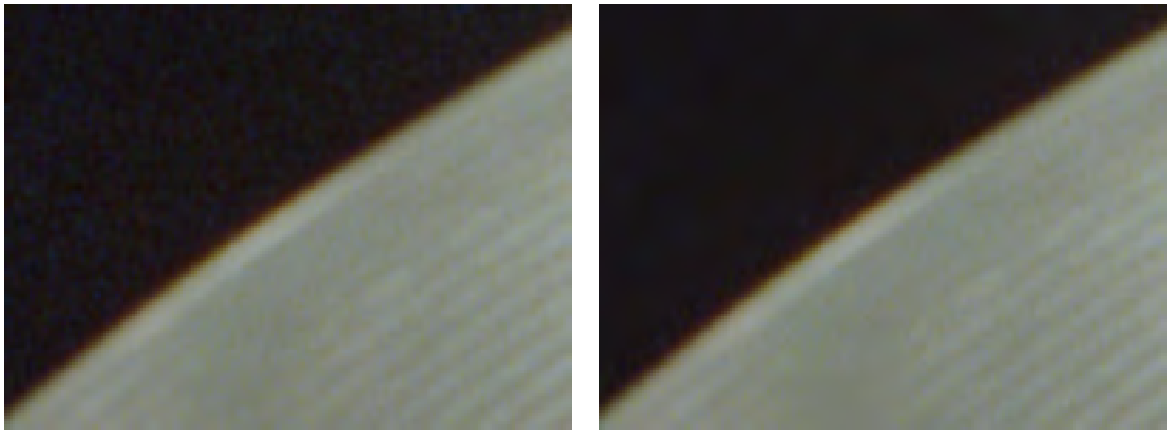
Figure 4.8: The edge-regions by threshold method (top), fine-texture region (middle) and final edge map (bottom).

density is projected to 1D, are shown in Fig. 4.10. We see that the TV minimization method outperforms the traditional smoothing filter and the spatio-temporal filter.

The superiority of the proposed TV algorithm can be stated below. First, the proposed TV method using spatial correlation and cross-color correlation detects film grain noise more accurately, especially for low-frequency energy of noise. Since most energy of film grain noise lies in the low-frequency band as mentioned in Sec. 4.3, the proposed TV method works well. Second, the power of extracted noise using the TV method is closest to the ground truth. To illustrate this point, we compare the power of noise extracted by



(a) Rolling tomatoes



(b) Playing cards

Figure 4.9: The close-up view of the original (left) and the denoised (right) images.

various algorithms in Table 4.1. To conclude, the proposed TV algorithm can efficiently suppress film grain noise and improve the coding gain, which will be discussed in Sec. 4.5.3

4.5.2 Film Grain Noise Synthesis

For synthesized noise evaluation, the conventional metrics such as the MSE or PSNR value are not useful. Here, we consider several criteria to evaluate the performance of different synthesis methods based on unique properties of film grain noise given in Sec.

Table 4.1: Comparison of extracted noise power by different algorithms.

		Temporally extracted	Spatio-temporal filter	Gaussian filter	general TV method	proposed TV method
Rolling tomatoes	G	5.10	3.95	3.75	4.56	4.89
	R	5.71	4.22	3.97	5.10	5.49
	B	16.72	7.44	5.98	13.50	15.28
Playing cards	G	9.86	4.97	4.40	6.74	6.80
	R	6.95	4.32	3.82	5.07	5.23
	B	38.20	19.56	18.23	28.56	29.64
Old town	Y	13.41	9.65	10.12	12.87	12.88

4.1. Out of the six properties, temporal independency and Gaussian distribution are automatically satisfied, since we use the i.i.d. Gaussian noise as the excitation signal. In the frequency domain, the power spectrum density determines the visual appearance of noise. Since the signal with stronger low-frequency components is more visible to human eyes while film grain noise has higher low-frequency components, it is important to re-synthesize noise to have a similar power spectrum density. Likewise, the spatial distribution of noise plays an important role for human perception in the spatial domain. In the RGB domain, the correlation between three color channels should be considered in noise generation. Even though pixels in the RGB domain have different values, the same film grain noise is physically created at each pixel and the cross-color correlation should be preserved. In addition, the background signal with a different intensity has different noise power, which is also perceptually visible. All these criteria will be considered and tested one by one in the following.

Among these criteria, the matching of cross-color correlation appears to be the most important one since it leads to intensity compensation between color channels. Due to this reason, we use the white signal scaling to preserve the cross-color correlation as mentioned

Table 4.2: Comparison of the cross-color correlation.

	Rolling tomatoes		Playing cards	
	extracted	synthesized	extracted	synthesized
G-R	0.85	0.92	0.21	0.25
G-B	0.95	0.97	0.80	1.23

in Sec. 4.4.3. The cross-color correlation values between the extract and synthesized noise are compared in Table 4.2. The signal dependency property is compared in Fig. 4.11 while the power spectrum density of the extracted and synthesized noise with the 7-coefficient AR model as described in Sec. 4.4.1 is compared in Fig. 4.12. As shown in these two figures, their power spectrum density plots are similar. We can make these curves closer to each other using a higher-order AR model or more complicated model such as ARMA model at the cost of higher complexity. Since the performance of a more complicated model is not significantly better, the 7-coefficient AR model is accurate enough as far as the power spectrum density is concerned.

Finally, we also evaluate the results by subjective quality testing method with Table. 4.3 as proposed in [81]. Based on the subjective quality rating, we selected 10 experts and 10 non-experts, and asked them to compare the original video with 1) conventionally decoded video with QP=24, and 2) output video by the proposed framework, *i.e.* denoised, decoded with QP=24 and noise added sequence. We show the subjective test results in Fig. 4.13, where each shaded bar and its middle dotted line represent the 95% confidence interval and its mean value, respectively.

Table 4.3: Quality rating on a 1 to 5 scale.

Rating	Impairment	Quality
5	Imperceptible	Excellent
4	Perceptible, not annoying	Good
3	Slightly annoying	Fair
2	Annoying	Poor
1	Very annoying	Bad

4.5.3 Coding Gain Improvement

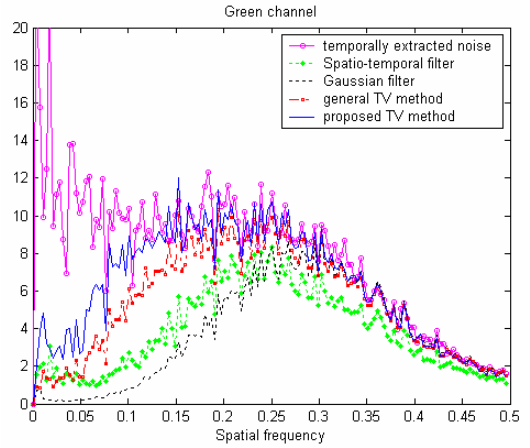
We show the bit-rate savings as a function of the quantization parameter (QP) in Fig. 4.14 for different denoising algorithms with similar output video quality. Here, we have tested 30 frames (1sec) for each case with one intra-coded (I) frame and 29 predicted coding (P) frames. In Fig. 4.14, we also provide the portion of smooth regions, since the bit-rate saving is directly affected by this portion. We see that the film grain denoising algorithm significantly reduces the coding bit-rate, especially for smaller QP values, and denoising with the proposed TV minimization method demands the lowest bit rate for the same QP.

Finally, we show parts of two image frames and their corresponding re-synthesized counterparts in Fig. 4.15 for visual performance comparison, where re-synthesized images are obtained by adding the synthesized film grain noise to decoded images with QP=24. Since the whole image is too big to reveal the advantage of the proposed algorithm, we only show the close-up views that cover the homogenous region (for the sequence of rolling tomatoes) and the edge region (for the sequence of playing cards), respectively.

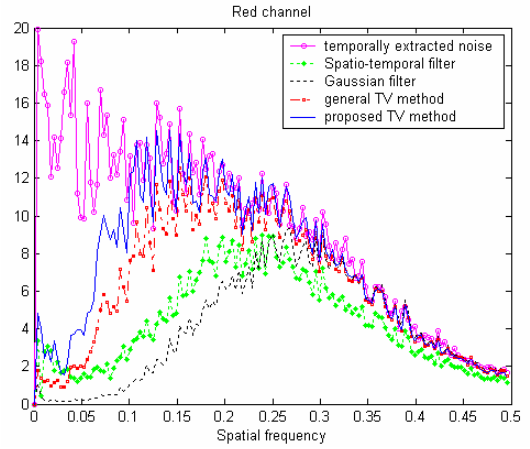
4.6 Conclusion and Discussion

A novel approach to HD video coding based on film grain noise extraction, modeling, resynthesis was presented. There are several important contributions in our proposed scheme. First, the edge-preserving denoising technique is used for film grain noise extraction. The edge-detection filters and the fine-texture detection algorithm allow us to separate non-smooth regions from smooth regions in an image. Second, the denoising method based on the modified total variation (TV) minimization method using specific film grain properties was designed to suppress film grain noise efficiently without distorting the original image. Third, noise characteristics are extracted by temporal difference, and they can be used in the denoising process. Last, a novel film grain noise synthesis algorithm is obtained by using the AR model excited by scaled white noise. It was shown that the coding gain is much improved by encoding the denoised image sequence. Furthermore, we show that the synthesized film grain noise is subjectively satisfactory to human eyes and argue that it is objectively similar to observed film grain noise in terms of its statistical properties.

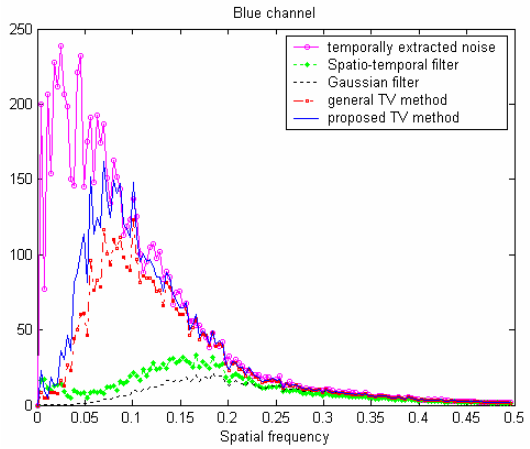
All time-consuming tasks are implemented at the encoder in our proposed algorithm. Only simple noise synthesis and addition is needed at the decoder. Thus, the additional complexity required by consumer electronic devices is negligible. Moreover, it demands only a small number of parameters per frame or per GOP as the overhead. As a result, the proposed film grain noise model can be easily added to the any current video coding standards.



(a) Green

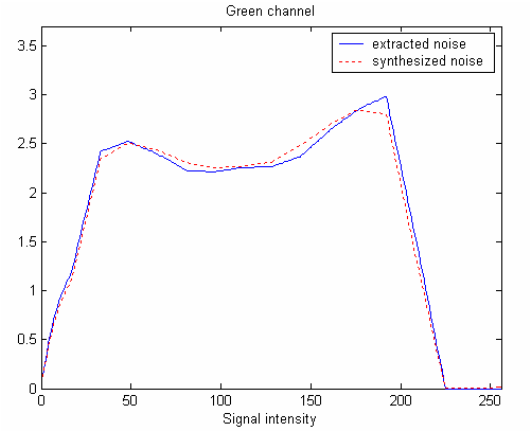


(b) Red

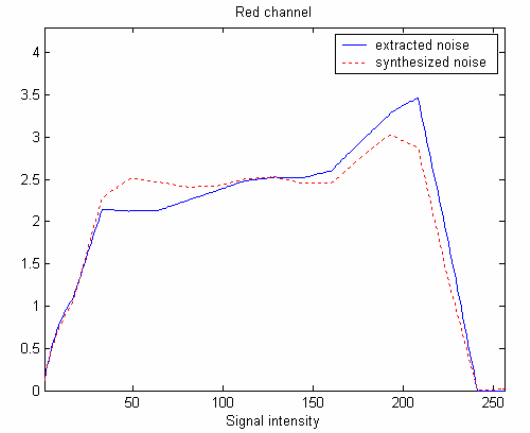


(c) Blue

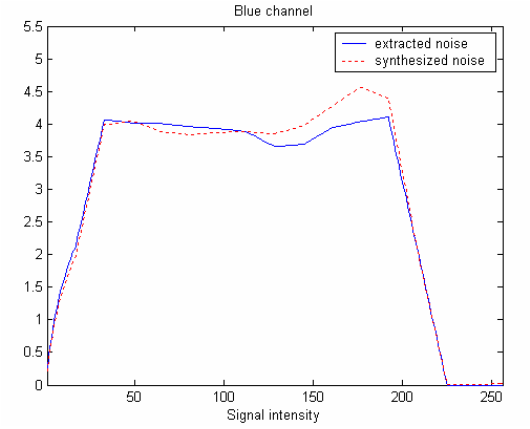
Figure 4.10: Comparison of the squared-root of the PSD for extracted noise using several algorithms.



(a) Green

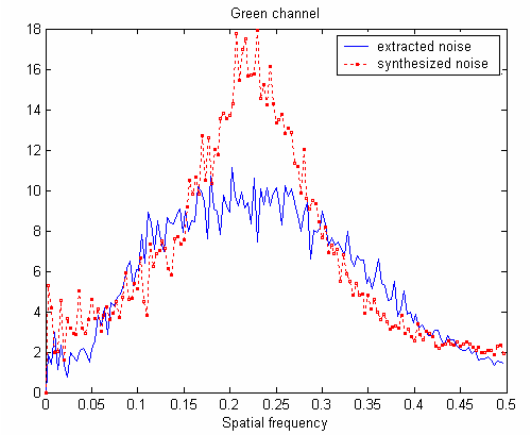


(b) Red

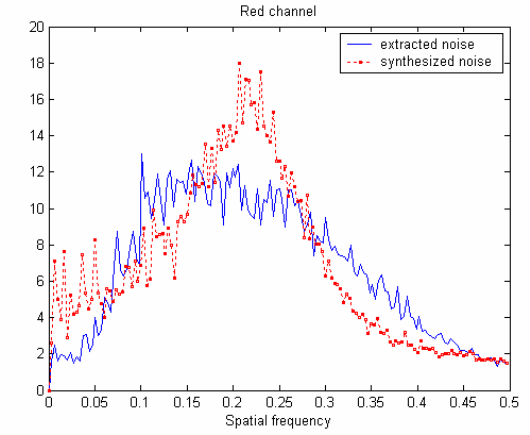


(c) Blue

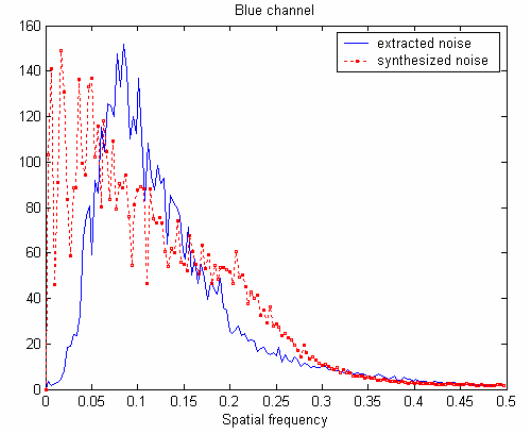
Figure 4.11: Comparison of signal dependency between extracted and synthesized noise for the rolling tomatoes sequence.



(a) Green



(b) Red



(c) Blue

Figure 4.12: Comparison of the square root of PSD between extracted and synthesized noise for the rolling tomatoes sequence.

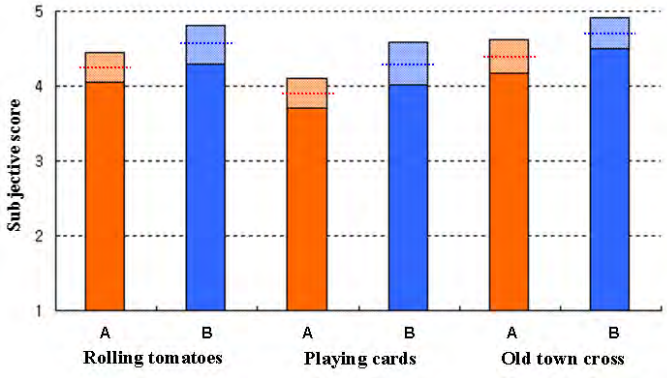
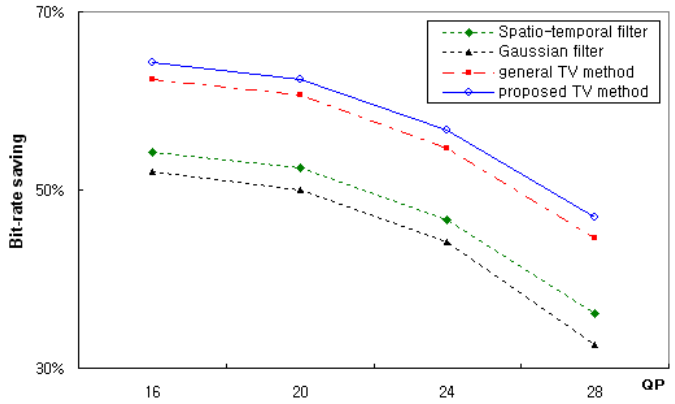
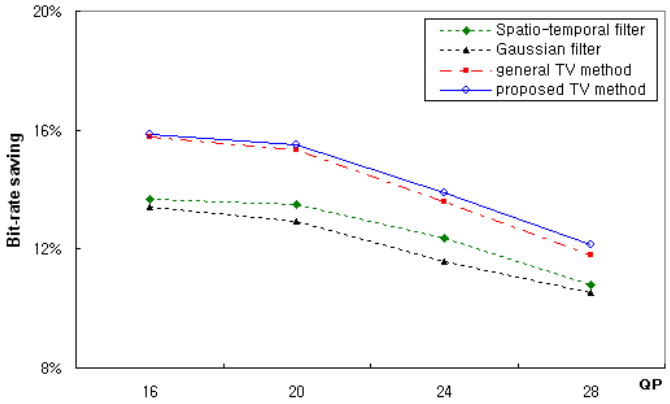


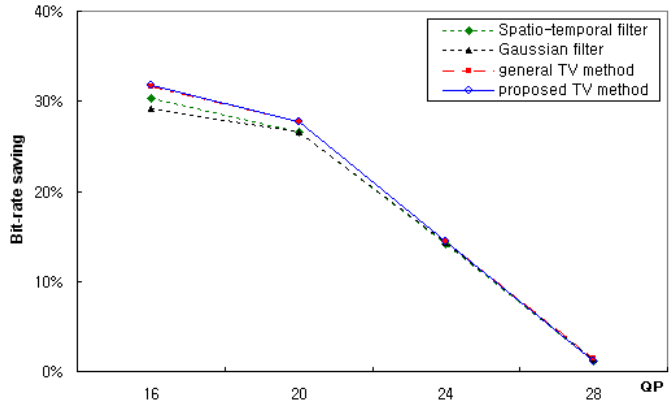
Figure 4.13: Comparison of subjective test results, where A denotes the coding result using the conventional H.264/AVC reference codes and B denotes the coding result using the proposed method.



(a) Rolling tomatoes, smooth regions : 72.5%

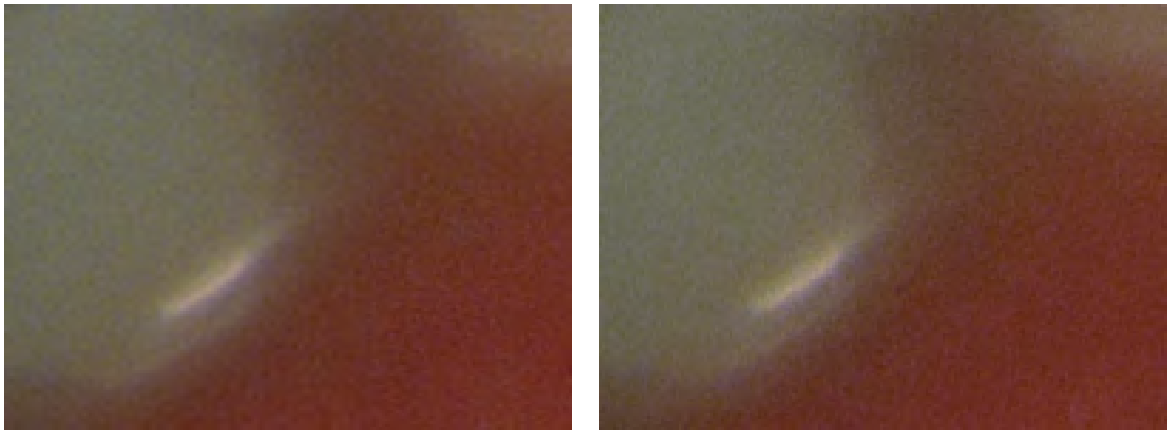


(b) Playing cards, smooth regions : 19.5%

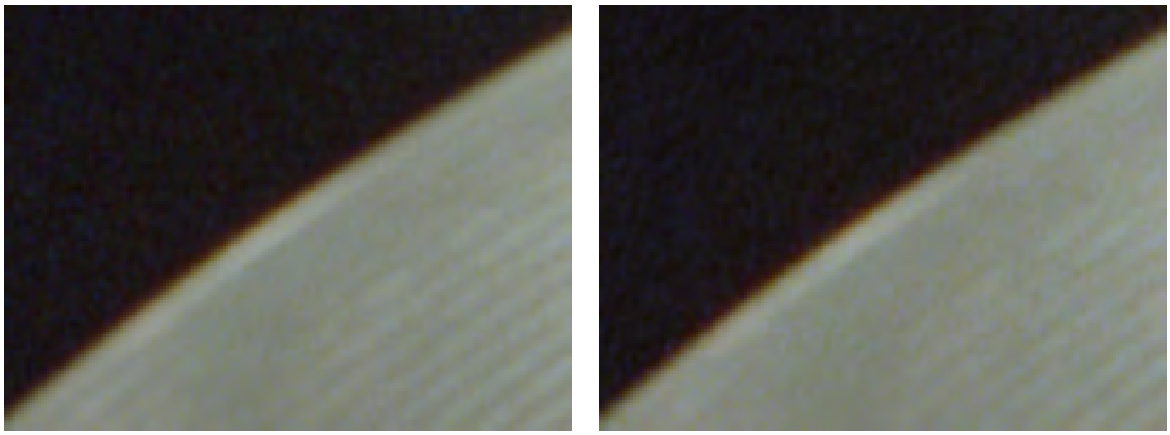


(c) Old Town Cross, smooth regions : 37.8%

Figure 4.14: The coding bit-rate savings comparison of different algorithms as a function of quantization parameters (QP).



(a) Rolling tomatoes



(b) Playing cards

Figure 4.15: Close-up of the original images (left) and the re-synthesized images (right).

Chapter 5

Super-Resolution of Stochastic Texture Image

5.1 Introduction

The task to generate a high-resolution image from a set of lower-resolution images is called image super-resolution. This problem has received a lot of attention for its enormous potential applications The well-known image interpolation (or upsampling) method can be viewed as one super-resolution technique. Image interpolation has been studied for about four decades. Polynomial interpolation algorithms, such as bilinear or bicubic interpolation methods, have been widely used due to their simplicity. However, they often yield blurred results around sharp edges and textured regions. A lot of efforts have been made to overcome blurring artifacts in image edges with a substantial amount of progress in recent years. However, an effective super-resolution technique for textured image is still a challenging issue.

Image super-resolution can be treated as an image restoration problem. However, most previous super-resolution algorithms focused on the processing in edge regions. Few of them targeted at texture processing, especially stochastic texture interpolation.

Edge-oriented super-resolution algorithms are not adequate for stochastic texture, since random texture is more complex to model and predict than image edges. The state-of-the-art non-local (NL) super-resolution algorithm does not produce satisfactory results for random texture, either. Since the block similarity measure of the conventional NL algorithm was conducted based on the pixel intensity difference, the upsampled image may contain artificial patterns and/or undesirable artifacts.

In this work, we propose a new image interpolation scheme, which is especially suitable for random texture. To interpolate a coarse-resolution input texture image, we adopt a piecewise auto-regressive (PAR) model and apply the NL strategy to estimate model parameters adaptively under the assumption that random texture is self-regular in model parameters. This new scheme, called the PAR/NL algorithm, can be viewed as a hybrid of the model-based (*i.e.,* PAR) approach and the learning-based (*i.e.,* NL) approach. In the proposed PAR/NL scheme, we first select a model structure and then model parameters adaptively using the learning-based approach with a self training set. Theoretical analysis is conducted to understand the consistency and robustness issues of the PAR/NL algorithm, and experimental results are given to demonstrate its performance.

The rest of this chapter is organized as follows. The basic NL algorithm for image denoising and interpolation is briefly reviewed in Sec. 5.2. Then, the new texture interpolation algorithm is proposed in Sec. 5.3. Experimental results are provided to demonstrate the performance of the proposed PAR/NL-based texture interpolation algorithm in Sec. 5.4. Finally, concluding remarks are given in Sec. 5.5.

5.2 Image Restoration with Non-Local (NL) Algorithm

We review the NL algorithm and its application to image restoration in this section. The objective of image restoration is to estimate the unknown original image, X, from an observed image, Y, in form of

$$Y = f(X) + N, \tag{5.1}$$

where f denotes an image distortion process and N is the noise image. For example, f can be a down-sampling operator for the image interpolation problem, or a blurring operator for the image deblurring problem. In the following sections, we will investigate the basic idea of the NL algorithm, and also provide its derivation and analysis for theoretic consistency.

5.2.1 Non-Local Means (NLM) Denoising Algorithm

The NL algorithm was first proposed to solve the image denoising problem [8]. If f is an identity operator, the image restoration problem is reduced to an image denoising problem as

$$Y = X + N. \tag{5.2}$$

The main idea of the NL-denoising algorithm is to restore a pixel with its local neighborhood as well as non-local pixels that are in a similar environment. Simply speaking, the target pixel will be obtained as a weighted average of local and non-local pixels, where the weight is computed based on the similarity between two image regions. That is, a

pixel surrounded by a more similar block will contribute more to the restoration of the target pixel than that surrounded by a less similar block. For this reason, NL-denoising algorithm is often called NL-means (NLM) algorithm.

Mathematically, the NLM algorithm to the solution of Eqn. (5.2) can be written as

$$\hat{X}(\underline{i}) = \frac{1}{Z_i} \sum_{j \in I} w_{i,j} Y(j), \tag{5.3}$$

where $w_{i,j}$ is a weight denoting the contribution from $Y(j)$ to $\hat{X}(i)$, Z_i is a normalization factor to make $\sum_j w_{i,j}/Z_i = 1$, and I is the image space. Specifically, an adaptive weight is determined by measuring the neighborhood distortion as

$$w_{i,j} = \exp\left(-\frac{D(Y(\mathcal{N}_i), Y(\mathcal{N}_j))}{h^2}\right), \tag{5.4}$$

where D is a distortion measure, $\mathcal{N}_i$ is a squared neighborhood of pixel $i = (i_1, i_2)$, and h is the filter coefficient that adjusts the decay of the weight. As shown above, the distortion measure directly affects the overall performance. In [8], it is determined by

$$D(Y(\mathcal{N}_i), Y(\mathcal{N}_j)) = || Y(\mathcal{N}_i) - Y(\mathcal{N}_j) ||_a^2, \tag{5.5}$$

where $a > 0$ is the standard deviation of the Gaussian kernel.

It is worthwhile to compare the NLM scheme with the well known bilateral filter [78], since the bilateral filter shares the same form as Eqn. (5.3) with a different definition of the weight factor as

$$\begin{aligned} w_{i,j} &= f_s(Y(i), Y(j)) \cdot f_c(i,j) \\ &= \exp\left(-\frac{||Y(i) - Y(j)||^2}{h_s^2}\right) \cdot \exp\left(-\frac{||i-j||^2}{h_c^2}\right), \end{aligned} \tag{5.6}$$

where f_s and f_c are functions used to compute the pixel-intensity similarity and the geometrical closeness between these two pixels, respectively. As an example, the Gaussian function of the Euclidean distance can be used for controlling parameters h_s and h_c. As given above, the bilateral filter could be viewed as one specific form of the NLM filter by taking the geometrical distance into account, and using the dirac delta kernel as the distortion measure (*i.e.* $a \to 0$ in Eqn. (5.5)). On the other hand, the NLM scheme does not consider the geometric closeness (*i.e.* $h_c \to \infty$ in Eqn. (5.6)), and measure the similarity based on its local information instead.

5.2.2 Derivation and Analysis of NLM

The NLM algorithm is fundamentally rooted on the 'block-wise regularity' assumption of images. This assumption has been shown experimentally to be valid for most common images. For two pixels surrounded by identical local neighborhoods, it is reasonable to

assume that the corresponding noise-free pixels will be identical as well. This idea can be mathematically formularized with the following objective function

$$J_{NLM} = \sum_{i \in I} \sum_{j \in I} w_{i,j} ||\hat{X}(i) - Y(j)||^2. \tag{5.7}$$

Then, the optimal solution of Eqn. (5.7) can be obtained by setting its derivative

$$\frac{\partial J_{NLM}}{\partial X(i)} = 2 \sum_{j \in I} w_{i,j} (\hat{X}(i) - Y(j)) \tag{5.8}$$

to zero. Then, we have

$$\hat{X(i)} = \frac{\sum_j w_{i,j} Y(j)}{\sum_j w_{i,j}}. \tag{5.9}$$

Actually, the above optimal solution is the same as that given in Eqn. (5.3).

Note that the objective function in Eq. (5.7) is generic and its adaptive weight values can be chosen based on a denoising filter kernel. For example, $w_{i,j}$ can be chosen using the isotropic Gaussian kernel which assigns a higher penalty value (*i.e.* a smaller weight) for far-away pixels by the Euclidean distance. This implies that a noise-free pixel would have the identical value with its local neighborhood, which is valid in the smooth background. In the sharp-edge region, the weight values should be changed according to the edge orientation to avoid the same weight across two different sides of an edge in spite of their geometric closeness. Likewise, the NLM filter is an adaptive filter which involves the non-local information (*i.e.* the size of the NLM kernel filter could be as large as the whole image).

To determine a good set of adaptive kernel weights is a challenging research issue. In the NLM scheme, it is obtained by measuring the block similarity. Although it would be more reasonable to measure the block distance between two noise-free blocks, we have no choice but measure the block distance between two noisy blocks. On the other hand, we can derive the following relationship under the *i.i.d.* white Gaussian noise assumption:

$$\begin{aligned} E||Y(\mathcal{N}_i) - Y(\mathcal{N}_j)||_a^2 &= E||(X(\mathcal{N}_i) - X(\mathcal{N}_j)) + (N(\mathcal{N}_i) - N(\mathcal{N}_j))||_a^2 \\ &= E||X(\mathcal{N}_i) - X(\mathcal{N}_j)||_a^2 + 2\sigma_N^2, \end{aligned} \tag{5.10}$$

where σ_N is the standard deviation of *i.i.d* white noise N. Actually, the impact of noise will be statistically cancelled out.

5.2.3 Adaptation for Image Super-Resolution

Although the NL algorithm was initially developed for image denoising, the same strategy can be extended to image super-resolution. The mathematical form for super-resolution can be expressed as [55]

$$Y_k = g_m(f_k(X)) + N_k, \quad k = 1, 2, ..., K, \tag{5.11}$$

where $f_k(\cdot)$ is a function of geometrical warping and blurring and $g_m(\cdot)$ is an m-to-1 down-sampling operation.

It is intuitive that more observations ($K \gg 1$) yield better results since one can predict the unknown pixel value with a large number of observations. Inaccuracy caused by noise, imperfect models, or false estimation can be compensated by these observations.

On the other hand, for a smaller K value, the high-resolution image is mostly estimated based on its prior model such as piece-wise smoothness. When $K = 1$, the accuracy of the prior model is critical, since the resultant image is only determined by its prior model. In this sense, the NL strategy is advantageous by assuming that the local information will be repeated in somewhere else to increase the value of K. The above discussion is particularly true if a given image has strong self-regularity such as tiles, blocks, windows, etc.

The adaptation of the NLM algorithm to the interpolation problem is straightforward. That is, it initializes the high-resolution image by an arbitrary interpolation algorithm, and then assumes the initially interpolated image is a noisy version of the ground-truth. Then, we can follow the same restoration procedure as described above. Mathematically, the weight can be chosen as

$$w_{i,j} = \exp\left(-\frac{D(Y_u(\mathcal{N}_i), Y_u(\mathcal{N}_j))}{h^2}\right), \tag{5.12}$$

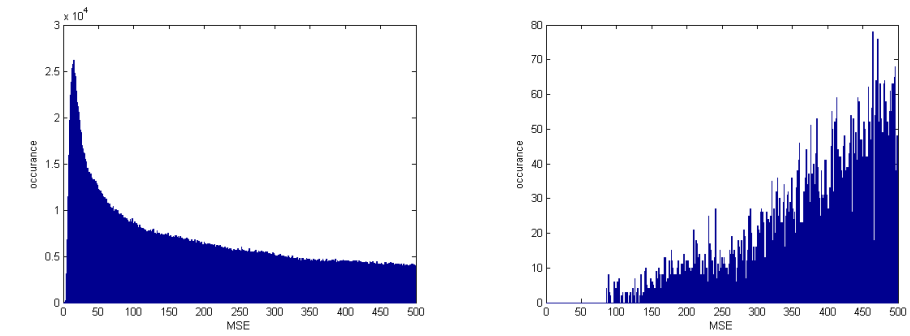
where subscript u represents the intermediate upsampled image that matches the desired resolution. A more generalized version of NL-based interpolation was formulated and the corresponding super-resolution methods were proposed in [60].

5.3 PAR/NL Texture Interpolation Algorithm

In this section, we will develop an interpolation algorithm for random texture such as sand, wool, fabric, etc. Note that there is another type of texture called structured texture

with bricks and tiles as examples. Structured texture is not considered here, since it can be well handled by the previously developed edge-oriented interpolation algorithms.

Due to the stochastic property of random texture, the application of the NL-based interpolation algorithm described above does not yield satisfactory results. Generally speaking, the NL-based interpolation using the NLM scheme strongly relies on the 'self-regularity' in the pixel domain as defined in Eqn. (5.5), which is however not valid for random texture. The following experiment shows the difference between these two cases clearly. First, we randomly select blocks from a common image ('Lena') and the random texture image ('Food #5'), compute the MSE of samples in each block, and then plot the MSE histogram in Fig. 5.1. As shown in the figure, the pixel-wise similarity does not hold for random texture.



(a) Lena (b) Food #5

Figure 5.1: Comparison of MSE histograms of arbitrarily chosen blocks.

To overcome this limitation, we use the NL algorithm to estimate and refine model parameters, which is the main contribution of this research. To be more specific, the conventional NL-based interpolation estimates unknown pixel values, $\hat{X}(i)$, by exploring other non-local similar image patches to maximize probability $P(\hat{X}(i)|X(\mathcal{N}_i))$. In

contrast, the proposed algorithm estimates the hidden model parameter set, $\hat{\theta}$, by maximizing probability $P(\hat{\theta}_i|X(\mathcal{N}_i))$. This idea is rooted on the assumption that the model parameters that govern the random texture behavior are self-regular.

In the following subsections, we first introduce the piece-wise auto-regressive (PAR) model as the selected linear model, and then show the detailed algorithm for the objective function and distortion measurement with its theoretic analysis next.

5.3.1 Piecewise Auto-regressive (PAR) Model

For random texture interpolation, we adopt the PAR model in form of

$$\begin{aligned} \hat{X}(2i_1+1, 2i_2+1) &= \sum_{k\in K} \theta_i(k_1,k_2) X(2(i_1+k_1), 2(i_2+k_2)) \\ &= \sum_{k\in K} \theta_i(k_1,k_2) Y(i_1+k_1, i_2+k_2), \end{aligned} \tag{5.13}$$

where θ_i is the adaptive interpolation coefficients at pixel i, Y and X are $M \times N$ low-resolution and $2M \times 2N$ high-resolution images, respectively, and $X(2i_1, 2i_2) = Y(i_1, i_2)$. Eqn. (5.13) shows that an unknown high-resolution pixel is obtained by a linear sum of its known low-resolution neighborhood pixels. Once we complete the whole interpolation process in Eqn. (5.13), all $X(i)$ ($i_1 + i_2$=even) are interpolated. To fill out remaining pixels (*i.e.*, $X(i)$ with $i_1 + i_2$ =odd), we can simply rotate the image by 90 degrees and repeat the above process. As a result, each dimension of the interpolated image becomes twice as large as that of the original image.

In fact, the above PAR model has been adopted for quite a few image interpolation algorithms. For example, the well-known bilinear or bicubic algorithms can be viewed as

one of the PAR methods with a fixed set of model parameters. However, the use of fixed parameters has a strong limitation and, consequently, the resulting model cannot capture the complex behavior of random texture and image edges well. Li [38] proposed a new edge-directed interpolation (NEDI) algorithm by selecting model parameters according to its low-resolution local covariance information adaptively. It improves the interpolation performance in strong edge regions. This idea was further improved in [89] by jointly estimating unknown pixels in groups.

However, we observe that the performance of these improved algorithms is still not adequate for random texture due to following reasons. First, adopting the low-resolution information directly for the high-resolution grid space would be too restrictive, since texture has weak regularity across scales than strong edges [14]. Second, estimating model parameters using the local neighborhood information is not sufficient due to the stochastic, and non-stationary nature of underlying texture. Third, estimating the target pixel with only its four neighbors (*e.g.*, $k_1, k_2 \in \{0, 1\}$) cannot capture the texture behavior well while increasing its model order, $|K|$, would involve more unreliable local information for the parameter estimation. To overcome these drawbacks, we propose to adopt the NL strategy to compute model parameters adaptively.

5.3.2 Model Parameter Estimation

The difficulty in texture prediction lies in that the increase of the model order will demand more samples for parameter prediction. For homogenous and stationary texture, a model of a larger order is not a problem. However, for spatially non-stationary texture, we are not able to get accurate estimation if far-away pixels are needed in the estimation

process. The proposed PAR/NL scheme can overcome this limitation using the non-local information for PAR model parameter estimation.

The way to use the non-local information is the same as that of the conventional NL-based interpolation algorithm. The main difference is that the objective of the PAR/NL model is to estimate model parameters, $\{\theta_i | i \in I\}$, instead of pixel values. We define the objective function of the PAR/NL algorithm as

$$J_{PAR/NL} = \sum_{i \in I} \sum_{j \in I} w_{i,j} ||Y_u(v_j)\, \hat{\theta}_i - Y_u(j)||^2, \tag{5.14}$$

where v_j is a vector consisting of the spatial neighborhood around pixel j and model parameter θ is a column vector. By this objective function, model parameters are constrained to satisfy the self-regularity property and the optimal parameters at pixel i can be obtained as

$$\frac{\partial J}{\partial \hat{\theta}_i} = 2 \sum_j w_{i,j} (Y_u(v_j))^T Y_u(v_j)\, \hat{\theta}_i - Y_u(j)) = 0 \tag{5.15}$$

Therefore, we get

$$\hat{\theta}_i = \Big[\sum\nolimits_j w_{i,j} (Y_u(v_j))^T Y_u(v_j)\Big]^{-1} \sum\nolimits_j w_{i,j} (Y_u(v_j))^T Y_u(v_j). \tag{5.16}$$

Once the whole set of model parameters, $\{\theta_i \,|\, i \in I\}$, are obtained, the estimated pixels $\hat{X}(i)$ can be automatically computed by Eqn. (5.13).

Under this framework, we only need to find the parameter set at odd grid $i = 2i' + 1$ in practice. That is, we can partition the whole image set into even-grid and odd-grid sets, *i.e.*,

$$I = I_e \cup I_o,$$

where I_e and I_o are the even-grid and odd-grid image sets, respectively. In a similar fashion, we can divide the high-resolution image X into the observed data X_e and unobserved data X_o, and the upsampled image will be union of these two, *i.e.*,

$$Y_u = X_e \cup \hat{X}_o,$$

since the odd pixel of the upsampled image Y_u is the estimated data. Then, the objective function can be re-written as

$$\begin{aligned} J_{PAR/NL} &= \sum_{i \in I_o} \sum_{j \in I} w_{i,j} ||Y_u(v_j)\, \hat{\theta}_i - Y_u(j)||^2 \\ &= \sum_{i \in I_o} \left[\sum_{j \in I_o} w_{i,j} ||X_e(v_j)\, \hat{\theta}_i - \hat{X}_o(j)||^2 + \sum_{j \in I_e} w_{i,j} ||\hat{X}_o(v_j)\, \hat{\theta}_i - X_e(j)||^2 \right]. \end{aligned} \tag{5.17}$$

For the further algorithm development, we only consider the first term of the above equation as our objective function as

$$J_{PAR/NL} = \sum_{i \in I_o} \sum_{j \in I_o} w_{i,j} ||X_e(v_j)\, \hat{\theta}_i - \hat{X}_o(j)||^2, \tag{5.18}$$

since this form is more convenient to compute and derive other formulas.

5.3.3 Determination of Intermediate Up-sampled Image

For the PAR/NL scheme, it is important to determine the intermediate upsampled image, Y_u, which will be the initial guess of the proposed PAR/NL algorithm. It even determines the consistency of the proposed algorithm to be discussed later. Basically, we assume that the inaccuracy of model parameters is caused by the non-stationarity of the spatial covariance. In other words, texture with a spatially-stationary covariance function will have constant model parameters. This is our start point.

Based on the local stationarity assumption, the optimal model parameters and the corresponding upsampled image will be obtained by minimizing the following objective function

$$J_{PAR/L} = \sum_{i \in I_o} \sum_{j \in \mathcal{N}_i \cap I_o} w_{i,j}^G ||X_e(v_j)\hat{\theta}_i^L - \hat{X}_o(j)||^2, \tag{5.19}$$

which is similar to Eqn. (5.18) except for two items. One is the range of index j, which is restricted to the local neighborhood of i, represented by $\mathcal{N}_i$, while the other is the weight factor, $w_{i,j}^G$, which is calculated based on the fixed Gaussian kernel and only a function of distance $||i-j||^2$.

However, the minimization of this objective function is not trivial, since it has two unknown variables, *i.e.*, θ^L and X_o. Here, we assume X_o as unobserved latent variables, and apply the expectation-maximization (EM) algorithm to determine the optimal parameters by following two iterative procedures.

E-step) Calculate the expected value of the log likelihood function with respect to the conditional distribution of X_o given observation X_e under the current estimate of parameter θ

$$Q(\theta, \theta^{(t)}) = E\left[\log p(X_e, X_o|\theta)|X_e, \theta^{(t)}\right].$$

M-step) Find the parameter that maximizes the following quantity

$$\theta^{(t+1)} = \arg\max_{\theta} Q(\theta, \theta^{(t)}).$$

In the E-step, the log likelihood function is expanded as

$$\begin{aligned} \log p(X_e, X_o|\theta) &= \log p(X_e|X_o, \theta) + \log p(X_o|\theta) \\ &= \log p(X_e|X_o, \theta) + \log p(X_o), \end{aligned} \tag{5.20}$$

since the X_o is independent of θ. It can be simplified as

$$\begin{aligned} \log p(X_e(i), X_o(i)|\theta) &= -\sum_{j \in \mathcal{N}_i \cap I_o} \frac{||X_e(v_j)\,\theta_i - X_o(j)||^2}{2\sigma^2} + A_1 \\ &= -\sum_{j \in \mathcal{N}_i \cap I_o} \frac{||X_e(v_j)\theta_i||^2 - 2\theta_i^T X_e(v_j) X_o(j)}{2\sigma^2} + A_2, \end{aligned} \tag{5.21}$$

where A_1 and A_2 are constants, whose values are independent of θ. This equation is linear in variable X_o so that the E-step is to compute the conditional expectation of X_o given observation X_e and current parameter estimate $\theta^{(t)}$ as

$$\begin{aligned} X_o(i)^{(t)} &= E[X_o|X_e, \theta^{(t)}] \\ &= \sum_{j \in \mathcal{N}_i \cap I_o} w_{i,j}^G \cdot X_e(v_i)\, \theta_j^{(t)}\,. \end{aligned} \tag{5.22}$$

Then, the Q-function can be given as

$$Q(\theta, \theta^{(t)}) = \sum_{i \in I_o} \sum_{j \in \mathcal{N}_i \cap I_o} -\frac{||X_e(v_j)\, \theta_i - X_o^{(t)}(j)||^2}{2\sigma^2} + A_1 \tag{5.23}$$

Finally, the M-step updates the parameter with given parameter estimate $\theta^{(t)}$ and computed latent data $X_o^{(t)}$ as

$$\begin{aligned} \theta^{(t+1)} &= \arg\max \left[\sum_{i \in I_o} \sum_{j \in \mathcal{N}_i \cap I_o} -\frac{||X_e(v_j)\, \theta_i - X_o^{(t)}(j)||^2}{2\sigma^2} + A_1 \right] \\ \therefore \theta_i^{(t+1)} &= \left[\sum_{j \in \mathcal{N}_i \cap I_o} w_{i,j}^G (X_e(v_j))^T X_e(v_j) \right]^{-1} \sum_{j \in \mathcal{N}_i \cap I_o} w_{i,j}^G (X_e(v_j))^T X_o(j)\,. \end{aligned} \tag{5.24}$$

It is advantageous to adopt the EM procedure since it always guarantees the convergence of the iterative solution. Experimentally, we find that the optimal set of parameters is not sensitive to the initial guess of the upsampled image. We use the bicubic algorithm for the initial upsampling in our implementation.

5.3.4 Analysis of PAR/NL Algorithm

This section will provide more analysis to the above-mentioned algorithm by comparing it with the conventional NL algorithm. Specifically, we are going to investigate the relationship between a locally obtained upsampled image with parameters and its ground-truth. Then, it will provide the objective and the expected improvement of the proposed PAR/NL algorithm.

The ground-truth upsampled image can be written as

$$\begin{aligned} X_o &= \hat{X}_o + \varepsilon_1 \\ &= X_e(v_i)\theta^o + \varepsilon_1 \\ &= X_e(v_i)\hat{\theta}^L + X_e(v_i)(\theta^o - \hat{\theta}^L) + \varepsilon_1 \\ &= \hat{X}_o^L + \varepsilon_1 + \varepsilon_2 \\ &= \hat{X}_o^L + \varepsilon, \end{aligned} \tag{5.25}$$

where X_o is the ground-truth high-resolution image, $\hat{X}_o$ is the optimal estimated data, and θ^o is the optimal parameter set that minimizes *i.i.d.* forecast error ε_1. On the other hand, the locally obtained parameter set, $\hat{\theta}^L$, will yield $\hat{X}_o^L$ and produces another type of error ε_2, which is called the 'miss-prediction' error.

Since the proposed framework involves two different types of errors, we will explain them more clearly below. The forecast error, ε_1, is caused by the projection process of the estimation since the dimension of the data is larger then the dimension of the parameters. Consequently, the forecast error can be reduced by increasing the model order, K. However, a larger order number tends to increase the miss-prediction error due

to the limit number of samples satisfying the stationary property. On the other hand, the miss-prediction error, ε_2, can be reduced by an advanced parameter estimation algorithm. This is achieved by the proposed PAR/NL scheme in our work. As a result, the proposed PAR/NL algorithm is meaningful when the miss-prediction error is much larger than the forecast error. Otherwise, the improvement by the proposed PAR/NL algorithm will not be perceptually significant. Thus, the prediction error by $\hat{\theta}_i^L$ at grid i becomes

$$\begin{aligned} E||\varepsilon_i||^2 &= E||X(v_i)\,\hat{\theta}_i^L - X(i)||^2 \\ &= E||X(v_i)\,\theta_i^o - X(i) + X(v_i)\,(\hat{\theta}_i^L - \theta_i^o)||^2 \\ &= E||X(v_i)\,\theta_i^o - X(i)||^2 + E||X(v_i)\,(\hat{\theta}_i^L - \theta_i^o)||^2 \\ &= \varepsilon_{1,i}^2 + \varepsilon_{2,i}^2, \end{aligned} \tag{5.26}$$

i.e., the sum of the forecast and the miss-prediction errors.

The objective function in Eqn. (5.18) can be derived and re-written as

$$\begin{aligned} J_{PAR/NL} &= \sum_{i\in I_o}\sum_{j\in I_o} w_{i,j}||X_e(v_j)\,\hat{\theta}_i - \hat{X}_o^L(j)||^2 \\ &= \sum_{i\in I_o}\sum_{j\in I_o} w_{i,j}||X_e(v_j)\,\hat{\theta}_i - (\hat{X}_o(j) + \varepsilon_{1,j})||^2 \\ &\approx \sum_{i\in I_o}\sum_{j\in I_o} w_{i,j}||X_e(v_j)\,\hat{\theta}_i - \hat{X}_o(j)||^2 + E_1 \\ &= \sum_{i\in I_o}\sum_{j\in I_o} w_{i,j}||X_e(v_j)\,(\hat{\theta}_i - \theta_j^o)||^2 + E_1, \end{aligned} \tag{5.27}$$

where E_1 is the error term caused by ε_1, and it becomes constant on average by its *i.i.d.* distribution. Finally, this equation becomes analogous to the conventional NL-based interpolation algorithm in the sense that it compares the self-similarity between

parameters. The major distinction is that its difference is projected to the space spanned by its sample set, which is one of the drawbacks of the least-square (LS) approach. In this work, it would be useful to compute the projected error, since the error computed in the pixel domain can be well handled and matched to the general NL-based interpolation algorithm than the error computed in the parameter domain.

5.3.5 Distortion Measure for Texture

It is important to investigate the distortion measure for random texture since the pixel-wise distance in the NL algorithm does not hold. In the pixel-wise NL algorithm, the local neighborhood information is used to measure the similarity for the central pixel adjustment. In a similar fashion, we define the similarity between two texture blocks by investigating the parameters of their local neighborhoods as

$$\begin{aligned} & D_\theta(Y_u(\mathcal{N}_i), Y_u(\mathcal{N}_j)|\hat{X}_o^L) \\ & = \sum_{p\in\mathcal{N}_i, q\in\mathcal{N}_j} ||X_e(v_p)\,\hat{\theta}_q^L - \hat{X}_o^L(p)||^2 + ||X_e(v_q)\,\hat{\theta}_p^L - \hat{X}_o^L(q)||^2 \\ & = \sum_{p\in\mathcal{N}_i, q\in\mathcal{N}_j} ||X_e(v_p)\,\hat{\theta}_q^L - X_e(v_p)\,\hat{\theta}_p^L||^2 + ||X_e(v_q)\,\hat{\theta}_p^L - X_e(v_q)\,\hat{\theta}_q^L||^2 \\ & = \sum_{p\in\mathcal{N}_i, q\in\mathcal{N}_j} ||X_e(v_p)\,(\hat{\theta}_q^L - \hat{\theta}_p^L)||^2 + ||X_e(v_q)\,(\hat{\theta}_p^L - \hat{\theta}_q^L)||^2. \end{aligned} \tag{5.28}$$

In words, it computes the distance in the projected domain of the parameter distance. It is worthwhile to point out that the pixel distance of the NL algorithm is analogous to the projected one of the parameter-distance of the proposed PAR/NL algorithm.

In the NL image restoration process, selection of the appropriate filtering parameter h in Eqn. (5.4) also plays an important role since it determines the contribution of the non-local information by adjusting the decay of weights. Buades *et al.* [8] proposed to set h according to its noise variance in the image denoising application. Here, we also use the noise variance to determine the filter coefficient. Since the PAR/NL model is applied to the intermediate upsampled image with the locally adaptive model parameters, θ^L, the miss-prediction error of model parameters as represented by δ in Eqn. (5.26) indicates the noise level in our framework conceptually. However, none of the error in Eqn. (5.26) is practically observable since both ground-truth X_o and θ^o are unknown. This situation is similar to the conventional NL-based algorithm without prior knowledge of noise variance. In the blind case, noise energy has to be estimated for further processing. In this work, we assume that the minimum value of the objective function in Eqn. (5.19) is equal to noise energy. Thus, we can compute h as

$$h = \sqrt{\frac{1}{|I|}\sum_{i\in I}\sum_{j\in\mathcal{N}_i} w_{i,j}^G ||X_e(v_j)(\hat{\theta}_i^L - \hat{\theta}_j^L)||^2}\,. \tag{5.29}$$

This approach is reasonable since the model parameter difference is caused by the non-stationarity of given texture.

5.3.6 Summary of Proposed PAR/NL Algorithm

The proposed PAR/NL algorithm can be summarized as follows, and graphically illustrated in Fig. 5.2

A1) Initialize high-resolution image Y_u from a given low-resolution image, Y, with an arbitrary interpolation scheme.

A2) For each grid point $i \in I_o$, estimate the local model parameters, $\hat{\theta}_i^L$, with its local neighborhood information only as given in Eqn. (5.24).

A3) Update Y_u with the estimated parameters $\hat{\theta}^L$ as given in Eqn. (5.23).

A4) Repeat Steps A2)-A3) until the change of Y_u is small enough.

A5) Compute all unknown pixels of the Y_u^L with $\hat{\theta}_i^L$ obtained from Step A4).

A6) Compute filter coefficient h with Eqn. (5.29)

A7) For each grid point $i \in I_o$, estimate non-local model parameters, $\hat{\theta}_i^{NL}$, with all available non-local information as given in Eqn. (5.16).

A8) Compute all unknown pixels of the Y_u^{NL} with $\hat{\theta}_i^{NL}$ obtained from Step A7).

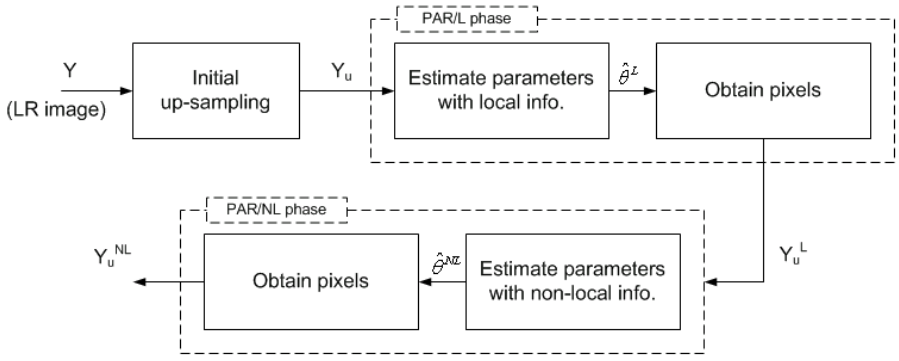


Figure 5.2: The overall structure of the proposed PAR/NL algorithm.

Since the PAR/NL algorithm exploits the information of distant as well as local pixels, it provides more flexibility than increasing the model order. In our implementation, we

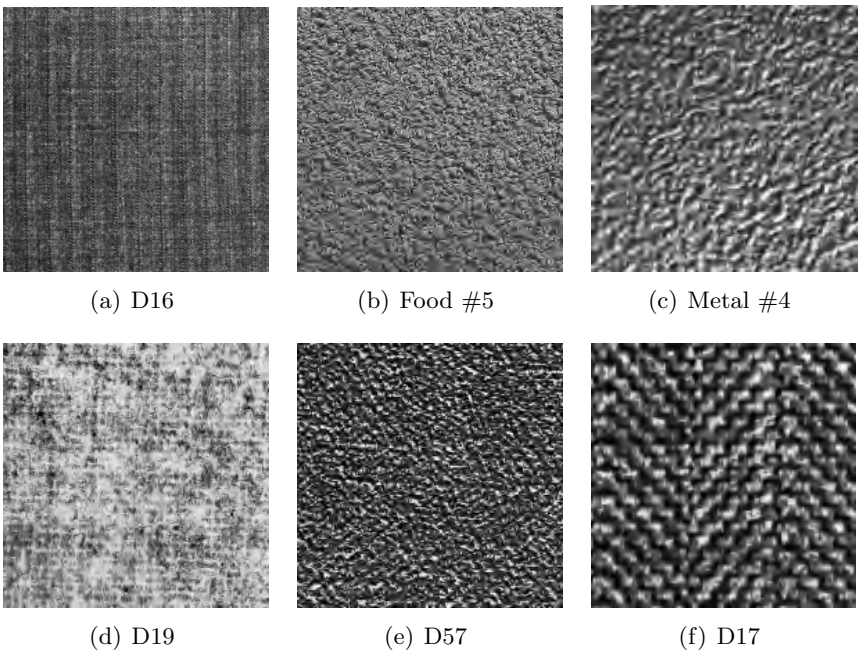
fix model order $|K| = 36$, *i.e.*, 6×6 neighborhood pixels to estimate the value of a target pixel. For robust estimation, we use 49 samples to estimate θ^L and θ^{NL}, *i.e.*, $|\mathcal{N}| = 49$.

5.4 Experimental Results

To evaluate the performance of the proposed PAR/NL interpolation algorithm, we selected several random texture images from the VisTex texture database [76] and the Brodatz album [7] as the test set. All test images were 8-bit gray images. They were chosen to allow a wide range of diversity. The selected textures are shown in Fig. 5.3 for visual inspection. Note that 'D16' has a fine but visible direction, while 'D57' and 'Metal #4' have more obvious directional patterns with fine random texture. 'Food #5' has no visible directional pattern with detail texture. 'D19' is composed by texture with varying background. 'D17' has some structure in the texture patten, which is included to test the limitation of the proposed PAR/NL algorithm.

For performance evaluation, we rely only on the subjective quality test for two reasons. First, there is no ground truth available. Second, it is generally agreed that the conventionally used quantitative measure, such as the mean-squared-error (MSE) or the peak-signal-to-noise-ratio (PSNR), cannot reflect the visual quality of textured images well, since the pixel-by-pixel accuracy in stochastic texture is not meaningful. For example, by shifting a texture image by 1 pixel horizontally or vertically, it is still the same texture image to human eyes but their MSE could be large.

For visual comparison, we show interpolated images in Figs. 5.4 $\sim$ 5.9 using (a) the bilinear, (b) the bicubic, (c) the NL-interpolation, (d) the edge-directed interpolation [38],



(a) D16 (b) Food #5 (c) Metal #4

(d) D19 (e) D57 (f) D17

Figure 5.3: Selected texture images for performance comparison.

(e) the optimized edge-directed interpolation [89], and (f) the proposed PAR/NL interpolation scheme. In these figures, the first two rows consist of the 2x2 upsampled images, and the bottom two rows consist of the 4x4 upsampled images. Due to the space limit and visual convenience, we only show the center part of upsampled images. We see that the bilinear and the bicubic algorithms lose many texture detail and blur interpolated images. The edge-directed algorithm is good in sharpening edges, but it tends to generate some directional patterns. Besides, it can generate spurious points caused by the mismatch between low- and high-resolution textures easily. The optimized edge-directed algorithm is developed to overcome this drawback by the soft-estimation method, but the results are still not satisfactory, and sometimes even worse as shown. The NL-interpolation appears to mislead the general outlook of the interpolated texture. The proposed PAR/NL scheme preserves texture detail well, and yields a better stochastic texture behavior than

other algorithms. It is also interesting to point out that the performance of the proposed PAR/NL is varying according to the randomness and stationarity of given texture. For example, the result for 'D17' is not as good as those for 'D16' and 'Food #5', since 'D17' is more structured.

We also evaluated the quality of interpolated images by a subjective test procedure. Each algorithm is evaluated by 10 subjects, and scored on a scale of $1 \sim 10$ with the bilinear interpolated image as the reference, whose score is set to 5. The subjective test results for three representative textures are shown in Fig. 5.10, where the shaded bar and its middle dotted line represent the 95% confidence interval and its mean value, respectively. We see that the bicubic, the edge-directed and the proposed PAR/NL method all outperform the bilinear interpolation scheme substantially. The proposed PAR/NL interpolation has the best performance among all schemes consistently.

5.5 Conclusion and Discussion

A new approach for texture interpolation, called the PAR/NL interpolation method, was proposed in this work. It uses the PAR model whose parameters are determined by the NL algorithm. It was shown that parameters of the PAR model can be efficiently obtained from the image itself, and the resultant PAR/NL interpolation scheme outperforms several benchmarking methods. Since the PAR/NL algorithm is an adaptation of the conventional NL-interpolation algorithm, we also provide the complete analogy between two algorithms with the theoretic analysis and derivation.

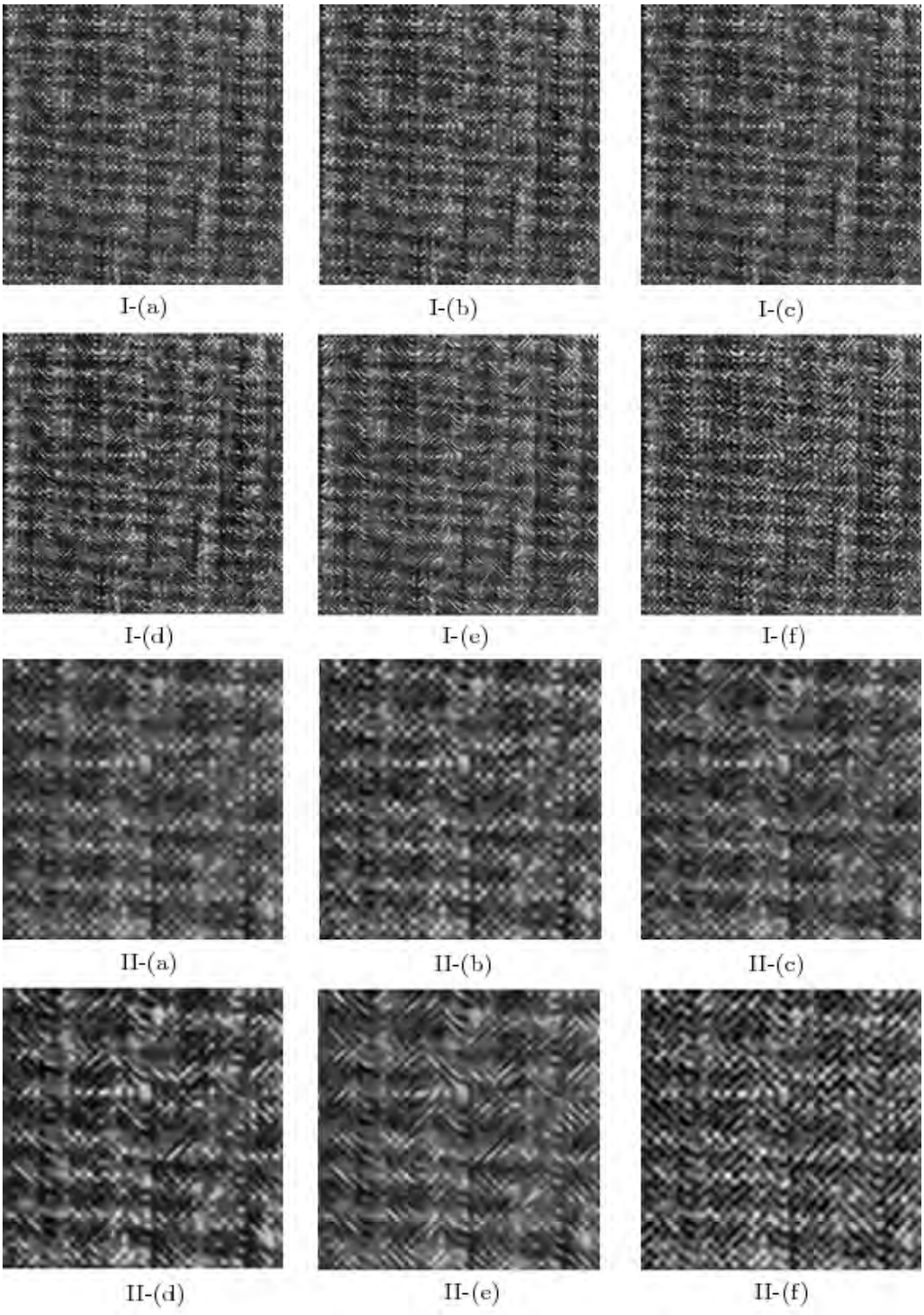


Figure 5.4: Upsampled image results for 'D16' for I. 2x2 zooming, II. 4x4 zooming with (a) bilinear, (b) bicubic, (c) NL-interpolation, (d) edge-directed [38], (e) optimized edge-directed [89], and (f) proposed PAR/NL schemes.

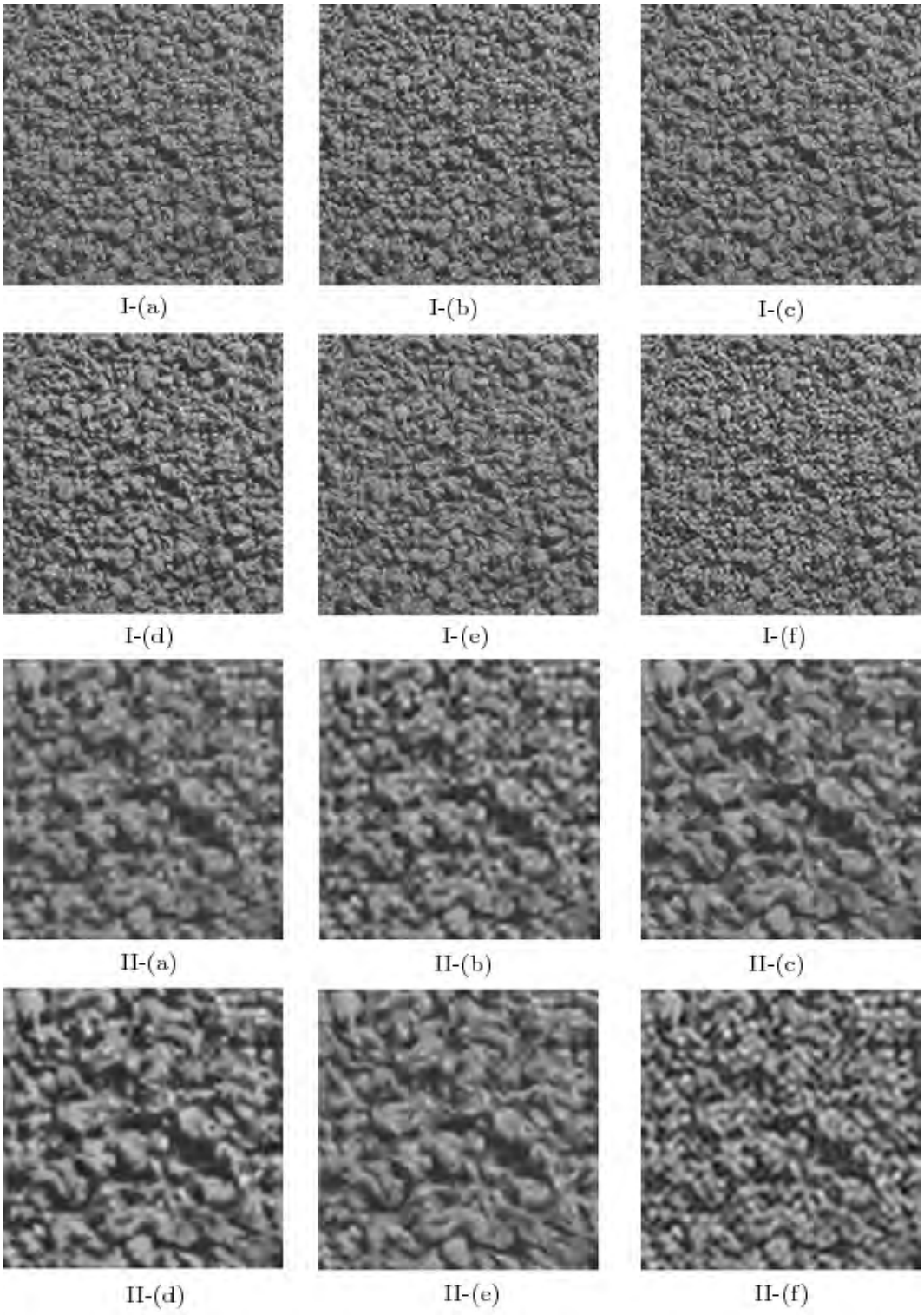


Figure 5.5: Upsampled image results for 'Food #5' for I. 2x2 zooming, II. 4x4 zooming with (a) bilinear, (b) bicubic, (c) NL-interpolation, (d) edge-directed [38], (e) optimized edge-directed [89], and (f) proposed PAR/NL schemes.

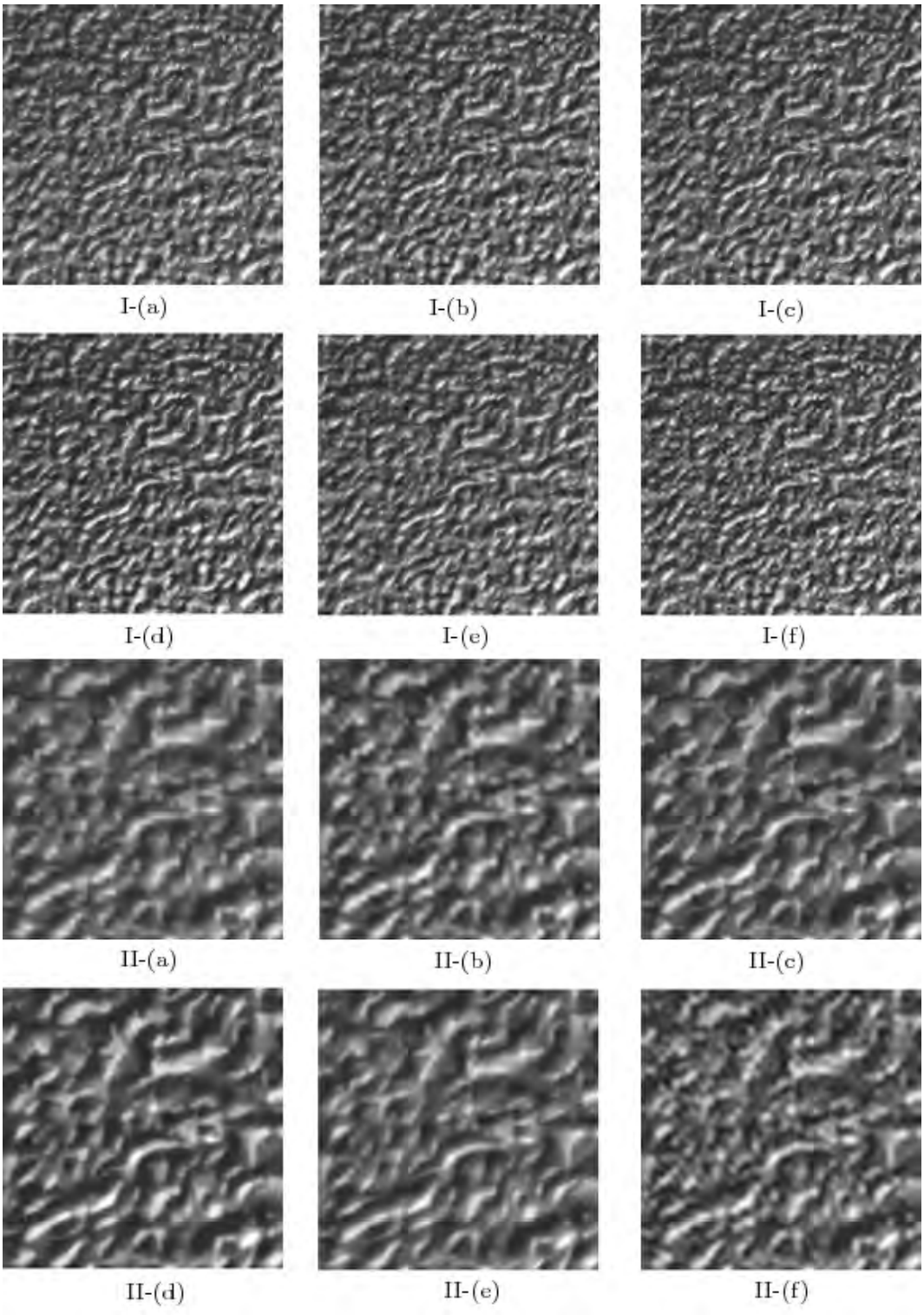


Figure 5.6: Upsampled image results for 'Metal #4' for I. 2x2 zooming, II. 4x4 zooming with (a) bilinear, (b) bicubic, (c) NL-interpolation, (d) edge-directed [38], (e) optimized edge-directed [89], and (f) proposed PAR/NL schemes.

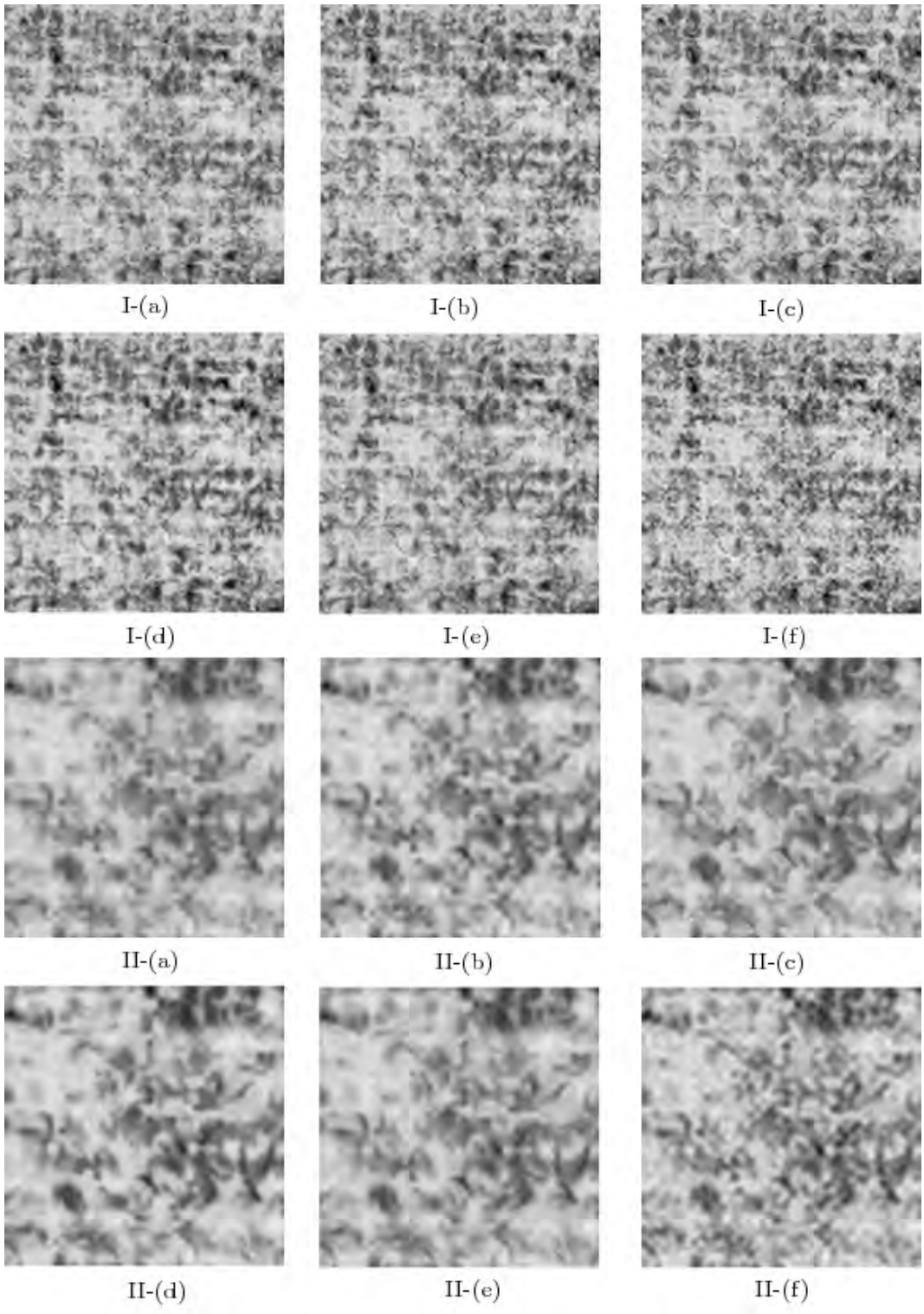


Figure 5.7: Upsampled image results for 'D19' for I. 2x2 zooming, II. 4x4 zooming with (a) bilinear, (b) bicubic, (c) NL-interpolation, (d) edge-directed [38], (e) optimized edge-directed [89], and (f) proposed PAR/NL schemes.

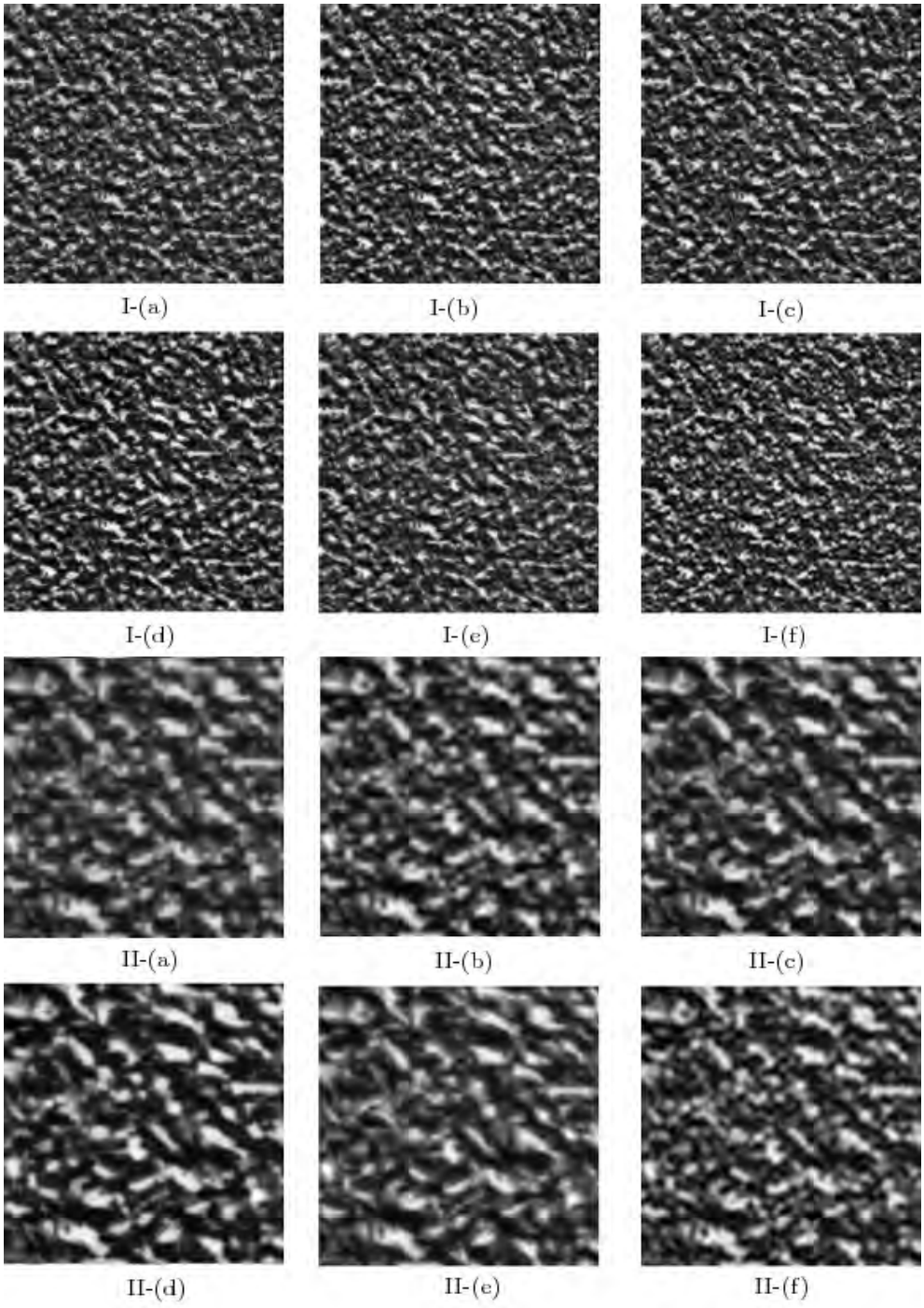


Figure 5.8: Upsampled image results for 'D57' for I. 2x2 zooming, II. 4x4 zooming with (a) bilinear, (b) bicubic, (c) NL-interpolation, (d) edge-directed [38], (e) optimized edge-directed [89], and (f) proposed PAR/NL schemes.

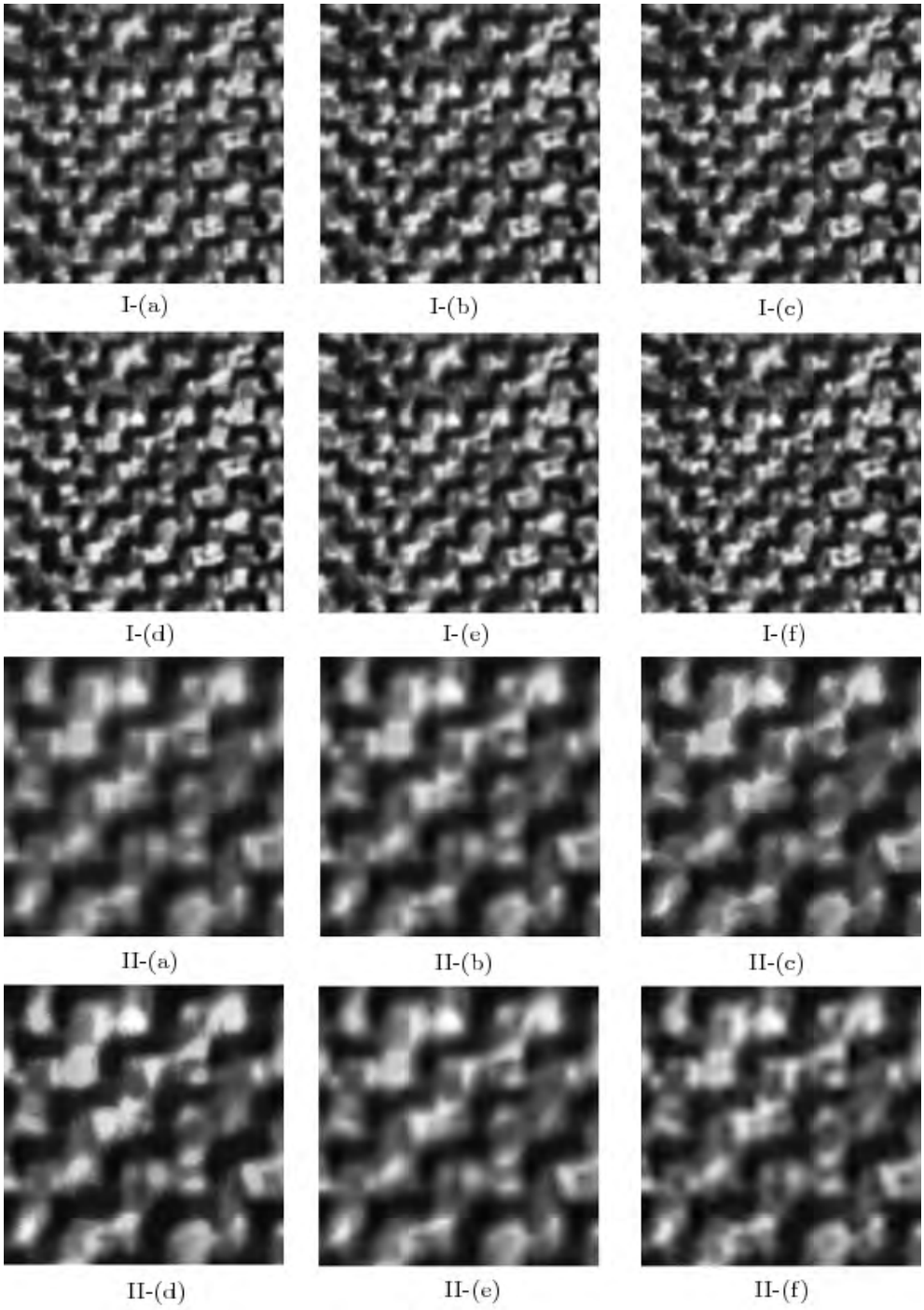


Figure 5.9: Upsampled image results for 'D17' for I. 2x2 zooming, II. 4x4 zooming with (a) bilinear, (b) bicubic, (c) NL-interpolation, (d) edge-directed [38], (e) optimized edge-directed [89], and (f) proposed PAR/NL schemes.

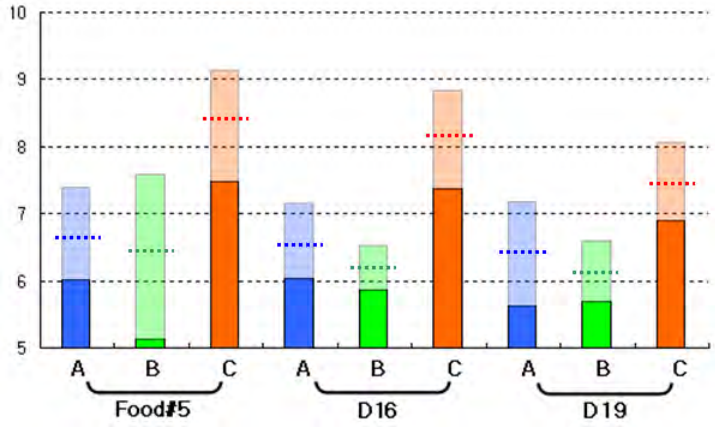


Figure 5.10: Comparison of subjective test results, where A : bicubic, B: edge-directed [38] and C: the proposed method.

Chapter 6

Conclusion and Future Work

6.1 Summary of the Research

Two main research topics have been examined in this dissertation. First, we studied texture processing techniques to improve the efficiency of state-of-the-art video coding standards, where the main idea was built upon synthesis-based texture coding. Second, we investigated the super-resolution method for textured image interpolation, which improves the prediction accuracy of the linear model.

A new framework to control the synthesized texture by sending the additional side information was proposed in Chapter 3. There are several important contributions in our proposed scheme. First, a novel framework to control the synthesized texture by sending additional side information was proposed, and it was shown that only a small amount of side information can efficiently guide the overall texture structure. The side information can be easily obtained by a standard encoder operating at a low bit-rate, and no additional coding tools for the side information are necessary. Second, we can further increase

the synthesis efficiency by adaptively selecting the side information based on the rate-distortion (R-D) optimization method. As a result, the locally adaptive side information improves the similarity and naturalness compared to using uniform side information for a given bit budget. Third, a texture decomposition algorithm was developed for the proposed texture synthesis algorithm to address the problem of illumination-variant texture, and its effectiveness was presented by experimental results.

Film grain noise modeling method was developed based on its unique properties in Chapter 4. The model can be used for two purposes: 1) film grain noise extracting in the encoder and 2) film grain noise rendering in the decoder. To encode high definition video with film grain noise, we proposed a video coding system as follows. In the encoder end, film grain noise was extracted as a pre-processing step, which consisted of two phases: smooth-region detection and film grain denoising. Since the pre-processing algorithm should cause little distortion to the original content, we extracted film grain noise from the smooth area only. Then, a non-linear denoising scheme based on total variation minimization was proposed. We exploited various film grain noise properties for noise estimation, which was proved to be more efficient than the conventional 2D Gaussian filtering and the simple TV method. In the decoder end, we propose a noise rendering (or synthesis) method using the spatial and spectral AR model. The AR model is selected due to its low decoding complexity and good rendering capability. It was shown by experimental results that the rendered film grain noise is similar to the original in terms of the noise power spectrum density, distribution, cross-color correlation and signal dependency.

Finally, a new approach for texture interpolation, called the PAR/NL interpolation method, was proposed in Chapter 5. It adopted a PAR model whose parameters are determined by the NL algorithm. We provided an analysis on the conventional NL algorithm and the proposed PAR/NL approach and compared the performance of these two schemes. It was shown that parameters of the PAR model can be efficiently obtained from the image itself, and the resultant PAR/NL interpolation scheme outperforms several benchmarking methods.

6.2 Future Research Topics

To make our current research more complete, we would like to investigate the following research topics in the near future.

- Complexity reduction for texture synthesis

  The complexity of synthesis-based video texture coding is very high, which could be a major bottleneck for its wide acceptance. This problem is even worse by considering the fact that computational power of the decoder is usually lower than the encoder. For these reasons, we need to develop a faster texture synthesis algorithm.

- Non-causal film grain noise rendering

  The proposed film grain noise rendering method uses the raster scanning order, which is a causal rendering method. Since causal rendering may result in an artificial directional pattern, it could be perceptually different than the original isotropic film grain noise. It is desirable to develop a non-causal rendering scheme without

increased complexity. One possible direction is the multi-resolution-based rendering method.

- More applications with PAR/NL texture model

  The proposed PAR/NL method has a potential to offer a good solution to challenging image processing problems such as texture classification, restoration, learning method, etc. Most today's image processing algorithms heavily depend on the locality assumption. In addition to the linear model, it is possible to integrate other models such as the higher-order model or the support vector machine (SVM) with the non-local strategy. It is interesting to find a more generic framework for textured image/video processing.

Bibliography

[1] O. K. Al-Shaykh and R. M. Mersereau, "Lossy compression of noisy images," *IEEE Transactions on Image Processing*, vol. 7, no. 12, pp. 1641–1652, Dec. 1998.

[2] J. F. Aujol, G. Gilboa, T. Chan, and S. Osher, "Structure-texture image decomposition - modeling, algorithms, and parameter selection," *International Journal of Computer Vision*, vol. 67, no. 1, pp. 111–136, Apr. 2006.

[3] A. Benyassine, E. Shlomot, H. Y. Su, D. Massaloux, C. Lamblin, and J. P. Petit, "ITU recommendation G.729 annex B: A silence compression scheme for use with G.729 optimized for V.70 digital simultaneous voice and data applications," *IEEE Communication Magazine*, vol. 35, no. 9, pp. 64–73, Sep. 1997.

[4] M. Bertalmio, L. Vese, G. Sapiro, and S. Osher, "Simultaneous structure and texture image inpainting," *IEEE Transactions on Image Processing*, vol. 12, no. 8, pp. 882–889, Aug. 2003.

[5] K. J. Boo and N. K. Bose, "A motion-compensated spatio-temporal filter for image sequences with signal-dependent noise," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 3, pp. 287–298, Jun. 1998.

[6] M. Bosch, F. Zhu, and E. J. Delp, "Spatial texture models for video compression," in *Proceedings of IEEE International Conference on Image Processing*, 2007, pp. 93–96.

[7] P. Brodatz, "Texture : A photographic album for artists and designers," New York, 1966.

[8] A. Buades, B. Coll, and J. Morel, "A non-local algorithm for image denoising," in *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, 2005, pp. 60–65.

[9] A. Buades, B. Coll, and J. M. Morel, "A review of image denoising algorithms, with a new one," *Multiscale Modeling and Simulation (SIAM interdisciplinary journal)*, vol. 4, no. 2, pp. 490–530, Jul. 2005.

[10] J. A. Cadzow, D. M. Wilkes, and R. A. Peters, "Image texture synthesis-by-analysis using moving-average models," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 29, no. 4, pp. 1110–1122, Oct. 1993.

[11] P. Campisi, D. Hatzinakos, and A. Neri, "A perceptually lossless, model-based, texture compression technique," *IEEE Transactions on Image Processing*, vol. 9, no. 8, pp. 1325–1336, Aug. 2000.

[12] P. Campisi and G. Scarano, "A multiresolution approach for texture synthesis using the circularharmonic functions," *IEEE Transactions on Image Processing*, vol. 11, no. 1, pp. 37–51, Jan. 2002.

[13] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, Nov. 1986.

[14] W. K. Carey, D. B. Chuang, and S. S. Hemami, "Regularity-preserving image interpolation," *IEEE Transactions on Image Processing*, vol. 8, no. 9, pp. 1293–1297, Sep. 1999.

[15] T. Chang and C.-C. Kuo, "Texture analysis and classification with tree-structured wavelet transform," *IEEE Transactions on Image Processing*, vol. 2, no. 4, pp. 429–441, Oct. 1993.

[16] D. Charalampidis, "Texture synthesis: textons revisited," *IEEE Transactions on Image Processing*, vol. 15, no. 3, pp. 777–787, Mar. 2006.

[17] R. Chellappa and R. L. Kashyap, "Texture synthesis using 2-D noncausal autoregressive models," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 33, no. 1, pp. 194–203, Feb. 1985.

[18] H. Y. Cheong, A. M. Tourapis, J. Llach, and J. Boyce, "Adaptive spatio-temporal filtering for video de-noising," in *Proceedings of IEEE International Conference on Image Processing*, 2004, pp. 24–27.

[19] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering," *IEEE Transactions on Image Processing*, vol. 16, no. 8, pp. 2080–2095, Aug. 2007.

[20] A. Dumitras and B. G. Haskell, "A texture replacement method at the encoder for bit-rate reduction of compressed video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 2, pp. 163–175, Feb. 2003.

[21] A. A. Efros and W. T. Freeman, "Image quilting for texture synthesis and transfer," in *International Conference on Computer Graphics and Interactive Techniques*, 2001, pp. 341–346.

[22] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," in *International Conference on Computer Vision*, 1999, pp. 1033–1038.

[23] K. B. Eom, "2-D moving average models for texture synthesis and analysis," *IEEE Transactions on Image Processing*, vol. 7, no. 12, pp. 1741–1746, Dec. 1998.

[24] M. A. T. Figueiredo and R. D. Nowak, "An EM algorithm for wavelet-based image restoration," *IEEE Transactions on Image Processing*, vol. 12, no. 8, pp. 906–916, Aug. 2003.

[25] W. T. Freeman, T. R. Jones, and E. C. Pasztor, "Example-based super-resolution," *IEEE Computer Graphics and Applications*, vol. 22, no. 2, pp. 56–65, Mar. 2002.

[26] C. Gomila, "SEI message for film grain encoding: syntax and results," *JVT 7th meeting, Doc. JVT-I013, San Diego*, 2003.

[27] C. Gomila and A. Kobilansky, "SEI message for film grain noise," *JVT 8th meeting, Doc. JVT-H022, Geneva*, 2003.

[28] A. B. Hamza, H. Krim, and G. B. Unal, "Unifying probabilistic and variational estimation," *IEEE Signal Processing Magazine*, vol. 19, no. 5, pp. 37–47, Sep. 2002.

[29] R. M. Haralick, "Statistical and structural approaches to texture," in *Proceedings of IEEE*, 1979.

[30] ITU-T, "Video codec for audiovisual service at p × 64 kbps/s," *ITU-T Rec. H.261*, vol. 2, 1990.

[31] ——, "Video coing for low bit rate communications," *ITU-T Rec. H.263*, vol. 2, 1998.

[32] A. K. Jain and F. Farrokhnia, "Unsupervised texture segmentation using gabor filters," *Pattern Recognition*, vol. 24, no. 12, pp. 1167–1186, Dec. 1991.

[33] L. M. Kaplan and C.-C. Kuo, "Texture roughness analysis and synthesis via extended self-similar (ESS) model," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 11, pp. 1043–1056, Nov. 1995.

[34] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra, "Texture optimization for example-based synthesis," in *ACM SIGGRAPH*, 2005, pp. 795–802.

[35] V. Kwatra, A. Schodl, I. Essa, G. Turk, and S. Bobick, "Graphcut textures : image and video synthesis using graph cuts," in *ACM SIGGRAPH*, 2003.

[36] K. I. Laws, "Textured image segmentation," *USCIPI Report 940*, 1980.

[37] X. Li, "Variational Bayesian image processing on stochastic factor graphs," in *Proceedings of IEEE International Conference on Image Processing*, 2008, pp. 1748–1751.

[38] X. Li and M. T. Orchard, "A new edge-directed interpolation," *IEEE Transactions on Image Processing*, vol. 10, no. 10, pp. 1521–1527, Oct. 2001.

[39] L. Liang, C. Liu, Y. Xu, B. Guo, and H. Shum, "Real-time texture synthesis by patch-based sampling," *ACM Transaction on Graphics*, vol. 20, no. 3, pp. 127–150, Jul. 2001.

[40] L. I. Ludin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D*, vol. 60, no. 1-4, pp. 259–268, Nov. 1992.

[41] S. Mallat and S. Zhong, "Characterization of signals from multiscale edges," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 7, pp. 710–732, Jul. 1992.

[42] J. Mao and A. K. Jain, "Texture classification and segmentation using multiresolution simultaneous autoregressive models," *Pattern Recognition*, vol. 25, no. 2, pp. 173–188, Feb. 1992.

[43] T. M. Moldovan, S. Roth, and M. J. Black, "Denoising archival films using a learned Bayesian model," in *Proceedings of IEEE International Conference on Image Processing*, 2006, pp. 2641–2644.

[44] T. K. Moon, "The expectation-maximization algorithm," *IEEE Signal Processing Magazine*, vol. 13, no. 6, pp. 47–60, Nov. 1996.

[45] MPEG-2, "Test Model 5 (TM5)," *Doc. ISO/ISE JTC1/SC92/WG11/93-225b*, Apr. 1995.

[46] P. Ndjiki-Nya, B. Makai, G. Blattermann, A. Smolic, and H. Schwarz, "Improved H.264/AVC coding using texture analysis and synthesis," in *Proceedings of IEEE International Conference on Image Processing*, 2003, pp. 849–852.

[47] P. Ndjiki-Nya, C. Stuber, and T. Wiegand, "A new generic texture synthesis approach for enhanced H.264/MPEG4-AVC video coding," *Lecture Note in Computer Science*, vol. 3893, pp. 121–128, 2006.

[48] ——, "Texture synthesis method for generic video sequences," in *Proceedings of IEEE International Conference on Image Processing*, 2007, pp. 397–400.

[49] A. Nealen and M. Alexa, "Hybrid texture synthesis," in *Proceedings of Eurographics workshop on Rendering*, 2003, pp. 97–105.

[50] B. T. Oh, Y. Su, A. Segall, and C.-C. J. Kuo, "Synthesis-based texture coding for video compression with side information," in *Proceedings of IEEE International Conference on Image Processing*, 2008, pp. 1628–1631.

[51] B. T. Oh, S. Sun, S. Lei, and C.-C. J. Kuo, "Film grain noise modeling in advanced video coding," in *Proceedings of SPIE, Visual Communications and Image Processing*, 2007.

[52] A. Ortega and K. Ramchandran, "Rate-distortion methods for image and video compression," *IEEE Signal Processing Magazine*, vol. 15, no. 6, pp. 23–50, Nov. 1998.

[53] M. K. Ozkan, M. I. Sezan, and A. M. Tekalp, "Adaptive motion-compensated filtering of noisy image sequences," *IEEE Transactions on Circuit and Systems for Video Technology*, vol. 3, no. 4, pp. 277–290, Aug. 1993.

[54] R. Paget and I. D. Longstaff, "Texture synthesis via a noncausal nonparametric multiscale Markov random field," *IEEE Transactions on Image Processing*, vol. 7, no. 6, pp. 925–931, Jun. 1998.

[55] S. C. Park, M. K. Park, and M. G. Kang, "Super-resolution image reconstruction: a technical overview," *IEEE Signal Processing Magazine*, vol. 20, no. 3, pp. 21–36, May 2003.

[56] A. Pizurica, V. Zlokolica, and W. Philips, "Noise reduction in video sequences using wavelet-domain and temporal filtering," in *SPIE Conference on Wavelet Applications in Industrial Processing*, 2003, pp. 48–59.

[57] J. Portilla and E. P. Simoncelli, "Texture modeling and synthesis using joint statistics of complex wavelet coefficients," in *IEEE Workshop on Statistical and Computational Theories of Vision*, 1999.

[58] ——, "A parametric texture model based on joint statistics of complex wavelet coefficients," *International journal of Computer Vision*, vol. 40, no. 1, pp. 49–71, Oct. 2000.

[59] W. K. Pratt, *Digital Image Processing.* 3nd Edition, Wiley, 2001.

[60] M. Protter, M. Elad, H. Takeda, and P. Milanfar, "Generalizing the nonlocal-means to super-resolution reconstruction," *IEEE Transactions on Image Processing*, vol. 18, no. 1, pp. 36–51, Jan. 2009.

[61] T. Randen and J. H. Husoy, "Filtering for texture classification: A comparative study," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, no. 4, pp. 291–310, Apr. 1999.

[62] T. Ryan, D. Sanders, H. Fisher, and A. Iverson, "Image compression by texture modeling in the wavelet domain," *IEEE Transactions on Image Processing*, vol. 5, no. 1, pp. 26–36, Jan. 1996.

[63] P. Schallauer and R. Morzinger, "Film grain synthesis and its application to re-graining," in *Proceedings of Conference Image Quality and System Performance III*, 2006.

[64] ——, "Rapid and reliable detection of film grain noise," in *Proceedings of IEEE International Conference on Image Processing*, 2006, pp. 413–416.

[65] M. Schlockermann, S. Wittmann, T. Wedi, and S. Kadono, "Film grain coding in H.264/AVC," *JVT 9th meeting, Doc. JVT-I034, San Diego*, 2003.

[66] J. Starck, M. Elad, and D. L. Donoho, "Image decomposition via the combination of sparse representations and a variational approach," *IEEE Transactions on Image Processing*, vol. 14, no. 10, pp. 1570–1582, Oct. 2005.

[67] L. Stroebel, J. Compton, I. Current, and R. D. Zakia, *Basic Photographic Materials and Processes.* 2nd Edition, Focal Press, 2000.

[68] D. M. Strong, P. Blomgren, and T. F. Chan, "Spatially adaptive local feature-driven total variation minimizing image restoration," in *Proceedings of the SPIE Annual Meeting*, 1997, pp. 222–233.

[69] G. J. Sullivan and T. Wiegand, "Rate-distortion optimization for video compression," *IEEE Signal Processing Magazine*, vol. 15, no. 6, pp. 74–90, Nov. 1998.

[70] J. Sun, L. Yuan, J. Jia, and H. Y. Shum, "Image completion with structure propagation," in *ACM SIGGRAPH*, 2005, pp. 861–868.

[71] J. Sun, N.-N. Zheng, H. Tao, and H.-Y. Shum, "Image hallucination with primal sketch priors," in *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, 2003, pp. 729–736.

[72] Y. Suzuki, C. S. Boon, and T. K. Tan, "Inter frame coding with template matching averaging," in *Proceedings of IEEE International Conference on Image Processing*, 2007, pp. 409–412.

[73] M. Szummer and R. W. Picard, "Temporal texture modeling," in *Proceedings of IEEE International Conference on Image Processing*, 1996, pp. 823–826.

[74] ITU-T Recommendation H.264 and ISO/IEC 14496-10, "Advanced video doding for generic audiovisual services," May. 2003.

[75] JVT reference software version JM, "http://iphome.hhi.de/suehring/tml/ download/old_jm."

[76] MIT Media Laboratory, "http://vismod.media.mit.edu/vismod/imagery/ visiontexture/vistex.html."

[77] SMPTE RDD 5-2006, "Film grain technology - Specifications for H.264 — MPEG-4 AVC bit-streams," Mar. 2006.

[78] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proceedings of IEEE international conference on Computer Vision*, 1998, pp. 839–846.

[79] M. K. Tsatsanis and G. B. Giannakis, "Object and texture classification using higher order statistics," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 7, pp. 733–750, Jul. 1992.

[80] L. A. Vese and S. J. Osher, "Image denoising and decomposition with total variation minimization and oscillatory functions," *Journal of Mathematical Imaging and Vision*, vol. 20, no. 1-2, pp. 7–18, Jan. 2004.

[81] A. Watson and M. A. Sasse, "Measuring perceived quality of speech and video in multimedia conferencing applications," in *Proceedings of ACM international conference on Multimedia*, 1998, pp. 12–16.

[82] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, pp. 1–19, Jul. 2003.

[83] 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*, vol. 7, no. 2, pp. 433–437, Apr. 1997.

[84] J. C. K. Yan, P. Campisi, and D. Hatzinakos, "Film grain noise removal and generation for color images," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1998, pp. 2957–2960.

[85] J. C. K. Yan and D. Hatzinakos, "Signal-dependent film grain noise removal and generation based on higher-order statistics," in *Proceedings of IEEE International Conference on Image Processing*, 1997, pp. 77–81.

[86] Y. Yang, N. P. Galatsanos, and A. K. Katsaggelos, "Regularized reconstruction to reduce blocking artifacts of block discrete cosine transform compressed images," *IEEE Transactions on Circuit and Systems for Video Technology*, vol. 3, no. 6, pp. 421–432, Dec. 1993.

[87] J. Zhang, D. Wang, and Q. N. Tran, "A wavelet-based multiresolution statistical model for texture," *IEEE Transactions on Image Processing*, no. 11, pp. 1621–1627, Nov. 1998.

[88] J. Zhang, K. Zhou, L. Velho, B. Guo, and H. Y. Shum, "Synthesis of progressively-variant textures on arbitrary surfaces," in *ACM SIGGRAPH*, 2003, pp. 295–302.

[89] X. Zhang and X. Wu, "Image interpolation by adaptive 2-D autoregressive modeling and soft-decision estimation," *IEEE Transactions on Image Processing*, vol. 17, no. 6, pp. 887–896, Jun. 2008.