An Examplar-Based Approach for Texture Compaction Synthesis and Retrieval

Paruvelli Sreedevi, Wen-Liang Hwang, and Shawmin Lei, *Fellow, IEEE*

***Abstract*—A texture representation should corroborate various functions of a texture. In this paper, we present a novel approach that incorporates texture features for retrieval in an examplar-based texture compaction and synthesis algorithm. The original texture is compacted and compressed in the encoder to obtain a thumbnail texture, which the decoder then synthesizes to obtain a perceptually high quality texture. We propose using a probabilistic framework based on the generalized EM algorithm to analyze the solutions of the approach. Our experiment results show that a high quality synthesized texture can be generated in the decoder from a compressed thumbnail texture. The number of bits in the compressed thumbnail is 400 times lower than that in the original texture and 50 times lower than that needed to compress the original texture using JPEG2000. We also show that, in terms of retrieval and synthesization, our compressed and compacted textures perform better than compressed cropped textures and compressed compacted textures derived by the patchwork algorithm.**

***Index Terms*—Examplar-based approach, texture compaction, texture compression, texture retrieval, texture synthesis.**

I. Introduction

THE rapid growth in the amount of digital media content and recent advances in technologies for sharing and learning information in a global manner have led to an on-demand need for storage space and the data retrieval [1]. Moreover, with the advent of high resolution digital cameras and advanced texture generation techniques, textures usually require a large amount of storage space. Reducing the storage space can improve real-time rendering [2], [3] as well as the performance of remote applications on low-end devices, such as using a mobile phone to retrieve textured data [4]. Usually, compaction or compression is applied to reduce a texture's space. The former reduces the size of a texture, but not the number of bits per pixel; while the latter reduces the number of bits per pixel, but not the size of the texture. Some texture compaction algorithms, e.g., those proposed in [5] and [6], use an examplar-based approach to generate a small compacted texture, carefully extracted from the original texture. Besides the space issue, efficient retrieval of textural data is essential because of the rapid increase in the number of digital libraries and databases [7], [8]. Traditional image retrieval systems use text for searching; however, the method is only useful for coarse texture retrieval because a texture usually contains several details that can not be well-described by text or keywords [9].

Although the above two issues have been widely studied, most approaches try to optimize one issue without considering the other. However, Vasconcelos and Lippman argue that the two issues are closely related and should therefore be considered jointly. To this end, they proposed a library-based coding approach that uses vector quantization techniques to execute compression and retrieval tasks [10], [11]. Library-based coding can be regarded as an implementation of the paradigm that a texture representation should be able to integrate various applications of textures and compression effectively. Motivated by Vasconcelos and Lippman's work, we propose an approach that incorporates the retrieval requirements into a compaction and synthesis algorithm. Under our approach, the goal of compacting and compressing a texture is twofold: 1) to reduce the size of the texture and the number of bits per pixel; and 2) to facilitate the retrieval of textural data. In other words, the compressed and compacted texture retains the textural features of the original texture for the retrieval task. Fig. 1(a) illustrates an application of our approach, where texture is compacted to a smaller thumbnail texture, and then compressed. The compressed thumbnail texture can then be used to synthesize a high quality texture and retrieve a texture from the target database. As shown in Fig. 1(b), the approach can also be used to generate a database of compressed thumbnail textures; then, any texture in the database can be used to retrieve the original texture from the original database.

To implement our approach, we propose using a probabilistic framework that explores the relations between the compaction and synthesis processes and incorporates the retrieval requirements into the framework. We show that an optimized solution of the proposed framework can be obtained and analyzed by applying the generalized EM approach [12]. In addition, we use a subjective test to evaluate the approach's performance in terms of the compaction, compression, and synthesization processes, and adopt an objective measurement to assess the retrieval performance.

In our implementation, compaction and synthesis are based on the examplar-based approach described in [13], which can generate perceptually high quality textures. The texture feature used for retrieval is the local binary pattern (LBP) [14]. It can be derived easily and incorporated into an examplar-based texture compaction and synthesis algorithm to support texture retrieval.

Manuscript received July 13, 2009; revised December 03, 2009. First published December 31, 2009; current version published April 16, 2010. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Sabine Susstrunk.

P. Sreedevi is with the Institute of Information Science, Academia Sinica, Nankang, Taipei, 11529 Taiwan, R.O.C.

W.-L. Hwang is with the Institute of Information Science, Academia Sinica, Nankang, Taipei, 11529 Taiwan, R.O.C., and also with the Department of Computer Science and Information Engineering, Kainan University, Taoyuan, Taiwan, R.O.C. (e-mail: whwang@iis.sinica.edu.tw).

S. Lei is with MediaTek, Inc., Hsinchu, Taiwan, R.O.C.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2009.2039665

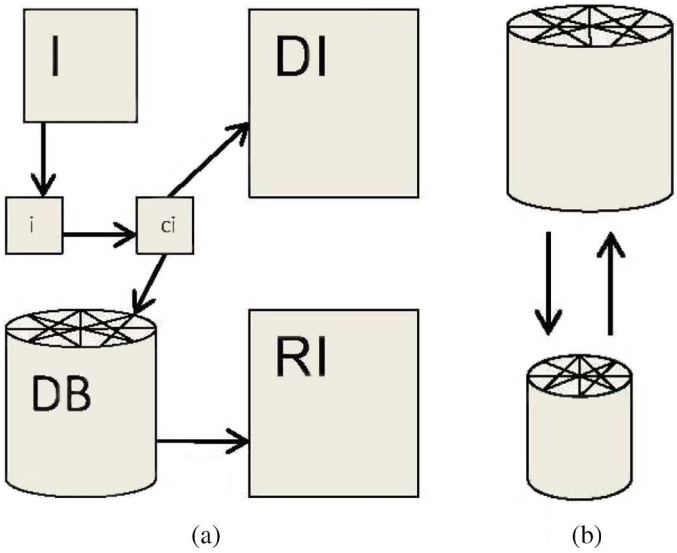

Fig. 1. Schematic diagram of the potential applications of the proposed approach. (a) Input texture I is compacted to i and compressed to obtain ci and DI is synthesized from ci, which is also used to retrieve the texture RI in DB that is perceptually similar to ci. (b) The original database is compacted and compressed to obtain a compressed and compacted database, which requires less storage space and can be stored in a low-end device. Any texture in the compacted database can be used to retrieve a texture in the original database at the server.

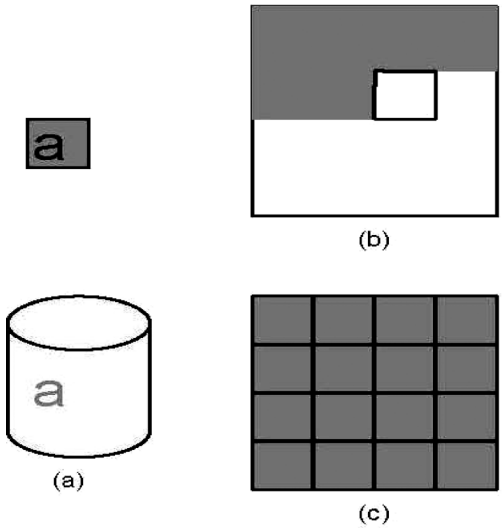

Fig. 2. Schematic diagram of the patchwork algorithm. (a) Patches of size a are sampled from the input texture. The pool of sampled patches is called the patch domain. (b) A new image is generated from the top left-hand corner to the bottom right-hand corner. In each step, the patch selected from the patch pool is pasted. The shaded area indicates the area already patched. (c) The cost of generating the new image is calculated from the overlapping area (the solid area).

We present several numerical results to demonstrate the effectiveness of the proposed algorithm compared to that of other methods. First, we show that our compressed thumbnail texture, in which the number of bits is 400 times lower than that in the original texture and 50 times lower than that needed to compress the original texture using JPEG 2000, can generate a texture that looks like the original texture. Then, we show that the probability of synthesizing a perceptually high quality texture from our thumbnail texture is higher than that of a texture synthesized by cropping a part of the original texture, and also higher than that of compacting the texture by the patchwork algorithm. Finally, the LBP features of different-sized compressed textures derived by our approach are used to retrieve relevant textures from a database. Our results demonstrate that incorporating the LBP feature into the texture compaction and synthesis approach not only helps synthesize high quality textures, but also facilitates texture retrieval.

The remainder of this paper is structured as follows. In Section II, we review the multipatch algorithm [15], which is a variation of the patchwork algorithm [13], [16], and the LBP feature. The multipatch algorithm is an implementation of the examplar-based texture synthesis approach and is used to demonstrate our numerical results. In Section III, we describe the probabilistic framework of the proposed paradigm as well as the generalized EM algorithm, which is used to analyze the solutions in the encoding and decoding phases. In Section IV, we discuss some implementation issues critical to the performance of our framework, and provide the numerical results of various experiments. Then, in Section V, we summarize our conclusions and consider possible avenues for future research.

II. Examplar-Based Approach and the LBP Texture Feature

In this section, we review the examplar-based approach and the LBP texture feature.

A. Multipatch Algorithm

Texture compaction or synthesis is a technique that creates a new image from a sample texture image such that, to an observer, the new image appears to be generated by the same underlying process as the sample texture. If the size of the new image is smaller than that of the sample texture, the technique is called *compaction*; otherwise, it is called *synthesis*. Although several synthesization methods can generate satisfactory results, e.g., [13], [17]–[22], the examplar-based approach proposed in [13] is fast and easy to implement, and it yields a high-quality texture for a broad range of textures. First, the algorithm samples patches of equal size from the original texture. The sampled patches form a domain for generating a new image. Then, a patch is selected and pasted to generate a new image in the scanning order. If the size of the patch is one pixel, the approach is called a *pixel-based* approach; otherwise, it is called a *patch-based* approach. A schematic diagram of the patchwork algorithm is shown in Fig. 2 [13], [16]. Irrespective of the type of implementation, the examplar-based approach is a random process because some randomness is usually imposed to create variations when many texture images are generated from the original texture [23], [24]. Thus, by running the algorithm several times, we can obtain different textures.

Since the approach models textures as homogeneous Markov random fields, it is extremely effective in generating homogeneous textures. Some researchers have modified the approach with the goal of making it work as well for near-homogeneous textures [6]. Moreover, some analytical results based on

Fig. 3. Five cropped textures are taken from the image, which is from the Outtex database.

nonparametric resampling of random fields and a variational point of view are presented in [25] and [26], respectively. One problem with the approach is the selection of the correct patch size because using different-sized patches tends to generate different textures [15]. In addition, the performance of many examplar-based algorithms has not been evaluated in terms of the probability metric. The multipatch algorithm, shown in Fig. 4, was developed to address these issues [15]. The idea is to simply sample different-sized patches and use a combination of them to generate a new image, thereby increasing the size of the patch domain and improving the synthesized result. However, using different-sized patches for synthesis also increases the synthesization time because there are many ways to combine patches. The algorithm assigns a cost to each legal combination of different-sized patches and iteratively modifies each combination until the cost cannot be reduced further. The results of the multipatch algorithm have been evaluated in terms of the probability of generating a perceptually high quality texture and the average time required to complete the process. It has been shown that the algorithm is superior to other methods [16], [23].

B. Local Binary Pattern of a Texture

The local binary pattern (LBP) is a texture descriptor whose performance in different applications has been evaluated in several works [14], [27], [28]. The LBP of a texture can be applied in various ways depending on the invariance property of the texture feature to be retained. In this paper, we use the 3×3 neighborhood of the LBP, as shown by the following example.

To calculate the LBP value of a center pixel, the value of each of the eight neighboring pixels $Np_{i=1,\ldots,8}$ of the center pixel is first labeled 0 or 1 by subtracting the neighboring pixel's value from the center pixel's value *Cp*

$$Np_i = \begin{cases} 1, & \text{if } (Cp - Np_i) \leq 0 \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Then, the obtained binary values are multiplied by their corresponding weights and aggregated to the LBP value of the center pixel: $\sum_{i=1}^{8} Np_i * 2^{i-1}$. Table I shows how the LBP value is computed at the center pixel. A histogram of the LBP values is normally used as the texture descriptor because, after the histogram has been appropriately modified, it is invariant under gray-scale transformation of a texture, as well as under translation and rotation of the texture [29].

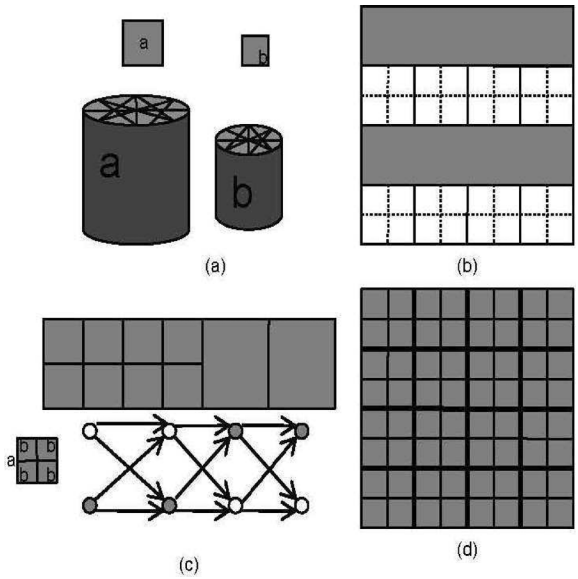

Fig. 4. Schematic diagram of the multipatch algorithm. (a) Two different-sized patches, a and b, are sampled from the input texture. The patch domain is the union of the domains of patches of sizes a and b. (b) New image is generated by modifying the rows in the image iteratively; even-numbered rows are modified when odd-numbered rows are fixed, and odd-numbered rows are modified when even-numbered rows are fixed. (c) Each row in (b) contains four patches. There are two ways to paste each patch: by using one size a patch or four size b patches. We represent the two methods by two nodes: one for size a patches and the other for size b patches. Each node is associated with a cost, which is the MSE of the overlapping area of pasting a patch to the node. The cost is calculated by (38) and λ is set to 0. A row can be represented as a directed graph, and dynamic programming is used to select the optimal path of each row. For example, the path that passes through the solid nodes in the graph is the optimal path, which is the path of the directed graph that has the minimum total cost, summed from the nodes on the path. (d) The cost of the generated texture is calculated from the overlapping area, indicated by the solid area in the image. The multipatch algorithm generates a texture whose cost is a local minimum.

TABLE I
FROM LEFT TO RIGHT, THE FOUR UNITS ARE NUMBERED 1, 2, 3, AND 4, RESPECTIVELY. UNIT 1 IS THE ORIGINAL UNIT. THE VALUE OF THE CENTER PIXEL OF UNIT 1 IS TAKEN AS THE THRESHOLD VALUE TO LABEL THE BINARY VALUES OF THE NEIGHBORING PIXELS, AS SHOWN IN UNIT 2. UNIT 3 COMPRISES THE WEIGHTING FACTORS ASSIGNED TO EACH PIXEL IN THE UNIT. MULTIPLYING THE CORRESPONDING PIXELS IN UNITS 2 AND 3 BY THE APPROPRIATE WEIGHTS YIELDS THE LPB VALUE OF THE CENTER PIXEL IN UNIT 4, I.E., $1 + 2 + 4 + 8 + 128 = 143$

5	4	3	1	1	1	1	2	4
4	*Cp*=3	1	1		0	8		16
2	0	3	0	0	1	32	64	128

1	2	4
8	*LBP*=143	0
0	0	128

III. FORMULATION AND ANALYSIS

In this section, we describe the proposed probabilistic framework used to design an examplar-based algorithm for the compaction, synthesis, and retrieval of textures. The framework is comprised of an encoding phase and a decoding phase, as shown in Fig. 5. The encoding phase compacts and compresses an original texture into a thumbnail texture, after which the decoding phase synthesizes the compressed thumbnail texture.

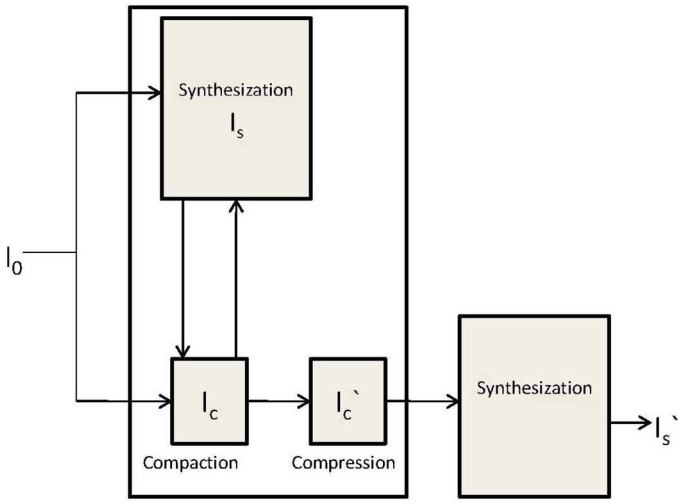

Fig. 5. Proposed system. The input texture I_o is used to generate a compacted texture I_c and a synthesized texture I_s in the encoding phase. The compacted texture is compressed to obtain I_c' before it is transmitted to the decoder, which performs texture synthesization to obtain the output texture I_s'. The two paths, $I_o \rightarrow I_c \rightarrow I_s$ and $I_o \rightarrow I_s \rightarrow I_c$, in the encoder indicate that the modification is performed on images iteratively to obtain the compacted and the synthesized textures.

A. Encoding

To retain the textural properties of the original texture in a compacted thumbnail texture, we use the approach applied in a closed loop video coding system. More specifically, the encoder contains the decoder so that it can adjust the coding parameters to obtain the optimum decoded frame. In addition to the compaction and compression components for generating a thumbnail, our encoder contains a synthesization component that can synthesize a texture from a thumbnail texture. Based on the Markov chain assumption of conditional probability, the proposed encoding algorithm iteratively modifies the synthesized and compacted textures in a probability framework. We will show that the Markov chain assumption helps us analyze the result of the encoding phase, as well as the result of the decoding phase.

Let I_o be the input texture, and let I_s (the subscript s stands for "synthesized") and I_c (the subscript c stands for "compacted") be the synthesized and compacted textures, respectively. In addition, let $P(I_s, I_c|I_o)$ be the probability that the pair of textures (I_s, I_c) can be generated from the given texture I_o. In the encoding phase, the goal is to generate the pair I_s and I_c that maximizes the probability

$$\max_{I_s, I_c} P(I_s, I_c|I_o). \tag{2}$$

Then, by using the chain rule of the probability and the Markov chain assumption $I_o \rightarrow I_s \rightarrow I_c$, we have

$$P(I_s, I_c|I_o) = P(I_s|I_o)P(I_c|I_o, I_s) \tag{3}$$
$$= P(I_s|I_o)P(I_c|I_s) \tag{4}$$

where (3) is obtained according to the chain rule of the probability, and (4) is derived by the Markov chain assumption. Similarly, based on the Markov chain assumption $I_o \rightarrow I_c \rightarrow I_s$, we have

$$P(I_s, I_c|I_o) = P(I_c|I_o)P(I_s|I_o, I_c) \tag{5}$$
$$= P(I_c|I_o)P(I_s|I_c). \tag{6}$$

We can use (4) and (6) iteratively to find the pair of textures that maximizes (2). Fig. 5 shows a diagram of the proposed encoder, which applies the following encoding algorithm.

Encoding Algorithm

Step 0: Let n be 0 and $I_s^{[n]}$ and $I_c^{[n]}$ be, respectively, the initial synthesized texture and the compacted texture.
Step 1: Generate $I_s^{[n+1]}$ by solving $\max P(I_s^{[n+1]}|I_o)$ with $I_s^{[n]}$ as the initial texture.
Step 2: Generate $I_c^{[n+1]}$ by solving $\max P(I_c^{[n+1]}|I_s^{[n+1]})$ with $I_c^{[n]}$ as the initial texture.
Step 3: Generate $I_c^{[n+2]}$ by solving $\max P(I_c^{[n+2]}|I_o)$ with $I_c^{[n+1]}$ as the initial texture.
Step 4: Generate $I_s^{[n+2]}$ by solving $\max P(I_s^{[n+2]}|I_c^{[n+2]})$ with $I_s^{[n+1]}$ as the initial texture.
Step 5: If $P(I_s^{[n+2]}, I_c^{[n+2]}|I_o) \leq P(I_s^{[n+1]}, I_c^{[n+1]}|I_o)$, terminate the process; otherwise, let $n \leftarrow n+2$ and go to **Step 1**.
End Algorithm.

Steps 1 and **2** of the above algorithm improve (4) and yield $I_s^{[1]}, I_c^{[1]}, I_s^{[3]}, I_c^{[3]}, \ldots$; while **Steps 3** and **4** improve (6) and yield $I_s^{[2]}, I_c^{[2]}, I_s^{[4]}, I_c^{[4]}, \ldots$. The algorithm iteratively improves $P(I_s, I_c|I_o)$ according to (4) and (6) until the value does not increase any further. Thus, the solution of the algorithm is the pair of textures that is the local maximum of $P(I_s, I_c|I_o)$.

The algorithm needs to calculate the maxima of $P(I_s|I_o), P(I_c|I_s), P(I_c|I_o)$, and $P(I_s|I_c)$ in **Steps 1, 2, 3**, and **4**, respectively. Next, we explain the method used to derive $P(I_c|I_s)$. We omit the derivation of the other three terms because they can be obtained in a similar manner.

1) Generalized EM Approximation: Because estimating the probability $P(I_c|I_s)$ is difficult, we introduce a hidden variable, C, for a texture feature to simplify the derivation of $\max \log\ P(I_c|I_s)$. For any function $T_\sigma(I_c, C|I_s)$, the above log probability is

$$\log\ P(I_c|I_s) = \log \sum_C P(I_c, C|I_s) \tag{7}$$
$$= \log \sum_C \frac{T_\sigma(I_c, C|I_s)}{T_\sigma(I_c, C|I_s)} P(I_c, C|I_s) \tag{8}$$
$$\geq \sum_C T_\sigma(I_c, C|I_s) \log \frac{P(I_c, C|I_s)}{T_\sigma(I_c, C|I_s)}. \tag{9}$$

The bound used in the above equations takes the form of Jensen's inequality

$$\log \sum_i t_i a_i \geq \sum_i t_i \log a_i \tag{10}$$

where $\sum_i t_i = 1$, and a_i are arbitrary scalars. The above derivation is also called the generalized EM approach [12]. The goal is to estimate $T_\sigma(I_c, C|I_s)$ and $P(I_c, C|I_s)$ jointly, so as to maximize (9). Let g be the texture feature extraction function. We define

$$T_\sigma(I_c, C|I_s) \propto \begin{cases} e^{-\frac{1}{\sigma^2}(d_i(I_s,I_c)+\lambda\, d_t(g(I_s),C))}, & \text{if } g(I_c) = C \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

where d_i denotes the cost incurred by using I_s to generate I_c; d_t measures the differences between the textural features; λ is the parameter that balances the two costs d_i and d_t; and σ is the variance of the distribution. $T_\sigma(I_c, C|I_s)$ is a function of two variables I_c and C. If the value of σ^2 is very small, then $T_\sigma(I_c, C|I_s)$ will peak at (I_c^*, C^*) with $C^* = g(I_c^*)$ and I_c^* will be the solution of

$$\min_{I_c}(d_i(I_s, I_c) + \lambda\, d_t(g(I_s), g(I_c))). \tag{12}$$

If $d_t(g(I_s), g(I_c)) = 0$, then the solution of the above equation is independent of the value of λ. Usually, the texture features $g(I_s)$ and $g(I_c)$ are different; therefore, I_c^* is λ dependent. In this case, (11) can be approximated by the 2-D Dirac function

$$T(I_c, C|I_s) = \begin{cases} 1, & \text{if } C = g(I_c) \text{ and } I_c \text{ minimizes} \\ d_i(I_s, I_c) + \lambda d_t(g(I_s), g(I_c)) \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

Then, by substituting (13) into (9), we can obtain the following approximation:

$$\sum_C T_\sigma(I_c, C|I_s) \log \frac{P(I_c, C|I_s)}{T_\sigma(I_c, C|I_s)} = \sum_C T(I_c, C|I_s) \log \frac{P(I_c, C|I_s)}{T(I_c, C|I_s)} \tag{14}$$

$$\sum_C T(I_c, C|I_s) \log P(I_c, C|I_s). \tag{15}$$

Since $T(I_c, C|I_s)$ is like a Dirac distribution and the entropy of the Dirac distribution is zero, we have $\sum_C T(I_c, C|I_s) \log T(I_c, C|I_s) = 0$ in the derivation of (15). To maximum (9), we have

$$\max_{I_c} \sum_C T_\sigma(I_c, C|I_s) \log \frac{P(I_c, C|I_s)}{T_\sigma(I_c, C|I_s)} = \max_{I_c} \sum_C T(I_c, C|I_s) \log P(I_c, C|I_s) \tag{16}$$

$$= \max_{I_c} T(I_c, g(I_c)|I_s) \log P(I_c, g(I_c)|I_s) \tag{17}$$

$$= T(I_c^*, g(I_c^*)|I_s) \log P(I_c^*, g(I_c^*)|I_s) \tag{18}$$

$$= \log P(I_c^*, g(I_c^*)|I_s). \tag{19}$$

The derivation of (16) is based on (14) and (15); and the derivations of (17), (18), and (19) are based on (13). Above derivation shows that the texture I_c^* that maximizes (13) also maximizes (9). Note that I_c^* is not the solution of $\max_{I_c} P(I_c|I_s)$. It is the optimal solution derived by the generalized EM approach for (9) with the function in (13).

From our derivations, we conclude that finding the solutions of $\max_{I_s} P(I_s|I_o), \max_{I_c} P(I_c|I_s), \max_{I_c} P(I_c|I_o)$, and $\max_{I_s} P(I_s|I_c)$ in our encoding algorithm corresponds, respectively, to finding the solutions of the following equations:

$$\min_{I_s}(d_i(I_o, I_s) + \lambda\, d_t(g(I_o), g(I_s))) \tag{20}$$

$$\min_{I_c}(d_i(I_s, I_c) + \lambda\, d_t(g(I_s), g(I_c))) \tag{21}$$

$$\min_{I_c}(d_i(I_o, I_c) + \lambda\, d_t(g(I_o), g(I_c))) \tag{22}$$

$$\min_{I_s}(d_i(I_c, I_s) + \lambda\, d_t(g(I_c), g(I_s))). \tag{23}$$

B. Decoding

The decoding algorithm simply up-synthesizes a compressed thumbnail texture I_c' to a given size. Based on our probability framework, the best estimation of both the original texture I_o and the synthesized texture I_s from the I_c' is

$$\max_{I_o, I_s} P(I_o, I_s|I_c'). \tag{24}$$

The Markov chain assumptions, $I_o \rightarrow I_s \rightarrow I_c$ and $I_o \rightarrow I_c \rightarrow I_s$, can be applied to analyze (24) [30]. According to $I_o \rightarrow I_s \rightarrow I_c$, we have

$$P(I_o, I_s, I_c) = P(I_o, I_s)P(I_c|I_o, I_s) \tag{25}$$

$$= P(I_o, I_s)P(I_c|I_s). \tag{26}$$

Similarly, from $I_o \rightarrow I_c \rightarrow I_s$, we have

$$P(I_o, I_s, I_c) = P(I_o, I_c)P(I_s|I_o, I_c) \tag{27}$$

$$= P(I_o, I_c)P(I_s|I_c). \tag{28}$$

From (26) and (28), we obtain

$$P(I_o, I_s) = P(I_o, I_c)\frac{P(I_s|I_c)}{P(I_c|I_s)} \tag{29}$$

$$= P(I_o, I_c)\frac{P(I_s)}{P(I_c)}. \tag{30}$$

Hence, for any I_c, $P(I_o, I_s)$ is factored into the product of two functions, $P(I_o, I_c)$ and $(P(I_s))/(P(I_c))$; i.e., the functions of I_o and I_s, respectively. By choosing I_c', we have

$$P(I_o, I_s) = P(I_o, I_c')\frac{P(I_s)}{P(I_c')}. \tag{31}$$

Substituting I'_c for I_c and $P(I_o, I'_c)(P(I_s))/(P(I'_c))$ for $P(I_o, I_s)$ in (26), we obtain

$$P(I_o, I_s, I'_c) = P(I_o, I'_c)\frac{P(I_s)}{P(I'_c)}P(I'_c|I_s) \quad (32)$$

$$= P(I_o|I'_c)P(I_s, I'_c). \quad (33)$$

Therefore

$$P(I_o, I_s|I'_c) = P(I_o|I'_c)P(I_s|I'_c). \quad (34)$$

Thus, for any given I'_c, we have

$$\max_{I_o, I_s} P(I_o, I_s|I'_c) = \max_{I_o} P(I_o|I'_c) \max_{I_s} P(I_s|I'_c). \quad (35)$$

The results in (35) demonstrate that the synthesized texture I_s can be estimated independently of the estimation of the original texture I_o. Thus, we can apply the procedure described in Section III-A1 to derive the synthesized texture, which is the solution of the maximum a posterior (MAP) estimation

$$\max_{I_s} P(I_s|I'_c). \quad (36)$$

Like the encoder, the decoder synthesizes a texture by minimizing the cost function

$$d_i(I'_c, I_s) + \lambda\, d_t(g(I_{c'}), g(I_s)) \quad (37)$$

from the compressed thumbnail texture I'_c. Our decoder simply synthesizes a texture from the given compressed thumbnail texture I'_c, as shown in Fig. 5.

IV. Implementation and Numerical Results

We introduced the theoretical structure of the proposed approach in Section III. In Fig. 7, we provide an implementation result, where an input texture is iteratively down-synthesized and up-synthesized to obtain a thumbnail texture, which is then compressed and used to synthesize a texture that looked similar to the input texture. Fig. 6 shows some textures derived by our approach. We now consider the issues that are critical to the approach's implementation and detail our experiment results.

A. Cost of the Multipatch Algorithm

Texture compaction and synthesis are implemented by the method used in the multipatch algorithm, but there is a slight difference in the calculation of the cost function during the patch selection phase. In the proposed model, the cost function for generating texture I_2 from texture I_1 is

$$d_i(I_1, I_2) + \lambda\, d_t(g(I_1), g(I_2)) \quad (38)$$

where d_i and d_t are, respectively, the cost of using I_1 to generate I_2 and the distance between the texture features of I_1 and I_2. The cost function d_i is defined as the mean squared error of the total overlapped area in the image domain when I_2 is generated [see Fig. 4(d)]. The cost function d_t is defined as the the Kullback-Leibler divergence of the histograms $g(I_1)$ and $g(I_2)$, where g is the LBP feature extraction function, and the parameter λ balances the two terms. Since we use color textures, the total cost of generating I_2 is the sum of three costs, obtained independently by generating the R, G, and B channels of I_2 from the corresponding channels of I_1.

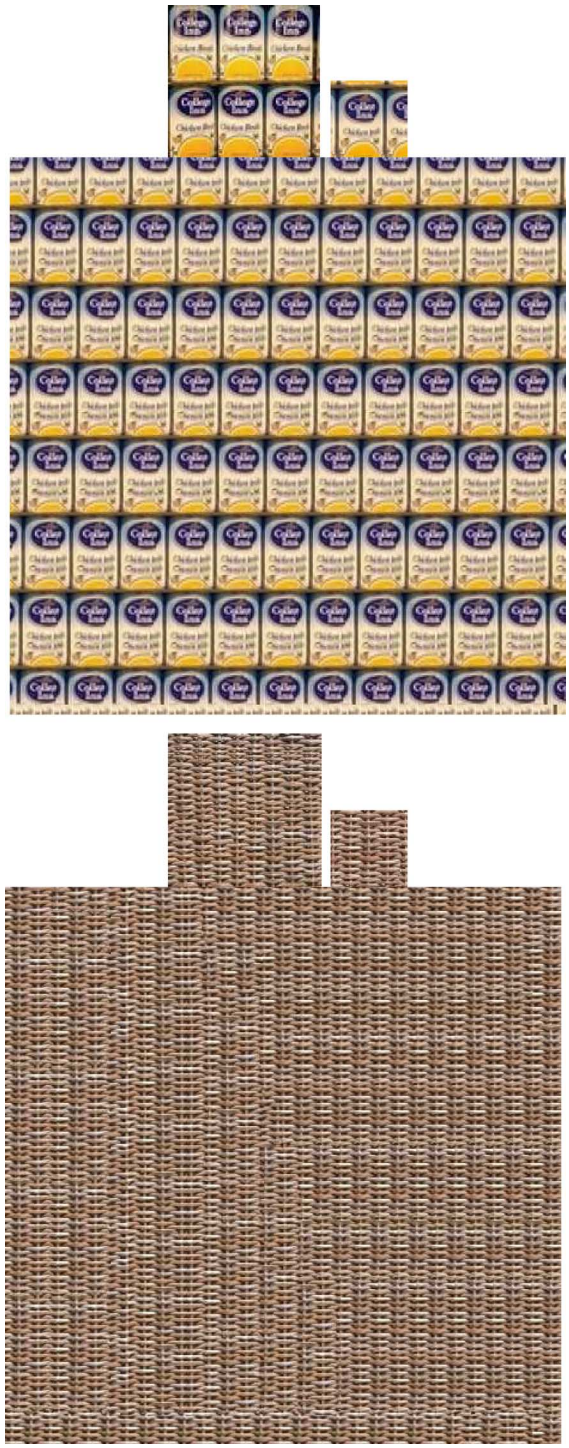

Fig. 6. Synthesized textures of size 464 × 464 (on the right) and the compressed compacted texture of size 64 × 64 (in the middle), where the synthesized textures are generated from the compressed compacted textures. Left: The original 128 × 128 textures. The top texture is from the Outtex database and the bottom texture is from the MIT VisTex database.

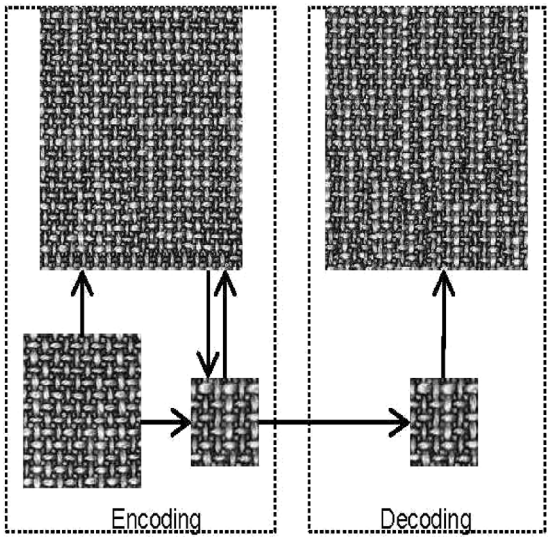

Fig. 7. Some textures generated by our method. The compacted thumbnail texture (size 64 × 64) and the synthesized texture (size 256 × 256) are generated from the input texture (size 128 × 128) in the encoding phase. The thumbnail compressed at a quality factor of 60% by JPEG 2000 and the synthesized texture (size 256 × 256) are shown in the decoding phase. Note that the synthesized textures in the encoder and the decoder are the same size. The texture is from the Outtex database.

B. Patch Domain (Patch Pool)

The sizes of the patches used for texture compaction and texture synthesis are 16 × 16 and 32 × 32, respectively [see Fig. 4(a)]. We sample patches of the above sizes from the input texture; that is, any patch of the above sizes whose top-left coordinate is dividable by α is sampled and added to the patch domain. We studied the parameter α by assigning its value as either 2 or 4. Clearly, sampling with α assigned to 2 in the patch domain will generate 4 times more patches than using α assigned to 4. Usually, we assign α to 4 to stabilize the algorithm when the input texture is 128 × 128, and assign it to 2 when the size of the input texture is 64 × 64.

C. Texture Compaction

When implementing Markovian chain iterations, we take the input image I_o and initialize the synthesized image $I_s^{[0]}$ and the compacted image $I_c^{[0]}$ to the sizes required by the patchwork algorithm. The multipatch algorithm is then applied to modify $I_s^{[0]}$ with I_o, and the generated $I_s^{[1]}$ is then used to modify $I_c^{[0]}$ and so on, as described in the **Encoding Algorithm** in Section III-A. Empirically, we find that our encoding algorithm usually terminates in less than three iterations. The sizes of our patches are 16 × 16 and 32 × 32. In our experiments on various input textures of size 128 × 128, we found that if the compacted texture was smaller than 64 × 64, then the number of candidate patches in the patch pool of the compacted texture was insufficient to synthesize a texture larger than 464 × 464. As a result, our algorithm tends to synthesize a texture with repeated patterns when the compacted texture is smaller than 64 × 64 and the synthesized texture is larger than 464 × 464. We experimented with different λ values and found that the perceptual quality of the synthesized textures was poor when a λ value was greater than 0.1. Thus, we sampled with λ values ranging from 0.02 to 0.1. We found that by setting the value of λ to 0.08, the thumbnail textures were perceptually better than those derived by using the other values.

D. Texture Compression and Performance Comparisons

We use a subjective test to evaluate the perceptual quality of the compressed and synthesized textures derived by our approach. In the encoder, the original texture is compacted to a thumbnail, which is then compressed and transmitted to the decoder, where it is synthesized to the required size. The compacted thumbnail texture is compressed by JPEG 2000, which uses the quality factor parameter to compress an image. In our experiment, to determine the quality factor for our system, we placed the compressed and uncompressed textures on a screen side by side so that they were displayed with the same screen luminance condition. The screen was a DEL LCD display with a resolution of 1280 × 1024 pixels with 8 bits/color/pixel; the diagonal dimension was 20 in and the screen luminance was 305.0 lux. We asked eight subjects to score each compressed texture as either *a good quality texture* or *a poor quality texture*. None of the subjects were color blind; and none were aware of the purpose of the experiment, or familiar with the image processing methods. Sitting no more than half a meter from the computer screen, each subject was asked to make a decision about each texture within 30 s. Fig. 8 shows the nine textures used in the experiments. They are taken from the Outtex database [31] and the MIT Vision Texture database [32]. The results shown in Fig. 9 indicate that the probability of deriving good quality textures is very high when the quality factor of JPEG 2000 is set at 60% or above for textures of size 464 × 464 and 64 × 64. Thus, we choose 60% as the quality factor in our study, since compressing a texture with a quality factor equal to or above that level will produce a compressed texture that is perceptually the same as the original texture.

The top sub-figure of Fig. 10 shows the number of bytes used to encode a compressed colored thumbnail texture of size 64 × 64 with different quality factors. The thumbnail texture was obtained by our encoding algorithm with the parameters described in Subsections A, B and C above. The sizes of the original texture, the thumbnail texture, and the synthesized texture were 128 × 128, 64 × 64, and 464 × 464, respectively. Note that at a quality factor of 60%, the number of bytes declines from 12.28 to 1.6 KB, which is about a factor of 8. The bottom sub-figure of Fig. 10 shows a 464 × 464 texture compressed by JPEG 2000 at different quality factors. The number of bits is only reduced by a factor of 8 at a quality factor of 60%. Since the size of the original texture is 464 × 464, which is approximately 50 times larger than the 64 × 64 compacted texture, and both the original and compacted textures are compressed by JPEG 2000 by a factor of 8 at a quality factor of 60%, our approach achieves approximately 50 times more bit reduction than that derived by using JPEG 2000. Moreover, the number of bits in the compressed thumbnail is $400 (= 8 * 50)$ times lower than that in the original texture. However, the tremendous saving in storage space and the reduction in the number of transmission bits is achieved at the cost of sacrificing the MSE error. If the texture quality is measured in terms of the MSE error, then a texture

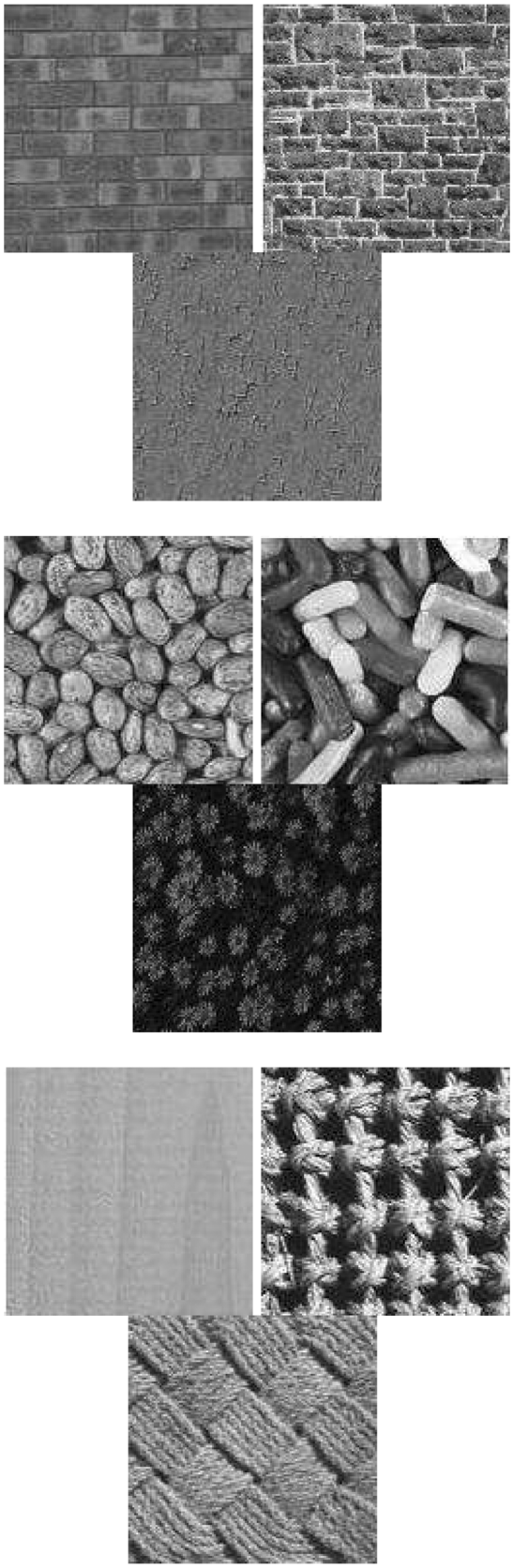

Fig. 8. We use OT to denote the Outtex database, and MIT to denote the MIT Vistex. Top left: Bricks, available in OT and MIT. Top middle: Bricks, available in OT. Top right: Wood, available in OT and MIT. Middle left: Food, available in OT and MIT. Middle left: Food, available in MIT. Middle right: Flowers, available in OT. Bottom left: Wood, available in OT and MIT. Bottom middle: Fabric, available in OT and MIT. Bottom left: Fabric, available in OT and MIT.

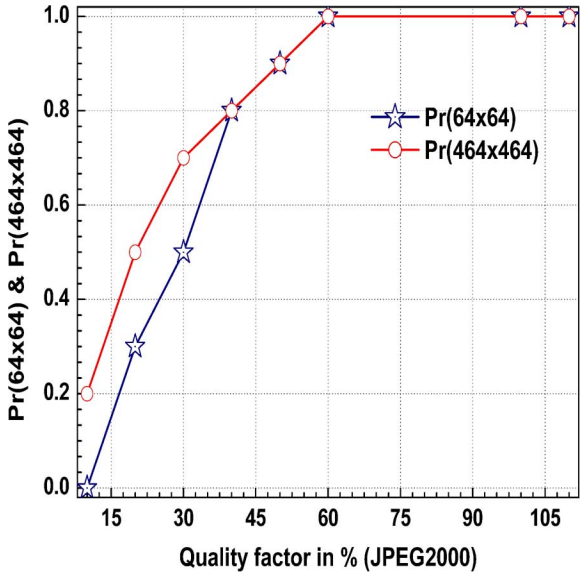

Fig. 9. Probability of generating a high quality texture, derived from the average scores of eight subjects, when a texture of size 64×64 or 464×464 is compressed at different quality factors by JPEG 2000. When the qualify factor is equal to or above 60%, the probability of generating a high quality compressed texture is 1, which implies that the compressed texture is perceptually the same as the original texture.

compressed by applying JPEG 2000 directly will definitely contain far fewer MSE errors than a texture compressed by our approach, as shown in Fig. 11.

E. Decoder Performance

Our system's decoder uses the multipatch algorithm to synthesize a texture from a compressed thumbnail texture. The decoder's parameters are given in Subsections A and B above. The λ value used to calculate the cost function is 0.08, which is the same as that used for encoding. Many examplar-based texture synthesization methods use randomization to create variations of a synthesized texture. The examplar-based multipatch algorithm is also a random algorithm that chooses a patch at random from a candidate set to paste on to the synthesized texture. Performance comparisons of various randomized texture-synthesization algorithms are detailed in [15] and given in Subsection F below. The performance benchmark that we use to evaluate our decoder is based on a subjective test, which measures the probability of deriving acceptable and good quality textures synthesized from a compressed thumbnail texture. The original 464×464 texture generated by our encoder was placed on a screen with the 464×464 texture synthesized by our decoder. The eight subjects who participated in the subjective evaluation described in Subsection D were asked to rate the quality of each decoder-synthesized texture as either *good* or *acceptable*. The other parameters of our subjective test are the same as those reported in Subsection D.

F. Our Thumbnails Versus Cropped Thumbnail Textures and Patchwork's Thumbnails

Intuitively, one might think that it is not necessary to compact a homogeneous texture in the encoding phase, since we could

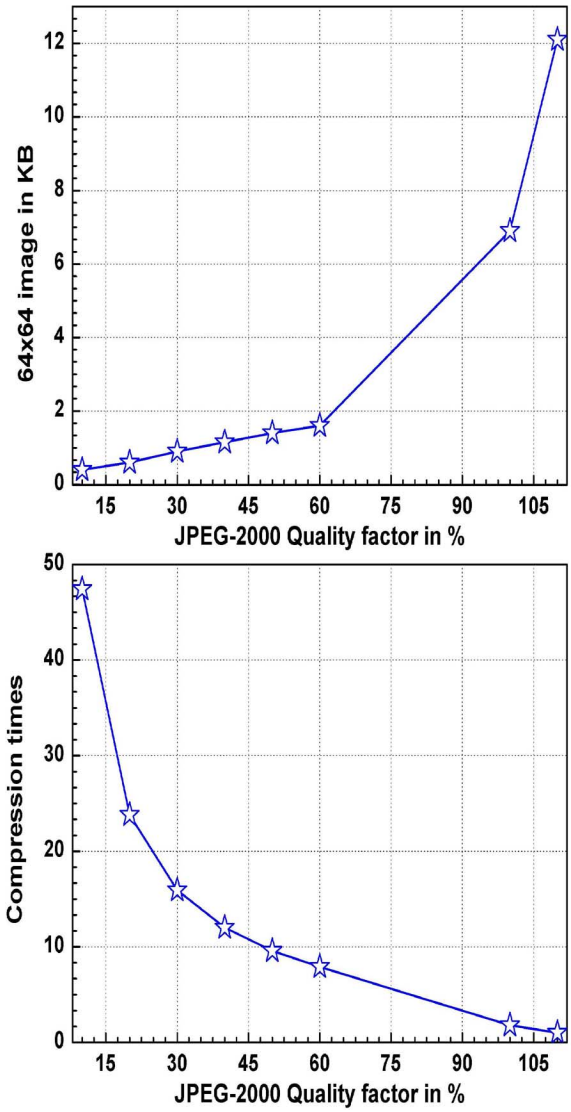

Fig. 10. Top: The compression value in KB of 64 × 64 thumbnail images at each quality factor ranging from 10%–100%. The quality factor greater than 100% relates to the original image, which is not compressed. Bottom: the average compression ratio of compressing a 464 × 464 texture by JPEG 2000 at different quality factors. Note that the 60% quality factor is the maximum compression for generating a texture without perceptual distortion, but the bit reduction is only eight times that of the original texture.

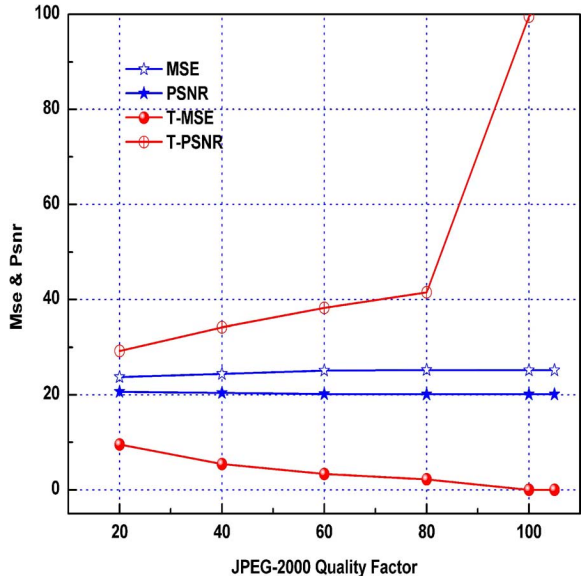

Fig. 11. Comparison of the MSE (mean squared error) and PSNR (peak-signal-to-noise ratio) of different approaches. T-MSE and T-PSNR are the results of compressing 464 × 464 textures with different quality factors by JPEG 2000. The MSE and PSNR are the results of our approach when the compacted 64 × 64 textures, compressed at different quality factors, are synthesized to textures of size 464 × 464. These results are the average of MSE and PSNR of the nine textures in Fig. 8.

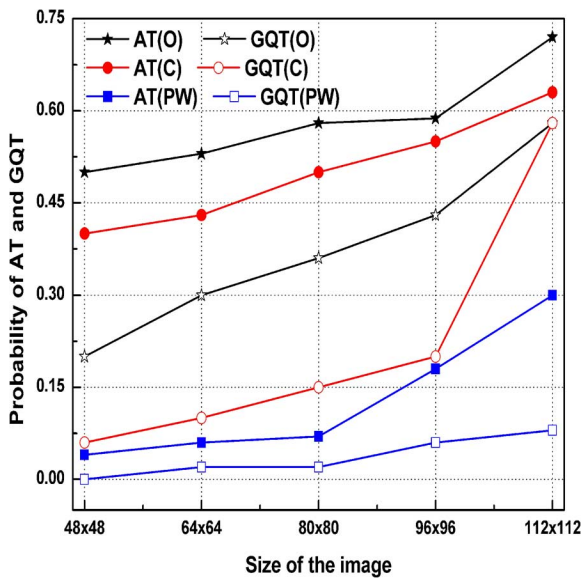

Fig. 12. Average probabilities of generating acceptable (AP) and good quality textures (GQT) synthesized by cropped (C), our compacted (O), and patchwork compacted (PW) textures, whose sizes range from 48 × 48 to 112 × 112. For each texture, we generated five synthesized textures. We also used the nine textures in Fig. 8. Note that our approach has a higher probability of generating both acceptable and high quality textures. The parameters of the patchwork algorithm are 16 × 16 and $\alpha = 4$, as described in Section IV-B.

simply crop the original texture to the required size [5]. Moreover, our approach's encoder may be considered too complex because we could simply use the patchwork algorithm to generate a compacted texture. To compare the three methods, we used our thumbnail textures, cropped thumbnail textures, and patchwork's thumbnail textures to synthesize images, and asked our subjects to rate the quality of each texture as either *good* or *acceptable*. In the experiment, all the synthesized textures were 464 × 464, while the thumbnail textures were in five different sizes compressed by JPEG 2000 at a quality factor of 60%. The cropped texture was taken from the center and the four corners of the input texture, as shown in Fig. 3. Fig. 12 shows that the average probabilities of obtaining acceptable and high quality textures vary as the sizes of cropped and compacted textures increase. The textures synthesized from our compacted textures have higher probabilities of generating both acceptable and high quality textures than those synthesized from the cropped textures and the patchwork algorithm's compacted textures. For a thumbnail of size 64 × 64, the textures synthesized by a decoder had a 30% probability of being high quality and a 50%

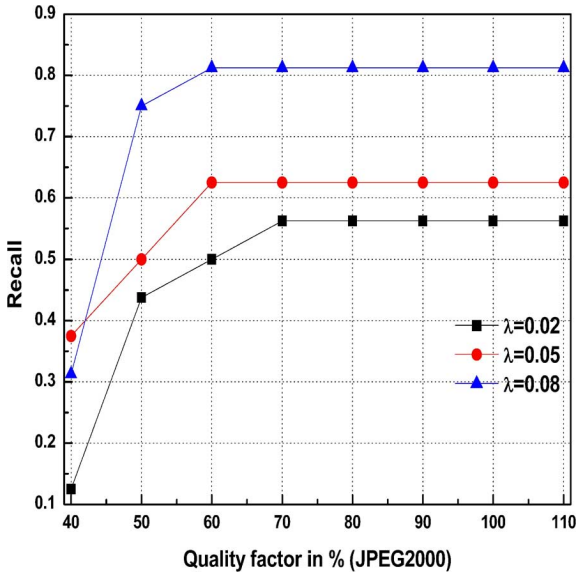

Fig. 13. Average recall, obtained by averaging seven query textures, versus the quality factor of JPEG 2000. The recall is maximized when $\lambda = 0.08$. When $\lambda > 0.1$, the perceptual quality of the synthesized images is poor; thus, we experimented with $\lambda = 0.02$, 0.05, and 0.08.

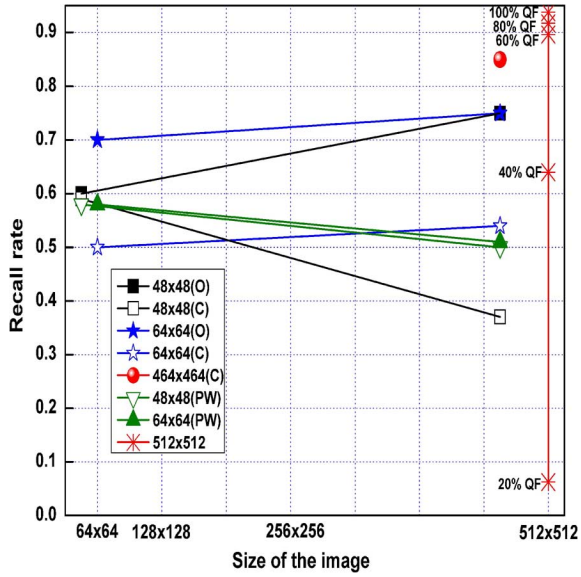

Fig. 14. Comparison of the average recall rates for the retrieval of our textures (O), the retrieval of the patchwork's compacted texture (PW), and the retrieval of cropped textures (C). In the decoder, a smaller-sized texture of size 48 × 48 or 64 × 64 compressed by JPEG 2000 at a quality factor of 60% is synthesized to a texture of size 464 × 464. We use line segments to connect the recall rates of two textures of different sizes. The rightmost end point of a segment indicates that the 464 × 464 texture is synthesized from the texture at the leftmost end point of the segment. Clearly, the recall rate of a 464 × 464 texture synthesized from our thumbnail textures is better than that of a 464 × 464 texture synthesized from any of the different-sized cropped textures and the patchwork algorithm's compacted textures. In addition, the recall rates of the 464 × 464 textures synthesized from our thumbnail textures are almost the same; however, there is a large difference in the recall rates of the 464 × 464 textures synthesized from different sized cropped textures. The compressed 464 × 464 cropped texture [464 × 464(C)] is not synthesized; instead, it is cropped from the original 512 × 512 texture. We use its recall rate as our reference point for the retrieval of the synthesized textures of that size. The rightmost line represents the retrieval results of using the compressed original textures at different qualify factors of JPEG 2000. The line is used as a reference to measure how much retrieval information is lost during the compaction process. Compared to the reference line, the recall rate of our compaction algorithm is reduced by 0.15 at a qualify factor of 60%.

probability of being acceptable. The other interesting measurement reported in [15] is the expected number of synthesizations required for a decoder to successfully synthesize an acceptable or a high quality texture. The concept of the expected number of synthesizations can be explained easily by the experiment of tossing a biased coin, with a probability p of coming up HEADS (SUCCESS) and a probability $1 - p$ of coming up TAILS (FAIL). The expected number of synthesizations refers to the average number of synthesizations required for the coin to come up HEADS the first time. This can be derived by the geometric distribution

$$p + 2p(1-p) + 3(1-p)^2 p + \cdots = \frac{1}{p}. \tag{39}$$

Thus, the expected number of synthesizations required for our decoder to generate an acceptable 464 × 464 texture from a compressed 64 × 64 texture is 2, and for a high quality texture it is $(10)/(3)$.

G. Texture Retrieval

We now present an analysis of the texture retrieval approach. The cost function in (38) is comprised of two terms, d_i and d_t, which are balanced by the parameter λ. The term d_i represents the perceptual quality of the generated texture, while the term d_t denotes the distance between the textural features of the generated texture and those of the original texture. The cost function d_t facilitates texture retrieval because relevant textures usually have similar features; therefore, their d_t is small. In general, it is difficult to determine the value of λ analytically, so we derive it from our experiments.

We took 40 textures of size 512 × 512 from the categories of bricks, fabrics, wood, food, flowers, and tiles in the MIT Vision Texture database and divided each one into 16 nonoverlapping sub-images. Thus, our texture set contained a total of 640 sub-images. The query textures, which must belong to one of the above categories, are taken from the database. For each query, the top-32 textures whose RGB-colored LBP features are most similar to those of the query texture are retrieved.

Recall and precision are usually used to measure the retrieval performance. We use A and B to denote the set of relevant images and the set of retrieved images respectively. Let the number of relevant images be $\#A = a + b$, where a is retrieved and b is not retrieved; and let $\#B = a + c$ be the number of retrieved images, where a is relevant and c is irrelevant. Recall is defined as $P(B|A)$; hence, it is formulated as $(a)/(a + b)$. Precision is defined as $P(A|B)$; hence, it is formulated as $(a)/(a + c)$. If $\#A$ and $\#B$ are fixed, it is easy to derive that the recall is $(\#B)/(\#A)$ times the precision. Since the number of relevant images in our experiment is 16 and the number of retrieved images is 32, the recall rate is twice the precision rate. Thus, we only discuss the experiment results for the recall metric. Fig. 13 shows the recall rates using different λ values. At $\lambda = 0.08$, the recall value is at the maximum. Thus, we used $\lambda = 0.08$ in the retrieval experiment.

Our final experiment compared the retrieval performance of different-sized textures obtained by various methods. The sizes

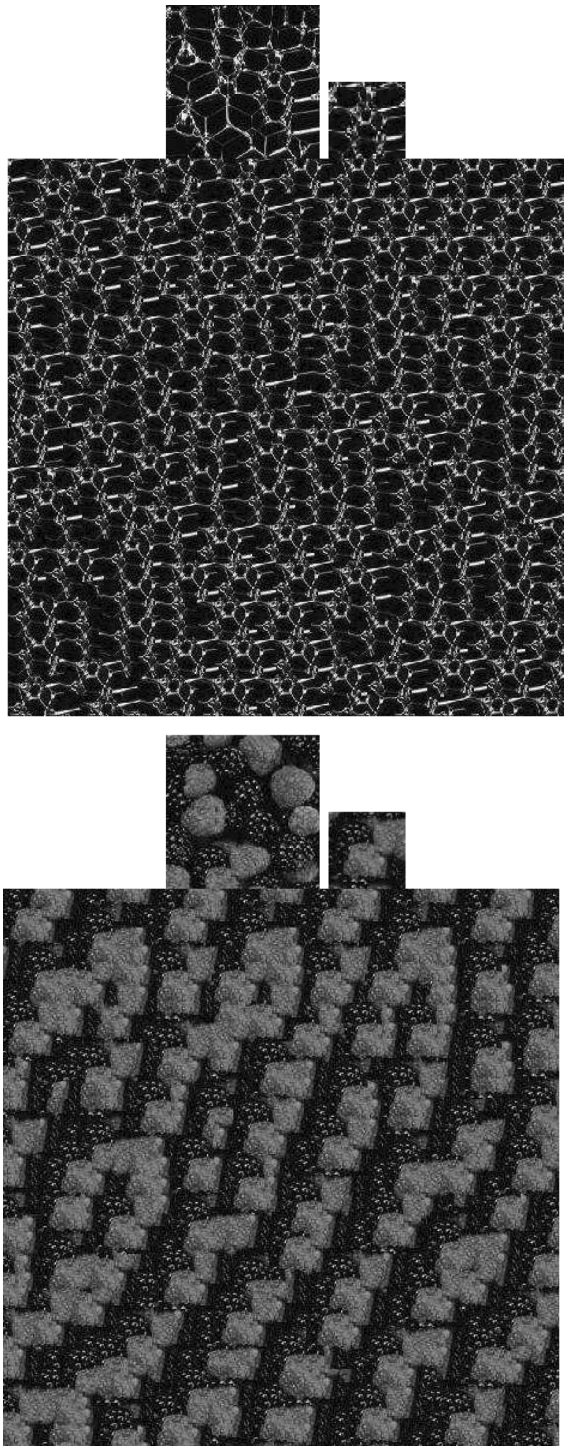

Fig. 15. Examples of a successful result and a failed result returned by the proposed algorithm. Top: The successful result. Bottom: The failed result. Both textures are from the Outtex database. Left: The original 128 × 128 texture. Middle: The 64 × 64 compressed and compacted texture. Right: The 464 × 464 synthesized texture.

of the thumbnail textures were 48 × 48 and 64 × 64, and they were compressed by JPEG 2000 at a quality factor of 60%. The size of the synthesized textures was 464 × 464. The average recall rates for the thumbnail textures and synthesized textures obtained by different methods are shown in Fig. 14. The recall results on the rightmost line of Fig. 14 are the references that we use to measure how much information is lost when compacting a texture by different methods. Without considering the reference recall values, the 464 × 464 cropped textures compressed at a quality factor of 60% achieved the highest recall, which is reasonable because that size retains the most information about the original texture. The 464 × 464 textures generated from 48 × 48 and 64 × 64 thumbnail textures by our method yielded the second highest recall rates. From the results shown in Fig. 14, it is clear that incorporating LBP features improves the retrieval performance of both compacted and synthesized textures.

H. Limitation

It is always useful to specify the limitation of a texture processing algorithm since textures are so difficult to represent and analyze. The proposed algorithm works well on periodic textures, but it is not effective on all nonperiodic textures, as shown in Fig. 15. It would be interesting to develop an algorithm that could work on nonperiodic textures. Some synthesization results on near-regular textures are reported in [33].

V. Conclusion

We propose a novel approach that integrates texture features for retrieval into an examplar-based texture compaction and synthesis approach, and show that the approach can be implemented by using the generalized EM algorithm. To validate the performance of our approach, we use the LBP feature and the multipatch algorithm. Our experiment results show that the proposed approach can achieve 50 times more bit reduction than that achieved by using JPEG 2000 to compress the input texture. We also compare the retrieval performance of textures derived by our method, by the patchwork algorithm, and by cropping the original texture. The recall results show that our method outperforms the other methods. The proposed approach is promising because it requires a much smaller space to represent a large texture and the representation is also very effective for texture retrieval. In future studies, we will incorporate more texture applications into the examplar-based approach and also improve our algorithm's performance on inhomogeneous textures.

Acknowledgment

The authors would like to thank the anonymous reviewers for their insightful comments and suggestions.

References

[1] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-based image retrieval at the end of the early years," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 12, pp. 1349–1380, Dec. 2000.

[2] R. Xu, S. N. Pattanaik, and C. E. Hughes, "High-dynamic-range still-image encoding in JPEG 2000," *IEEE Comput. Graph. Appl.*, vol. 25, no. 6, pp. 57–64, Nov./Dec. 2005.

[3] K. Roimela, T. Aarnio, and J. Itäranta, "High dynamic range texture compression," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 707–712, 2006.

[4] J. Ström and T. Akenine-Möller, "iPACKMAN: High-quality, low-complexity texture compression for mobile phones," in *Proc. ACM SIGGRAPH/EUROGRAPHICS Conf. Graphics Hardware*, Los Angeles, CA, 2005, pp. 63–70.

[5] L.-Y. Wei, J. Han, K. Zhou, B. Guo, and H.-Y. Shum, "Inverse texture synthesis," *ACM Trans. Graph.*, vol. 27, no. 3, Aug. 2008.

[6] J. Lu, A. S. Georghiades, A. Glaser, H. Wu, L.-Y. Wei, B. Guo, J. Dorsey, and H. Rushmeier, "Context-aware textures," *ACM Trans. Graph.*, vol. 26, no. 1, Jan. 2007.

[7] A. P. Berman and L. G. Shapiro, "Efficient content-based retrieval: Experimental results," in *Proc. IEEE Workshop on Content-Based Access of Image and Video Databases*, Jun. 1999, pp. 55–61.
[8] M. N. Do and M. Vetterli, "Wavelet-based texture retrieval using generalized Guassian density and kullback-leibler distance," *IEEE Trans. Image Process.*, vol. 11, no. 2, pp. 146–158, Feb. 2002.
[9] M. S. Lew, "Next-generation web searches for visual content," *Computer*, vol. 33, no. 11, pp. 46–53, Nov. 2000.
[10] N. Vasconcelos and A. Lippman, "Library-based image coding," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, 1994, vol. 2, pp. 489–492.
[11] N. Vasconcelos and A. Lippman, "Library-based coding: A representation for efficient video compression and retrieval," presented at the Data Compression Conf., Snowbird, UT, 1997.
[12] B. J. Frey, *Graphical Models for Machine Learning and Digital Communication*. London, U.K.: The MIT Press, 1997, 1998.
[13] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," in *Proc. Int. Conf. Computer Vision*, Sep. 1999, pp. 1033–1038.
[14] T. Ojala and M. Pietikäinen, "Unsupervised texture segmentation using feature distributions," *Pattern Recognit.*, vol. 32, pp. 477–486, 1999.
[15] L.-Y. Lai, W.-L. Hwang, and P. Sreedevi, "Performance evaluation of a novel sampling-based textures synthesis technique using different sized patches," *Signal Image Video Process.*, vol. 2, no. 3, pp. 275–286, Sep. 2008.
[16] L. Liang, C. Liu, Y. Q. Xu, B. Guo, and H. Y. Shum, "Real-time texture synthesis by patch-based sampling," *ACM Trans. Graph.*, vol. 20, no. 3, pp. 127–150, Jul. 2001.
[17] D. J. Heeger and J. R. Bergen, "Pyramid-based texture analysis/synthesis," in *Proc. 22nd Annu. Conf. Computer Graphics and Interactive Techniques*, New York, 1995, pp. 229–238.
[18] G. Jacovitti, A. Neri, and G. Scarano, "Texture synthesis-by-analysis with hard-limited Gaussian process," *IEEE Trans. Image Process.*, vol. 7, no. 2, pp. 1615–1621, Nov. 1998.
[19] P. Campisi, D. Hatzinakos, and A. Neri, "A perceptually lossless, model-based, texture compression technique," *IEEE Trans. Image Process.*, vol. 9, no. 8, pp. 1325–1336, Aug. 2000.
[20] R. Paget and I. D. Longstaff, "Texture synthesis via a noncausal nonparametric multiscale Markov random field," *IEEE Trans. Image Process.*, vol. 7, no. 6, pp. 1–9, Jun. 1998.
[21] J. Portilla and E. P. Simoncelli, "A parametric texture model based on joint statistics of complex wavelet coefficients," *Int. J. Comput. Vis.*, vol. 40, no. 1, pp. 49–71, Dec. 2000.
[22] X. Qin and Y. H. Yang, "Aura 3D textures," *IEEE Trans. Vis. Comput. Graph.*, vol. 13, no. 2, pp. 379–389, Mar.–Apr. 2007.
[23] L.-Y. Wei and M. Levoy, "Fast texture synthesis using tree-structured vector quantization," in *Proc. 27th Annu. Conf. Computer Graphics and Interactive Techniques*, New York, 2000, pp. 479–488.
[24] A. A. Efros and W. T. Freeman, "Image quilting for texture synthesis and transfer," in *Proc. 28th Annu. Conf. Computer Graphics and Interactive Techniques*, New York, 2001, pp. 341–346.
[25] E. Levina and P. J. Bickel, "Texture synthesis and nonparametric resampling of random fields," *Ann. Statist.*, vol. 34, no. 4, pp. 1751–1773, 2006.
[26] J.-F. Aujol, S. Ladjal, and S. Masnou, Exemplar-Based Inpainting From a Variational Point of View, CAM Rep. 09-04, UCLA, 2009.
[27] R. W. Picard, "A society of models for video and image libraries," *IBM Syst. J.*, vol. 35, no. 3 & 4, pp. 292–312, 1996.
[28] M. A. Akhloufi, X. Maldague, and W. B. Larbi, "A new color-texture approach for industrial products inspection," *J. Multimedia*, vol. 3, no. 3, pp. 44–51, Jul. 2008.
[29] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–988, Jul. 2002.
[30] S. L. Lauritzen, *Graphical Models*. Oxford, U.K.: Clarendon, 1996.
[31] *OuTex Database*, [Online]. Available: http://www.outex.oulu.fi/
[32] *MIT Media Lab. Vision Texture—VisTex Database*, [Online]. Available: http://www-white.media.mit.edu/vismod/imagery/Vision-Texture/vistex.html
[33] Y. Liu, W.-C. Lin, and J. H. Hays, "Near-regular texture analysis and manipulation," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 368–376, Aug. 2004.

Paruvelli Sreedevi received two M.S. degrees in education and communication engineering from Osmania Unversity, Hyderabad, India, and from the Graduate Institute of Communication Engineering Department, National Taiwan University, Taipei, Taiwan, R.O.C., in 1994 and 2005, respectively.

From November 2000 to December 2005, she was a Research Assistant on switch design of an optical router in the communication area at the National Taiwan University, Graduate Institute of Communication Engineering Department, Taipei. From January to July 2006, she was with the Networking Lab, Academia Sinica, Taipei, for a short period of time working on delayed torrent networks, and in July 2006, she shifted to the Multi Media Technology Lab, working on video/image coding and processing areas.

Wen-Liang Hwang received the B.S. degree in nuclear engineering from National Tsing Hua University, Hsinchu, Taiwan, R.O.C., the M.S. degree in electrical engineering from the Polytechnic Institute of New York, New York, and the Ph.D. degree in computer science from New York University in 1993.

He was a postdoctoral researcher with the Department of Mathematics, University of California, Irvine, in 1994. In January 1995, he became a member of the Institute of Information Science, Academia Sinica, Taipei, Taiwan, R.O.C., where he is currently a Research Fellow. He is the coauthor of the book: *Practical Time-Frequency Analysis* (Academic Press, 1998). He is currently an associate editor of the *Journal of Wavelet Theory and Applications* and the *International Journal of Wavelets, Multiresolution and Information Processing*. His research interests include wavelet analysis, signal and image processing, and multimedia compression and transmission. In 2001, he was awarded the Academia Sinica Research Award for Junior Researchers.

Shawmin Lei (S'87–M'88–SM'95–F'06) received the B.S. and M.S. degrees from the National Taiwan University, Taipei, Taiwan, R.O.C., in 1980 and 1982, respectively, and the Ph.D. degree from the University of California, Los Angeles, in 1988, all in electrical engineering.

From August 1988 to October 1995, he was with Bellcore (Bell Communications Research), Red Bank, NJ, where he had worked mostly in video compression and communication areas and for a short period of time in wireless communication areas. From October 1995 to March 2007, he was with Sharp Laboratories of America, Camas, WA, where he was a manager of the Video Coding and Communication Group. Since March 2007, he has been with MediaTek, Hsinchu, Taiwan, as a Director in the Advanced Technology Division, working in video/image coding and processing areas. His research interests include video/image compression, processing and communication, picture quality enhancement, multimedia communication, and data compression. He has published more than 60 peer-reviewed technical papers and more than 40 contributions to MPEG4, JPEG2000, H.263+, and H.264 international standard meetings. He has been awarded more than 35 patents.

Dr. Lei received the Best Paper Award (corecipient) from the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY in 1993. He was elevated to an IEEE Fellow in 2006.