EDGE-BASED INPAINTING AND TEXTURE SYNTHESIS FOR IMAGE COMPRESSION

*Dong Liu**

Dept. of Electronic Science and Technology
University of Science and Technology of China
Hefei 230026, China
Email: liud@mail.ustc.edu.cn

Xiaoyan Sun, Feng Wu

Internet Media Group
Microsoft Research Asia
Beijing 100080, China
Email: {xysun, fengwu}@microsoft.com

ABSTRACT

Towards visual quality rather than pixel-wise fidelity, we propose an image coding scheme integrated with edge-based inpainting and texture synthesis. In this scheme, an original image is analyzed at encoder side so that some blocks are removed during encoding. The edges related to these removed blocks will be compressed and transmitted. At decoder side, we propose an image restoration method, which consists of edge-based inpainting and texture synthesis, in order to fully utilize the transmitted edges and naturally restore the removed blocks. Experimental results show that our scheme can achieve up to 32% bit-rate saving at similar visual quality levels, compared with H.264/AVC intra coding.

1. INTRODUCTION

Mainstream image and video compression schemes, exemplified by JPEG2000 and H.264/AVC, mainly aim at optimizing pixel-wise fidelity such as peak signal-to-noise ratio (PSNR) given the bit-rate budget. It has been noticed that PSNR is not always a good metric for the visual quality of reconstructed images, while the latter is regarded as the ultimate objective of compression schemes [1]. Therefore, many efforts have been made to design compression systems towards visual quality, in which some image analysis tools such as segmentation and texture modeling are utilized to remove the perceptual redundancy [2].

Recently, there is emerging a new direction to exploit perceptual redundancy and to improve visual quality. This approach is inspired by the remarkable progresses in image inpainting and texture synthesis [3–6]. The basic idea is to remove some image regions at encoder, and to restore them at decoder by inpainting or synthesis methods. A straightforward realization of this idea is proposed in [7]. Moreover, since source images are available at encoder, various distinctive features can be extracted from removed regions and transmitted as assistant information, which may greatly empower the inpainting or synthesis methods [8]. Here, the assistant information can be regarded as a compact description of some image regions. From the inpainting point of view, assistant information makes inpainting a *guided* optimization for visual quality instead of a *blind* optimization.

In this paper, we try to investigate the capabilities of inpainting as well as synthesis in image compression, given edges as assistant information. We propose an image coding scheme, in which some blocks are removed at encoder side but the edges relating to them are transmitted. At decoder side, we design edge-based inpainting and employ texture synthesis for our scheme, in order to fully utilize edges to restore the removed blocks. Experimental results demonstrate the efficiency of our scheme in terms of great reduction in bit-rate.

The remainder of this paper is organized as follows. In Section 2 we introduce the entire scheme. Section 3 concentrates on the edge extraction and block removal methods. Section 4 describes edge-based inpainting and texture synthesis in our scheme. Section 5 provides experimental results and Section 6 concludes this paper.

2. OUR PROPOSED IMAGE CODING SCHEME

Fig. 1 depicts our proposed image compression scheme, in which inpainting and synthesis methods are integrated with normal image encoder/decoder such as JPEG codec. In our scheme, edge extraction is first performed on the original image. Then, according to exemplar selection, some blocks will be removed and the others will be encoded. Here, the coded blocks are called *exemplars* because they will be used as exemples in inpainting and synthesis. For the removed blocks, corresponding edges will be encoded and transmitted. At decoder side, edge-based inpainting and texture synthesis are

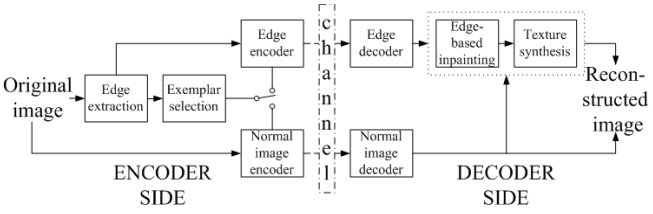

Fig. 1. Our proposed image coding scheme.

*This work has been done while D. Liu is with Microsoft Research Asia as an intern.

performed successively, both of which utilize edges and exemplars to restore the removed blocks.

Our scheme can be viewed as a special case of the general framework proposed in [8]. Simple edges, which only record the locations of edge pixels, are regarded as the assistant information for inpainting or synthesis for the following reasons. First, edges are crucial to the restoration of salient structures in images, as have been analyzed with some mathematical models [4]. Second, edges are concise and easy to be described in the compressed fashion. Last but not least, edge information is a low-level image feature which can be easily extracted compared with some semantic image features, so the adoption of edges enables a fully-automatic system.

3. EDGE EXTRACTION AND EXEMPLAR SELECTION

3.1. Edge Extraction

At encoder side, edges are first detected from the original image using the algorithm presented in [9]. Then, for the purpose of efficient compression, we propose a thinning process which condenses edges into one-pixel-width curves. In this process, we consider the following factors: the second derivative of each edge pixel calculated by the Laplacian operator; the difference in values between neighboring edge pixels; and the curvature of the edge at each pixel. These three terms are minimized to yield a smooth edge whose intensity is also continuous. Details of the thinning process are not presented due to page limitations.

3.2. Exemplar Selection

After edges are extracted, exemplar selection is performed to remove some blocks from an original image, and the remaining ones are regarded as exemplars which will be coded by normal image encoder. The exemplar selection is conducted at non-overlapped 8×8 block level. Each 8×8 block is classified into *structural* or *textural*. In specific, if a block contains more than one-quarter pixels which are located within a short distance (e.g. 5-pixel) from the edges, it will be regarded as structural, otherwise textural. Two kinds of blocks are processed independently.

Structural exemplar selection The selection of exemplars from structural blocks is accomplished in two steps. First, some blocks are regarded as *necessary* because inpainting can hardly restore them. Second, some *additional* blocks are selected in order to improve the visual quality of reconstructed images.

As illustrated in Fig. 2 (a) by the white blocks, the blocks located at the end points or conjunctions of edges are regarded as necessary blocks, because they contain the transition between different image partitions and thus are hard to be restored by inpainting. For a circle edge, two necessary blocks are identified in its inner and outer regions, respectively. These two are selected to provide exemples for inpainting.

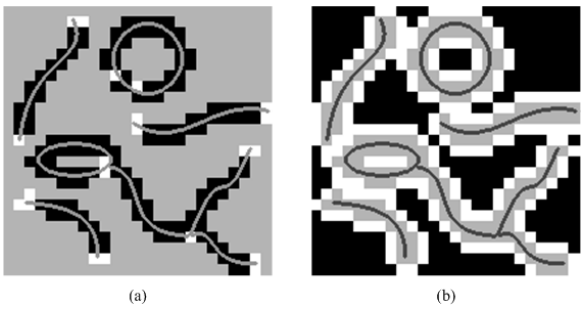

Fig. 2. Illustration of exemplar selection. Thick curves denote edges.

Moreover, additional structural blocks are selected to represent local variations. If a block contains obvious variation, it has the priority to be preserved. In practice, each black block in Fig. 2 (a), denoted as B_i, is assigned a variation parameter V_i defined as

$$V_i = w_1 Var(B_i) + w_2 \sum_{B_j \in \mu_4(B_i)} |E(B_i) - E(B_j)|, \quad (1)$$

where w_1 and w_2 are weighting factors, and $\mu_4()$ indicates four neighbors of a certain block. The functions $Var()$ and $E()$ are the variance and the mean of pixel values, respectively. Note that in one block, the different partitions separated by edges are independent in calculating the variance and the mean; then the resulting parameters of different partitions are summed up to get the total variation parameter of the block. Given an input threshold, the blocks with higher variation parameters will be selected as exemplars.

Textural exemplar selection Similarly, textural exemplars are also selected in two steps. Denoted by the white blocks in Fig. 2 (b), necessary textural blocks are selected at the border of textural regions. In specific, if a textural block is next to a structural one, it is considered as necessary. Such blocks are preserved because they can clearly separate textural blocks from structural ones and thus facilitate texture synthesis at the decoder. Additional blocks are also selected according to their variation parameters which can be calculated by (1). The textural blocks with higher variation parameters will be preserved as exemplars.

4. IMAGE RESTORATION

At the decoder side, we propose a compression-oriented image restoration method to reconstruct the removed blocks. Different from the previous work on inpainting or synthesis, our image restoration method tries to fully utilize the transmitted edges. In specific, edge-based inpainting (EBI) is performed to restore the pixel values on the edges and neighbor-

ing the edges, and texture synthesis is utilized for the restoration of textural regions.

4.1. Edge-Based Inpainting

Our proposed EBI method is performed based on individual edges. For each edge, EBI is completed in two steps as shown in Fig. 3. First, a linear interpolation is adopted to generate the unknown pixels on the edge from the known ones on the same edge. Second, the neighborhood of an edge, known as *influencing region*, is progressively filled-in by pixel generation. In the following we will concentrate on the second step.

In general, edges represent the structural information that consists of the discontinuities in images. But the textural information, referred to as kinds of regularities in statistics, geometric shapes, etc., also exists in the neighborhoods of edges. How to simultaneously restore the two kinds of information is an important issue [10].

In our scheme, we approach the problem by the pixel generation method. For each unknown pixel in the influencing region, known as the target pixel, we modify the pair matching method [5] to find out two candidate pixels. One is denoted as structural candidate (S-candidate), which lies within the influencing region; the other, textural candidate (T-candidate), locates within the neighborhood of the target pixel.

S-candidate is chosen to minimize the following weighted sum of squared difference (SSD):

$$D_S = \sum_i (|d(x_S^i) - d(x_t^i)| + 1) \times |f(x_S^i) - f(x_t^i)|^2, \quad (2)$$

where x_S^i is the ith pixel (e.g. in scan-line order) in the neighborhood of the S-candidate as shown in Fig. 3 (b). x_t^i is the ith pixel in the neighborhood of the target pixel. $d()$ means the distance from each pixel to the edge and $f()$ is the reconstructed image. By minimizing (2), we can find the S-candidate whose distance from the edge is similar to that of the target pixel. Differently, ordinary SSD is considered as the criterion to choose T-candidate:

$$D_T = \sum_i |f(x_T^i) - f(x_t^i)|^2, \quad (3)$$

x_T^i represents the ith pixel in the neighborhood of the T-candidate. At last, one of the two candidates will be chosen to fill-in the target pixel. The choice is made by comparing D_T and D'_S, which are defined in (3) and (4) respectively.

$$D'_S = (d/d_0 + pd/pd_0) \times \sum_i |f(x_S^i) - f(x_t^i)|^2, \quad (4)$$

where d_0 and pd_0 are constants. $d = d(x_t)$ stands for the distance from the target pixel to the edge, and pd indicates the distance from the target pixel to the S-candidate, as shown in Fig. 3 (b). If D_T is smaller than D'_S, T-candidate is selected to fill-in the target pixel, otherwise S-candidate is selected.

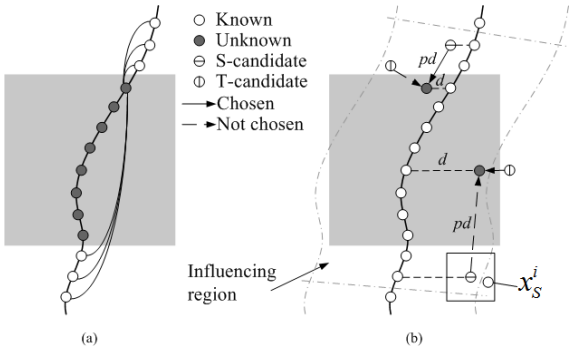

Fig. 3. Edge-based inpainting. (a) Unknown edge pixels are restored by interpolation; (b) the other unknown pixels are generated one by one, each of which is filled-in by one of two candidate pixels.

4.2. Texture Synthesis

After EBI, a patch-wise texture synthesis algorithm is adopted to restore the remaining regions, where patches are blocks with fixed size (e.g. 7×7). The patch furthest from the edges will be generated first, so as to prevent the interference of edges in texture synthesis. For a target patch which is partially unknown, a known patch is searched out from its neighborhood in terms of the least SSD between two patches. Graph-cut [6] and Poisson editing [11] are utilized to merge the known patch into the position of the target patch. This process is repeated until no unknown pixel exists.

5. EXPERIMENTAL RESULTS

In our proposed scheme as shown in Fig. 1, the normal image encoder/decoder can be any of the existing compression systems. Block-based coding methods are more suitable because our exemplar selection is also block-based. In this paper, we test our scheme on JPEG and H.264/AVC intra coding [12]. The edges related to the removed blocks are collected and coded by JBIG method. One binary map, which indicates whether a block is removed or not, is also coded into the bitstream.

In the comparison with standard JPEG, we use the Lena image and set the quality parameter from 50 to 95. Since PSNR is not quite suitable here, the method proposed in [1] is adopted to evaluate the quality of decoded images. The quality score is independently calculated for R/G/B color components. As shown in Fig. 4, our scheme outperforms JPEG in terms of the quality score. We would like to remark that the method in [1] is specially designed to capture ringing and block artifacts. How to evaluate the artifacts introduced by inpainting or synthesis is still an open problem.

In the comparison with H.264/AVC intra coding, we use some standard test images and quantization parameter (QP) is

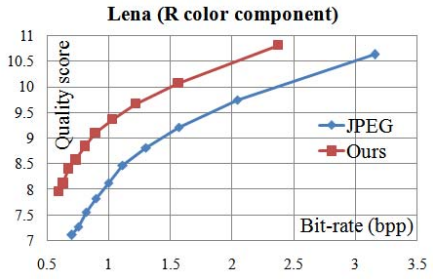

Fig. 4. Coding performance in terms of quality score [1] versus bit-rate.

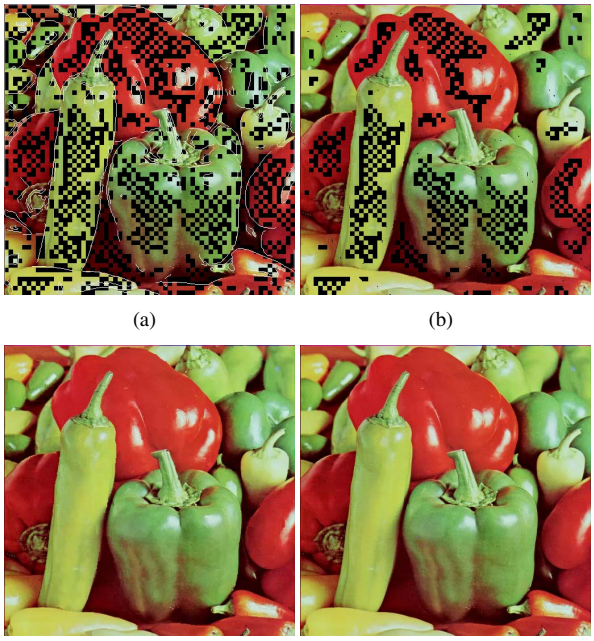

Fig. 5. The results on image Peppers.

set to 24. Fig. 5 shows some decoding results of the Peppers image. In Fig. 5 (a), the decoded exemplars as well as edges (white curves) are depicted. EBI is first performed to give out (b), then texture synthesis fills-in other unknown regions and presents the final result (c). (d) is the decoded image by standard H.264 intra coding. It can be observed that (c) and (d) have similar visual quality. Table 1 lists the bit-rate of each image, which shows that our scheme can save up to 32% bits compared with H.264 intra coding.

6. CONCLUSION AND FUTURE WORK

In this paper, we have investigated a compression approach towards visual quality rather than pixel-wise fidelity. An image compression scheme is proposed to take advantage of inpainting as well as synthesis techniques. In our scheme, some blocks are intentionally removed during encoding, while simple edges are transmitted as assistant information. To fully utilize these edges, we design edge-based inpainting and adopt texture synthesis to restore the removed blocks. Experimental results show that our scheme can achieve bit-rate saving as high as 32% at similar visual quality levels.

Currently we have not implemented any optimization at either algorithm or realization level, the decoding will cost a few minutes for a 512×512 image. It requires our future work on the acceleration of inpainting and synthesis algorithms.

Table 1. Bit-rate saving of our scheme compared with standard H.264/AVC intra coding (QP=24)

Image (512×512)	Block Removal	Bit-Rate (bpp)		Bit-Rate Saving
		Standard	Ours	
Jet	39.7%	0.984	0.881	**10.5%**
Lena	32.7%	0.990	0.868	**12.4%**
Milk	52.8%	0.615	0.416	**32.4%**
Peppers	33.6%	1.308	1.080	**17.4%**

REFERENCES

[1] Z. Wang, H. R. Sheikh, and A. C. Bovik, "No-reference perceptual quality assessment of JPEG compressed images," in *Proc. Int. Conf. Image Process., ICIP*, 2002, pp. 477–480.

[2] M. M. Reid, R. J. Millar, and N. D. Black, "Second-generation image coding: an overview," *ACM Computing Surveys*, vol. 29, no. 1, pp. 3–29, Mar. 1997.

[3] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," in *Proc. ACM SIGGRAPH*, 2000, pp. 417–424.

[4] C. Ballester, M. Bertalmio, V. Caselles, G. Sapiro, and J. Verdera, "Filling-in by joint interpolation of vector fields and gray levels," *IEEE Trans. Image Process.*, vol. 10, no. 8, pp. 1200–1211, Aug. 2001.

[5] M. Ashikhmin, "Synthesizing natural textures," in *Proc. ACM Symposium on Interactive 3D Graphics, SI3D*, 2001, pp. 217–226.

[6] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick, "Graphcut textures: image and video synthesis using graph cuts," in *Proc. ACM SIGGRAPH*, 2003, pp. 277–286.

[7] S. D. Rane, G. Sapiro, and M. Bertalmio, "Structure and texture filling-in of missing image blocks in wireless transmission and compression applications," *IEEE Trans. Image Process.*, vol. 12, no. 3, pp. 296–303, Mar. 2003.

[8] X. Sun, F. Wu, and S. Li, "Compression with vision technologies," in *Proc. Picture Coding Symposium, PCS*, 2006.

[9] C. A. Rothwell, J. L. Mundy, W. Hoffman, and V.-D. Nguyen, "Driving vision by topology," in *Proc. Int. Symposium on Computer Vision*, 1995, pp. 395–400.

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

[11] P. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," in *Proc. ACM SIGGRAPH*, 2003, pp. 313–318.

[12] "H.264/AVC Reference Software, JM ver. 11.0," available online: http://iphome.hhi.de/suehring/tml/download/.