# Constrained Texture Synthesis for Image Post Processing

Yu Hen Hu and Rajas A. Sambhare
University of Wisconsin - Madison
Dept. Electrical and Computer Engineering
Madison, WI 53706 USA
Email: sambhare@cae.wisc.edu, hu@engr.wisc.edu

***Abstract*— A novel constrained texture synthesis approach is proposed to enhance the visual quality of a degraded image by reconstructing its high frequency texture content. In low-bit-rate image and video compression and communication systems, high frequency transformed coefficients are often lost due to aggressive quantization or un-even error protection schemes. However, with the block-based encoding and transmission methods, the amount of high frequency texture loss is uneven between adjacent blocks. As such, it is possible to exploit this texture-level correlation between adjacent image blocks to re-construct the lost texture. This is accomplished by applying a state of art patch-based quilting texture synthesis algorithm in this paper.**

**Keywords— texture synthesis; texture recovery, texture restoration, image post processing, coding artifact, JPEG2000.**

## I. INTRODUCTION

Loss of high frequency texture details is an anomaly that often occurs in images or video frames that has been subject to very low-bit-rate compression. In modern transform-based image and video coding algorithms, it is well known that most energy in a naturally generated image or video frame will concentrate on few low-frequency transformed coefficients with large amplitudes. On the other hand, many high frequency transformed coefficients often have small amplitudes and are likely to be discarded during the encoding process to achieve desired compression ratio. The result is is a *washout* effect due to the loss of fine textures in the decoded image. One such example is depicted in Figure 1 where a portion of the decoded Barbara image is shown. This image has been subject to JPEG2000 compression at a compression ratio of 94:1 (0.085 bpp) using the Elecard wavelet image compression software (http://www.elecard.com). Note that around both sides of the stripped pant, the fine texture details are lost. However, at the center of the pant, the strip texture is largely intact. One plausible explanation to this phenomenon is the uneven distribution of fine texture contents and uneven allocation of bit budgets at different coding tiles in the JPEG2000 compression algorithm. Nonetheless, the presence of well-preserved texture adjacent to the washout region offers an opportunity to reconstruct the lost fine texture details using the adjacent texture patterns as a reference. How to seize this opportunity to enable fine texture reconstruction in a highly compressed image or video frame is the focus of this paper.

Although the washout effect is a type of image and video coding artifact, existing coding artifact reduction algorithms

**Figure 1. Illustration of the washout artifact. Note the loss of stripe texture at portions the pant.**

can not adequately address this problem. Basically, there are two distinct approaches to image and video coding artifact reduction: the enhancement techniques, and the restoration techniques. With the enhancement approach, a context sensitive, and sometimes nonlinear filter is applied to clean up the visible coding artifact, such as blocking [1], [2], or ringing [3], [4]. These filters are designed to remove unsightly blocking artifacts, rather than to recover lost high frequency textures. With the restoration approach [5], the "original" image $f$ is to be estimated from the degraded image $g$ by minimizing two competing cost functions: (i) the deviation between $f$ and $g$, $||f - g||$, and (ii) a regularization term that represents our *a prior knowledge* about the image. Often this regularization term is modeled as a Gibb's energy distribution over a local *clique* of each pixel and can be interpreted as a smoothness criterion. Being a very general model, the Gibb's distribution cannot adequately capture the local statistics representing detailed fine textures.

Previously, Yang and Hu have reported a dithering based approach [6] to recover some high frequency components from a JPEG-encoded image. This approach appears to be effective only for fine-grain textures such as the photo granular noise. It is not applicable to the recovery of coarser granular textures. In [7], Krishnamurthy et al. reported a texture restoration method based on a Wold-decomposition of the image spectra. An EM algorithm is developed to iteratively restore both the deterministic and non-deterministic portion of the image. This method is applied to a uniformly blurred image and hence is not directly applicable to the uneven texture loss situation considered here.

In this paper, we propose a novel approach to address this texture restoration problem. We argue that since the texture loss is in general not uniform over the entire textural region, it is possible to restore the missing texture by extending the texture from surrounding regions with similar texture. This texture extension approach is similar to the smooth surface extension approach employed in image/video error concealment techniques. Here, the challenge is extend texture, rather than a smooth surface.

Recently, the field of computer graphics has made significant advances in texture synthesis. In particular, it has been demonstrated [8], [9] that a recently developed family of patch based quilting algorithms outperform many state of the art texture synthesis algorithms (e.g. [10], [11]), in both the quality of the synthesized texture and speed. However, in the context of 3D rendering applications, these methods are applied to synthesize texture in an unconstrained manner. In this paper, we extend the algorithms presented in [8] and [9] and apply them to recover lost fine texture in an image suffering from washout anomaly.

This form of post processing is different from the usual approach and is essentially a clever form of "plastic surgery". Specifically, portions of the visual material, which are undamaged, are used to repair artifacts. High frequency texture information is thus preserved and put to good use. In this paper, our focus is on the task of adapting the patch-based texture synthesis method to mend image washout defects. We will report the procedure to achieve this goal and preliminary yet extremely encouraging experiment results.

As of now, we leave the tasks of identifying the locations of the source texture and the artifact region to human user supervision. We maintain that, however, a fully automated solution to these tasks would require our algorithm to understand the semantics of the underlying image and therefore are unlikely to be completely resolved in the near future.

In the rest of this paper, the constrained texture synthesis algorithm will be first reviewed. We present a variation of the algorithm that uses the underlying brightness information present in the artifact, for better results in some cases.

## II. Patch-Based Texture Synthesis

### A. Algorithm

In this section, we use the algorithm presented in [8] to illustrate the patch-based texture synthesis algorithm.

The basic objective of this algorithm is to take a small sample of source texture $S$, and generate an arbitrarily large amount of image data that, while not exactly like the original, will be perceived by humans to be of the *same texture*. Synthesis proceeds by randomly drawing blocks $s_i$ from a set of all such blocks $S_B$ in $S$. Blocks drawn from $S_B$ are placed next to each other with some overlap between blocks. We will search $S_B$ for such a block that by some measure agrees with its neighbors. The measure used is the error $e$ between overlapping pixel values. The block is pasted as long as $e$ is less than a threshold $t$. This error surface is the also used to calculate a minimum error cut boundary (MEBC), so that we can give our blocks ragged edges, which produce best results. Refer to Figure 2 for a summary of the algorithm. Thus we can generate a large texture sample, given a smaller sample. The algorithm in [9] is similar with feathering [12] instead of MEBC.

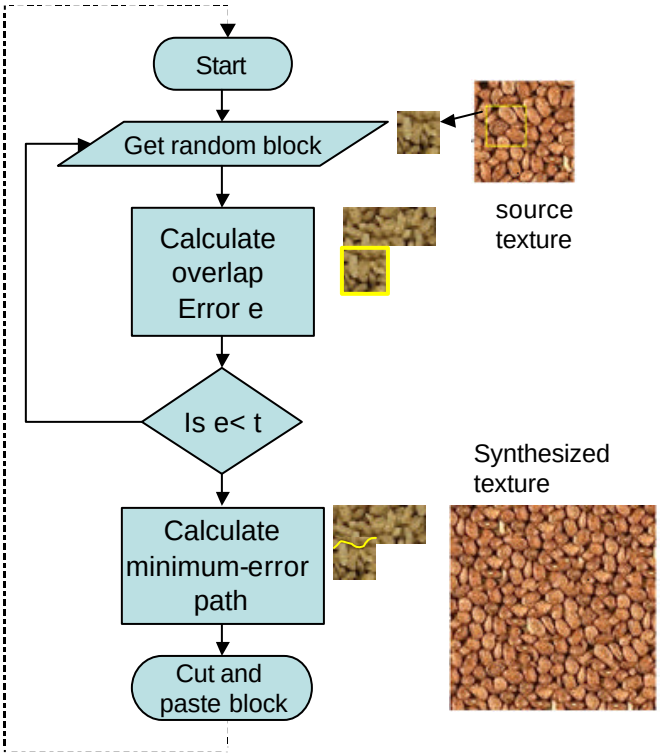



**Figure 2. A flow chart of the patch-based texture synthesis algorithm [Efros, 2001 #388].**

## III. Constrained Texture Synthesis for Washout Effect Reduction

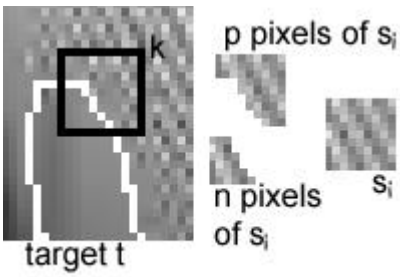
To apply the patch-based texture synthesis to reduce the washout artifact, one faces several challenges:

1. The washout region is not a rectangular.
2. The brightness of the washout region is not uniform.

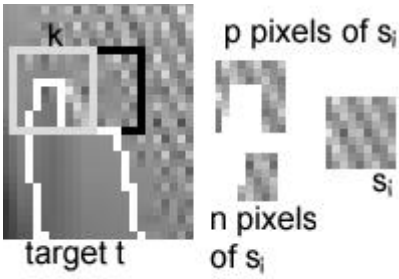
### A. Irregular Washout Region

Consider an arbitrarily shaped target region $t$ that has to be filled with a source texture $S$. Again, as a basic unit of synthesis we consider a square block $s_i$ drawn from the source texture $S$. In Figure 3(a), $t$ is the irregular region enclosed in white, while the large rectangle enclosed in white, forms the source region $S$. A few sample blocks $s_i$ are shown in $S$ as enclosed by white squares. We begin synthesis near those borders of $t$ where high frequency texture content is preserved. This border constraint gives us a starting point for our algorithm, as any block we paste in should match the preexisting texture.


(a) target region *t* and source region *S* are manually determined.




(b) find $s_i$ whose *p* portion best match existing *p* portion. Use the *n* portion of $s_i$ to synthesize the lost texture.




(c) The target area is now reduced by the area of the *n* portion that is synthesized in the last step.

**Figure 3. Illustration of proposed constrained texture synthesis procedure to reduce washout artifact.**

Thus for synthesis, we consider a square block *k* enclosing a portion of both *t* and pixels outside *t* where high frequency texture exists. This is shown in Figure 3(b). Pixels enclosed by *k* outside *t* have to be preserved (these are called *p* pixels) while pixels enclosed by *k* inside *t* have to be synthesized (these are called *n* pixels). To proceed with synthesis we randomly select a block $s_i$ from the source texture *S*, and determine the error *e* between the *p* pixels in target block and corresponding pixels in $s_i$ (these pixels are extracted via masking $s_i$). Error here is defined as the sum of squares of the difference between corresponding pixels. Thus we keep randomly drawing blocks from *S*, until the error drops below a certain threshold or we have searched for a long enough time (i.e. a counter set initially is exhausted). We then paste into *t*, the corresponding *n* pixels from the best $s_i$. (We use feathering or MEBC at the boundary of *t* to improve results. This is explained later.). We notice that in case there are a large number of *p* pixels, then the error may never drop below the threshold, which is why the counter has been added to prevent excessively long (and fruitless) searches, which usually happen in the case of stochastic textures. Also when there are a large number of *p* pixels, the *n* pixels will form a small area, and thus errors (caused by not having an exhaustive search) are less noticeable. We thus naturally exploit this psychophysical phenomenon.

After one block area has been successfully synthesized, we select the next block area in *t* for synthesis. Notice in Figure 3(c) that *t* has now reduced in size, as we have synthesized part of it. (The black boundary of the previously synthesized block has been left intact just for demonstration purposes). In this case we do not completely move past the newly synthesized block, but rather we allow for some of overlap (just like in [8] ). The amount of overlap can be varied. After a lot of testing, it was noticed that an overlap of $1/6^{th}$ of block dimensions seems to work best. This synthesis process is repeated until the entire target region *t* has been filled. Thus by allowing for an overlap between adjoining blocks we can seamlessly fill the entire target region.

### *B. Edge mismatch handling*

Edge mismatch along the overlap (the boundary of *t*) can be handled in two ways. While [8] uses the minimum error boundary cut (MEBC), [9] uses feathering along the overlap. We tested both methods and came to the following conclusions. While MEBC provides slightly more accurate results, it is much slower than feathering, and significant differences between the two are visible only for a small subset of textures. Specifically differences are observed only for textures with a significant amount high frequency content and with two or more histogram peaks. We first present the user with results based on feathering. If the user is not satisfied with the results, he/she can opt for the costlier MEBC based algorithm that can provide better results. However note that in both cases feathering is done along the original (irregular) boundary of target *t*, as a minimum error cut along an arbitrarily shaped curve is difficult.

### *C. Quilting with Brightness Transfer*

This is a variation of our algorithm which produces better results when the artifact is caused by coding or in some cases transmission errors. Here we exploit the fact that while high frequency information is lost or distorted, the low frequency content of the image is usually intact and can be exploited to produce better results. We proceed along the lines of the texture transfer algorithm described in [8] and we extend the synthesis algorithm constraining it to match not only the texture information but also the low frequency luminance information. We do this by modifying the error *e* to be the weighted sum of *a* times the block overlap error plus (1-*a*) times the squared error between the average luminance of the pixels of the target image and the newly picked texture. Varying *a* empirically, we notice that best results are obtained at $a = 0.75$. This brightness transfer step has been made an integral part of the algorithm and all results shown have incorporated this step unless otherwise noted.

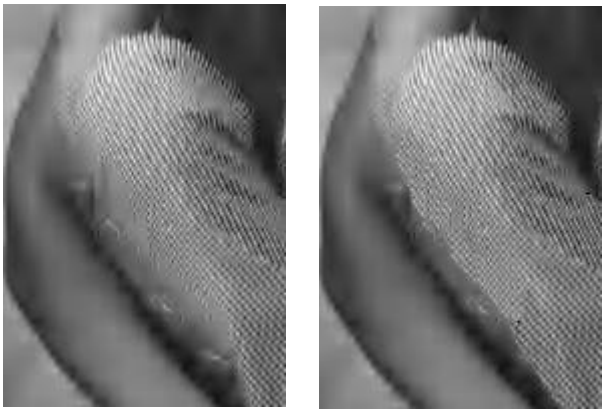


**Figure 4. (a) washout artifact (b) texture recovered**

## IV. IMPLEMENTATION EXAMPLES

### *A. Example 1. Barbara Image*

The Barbara image is compressed using the Elecard package at 94:1 compression ratio. As shown in Figure 1, there is significant loss of fine-grain textures at the both sides of the pants. After identifying the washout region manually, the constrained texture synthesis method is applied, and the results are shown in Figure 4. Note that the textures in both sides of the pants have been well reconstructed based on the texture at the center of the pants where it is better preserved. To achieve this, more than one target regions on both sides of the pants have been manually identified and more than one source regions have been used to reconstruct the lost texture. We also note that there are visible boundaries between the synthesized texture and its smeared background at the region closer to the lower arm. This makes the synthesized texture appears to be *artificial*. This is because the target region has not been adequately selected. If the target region *t* extends into the dark region between the arm and the pant, than such a spurious edge will be much less visible.

### *B. Example 2. Mountain Image*

The test image "mountain" (Figure 5a) contains lots of high frequency texture regions. It is compressed using the Elecard software at 20:1 compression ratio or 0.4 bpp (Figure 5b). The target region is identified as the dark region (Figure 5c), and the source region is identified as the rectangular region to the right side of the target region (Figure 5d). Note that the texture recovered result (Figure 5e) looks quite "natural" even its texture is quite different from that of the original (Figure 5a).

## V. CONCLUSION AND DISCUSSIONS

In this paper, we report preliminary result of applying patch-based texture synthesis method to reconstruct lost high frequency textures from an image suffering from washout coding artifact. We are working on texture-based segmentation algorithms to identify the target region automatically. We are also working on improving the texture synthesis results. The results so obtained can be used as an initial condition so that the mathematically rigorous Bayesian image restoration method can be applied. Nevertheless, this initial results are very encouraging and pointed to a new direction for image post-processing.

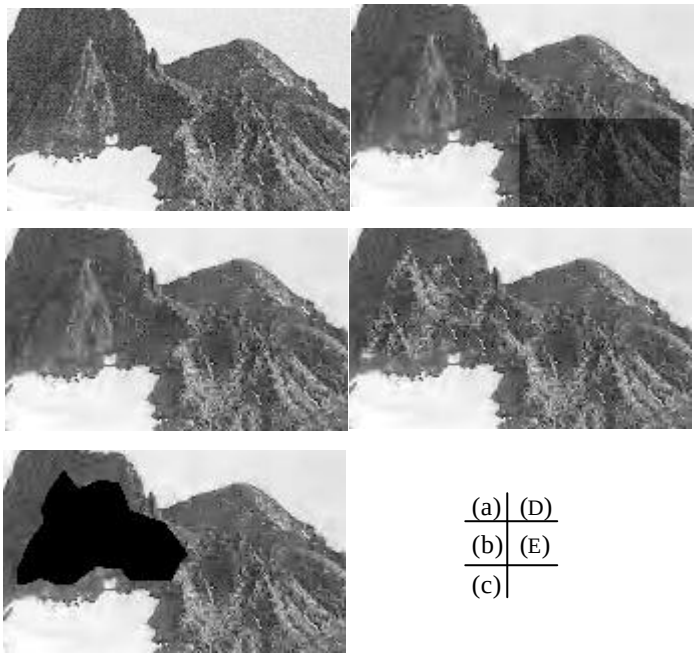



**Figure 5. Mountain Example**

## VI. REFERENCES


[1] S. H. Oguz, T. Q. Nguyen, and Y. H. Hu, "Critical quantization decisions in transform coding and blocking artifacts," Proc. ISCAS'99, Orlando, FL, pp. 63.19, 1999.

[2] S. Yang, Y. H. Hu, and D. L. Tull, "Blocking artifact removal using robust statistics and line process," IEEE Int'l Workshop on Multimedia Signal Processing, 1999.

[3] S. Yang, Y. H. Hu, D. L. Tull, and T. Q. Nguyen, "Maximum likelihood parameter estimation for image ringing artifact removal," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 11, pp. (to appear), 2001.

[4] S. H. Oguz, Y. H. Hu, and T. Q. Nguyen, "Image coding ringing artifact reduction using morphological post-filtering," IEEE Second Workshop on Multimedia Signal Processing, Los Angeles, CA, pp. 628-633, 1998.

[5] Y. Yang, and N. P. Galatsanos, and A. K. Katsaggelos, "Projection-based spatially adaptive reconstruction of block transform compressed images," *IEEE Trans. on Image Processing*, vol. 4, pp. 896-908, 1995.

[6] S. Yang and Y. H. Hu, "Block Effect Removal using Regularization and Dithering," Proc. ICIP'98, Chicago, IL, pp. 346-349, 1998.

[7] R. Krishnamurthy, J. W. Wood, and J. M. Francos, "Adaptive restoration of textured im ages with mixed spectra," *IEEE Trans. Image Processing*, vol. 5, pp. 648-652, 1996.

[8] A. A. Efros, and W. T. Freeman, "Image quilting for texture synthesis and transfer," Proc. ACM SIGRAPH, pp. 341-346, 2001.

[9] L. Liang, C. Liu , Y.-Q. Xu , B. Guo, H.-Y. Shum, "Real-time texture synthesis by patch-based sampling," *ACM Trans. on Graphics*, vol. 20, pp. 127-150, 2001.

[10] J. S. De Benet, "Multiresolution sampling procedure for analysis and synthesis of texture," Computer Graphics Proceedings, SIGGRAPH, pp. 361-368, 1997.

[11] K. Sivakumar and J. Goutsias, "Morphologically constrained GRFs: applications to texture synthesis and analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, pp. 99-113, 1999.

[12] R. Szeliski, and H. Y. Shum, "Creating full view panoramic mosaics and environmnet maps," Proc. ACM SIGGRAPH'97, pp. 251-258, 1997