MATCHED-TEXTURE CODING FOR STRUCTURALLY LOSSLESS COMPRESSION

Guoxin Jin[1], *Yuanhao Zhai*[2], *Thrasyvoulos N. Pappas*[1], *David L. Neuhoff*[2]

[1]EECS Department, Northwestern Univ., Evanston, IL 60208
[2]EECS Department, Univ. of Michigan, Ann Arbor, MI 48109

ABSTRACT

We propose a new texture-based compression approach that relies on new texture similarity metrics and is able to exploit texture redundancies for significant compression gains without loss of visual quality, even though there may visible differences with the original image (structurally lossless). Existing techniques rely on point-by-point metrics that cannot account for the stochastic and repetitive nature of textures. The main idea is to encode selected blocks of textures – as well as smooth blocks and blocks containing boundaries between smooth and/or textured regions – by pointing to previously occurring (already encoded) blocks of similar textures, blocks that are not encoded in this way, are encoded by a baseline method, such as JPEG. Experimental results with natural images demonstrate the advantages of the proposed approach.

Index Terms— structural similarity metric, blending, side-matching, direct block matching

1. INTRODUCTION

With impressive advances in image and video compression over the past decades, existing approaches to image compression are approaching saturation. However, they are still far from approaching the efficiency of the human brain in storing visual information. One of the keys to further advances is a more efficient handling of texture [1]. Current compression techniques rely on point-by-point similarity metrics that cannot account for the stochastic nature of textures, which can appear virtually identical even though they have substantial (visible when one looks carefully) point-by-point differences. This paper proposes a new texture-based compression approach that relies on new similarity metrics for textured images [2–4] that overcome limitations of existing metrics.

State of the art techniques rely on (temporal or spatial) prediction in combination with transform/subband/wavelet-based compression of the residual, or the original image if a good prediction cannot be found. Such techniques are quite efficient in smooth and piecewise smooth image regions, but are inefficient in regions that contain textures. This is because limitations of current metrics cause prediction to fail in textured regions, while the direct encoding of textures requires high bitrates due to the typically high frequency content of textured regions. However, because of their stochastic and repetitive nature, textures contain a lot of redundancy. The main idea of the proposed approach is to encode selected blocks of textures – as well as smooth blocks and blocks containing boundaries between smooth and/or textured regions – by pointing to previously occurring (already encoded) blocks of similar textures. Blocks that are not encoded in this way, are encoded by a baseline method, such as JPEG. The key to accomplishing this is the use of a new *structural texture similarity metric (STSIM)* [2–4], which allows visible point-by-point variations in textures that are virtually indistinguishable [due to the stochastic nature of the variations]. Another key to the success of the proposed approach is the use of texture blending [5] to eliminate stitching artifacts.

By allowing substantial point-by-point variations in textured areas the proposed approach deviates from the "gold standard" of *perceptually lossless* compression, whereby the compressed image is visually indistinguishable from the original in a side-by-side comparison. The new goal is *structurally lossless compression* [1], whereby the original and compressed images have the same high visual quality, even though in a side-by-side comparison there are visible differences – that do not affect the structure of the image. Such differences would be difficult to detect when the reproduction is viewed by itself, while in a side-by-side comparison, it should not be obvious which image is the original. Thus, structurally lossless compression bridges the gap between perceptually lossless and perceptually lossy compression.

The proposed approach is one way to achieve structurally lossless compression. Of course, there may be other ways to do that, e.g., ways that allow smoothly modifying or moving image edges. The goal of this paper is to show that by allowing visible texture variations that do not affect visual quality, the proposed scheme can achieve acceptable quality at rates at which conventional approaches (JPEG) break down.

The signal processing community has identified exploiting texture as a key to increasing compression efficiency (e.g., see selected papers in [6, 7]); one approach is "compression-by-synthesis" whereby the encoder tries to detect pure texture regions which it then represents with appropriate model parameters. What distinguishes our approach from those of other authors is the use of an explicit texture similarity metric and the idea of structurally lossless compression.

This research was funded in part by Sony Electronics Inc.

In Section 2 we discuss texture similarity metrics while in Section 3 we present the new coding approach.

2. STRUCTURAL TEXTURE SIMILARITY METRICS

One of the keys for the proposed approach is the use of SSIM-type metrics for comparing textured patches [8, 9]. The guiding spirit of SSIM metrics is to replace traditional point-by-point similarity measurements with measurements based on the similarities of local statistics computed separately for all positions of a sliding window. SSIMs can be implemented in the spatial or subband domain. However, close inspection of SSIMs reveals that they include point-by-point terms. In order to overcome such limitations, Zhao *et al.* [2] proposed a *structural texture similarity metric (STSIM)* that relies entirely on local image statistics, and is thus completely decoupled from point-by-point comparisons. Zujovic *et al.* further developed this idea in [3].

When using a similarity metric, it is important to realize that different applications impose different requirements on metric performance. While in image retrieval it may be sufficient to distinguish between similar and dissimilar textures, in image compression it is important to ensure a monotonic relationship between measured and perceived distortion. In [4], Zujovic *et al.* conducted subjective experiments with synthesized texture distortions that model natural texture variations in order to evaluate the performance of different metrics, including PSNR, SSIM, and STSIM. They used different original texture images and different types and degrees of distortions. They found that STSIM metrics provide the best performance. In the following, we use the STSIM metric proposed in [3], which exhibits an approximately monotonic relationship between measured and perceived distortion for natural textures. In smooth regions and near region boundaries, the metric behavior is still approximately monotonic, but the metric values are not consistent among smooth, pure texture, and boundary regions, and thus an absolute performance threshold cannot be established. Metric limitations also include the inability to distinguish small but perceptible changes in texture orientation and inconsistencies when applied with different sliding window sizes. However, as we will see in Section 3, when used in combination with MSE for "side-matching," the metric behavior improves considerably.

3. MATCHED-TEXTURE CODING (MTC)

In this section we describe a new approach to image compression that we call *Matched Texture Coding (MTC)*, which relies on STSIM with the goal of dramatically decreasing the number of bits required to encode textured regions of images.

The principal idea is to choose a baseline coding method, such as JPEG, and then to encode each block of the image either by the baseline method or by indicating to the decoder that the block is well approximated by the decoded reproduction of some block in the previously encoded section of the image. In the latter case, the encoding rate for the block becomes very small.

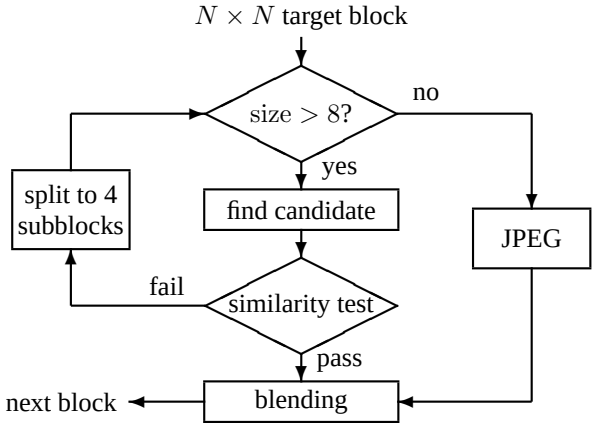

Fig. 1. Algorithm

Aside from the baseline coding method, the principal components of this approach are a similarity metric with which to compare the current block, which we will call the *target* block, to the reproductions of various blocks in the previously encoded region, from now on called *candidates*, a mechanism for identifying which candidates are to be tested, a threshold to determine if the most similar candidate found is sufficiently similar, a mechanism for identifying the chosen candidate to the decoder, and a method for blending the chosen candidates with the reproductions that surround them.

There are two basic versions, *Side Matching (SM)* and *Direct Block Matching (DBM)*, as well as combinations. In both versions, the method operates on successive nonoverlapping $N \times N$ blocks; we found $N = 32$ to be a good choice. As illustrated in Fig. 1, given the $N \times N$ block to be encoded next (target), it first seeks an $N \times N$ candidate that sufficiently matches the target in the sense that the STSIM value between the target and candidate is above a specified threshold. (If several candidates meet the threshold, then the best is chosen.) If it finds one, it must indicate to the decoder what candidate has been chosen. If it does not find one, it divides the target into four $\frac{N}{2} \times \frac{N}{2}$ subblocks and repeats the process, until the block size is equal to 8×8, in which case the block is encoded with the baseline coder, e.g., JPEG. The search for candidates is not extended to smaller blocks because the potential coding gains and computational complexity and reliability of finding a good match are not worth it. The rate-quality tradeoff of both versions can be controlled via the JPEG rate and the quality metric thresholds.

3.1. Side Matching (SM)

In Side Matching the encoder searches for the K candidates (4 to 16 is typical) in the already coded region whose left and upper borders regions, called their *contexts*, most closely match the corresponding context of the current block, where the matching metric is the sum of squared pixel differences. As illustrated in Fig. 2, for an $N \times N$ current block (target "T"

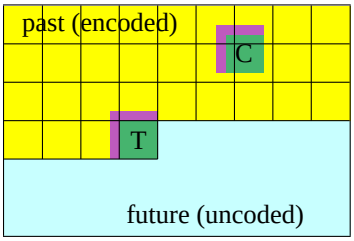

Fig. 2. Side Matching

shown in green), the context (shown in purple) is typically taken to be L-shaped with arms of width $N/4$. The candidates ("C" shown in green) corresponding to the K best contexts (shown purple above and to the left of "C") are then tested for similarity to the current block using STSIM. The candidate with the highest STSIM value then becomes the candidate for the test in Fig. 1. The idea is that if the context of a candidate matches that of the current block, then the candidate itself is likely to match the current block. One advantage of SM is that only $\log_2 K$ bits are needed to indicate the selected candidate, since the decoder can itself find the K best contexts. For an $N = 32$ and $K = 16$, this requires 0.004 bpp, which is a striking reduction over the 0.5–2.0 bpp typically required by JPEG. Such reductions are all the more striking because they usually happen for textured current blocks, for which JPEG is likely to generate more than its average rate in bits/pixel. Since only K STSIM computations are required per block, the main effort in this method is computing the context matches.

A key element of SM is **blending** [5]. Even when a candidate is very, very similar to the candidate it replaces, there will often be a noticeable discontinuities between the block and surrounding pixels. For this reason, we use blending to reduce or eliminate the appearance of such discontinuities. Context matching is an essential part of blending [5], thus serving a dual purpose, helping identify suitable candidates for target encoding, and ensuring smooth blending.

Note that the above approach does not have to be limited to blocks with uniform textures. It can also be used for blocks containing region boundaries (between smooth and/or textured regions). While such blocks are not as common, there are nevertheless gains to be made, provided the similarity metric (STSIM) can accurately predict the perceived similarity of the target block in such cases. In practice, a good match is secured by a combination of side-matching (MSE) and STSIM. As we will discuss below, by eliminating unsuitable candidates, side-matching, not only reduces the number of STSIM computations, but also helps obtain better target matches than those that could be obtained by STSIM alone. (This is because STSIM is not yet perfect!) This is true for both uniform blocks and blocks that contain region boundaries.

3.2. Direct Block Matching (DBM)

In Direct Block Matching the goal is to find the best candidate for the current block among all possible candidates in the previously encoded region of the image. Then, if the STSIM similarity of this candidate to the current block is above a specified threshold, its location is encoded for the decoder, and the candidate itself becomes the decoded reproduction of the current block.

For an $N \times N$ image, encoding the location of a successful candidate requires at most $2 \log_2 N$ bits. For example, for a 32×32 block of a 512×512 image, this is at most $18/1024 = 0.018$ bits/pixel, which is still a marked reduction over the typical 0.5–2.0 bpp typically required by JPEG.

Since computing STSIM for every candidate location is computationally expensive, DBM uses progressive location search and progressive metric calculation. The former is similar to hierarchical motion search in video coding, in that a subset of locations, e.g., a coarse lattice, are tested first, and then further tests are performed in the vicinity of the best found locations. The latter exploits the fact that most locations produce poor candidates, many of which can be ruled out based on a very simple metric, leaving only a small number of candidates for the full STSIM. For example, a simple first metric is the percentage difference between the variance of the current block and that of the candidate: $\left|\sigma_T^2 - \sigma_C^2\right| / \sigma_T^2$.

3.3. Additional MTC Issues

STSIM Details: We used a complex steerable filter implementation that decomposes each image into three scales and four orientations as in [3]. To limit the amount of computation, we used a 16×16 sliding window with no overlap. We also found that, in order to meet quality targets across each block, the most effective spatial pooling strategy is to use the minimum value across the windows instead of the average.

Decision Bits: These are bits that are necessary for the encoder to describe the split or not decisions. A simple method requires at most 5 bits for each 32×32 block of the image, i.e., 0.005 bpp per block.

Low Frequency Coding (LFC) of Background Gray Level: In preliminary tests, we found that candidates are sometimes rejected due to a mismatch of overall gray level, even though the textures are otherwise well-matched. To increase the number of viable candidates, we have found it best to first encode the very low frequency components of the image and then apply MTC to the residual. Several techniques have been tried and the best performance is attained with a form of QPC [10], which typically encodes with approximately 0.05 bpp.

Blending for JPEG-encoded Blocks: For JPEG encoded blocks, blending with the previously encoded MTC-coded blocks (above and to the left) must be done inside the JPEG block. To be able to do this smoothly target matching must include the context below and to the right of the block. Thus, target matching includes a border all around the target block; with typical width equal to 1/4 of the block dimension. While this may slightly reduce the number of acceptable candidates, it does have a significant effect on continuity.

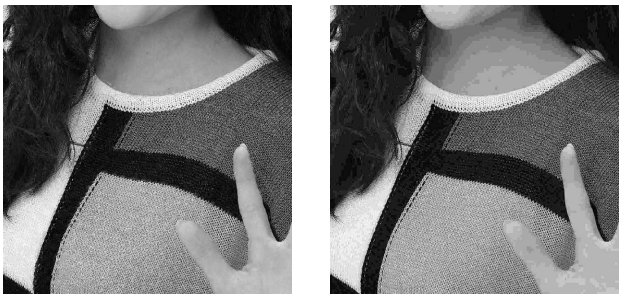

Fig. 3. Coding of "woman" at 0.34 bpp. (a) MTC, (b) JPEG

Fig. 4. Coding of "house" at 0.34 bpp. (a) MTC, (b) JPEG

3.4. Discussion and Experimental Results

Our initial thinking was that DBM would yield better quality candidates than SM at the expense of higher computational complexity. The higher quality candidates would offset the small additional bit rate resulting in similar or slightly better rate vs. distortion performance. However, we discovered that SM worked significantly better, A closer look revealed that SM, not only minimizes block boundary discontinuities by facilitating blending, but most interestingly, also has the effect of finding candidates whose texture patterns more closely matched those of the current block. In particular, the fact that candidates were being chosen based on a point-by-point context match tended to increase the likelihood that candidate itself contained texture with features similar to those of the current block, e.g., its basic elements would be of similar size, shape and orientation. This, of course, points to a limitation of STSIM. When side-matching was added as a filter for eliminating bad candidates, there was a substantial improvement in performance. Overall, an intertwining of progressive location search, side-matching, and progressive metric evaluation leads to an effective, computationally efficient method of finding candidates for the current block.

Taking into account compression, reproduced image quality, and computational complexity, the best results so far have been attained with a combination of SM and DBM. Space does not permit a description of how they were combined, but representative results are shown in Figs. 3 and 4, which show the results of two images encoded by MTC and JPEG at the same rate, 0.34 bpp for both figures. By magnifying the file, you may note that in Fig. 3, JPEG does a little better job encoding the texture in the sweater, but this causes it to run out

	32×32 MTC	16×16 MTC	8×8 JPEG	LFC	Overall
image coded	21.3%	22.5%	56.2%		100%
coding rate	0.02 bpp	0.09 bpp	0.48 bpp		
contribution	0.005 bpp	0.02 bpp	0.27 bpp	0.04 bpp	0.34 bpp

Table 1. Statistics for MTC coding of "woman" (Fig. 3).

	32×32 MTC	16×16 MTC	8×8 JPEG	LFC	Overall
image coded	25.7%	8.26%	66.1%		100%
coding rate	0.02 bpp	0.09 bpp	0.43 bpp		
contribution	0.006 bpp	0.008 bpp	0.28 bpp	0.04 bpp	0.34 bpp

Table 2. Statistics for MTC coding of "house" (Fig. 4).

of bits, thus resulting in false contours in the neck and the hair areas. While the MTC-coded texture is different, there are no apparent artifacts, just increased roughness. In Fig. 4, MTC does a better job in the roof and grass areas. A careful examination of the windows reveals some artifacts. JPEG on the other hand, spends a lot of bits in the windows at the expense of false contours in the roof and grass areas.

Table 1 shows that for 44% of the image, JPEG is replaced by MTC. On the portions where JPEG is used, it encodes at 0.48 bpp, whereas where MTC is used, only 0.02 bpp and 0.09 bpp are needed for 32×32 and 16×16 blocks, respectively. Moreover, the savings in the MTC areas enable the use of a higher JPEG quality in the rest of the image than would be used if JPEG needed to average to 0.34 bpp. This improves the quality of the candidates available for the 44% where MTC is used.

4. REFERENCES

[1] T.N. Pappas, *et al.*, "Image analysis and compression: Renewed focus on texture," *Visual Inf. Proc. Comm.*, San Jose, CA, Jan. 2010, Proc. SPIE, Vol. 7543.

[2] X. Zhao, *et al.*, "Structural texture similarity metrics for retrieval applications," *ICIP-08*, pp. 1196–1199.

[3] J. Zujovic, *et al.*, "Structural similarity metrics for texture analysis and retrieval," *ICIP-09*, pp. 2225–2228.

[4] J. Zujovic, *et al.*, "Subjective and objective texture similarity for image compression," *ICASSP-12*. Accepted.

[5] A.A. Efros, W.T. Freeman, "Image quilting for texture synthesis and transfer," *SIGGRAPH-01*, pp. 341–346.

[6] Special session on "Next generation image and video coding through texture analysis and synthesis," *ICIP-09*, Nov. 2009.

[7] "Special issue on emerging technologies for video compression," *IEEE J. Sel. Topics Sig. Proc.*, vol. 5, no. 7, Nov. 2011.

[8] Z. Wang, *et al.*, "Image quality assessment: From error visibility to structural similarity," *IEEE Tr. Image Proc.*, vol. 13, pp. 600–612, Apr. 2004.

[9] Z. Wang, E.P. Simoncelli, "Translation insensitive image similarity in complex wavelet domain," *ICASSP-05*, vol. II, pp. 573–576.

[10] C.-Y. Teng, D.L. Neuhoff, "A new quadtree predictive image coder," *ICIP-95*, vol. II, pp. 73–76.