Fast Texture Synthesis using Tree-structured Vector Quantization

Li-Yi Wei Marc Levoy

Stanford University

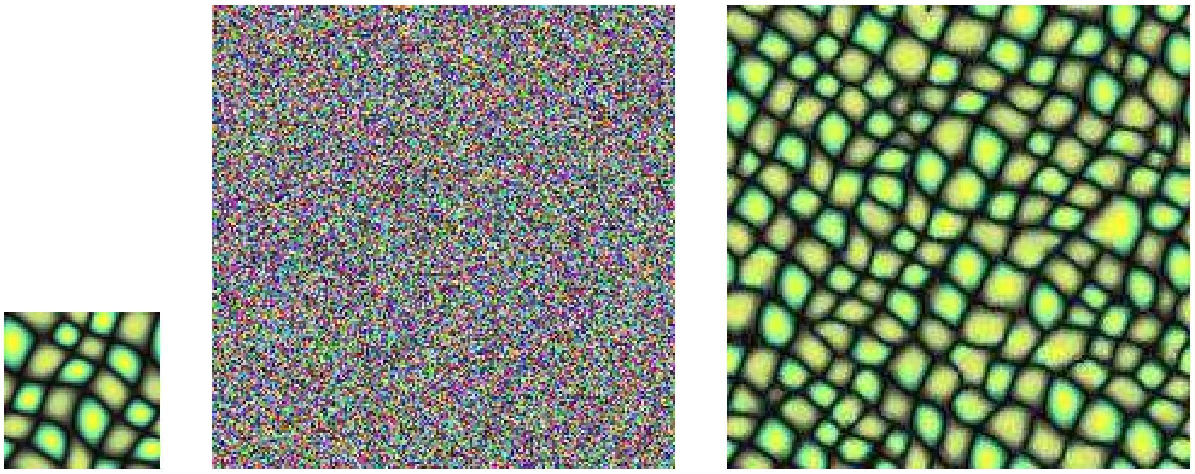

Figure 1: *Our texture generation process takes an example texture patch (left) and a random noise (middle) as input, and modifies this random noise to make it look like the given example texture. The synthesized texture (right) can be of arbitrary size, and is perceived as very similar to the given example. Using our algorithm, textures can be generated within seconds, and the synthesized results are always tileable.*

Abstract

Texture synthesis is important for many applications in computer graphics, vision, and image processing. However, it remains difficult to design an algorithm that is both efficient and capable of generating high quality results. In this paper, we present an efficient algorithm for realistic texture synthesis. The algorithm is easy to use and requires only a sample texture as input. It generates textures with perceived quality equal to or better than those produced by previous techniques, but runs two orders of magnitude faster. This permits us to apply texture synthesis to problems where it has traditionally been considered impractical. In particular, we have applied it to constrained synthesis for image editing and temporal texture generation. Our algorithm is derived from Markov Random Field texture models and generates textures through a deterministic searching process. We accelerate this synthesis process using tree-structured vector quantization.

To appear in Proceedings of SIGGRAPH 2000

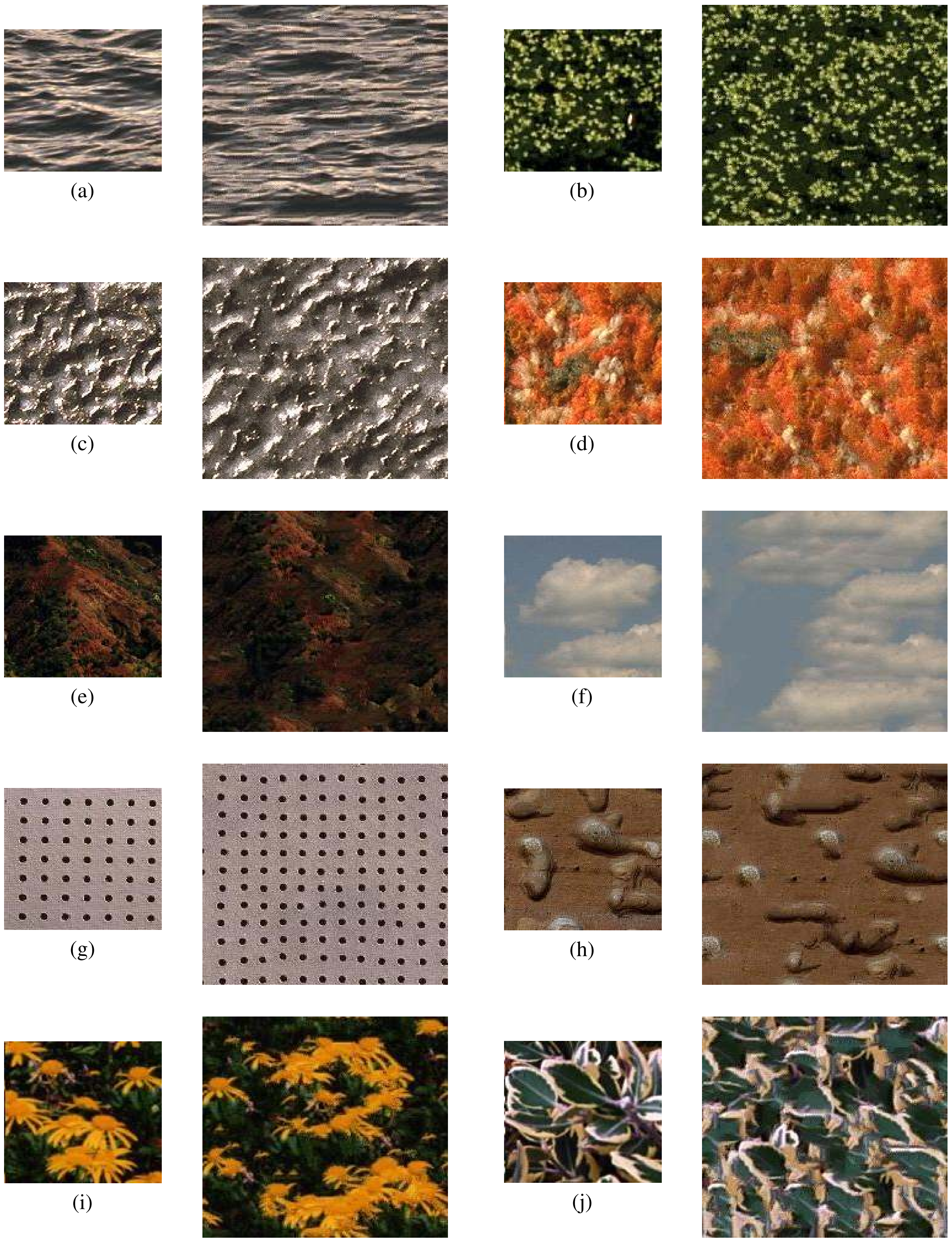

Figure 2: *Texture synthesis results. The smaller patches (size 128x128) are the input textures, and to their right are synthesized results (size 200x200). Each texture is generated using a 4-level Gaussian pyramid, with neighborhood sizes* {*3x3,1*}, {*5x5,2*}, {*7x7,2*}, {*9x9,2*}, *respectively, from lower to higher resolutions. VisTex textures: (a) Water 0000 (b) Misc 0000 (c) Metal 0004 (d) Fabric 0015 (e) Terrain 0000 (f) Clouds 0000 (g) Tile 0007 (h) Stone 0002 (i) Flowers 0000 (j) Leaves 0009.*

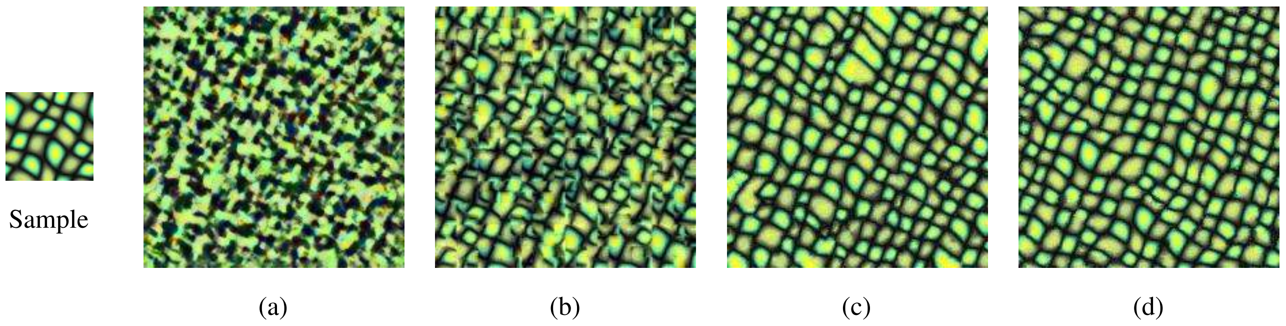

Figure 3: *A comparison of texture synthesis results using different algorithms: (a) Heeger and Bergen's method (b) De Bonet's method. (c) Efros and Leung's method. (d) Our method. Only Efros and Leung's algorithm produces results comparable with ours. However, our algorithm is 2 orders of magnitude faster than theirs. The sample texture patch has size 64x64, and all the result images are of size 192x192.*

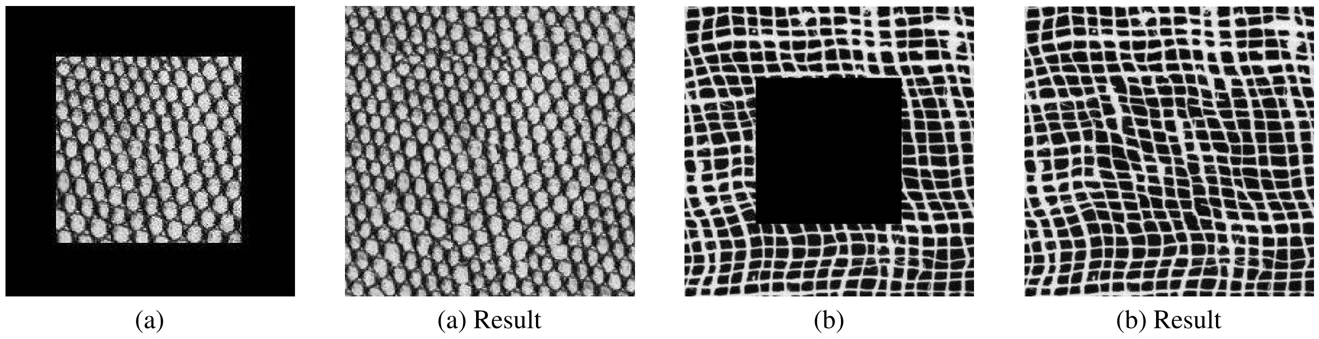

Figure 4: *Constrained synthesis examples. In each pair of figures, the original image is on the left and the synthesized result is on the right. The goal is to fill in the black regions without changing the rest of the image. Examples shown are Brodatz textures with image extrapolation (a) D36 and hole filling (b) D103.*

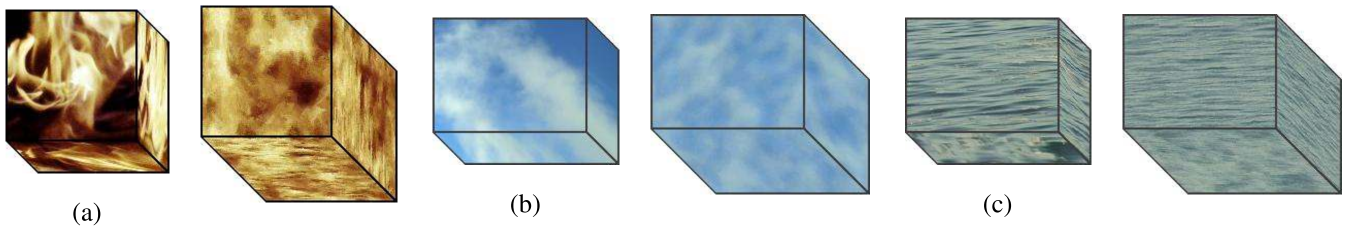

Figure 5: *Temporal texture synthesis results. (a) fire (b) smoke (c) ocean waves. In each pair of images, the spatial-temporal volume of the original motion sequence is shown on the left, and the corresponding synthesis result is shown on the right. A 5x5x5 causal neighborhood is used for synthesis. The original motion sequences contain 32 frames, and the synthesis results contain 64 frames. The individual frame sizes are (a) 128x128 (b) 150x112 (c) 150x112. Accelerated by TSVQ, the training time are (a) 1875 (b) 2155 (c) 2131 seconds and the synthesis time per frame are (a) 19.78 (b) 18.78 (c) 20.08 seconds. To save memory, we use only a random 10 percent of the input neighborhood vectors to build the (full) codebooks.*