Texture Compression using Wavelet Decomposition : A Preview

Pavlos Mavridis* Georgios Papaioannou†

Department of Informatics, Athens University of Economics & Business

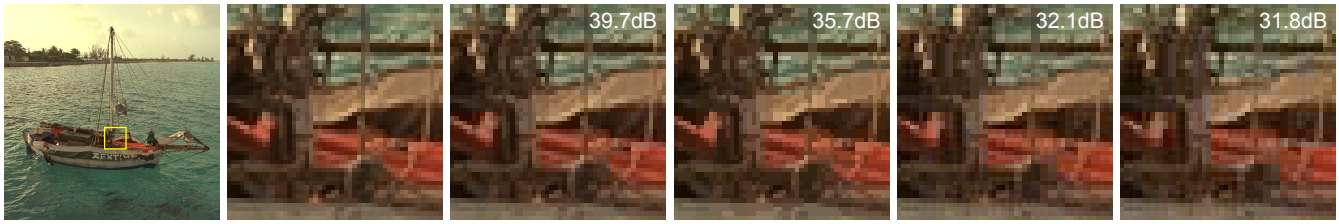

Figure 1: *Demonstration of the proposed method when encoding a color image at various bitrates. Our method runs on commodity graphics hardware and supports efficient anisotropic texture filtering at fast decoding speeds. The numbers denote the Peak Signal to Noise Ratio.*

Abstract

We present a new fixed-rate texture compression scheme based on the energy compaction properties of the Discrete Wavelet Transform. Targeting existing commodity graphics hardware and APIs, our method is using the DXT compression formats to perform the quantization and storage of the wavelet transform coefficients, ensuring very fast decoding speeds. An optimization framework minimizes the quantization error of the coefficients and improves the overall compression quality. Our method provides a variety of low bitrate encoding modes for the compression of grayscale and color textures. These encoding modes offer either improved quality or reduced storage over the DXT1 format. Furthermore, anisotropic texture filtering is performed efficiently with the help of the native texture hardware. The decoding speed and the simplicity of the implementation make our approach well suited for use in games and other interactive applications.

1 Overview

The goal of our method is to extend the currently available DXT compression formats by offering more flexibility in terms of compression rates, as shown in Figure 1.

Wavelet Transform. The first step in our encoding framework is to decompose each channel of an input texture into four coefficient subbands using the two dimensional *Haar* transform. After this transform, each 2×2 block of pixels is transformed to one scale coefficient (ll) and three wavelet coefficients (lh, hl and hh).

Coefficient Packing. The spectral energy of the input image is mostly concentrated in the scale coefficients, while the wavelet coefficients are mostly zero. Our compression scheme stores the coefficients as regular compressed textures on the GPU, to allow for fast random access. The alpha channel of the DXT5 format is compressed at a higher bitrate than color. Thus, we pack there the ll coefficient of each 2×2 pixel block, and the three wavelet coefficients in the RGB channels. This packing gives preferential treatment to the ll coefficients, since they have the highest contribution to the final image. A second storage mode gives additional precision to the hl and lh coefficients, by packing them along with ll in the RGB channels of a BPTC texture, while dropping the hh coefficients. Both formats result in a 2bpp encoding for a single channel. The offline encoder chooses the packing mode that gives the best PSNR.

Coefficient Optimization. Most of the wavelet coefficients are clustered towards zero, while a very few have values at the edges of the color spectrum. To improve the quantization we perform a series of optimizations. First, we clamp a number of coefficients at the edges of the spectrum, and we perform the quantization based on the resulting range of coefficient values. Furthermore, in order to redistribute the coefficient values more evenly in the spectrum, we perform an exponential scale of the data. The optimal amount of scaling and clamping, that gives the highest PSNR, is calculated by an off-line encoder using a hill-climbing optimization process.

YCoCg Transform. The human visual system is much more sensitive to variations of luminance than chrominance. Our method, much like the JPEG standard and many other image and video encoders, exploit this fact in order to encode color images at lower bitrates. Our method encodes color images in the YCoCg color space [Malvar and Sullivan 2003]. The luma channel (Y) is encoded at full resolution, while the chrominance channels (CoCg) are downsampled, depending on the desired quality and bitrate.

Decoding. The actual decoding of the compressed data is straightforward and extremely efficient. With a single texture fetch per channel, the four Haar coefficients are fetched and de-quantized via the dedicated texture hardware and the final texture color is computed by the shader using the inverse Haar and YCoCg transform.

Texture Filtering. Texture filtering must be performed after decompressing the textures, thus preventing the use of native texture filtering. Bilinear texture filtering can be performed in the shader with four texture fetches per channel. Trilinear filtering also costs four fetches, since the ll coefficient corresponds to the lower mip level. We also prove that anisotropic minification can be approximated directly by the hardware, thus a full anisotropic implementation can use the hardware filtering for minification, and gradually switch to software bilinear filtering for magnification, at the cost of only four fetches per channel.

References

MALVAR, H., AND SULLIVAN, G. 2003. *YCoCg-R: A Color Space with RGB Reversibility and Low Dynamic Range*. Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG, Document No. JVTI014r3, July 2003.

*e-mail: pmavridis@gmail.com

†e-mail:gepap@aueb.gr