Pacific Graphics 2012
C. Bregler, P. Sander, and M. Wimmer
(Guest Editors)

Volume 31 (2012), Number 7

Texture Compression using Wavelet Decomposition (Supplemental Material)

Pavlos Mavridis and Georgios Papaioannou

Department of Informatics, Athens University of Economics & Business

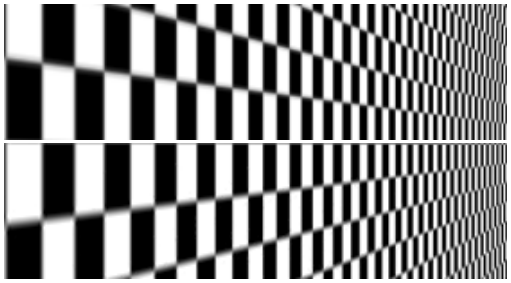


Figure 1: *Our method supports fast and accurate anisotropic filtering. Up: Native hardware anisotropic filtering. Bottom: Anisotropic filtering with wavelet encoded textures. The challenging anisotropic minification case is handled by the native hardware, while our shader only computes bilinear magnification.*

1. Anisotropic Filtering

An accurate implementation of *elliptical anisotropic filtering* in the shader is possible, as shown in [MP11], but the performance overhead can be prohibitive for some applications. We claim that when a lot of texels are projected to a single pixel, which is always the case when anisotropic minification is involved, direct filtering of the wavelet coefficients, as performed by the hardware, provides a very good approximation of the actual anisotropic filtering. A proof of this claim is provided in the Appendix. Since this approximation is only valid when anisotropic minification is involved, for the magnification case our implementation gradually switches to bilinear filtering, thus the total cost of anisotropic filtering is also only four fetches per input channel. In Figure 1 we can observe that the texture filtering produced by our method closely matches the native hardware filtering, while any expensive anisotropic computations in the shader are avoided, since the shader has only to perform the less expensive bilinear filtering for the magnification case. One potential optimization to further improve performance and reduce the consumed bandwidth is to perform the filtering calculations only on the luminance channel.

References

[MP11] MAVRIDIS P., PAPAIOANNOU G.: High quality elliptical texture filtering on GPU. In *Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2011), I3D '11, ACM, pp. 23–30. 1

Appendix A: Anisotropic Filtering of Wavelet Textures

A texture filtering algorithm should compute the following convolution sum of the texels C inside the projected pixel footprint S

$$C_f(s,t) = \sum_{u,v \in S(s,t)} H(u,v)C(u,v) \tag{1}$$

where H is the *normalized* projected reconstruction filter. Substituting $C(u,v)$ from Equation 3(paper) we get

$$C_f(s,t) = \sum_{u,v} H(u,v)LL - \sum_{u,v} H(u,v)f((u\&1) \wedge (v\&1))HH$$
$$- \sum_{u,v} H(u,v)f(v\&1)LH - \sum_{u,v} H(u,v)f(u\&1)HL \tag{2}$$

Instead of this, our method directly filters the wavelet coefficients using the native texture hardware, effectively calculating

$$C_f(s,t) = \sum_{u,v} H(u,v)LL - f((s\&1) \wedge (t\&1)) \sum_{u,v} H(u,v)HH$$
$$- f(t\&1) \sum_{u,v} H(u,v)LH - f(s\&1) \sum_{u,v} H(u,v)HH \tag{3}$$

The wavelet coefficients are mostly values near zero. When S contains many texels, which always happens when anisotropic minification is involved, the weighted sum of the wavelet coefficients inside S is close to zero. This can be seen in the histogram of the coefficients in Figure 2(paper). Therefore, the dominant term in both Eq. 2 and 3 is the first one, which corresponds to the scale coefficient. Therefore, Equation 3 is a good approximation of Equation 2.

Original Scaled DXT1 (2.25 bpp) Proposed (2.25 bpp)

Figure 2: *Visual comparison of our method with a simpler alternative, scaling down the DXT1 textures to achieve the same bitrate reduction (see also Section 4.1, full paper). As demonstrated in the above example this can result in over-smoothing of the high frequency content. In contrast, our method preserves the high frequency details better.*

	2 bpp / gray	5 bpp	4 bpp / dxt1	3 bpp	2.25 bpp
kodim01	31.1432	38.7518	34.4894	30.7634	30.4717
kodim02	36.4652	39.2994	37.0504	35.9071	33.9479
kodim03	37.8101	40.8268	39.1485	36.9557	35.7302
kodim04	35.8738	39.3124	37.7029	35.3152	34.0343
kodim05	30.013	36.3599	33.1634	29.4593	28.8259
kodim06	32.5186	39.7986	35.7663	32.1583	31.8676
kodim07	36.2838	39.7585	37.563	35.4233	34.2811
kodim08	30.0863	36.7518	32.6459	29.6788	29.2497
kodim09	36.6909	40.9917	38.338	35.8404	35.1356
kodim10	36.2798	41.2303	38.4218	35.4868	34.8902
kodim11	33.214	39.8562	36.274	32.7969	32.1949
kodim12	37.8345	41.6034	39.1907	36.8957	36.1157
kodim13	27.3916	36.6417	32.1403	27.1601	27.0072
kodim14	32.0945	36.0656	34.4633	31.4747	29.9906
kodim15	35.7609	39.4109	37.4821	34.9631	33.8123
kodim16	35.9828	42.1936	38.7144	35.8435	35.4833
kodim17	35.3075	41.5126	37.966	34.7047	34.3659
kodim18	30.9488	37.8639	34.7862	30.3808	29.8673
kodim19	34.185	40.3032	36.4587	33.8173	33.3004
kodim20	35.4739	40.0807	38.0965	34.5293	33.8235
kodim21	32.3534	39.5956	35.7739	31.946	31.5754
kodim22	34.2758	38.8693	36.6667	33.4103	32.6802
kodim23	38.0645	39.2894	38.9703	36.8581	35.483
kodim24	30.717	37.0262	34.3387	29.6258	29.2453

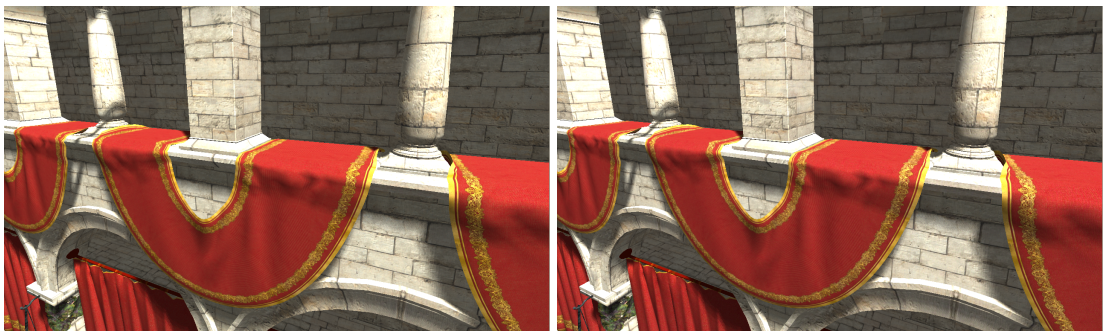
Figure 3: *Peak Signal to Noise Ratio (PSNR) of our method in the kodak set.*

	2 bpp / gray	3.0bpp	2.25bpp
kodim01	2.9537	1.0584	2.3669
kodim02	1.9467	1.048	0.7489
kodim03	2.3506	0.5134	1.088
kodim04	0.6077	-0.0619	-0.0263
kodim05	1.3744	-0.1601	0.7012
kodim06	2.942	1.3522	2.4697
kodim07	1.1604	0.041	0.3367
kodim08	4.4399	2.545	3.6704
kodim09	2.3871	0.6046	1.3907
kodim10	2.1251	0.5068	1.3237
kodim11	2.0687	0.5322	1.4148
kodim12	2.711	1.4628	1.9836
kodim13	1.5345	0.0778	1.3037
kodim14	1.1954	-0.0377	-0.0366
kodim15	2.4715	1.5135	1.2899
kodim16	3.4164	1.621	3.0104
kodim17	1.3201	0.1757	1.105
kodim18	1.0598	-0.3205	0.4654
kodim19	3.9093	2.0394	3.1239
kodim20	2.8505	0.4242	1.3982
kodim21	1.8791	0.2418	1.4491
kodim22	1.8046	0.1373	0.8904
kodim23	1.3664	-0.1168	0.0182
kodim24	2.0268	-0.0719	0.9173

Figure 4: *The coding gain of our format over a scaled down version of DXT1 and DXT5/A(first column), as discussed in the paper. (Section 4.1).*

	2 bpp	4 bpp	3 bpp	2.25 bpp
kodim01	28.1895	34.4894	29.705	28.1048
kodim02	34.5185	37.0504	34.5033	33.199
kodim03	35.4595	39.1485	36.2244	34.6416
kodim04	35.2661	37.7029	35.3771	34.0606
kodim05	28.6386	33.1634	29.6194	28.1247
kodim06	29.5766	35.7663	30.8061	29.3979
kodim07	35.1234	37.563	35.3823	33.9444
kodim08	25.6464	32.6459	27.1338	25.5793
kodim09	34.3038	38.338	35.1966	33.7449
kodim10	34.1547	38.4218	34.98	33.5665
kodim11	31.1453	36.274	32.2647	30.7801
kodim12	35.1235	39.1907	35.4329	34.1321
kodim13	25.8571	32.1403	27.0823	25.7035
kodim14	30.8991	34.4633	31.466	30.0272
kodim15	33.2894	37.4821	33.2981	32.5224
kodim16	32.5664	38.7144	34.2225	32.4729
kodim17	33.9874	37.966	34.529	33.2609
kodim18	29.889	34.7862	30.7013	29.4019
kodim19	30.2757	36.4587	31.7779	30.1765
kodim20	32.6234	38.0965	34.1051	32.4253
kodim21	30.4743	35.7739	31.7042	30.1263
kodim22	32.4712	36.6667	33.1623	31.7878
kodim23	36.6981	38.9703	36.9749	35.4651
kodim24	28.6902	34.3387	29.6977	28.328

Figure 5: *Peak Signal to Noise Ratio (PSNR) of the DXT1 format in the kodak set. For the grayscale results, DXT5/A was used. We have also included the results for the scaled down version, as discussed in the paper. (Section 4.1)*



Uncompressed textures Wavelet compressed textures at 2.25 bpp

Figure 6: *Wavelet encoded textures at 2.25 bpp on a typical game scene. (Model provided by Crytek). Due to the perspective projection, the texture filtering and the modulation of the textures by the lighting and other effects, it is difficult to evaluate a texture compression algorithm in a running application, in terms of quality. Nevertheless, we provide such a comparison here for completeness. Both images are high resolution and can be examined at high zoom levels.*

Figure 7: *Compression of color textures and images with our method using various compression modes. Please note that we have cropped the images at square dimensions for easy inclusion in the pdf, but the PSNR measurements are on the full uncropped images. All images are high resolution and can be examined at high zoom levels.*