Kakadu's Distortion-Length Slopes

David Taubman

31 Mar, 2014

1 Introduction and Conventions

Let $y_b[\mathbf{k}]$ denote the subband samples in code-block b of some image $x[\mathbf{n}]$. Also, let $y'_b[\mathbf{k}]$ denote the quantised subband samples in code-block b and $x'[\mathbf{n}]$ the corresponding synthesized image samples, having a total squared error distortion of

$$D = \sum_{\mathbf{n}} |x[\mathbf{n}] - x'[\mathbf{n}]|^2$$

This distortion may be well approximated by

$$D \approx \sum_b D_b$$

where the sum is over all code-blocks in the image, and the distortion contribution from a single code-block is given by

$$D_b = G_b \sum_{\mathbf{k}} |y_b[\mathbf{k}] - y'_b[\mathbf{k}]|^2$$

Here, G_b denotes the energy gain factor of the wavelet synthesis basis functions associated with the subband to which code-block b belongs. One way to calculate the value of G_b is to create an image in which all subband samples are 0 except for a single sample[1] in code-block b that is set to 1, perform the inverse wavelet transform and then measure the energy (sum of squared sample values) of the resulting image. However, there are much more efficient procedures for calculating G_b that consume negligible memory or computational resources.

Moreover, the total number of bytes (not bits) spent coding the image can be written

$$L = \sum_b L_b$$

where L_b is the number of bytes spent coding the quantised values of code-block b and the sum is over all code-blocks in the image.

[1] This sample should be sufficiently far from the image boundaries that its location has no impact on the procedure.

The operational distortion-length characteristic of a single code-block is the set of distortion-length pairs $\left(D_b^{(p)}, L_b^{(p)}\right)$ that correspond to the truncation of the block coding procedure for code-block b after p coding passes. Roughly speaking, we expect the distortion-length characteristic to be monotonically decreasing and convex $\cup$. However, neither of these properties always holds, so the block coder identifies the subset of coding pass end points

$$0 = p_b^0 < p_b^1 < p_b^2 < \cdots$$

that are potentially optimal truncation points for the code-block. These have the property that $\left\{\left(D_b^{(p_b^i)}, L_b^{(p_b^i)}\right)\right\}_{i=0,1,\ldots}$ define the convex hull of the operational distortion-length characteristic. For optimal truncation of the code-stream, and also for optimal construction of each quality layer, the key feature of each such p_b^i is its distortion-length slope, which is defined as follows:

$$S_b^{(p_b^i)} = \begin{cases} \frac{D_b^{(p_b^{i-1})} - D_b^{(p_b^i)}}{L_b^{(p_b^i)} - L_b^{(p_b^{i-1})}}, & i > 0 \\ \infty & i = 0 \end{cases}$$

Note that the slopes are monotonically decreasing:

$$\infty = S_b^0 > S_b^{(p_b^1)} > S_b^{(p_b^2)} > \cdots$$

A rate-distortion optimal quality layer λ may be defined by a single slope threshold S_λ, such that the cumulative number of coding passes included from code-block b within all layers up to and including layer λ is given by

$$p_b(\lambda) = \max\left\{p_b^i \mid S_b^{(p_b^i)} \geq S_\lambda\right\}$$

2 Log Slopes in Kakadu

The Kakadu implementation of JPEG2000 adopts an unsigned 16-bit logarithmic representation for distortion-length slopes $S_b^{(p_b^i)}$. The regular Kakadu block encoding procedure uses an LUT-based procedure to accurately approximate pass-to-pass distortion changes

$$\Delta D_b^p = D_b^p - D_b^{p-1}$$

After this, the convex hull analysis is readily performed and the slopes $S_b^{(p_b^i)}$ are calculated. Finally, the block coder assigns a log-slope value $\mathcal{K}_b^{(p)}$ to each coding pass, where

$$\mathcal{K}_b^{(p)} = \begin{cases} \max\left\{2, 2^{16} + \min\left\{-1, 2^8 \log_2\left(\frac{W_b S_b^{(p_b^i)}}{2^{80}}\right)\right\}\right\}, & \text{if } p = p_b^i \\ 0, & \text{otherwise} \end{cases}$$

Note that the non-zero values for $\mathcal{K}_b^{(p)}$ correspond to points that lie on the convex hull of the distortion-length characteristic. These values are formed by evaluating

$$2^{16} + 2^8 \log_2 \left(\frac{W_b S_b^{(p_b^i)}}{2^{80}} \right) = 2^{16} + 2^8 \left[\log_2 \left(W_b S_b^{(p_b^i)} \right) - 80 \right]$$
$$= 2^8 \left[176 + \log_2 \left(W_b S_b^{(p_b^i)} \right) \right] \quad (1)$$

and truncating any values that are smaller than 2 or exceed $2^{16} - 1$.

In the above expression, W_b refers to a "visual" weighting factor that may apply to the code-block. For rate-distortion optimisation with MSE as the distortion metric, $W_b = 1$. Other values for W_b may be supplied on a subband-by-subband basis via Kakadu's "Cband_weights" coding parameter attribute; more specifically, the values supplied to "Cband_weights" correspond to the square-roots of the W_b values in equation (1).

The linear slope values $S_b^{(p)}$ can be recovered from non-zero log-slope values $\mathcal{K}_b^{(p)}$ via

$$W_b S_b^{(p)} = 2^{2^{-8} \mathcal{K}_b^{(p)} - 176} \quad (2)$$

It is important to appreciate that the distortion and hence slope values are derived for normalised image sample values that have a unit nominal range. That is, the image samples $x[\mathbf{n}]$ have effectively been pre-scaled and level adjusted so that

$$-\frac{1}{2} \leq x[\mathbf{n}] \leq \frac{1}{2}$$

Since $S_b^{(p_b^i)}$ is calculated for normalized data, the MSE distortion-length slope for an image whose samples have B bit precision is actually $2^{2B} \cdot S_b^{(p_b^i)}$.

3 Experimental Validation

Since there have been various conflicting descriptions of Kakadu's log-slope values, it is worth having an experimental procedure to verify that the definition agrees with practice. To this end, we compress the 8-bit monochrome version of the well-known 512×512 image Lenna using Kakadu, with various bit-rate constraints and report the MSE distortions and log-slope values produced by KDU-7.3.3 in Table 1. The MSE values correspond to $2^{2B} D/A$, where $B = 8$ is the bit-depth, $A = 512^2$ is the number of image samples, and D is our normalized distortion. Similarly, the bit-rate R corresponds to $8L/A$ where L is the total byte count. This allows us to estimate the actual distortion-length slopes S from $\Delta D/\Delta L = 2^{-13} \Delta_{\mathrm{MSE}}/\Delta R$ where Δ refers to the difference between each operating point and the immediately lower rate operating point.

Table 1: Compression results for Lenna

$R = 8L/A$ (bpp)	MSE= $2^{16}D/A$	$\frac{\Delta D}{\Delta L} \approx S$	log-slope threshold $\mathcal{K}$	$2^{2^{-8}\mathcal{K}-176} = S$
0.7	8.69		42603	0.0013
0.8	7.72	0.0012	42550	0.0011
0.9	6.79	0.0011	42522	0.0010
1.0	5.97	0.0010	42488	0.0010
1.1	5.32	0.0008	42352	0.0007

4 Historical Notes for Kakadu

- In recent versions of Kakadu (7.3.3 and earlier), the API documentation states that the log-slope values are of the form

  $$256 * (256 - 64) + 256 \cdot \log_2(\Delta D/\Delta L)$$

  where ΔD is assessed from the perspective of an image whose samples have a nominal range from -128 to 128, regardless of the actual bit-depth of the image. This description was, regrettably, incorrect. The different nominal range given in the documentation is equivalent to an offset of $256 * (256 - 48)$ for the unit nominal range, rather than the intended $256 * (256 - 80)$. The documentation has been corrected from Kakadu version 7.3.4 onwards.

- In Kakadu versions 6.3.1 and earlier, the log-slope values were larger than those of later versions by exactly 2^{13}.

- The speed-pack version of Kakadu has always used a model-based distortion estimation strategy rather than the LUT-based approach of regular Kakadu. The model-based approach tended to slightly under-estimate the true distortion-length slopes up until version KDU-S7.2, from which point a significantly more accurate model has been employed so that there is little discrepancy between regular Kakadu and the speed-pack version of Kakadu.