Further Performance Results

results for independent channel coding, default strength

results for joint-ch. coding, strength 4 - perceptMode

Changes to Specification Text

In Clause 8.2, correct text "the variable minVall is set to – maxVal- 1" *to* "the variable minVal is set to –maxVal – 1".

In Clause 8.5.1, extend / correct text "For 0 <= j < szExt, the" *as follows:* "For 0 <= j < szExt and if szArr > 4, the".

Add the following text after Clause 8.11.8:

8.12 Deblocking process

The time-domain deblocking process is the last operational step of the block-wise and channel-wise decoding process. It is applied only when transform_present_flag associated with the given block and channel index currCh is equal to 1 and currBlockPos > 0 and when the global parameter percept_mode associated with the given channel group is greater than 0. Otherwise, all four steps below are bypassed and the reconstructed sample values rec[currCh] left unchanged.

Input to this process are the following, all of which are as defined in clause 8.2:
- the bit depth value BitDepthMax,
- the variables maxVal and minVal,
- the current channel index currCh,
- the current block size blockSize,
- the reconstructed sample values rec[currCh][i] with –4 <= i < blockSize.

Output to this process are the deblocked newly reconstructed sample values rec[currCh][i] with 0 <= i < blockSize.

The deblocking post-processor employs a three-step blocking detection algorithm, followed by the actual deblocking. When transform_present_flag equals 1 and currBlockPos >= 4, the variable thresh is defined as the output tCoeff[0] resulting from invoking clause 8.6.1 on QuantIndices[0] = 1 with Log2BlockSize = 0 and transform_present_flag = 0, scaled by 1 << max(0, BitDepthMax–12), and the following steps are applied to deblock sample values rec[currCh]:

- **Step 1: linear fitting** via fixed-point linear regression, on the four channel waveform samples on each side of the left boundary of the currently decoded block. Using each of the two linear models, the sample value between the first sample of the current and last sample of the previous block is estimated, and difference value valDiff is calculated. If valDiff ≠ 0 and |valDiff·2| is smaller than input bit depth dependent threshold thresh, proceed to step **2**, otherwise abort.

  The linear fit is applied by invoking the linear extrapolation process of clause 8.5.1 twice, first with array p = {rec[currCh][–4], rec[currCh][–3], rec[currCh][–2], rec[currCh][–1]}, szArr = 4 and szExt = 0, yielding slope, offset and meanValExtr of the left boundary side, and then with array p = {rec[currCh][3], rec[currCh][2], rec[currCh][1], rec[currCh][0]}, szArr = 4 and szExt = 0, yielding slope, offset and meanValExtr of the right boundary side. valDiff is set to the difference between the two meanValExtr values (first minus second value).

– **Step 2: evaluation** of each of the two linear models on the four samples on the corresponding side of the current block's left boundary and calculation of the 2 fitting errors (sum of absolute distance-weighted differences between each line value and its associated actual sample value). The evaluation is performed separately for each boundary side, with scale = 3 – percept_mode and respective slope, offset and array p associated with that boundary side, as defined in step 1:

fitting error = 0;
for (ix = –15, i = 0; i < 4; ix += 10, i += 1) {
  int val = Clip3(–(2^{31}–1), (2^{31}–1), (offset + slope · ix) >> 9);
  fitting error += (scale << i) · | val – p[i] |;
}
If the product of both fitting error values is greater than or equal to zero, both errors are scaled by 2. Then, if both of the two errors are smaller than thresh, proceed to step **3**, otherwise abort.

– **Step 3: calculation of L1 norm** of vector rec[currCh][i], yielding value sumAbsIn, and of a 1st-order high-pass filtered version of vector rec[currCh][i] using FIR filter [1 $–z^{-1}$], yielding

sumAbsHP = L1norm(rec[currCh][i] – rec[currCh][i – 1]), $0 < i <$ blockSize,

where L1norm(x) = sum(| x |) and the values resulting for i < 1 are excluded in the summations. Value sumAbsHP is then modified as follows: sumAbsHP = max(sumAbsIn, sumAbsHP · 16). If, for each fitting error of step 2, error · sumAbsHP < thresh · sumAbsIn (current block signal is spectrally sufficiently low-pass to justify a deblocking), proceed to step **4**, otherwise abort.

– **Step 4: execution of deblocking** algorithm in-place on the first 4 reconstructed samples of the currently decoded block, by adding a fading (away from boundary) ramp signal using valDiff:

rec[currCh][j] = Clip3(minVal, maxVal, rec[currCh][j] + ((valDiff · (7 – 2·j) + 4) >> 3)),

$0 <= j < 4$.