described in SMPTE ST 2084, and the "HybridLog-gamma" or "HLG" curve described in Rec. ITU-R BT. 2100.

[0006] As used herein, the term "forward reshaping" denotes a process of sample-to-sample or codeword-to-codeword mapping of a digital image from its original bit depth and original codewords distribution or representation (e.g., gamma or PQ or HLG, and the like) to an image of the same or different bit depth and a different codewords distribution or representation. Reshaping allows for improved compressibility or improved image quality at a fixed bit rate. For example, without limitation, reshaping may be applied to 10-bit or 12-bit PQ-coded HDR video to improve coding efficiency in a 10-bit video coding architecture. In a receiver, after decompressing the reshaped signal, the receiver may apply an "inverse reshaping function" to restore the signal to its original codeword distribution. As appreciated by the inventors here, as development begins for the next generation of a video coding standard, improved techniques for the integrated reshaping and coding of images are desired. Methods of this invention can be applicable to a variety of video content, including, but not limited, to content in standard dynamic range (SDR) and/or high-dynamic range (HDR).

[0007] The approaches described in this section are approaches that could be pursued, but not necessarily approaches that have been previously conceived or pursued. Therefore, unless otherwise indicated, it should not be assumed that any of the approaches described in this section qualify as prior art merely by virtue of their inclusion in this section. Similarly, issues identified with respect to one or more approaches should not assume to have been recognized in any prior art on the basis of this section, unless otherwise indicated.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] An embodiment of the present invention is illustrated by way of example, and not in way by limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

[0009] FIG. 1A depicts an example process for a video delivery pipeline;

[00010] FIG. 1B depicts an example process for data compression using signal reshaping according to prior art;

[00011] FIG. 2A depicts an example architecture for an encoder using hybrid in-loop reshaping according to an embodiment of this invention;

[00012] FIG. 2B depicts an example architecture for a decoder using hybrid in-loop reshaping according to an embodiment of this invention;

[00013] FIG. 2C depicts an example architecture for intra-CU decoding using reshaping according to an embodiment;

[00014] FIG. 2D depicts an example architecture for inter-CU decoding using reshaping according to an embodiment;

[00015] FIG. 2E depicts an example architecture for intra-CU decoding within inter-coded slices according to an embodiment for luma or chroma processing;

[00016] FIG. 2F depicts an example architecture for intra-CU decoding within inter-coded slices according to an embodiment for chroma processing;

[00017] FIG. 3A depicts an example process for encoding video using a reshaping architecture according to an embodiment of this invention;

[00018] FIG. 3B depicts an example process for decoding video using a reshaping architecture according to an embodiment of this invention;

[00019] FIG. 4 depicts an example process for reassigning codewords in the reshaped domain according to an embodiment of this invention;

[00020] FIG. 5 depicts an example process for deriving reshaping thresholds according to an embodiment of this invention;

[00021] FIG. 6A, 6B, 6C, and 6D depict example data plots for deriving reshaping thresholds according to the process depicted in FIG. 5 and an embodiment of this invention; and

[00022] FIG. 6E depicts examples of codeword allocation according to bin variance according to embodiments of this invention.

DESCRIPTION OF EXAMPLE EMBODIMENTS

[00023] Signal reshaping and coding techniques for compressing images using rate-distortion optimization (RDO) are described herein. In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are not described in exhaustive detail, in

order to avoid unnecessarily occluding, obscuring, or obfuscating the present invention.

OVERVIEW

[00024] Example embodiments described herein relate to signal reshaping and coding for video. In an encoder, a processor receives an input image in a first codeword representation to be reshaped to a second codeword representation, wherein the second codeword representation allows for a more efficient compression than the first codeword representation, and generates a forward reshaping function mapping pixels of the input image to a second codeword representation, wherein to generate the forward reshaping function, the encoder: divides the input image into multiple pixel regions, assigns each of the pixel regions to one of multiple codeword bins according to a first luminance characteristic of each pixel region, computes a bin metric for each one of the multiple codeword bins according to a second luminance characteristic of each of the pixel regions assigned to each codeword bin, allocates a number of codewords in the second codeword representation to each codeword bin according to the bin metric of each codeword bin and a rate distortion optimization criterion, and generates the forward reshaping function in response to the allocation of codewords in the second codeword representation to each of the multiple codeword bins.

[00025] In another embodiment, in a decoder, a processor receives coded bitstream syntax elements characterizing a reshaping model, wherein the syntax elements include one or more of a flag indicating a minimum codeword bin index value to be used in a reshaping construction process, a flag indicating a maximum codeword bin index value to be used in a reshaping construction process, a flag indicating a reshaping model profile type, wherein the model profile type is associated with default bin-relating parameters, including bin importance values, or a flag indicating one or more delta bin importance values to be used to adjust the default bin importance values defined in the reshaping model profile. The processor determines based on the reshaping model profile the default bin importance values for each bin and an allocation list of a default numbers of codewords to be allocated to each bin according to the bin's importance value. Then,
for each codeword bin, the processor:

generates second reshaped reconstructed samples of the coding unit based on reshaped residuals in the coded CU and the second reshaped prediction samples;

generates a reshaped loop filter output based on the second reshaped reconstructed samples and loop-filter parameters;

applies the inverse reshaping function to the reshaped loop filter output to generate samples of the coding unit in the second codeword representation; and

stores the samples of the coding unit in the second codeword representation in a reference buffer. Finally, the processor generates a decoded image based on the stored samples in the reference buffer.

[00027] In another embodiment, in a decoder, a processor receives a coded bitstream comprising one or more coded reshaped images in an input codeword representation and

reshaping metadata (207) for the one or more coded reshaped images in the coded bitstream. The processor generates a forward reshaping function (282) based on the reshaping metadata, wherein the forward reshaping function maps pixels of an image from a first codeword representation to the input codeword representation. The processor

generates an inverse reshaping function (265-3) based on the reshaping metadata or the forward reshaping function, wherein the inverse reshaping function maps pixels of a reshaped image from the input codeword representation to the first codeword representation. The processor extracts from the coded bitstream a coded reshaped image comprising one or more coded units, wherein:

for an intra-coded coding unit (intra-CU) in the coded reshaped image, the processor:

generates reshaped reconstructed samples of the intra-CU (285) based on reshaped residuals in the intra-CU and intra-predicted reshaped prediction samples;

applies the inverse reshaping function (265-3) to the reshaped reconstructed samples of the intra-CU to generate decoded samples of the intra-CU in the first codeword representation;

applies a loop filter (270) to the decoded samples of the intra-CU to generate output samples of the intra-CU; and

stores the output samples of the intra-CU in a reference buffer;

for an inter-coded CU (inter-CU) in the coded reshaped image, the processor:

else

$M_k = M_a$;

end

where TH_L denotes a predetermined lower threshold.

[00048] The above examples show how to select the number of codewords for each bin from two pre-select numbers M_f and M_a. Thresholds (e.g., TH_U or TH_L) can be determined based on optimizing the rate distortion, e.g., through exhaustive search. Thresholds may also be adjusted based on the quantization parameter values (QP). In an embodiment, for B=10, thresholds may range between 1,000 and 10,000.

[00049] In an embodiment, to expedite processing, a threshold may be determined from a fixed set of values, say, {2,000, 3,000, 4,000, 5,000, 6,000, 7,000}, using a *Lagrangian* optimization method. For example, for each TH(i) value in the set, using pre-defined training clips, one can run compression tests with fixed QP, and compute values of an objective function J defined as

$$J(i) = D + \lambda R. \tag{7}$$

Then, the optimal threshold value may be defined as the TH(i) value in the set for which $J(i)$ is minimum.

[00050] In a more general example, one can predefine a look-up table (LUT). For example, in Table 1, the first row defines a set of thresholds dividing the full range of possible bin metrics (e.g., $var_{bin}(k)$ values) into segments, and the second row defines the corresponding number of codewords (CW) to be assigned in each segment. In an embodiment, one rule to build such a LUT is: if the bin variance is too big, one may need to spend lots of bits to reduce the SSE, therefore one can assign codeword (CW) values less than M_a. If the bin variance is very small, one can assign a CW value larger than M_a.

[00051]

Table 1: Example LUT of codeword allocation based on bin variance thresholds

TH_0	…	TH_{p-1}	TH_p	TH_{p+1}	…	TH_{q-1}	
CW_0	…	CW_{p-1}	CW_p	CW_{p+1}	…	CW_{q-1}	CW_q

[00052] Using Table 1, the mapping of thresholds into codewords may be generated as follows:

For the k-th bin,

if there are no pixels in the bin

desired reshaped dynamic range is divided into *N* bins. After the input image is divided into non- overlapping blocks (step 410), for each block:

- Step 415 computes its luminance characteristics (e.g., mean and variance)
- Step 420 assigns each image block to one of the *N* bins
- Step 425 computes the average luminance variance in each bin

Given the values computed in step 425, in step 430, each bin is assigned a number of codewords according to one or more thresholds, for example, using any of the codeword allocation algorithms depicted in equations (4) to (8). Finally, in step (435), the final codeword allocation may be used to generate a forward reshaping function and/or an inverse reshaping function.

[00057] In an embodiment, as an example and without limitation, the forward LUT (FLUT) can be built using the following *C* code.

```
tot_cw = 2^B;
hist_lens = tot_cw/N;
for (i = 0; i < N; i++)
 {
  double temp = (double) M[i] / (double)hist_lens; //M[i] corresponds to Mk
  for (j = 0; j < hist_lens; j++)
  {
   CW_bins_LUT_all[i*hist_lens + j] = temp;
  }
 }

 Y_LUT_all[0] = CW_bins_LUT_all[0];
 for (i = 1; i < tot_cw; i++)
 {
  Y_LUT_all[i] = Y_LUT_all [i - 1] + CW_bins_LUT_all[i];
 }

for (i = 0; i < tot_cw; i++)
 {
  FLUT[i] = Clip3(0, tot_cw - 1, (Int)(Y_LUT_all[i] + 0.5));
 }
```

[00058] In an embodiment, the inverse LUT can be built as follows:

```
 low = FLUT[0];
 high = FLUT[tot_cw - 1];
 first = 0;
 last = tot_cw - 1;
 for ( i = 1; i < tot_cw; i++)
  if (FLUT[0] < FLUT[i])
  {
   first = i - 1;
   break;
```

else

bin_importance[k] = 0;

3) Automatically pre-assign codewords based on bin importance:

for the *k*-th bin,

if bin_importance[k] == 0

$M_k = 0$;

else if bin_importance[k] == 2

$M_k = M_f$;

else

$M_k = M_a$;

end

4) Build forward reshaping LUT based on codeword assignment for each bin, by accumulating the codeword assigned for each bin. The sum up should be less or equal to total codeword budget (e.g. 1024 for 10-bit full range). (E.g., see earliest C code).
5) Build inverse reshaping LUT (e.g., see earliest *C* code).

[00066] From a syntax point of view, alternative methods can also be applied. The key is to specify the number of codewords in each bin (e.g., M_k, for $k = 0, 1, 2, \ldots, N-1$) either explicitly or implicitly. In one embodiment, one can specify explicitly the number of codewords in each bin. In another embodiment, one can specify the codewords differentially. For example, the number of codewords in a bin can be determined using the difference of the number of codewords in the current bin and the previous bin (e.g., M_Delta(k) = M(k) – M(k-1)). In another embodiment, one can specify the most commonly used number of codewords (say, M_M) and express the number of codewords in each bin as the difference of the codeword number in each bin from this number (e.g., M_Delta(k) = M(k)- M_M).

[00067] In an embodiment, two reshaping methods are supported. One is denoted as the "default reshaper," where M_f is assigned to all bins. The second, denoted as "adaptive reshaper," applies the adaptive reshaper described earlier. The two methods can be signaled to a decoder as in Ref. [6] using a special flag, e.g., sps_reshaper_adaptive_flag (e.g., use sps_reshaper_adaptive_flag = 0 for the default reshaper and use sps_reshaper_adaptive_flag = 1) for the adaptive reshaper.

equal segments as well; however, forcing such a representation may cause loss in coding efficiency. As a compromise, in an embodiment one may be able to construct an inverse reshaping function with a "mixed" PWL representation, combining both equal and unequal segments. For example, when using 8 segments, one may first divide the whole range to two equal segments, and then subdivide each of these into 4 unequal segments. Alternatively, one may divide the whole range into 4 equal segments and then subdivide each one into two unequal segments. Alternatively, one may first divide the whole range into several unequal segments, then subdivide each unequal segment into multiple equal segments. Alternatively, one may first divide the whole range into two equal segments, and then subdivide each equal segment into equal sub-segments, where the segment length in each group of sub-segments is not the same.

[000119] For example, without limitation, with 1,024 codewords, one could have: a) 4 segments with 150 codewords each and two segments with 212 codewords each, or b) 8 segments with 64 codewords each and 4 segments with 128 codewords each. The general purpose of such a combination of segments is to reduce the number of comparisons required to identify the PWL-piece index given a code value, thus simplifying hardware and software implementations.

[000120] In an embodiment, for a more efficient implementation related to chroma residue scaling, the following variations may be enabled:

- Disable the chroma residual scaling when separate luma/chroma trees are used
- Disable the chroma residual scaling for 2x2 chroma; and
- Use the prediction signal rather than the reconstruction signal for intra as well as inter coded units

[000121] As an example, given the decoder depicted in FIG. 2E (200D_D) to process the luma component, FIG. 2F depicts an example architecture (200D_DC) for processing the corresponding chroma samples.

[000122] As depicted in FIG. 2F, compared to FIG. 2E, the following changes are made when processing chroma:

- The forward and reverse reshaping blocks (282 and 265-3) are not used

EEE 1. A method for adaptive reshaping of a video sequence with a processor, the method comprising:

accessing with a processor an input image in a first codeword representation; and

generating a forward reshaping function mapping pixels of the input image to a second codeword representation, wherein the second codeword representation allows for a more efficient compression than the first codeword representation, wherein generating the forward reshaping function comprises:

dividing the input image into multiple pixel regions;

assigning each of the pixel regions to one of multiple codeword bins according to a first luminance characteristic of each pixel region;

computing a bin metric for each one of the multiple codeword bins according to a second luminance characteristic of each of the pixel regions assigned to each codeword bin;

allocating a number of codewords in the second codeword representation to each codeword bin according to the bin metric of each codeword bin and a rate distortion optimization criterion;

and generating the forward reshaping function in response to the allocation of codewords in the second codeword representation to each of the multiple codeword bins.

EEE 2. The method of EEE 1, wherein the first luminance characteristic of a pixel region comprises the average luminance pixel value in the pixel region.

EEE 3. The method of EEE 1, wherein the second luminance characteristic of a pixel region comprises the variance of luminance pixel values in the pixel region.

EEE 4. The method of EEE 3, wherein computing a bin metric for a codeword bin comprises computing the average of the variances of luminance pixel values for all pixels regions assigned to the codeword bin.

EEE 5. The method of EEE 1, wherein allocating a number of codewords in the second codeword representation to a codeword bin according to its bin metric comprises:

applying the forward reshaping function to prediction samples stored in the reference buffer in the second codeword representation to generate second reshaped prediction samples;

generating second reshaped reconstructed samples of the coding unit based on reshaped residuals in the coded CU and the second reshaped prediction samples;

generating a reshaped loop filter output based on the second reshaped reconstructed samples and loop-filter parameters;

applying the inverse reshaping function to the reshaped loop filter output to generate samples of the coding unit in the second codeword representation; and

storing the samples of the coding unit in the second codeword representation in a reference buffer;

and

generating a decoded image based on the stored samples in the reference buffer.

EEE 23. An apparatus comprising a processor and configured to perform a method as recited in any one of the EEEs 1-22.

EEE 24. A non-transitory computer-readable storage medium having stored thereon computer-executable instruction for executing a method with one or more processors in accordance with any one of the EEEs 1-22.