NETVC (Internet Video Codec)
Internet-Draft
Intended status: Informational
Expires: May 4, 2017

Y. Cho
Mozilla Corporation

October 31, 2016

Applying PVQ Outside Daala
draft-cho-netvc-applypvq-00

Abstract

This document describes the Perceptual Vector Quantization (PVQ) outside of the Daala video codec, where PVQ was originally developed. It discusses the issues arising while integrating PVQ into a traditional video codec, AV1.

Table of Contents

1. Background

Perceptual Vector Quantization (PVQ) [I-D.valin-netvc-pvq] has been proposed as a quantization and coefficient coding tool for an internet video codec. PVQ was originally developed for Daala video codec [1], which does a gain-shape coding of transform coefficients instead of more traditional scalar quantization. (The original abbreviation of PVQ, "Pyramid Vector Quantizer", as in [I-D.valin-netvc-pvq] is now commonly expanded as "Perceptual Vector Quantization".)

The most distinguishing idea of PVQ is the way it referneces a predictor. With PVQ, we do not subtract the predictor from the input to produce a residual, which is then transformed and coded. Both the predictor and the input are transformed into the frequency domain. Then, PVQ applies a reflection to both the predictor and the input such that the prediction vector lies on one of the coordinate axes, and codes the angle between them. By not subtracting the predictor from the input, the gain of the predictor can be preserved and is explicitly coded, which is one of the benefits of PVQ. Since DC is not quantized by PVQ, the gain can be viewed as the amount of contrast in an image, which is an important perceptual parameter.

Also, an input block of transform coefficients is split into frequency bands based on their spatial orientation and scale. Then, each band is quantized by PVQ separately. The 'gain' of a band indicates the amount of contrast in the corresponding orientation and scale. It is simply the L2 norm of the band. The gain is non-linearly companded and then scalar quantized and coded. The remaining information in the band, the 'shape', is then defined as a point on the surface of a unit hypersphere.

Another benefit of PVQ is activity masking based on the gain, which automatically controls the quantization resolution based on the image contrast without any signaling. For example, for a smooth image area (i.e. low contrast thus low gain), the resolution of quantization will increase, thus fewer qunatization errors will be shown. Succint summary on the benefits of PVQ can be found in the Section 2.4 of [Terriberry_16].

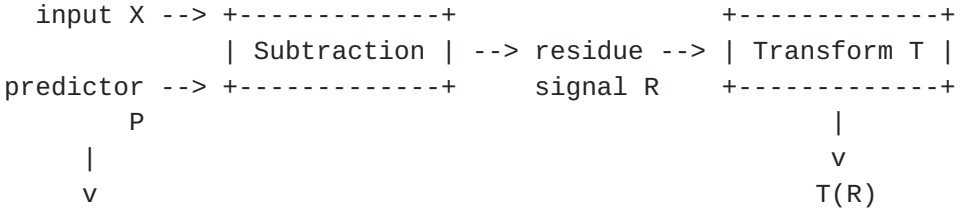
Since PVQ has only been used in the Daala video codec, which contains many non-traditional design elements, there has not been any chance to see the relative coding performance of PVQ compared to scalar quantization in a more traditional codec design. We have tried to apply PVQ in AV1 video codec, which is currently being developed by Alliance for Open Media (AOM) as an open source and royalty-free video codec. While most of benefits of using PVQ arise from the enhancement of subjective quality of video, compression results with activity masking enabled are not available yet in this draft because the required parameters, which were set for Daala, have not been adjusted to AV1 yet. These results were achieved optimizing solely for PSNR.

2. Integration of PVQ into non-Daala codec, AV1

Adopting PVQ in AV1 requires replacing both the scalar quantization step and the coefficient coding of AV1 with those of PVQ. In terms of inputs to PVQ and the usage of trasnforms as shown in Figure 1 and Figure 2, the biggest conceptual change required in a traditional coding system, such as AV1, is

- Introduction of a transformed predictor both in encoder and decoder. For this, we apply a forward transform to the predictors, both intra-predicted pixels and inter-predicted (i.e., motion-compensated) pixels. This is because PVQ references the predictor in the transform domain, instead of using a pixel-domain residual as in traditional scalar quantization.

- Absence of a difference signal (i.e. residue) defined as "input source - predictor". Hence AV1 with PVQ does not do any 'subtraction' in order for an input to reference the predictor. Instead, PVQ takes a different approach to referencing the predictor which happens in the transform domain.



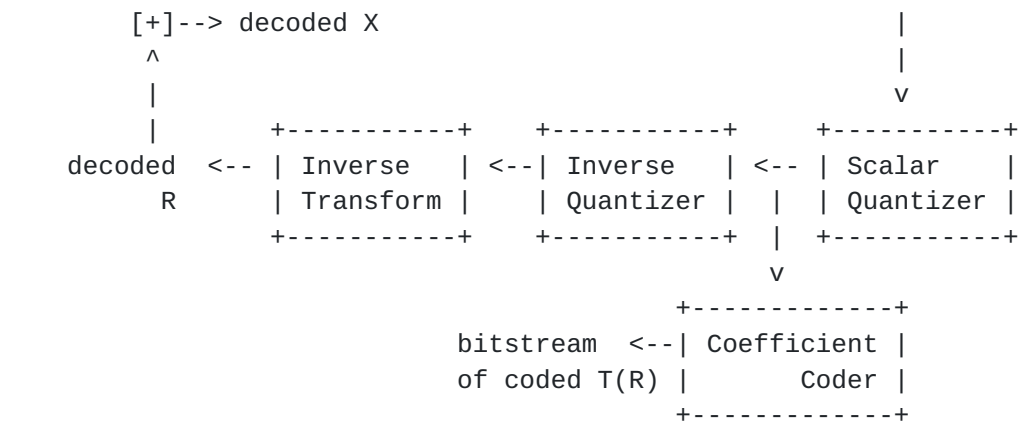


Figure 1: Traditional architecture containing Quantization and Transforms

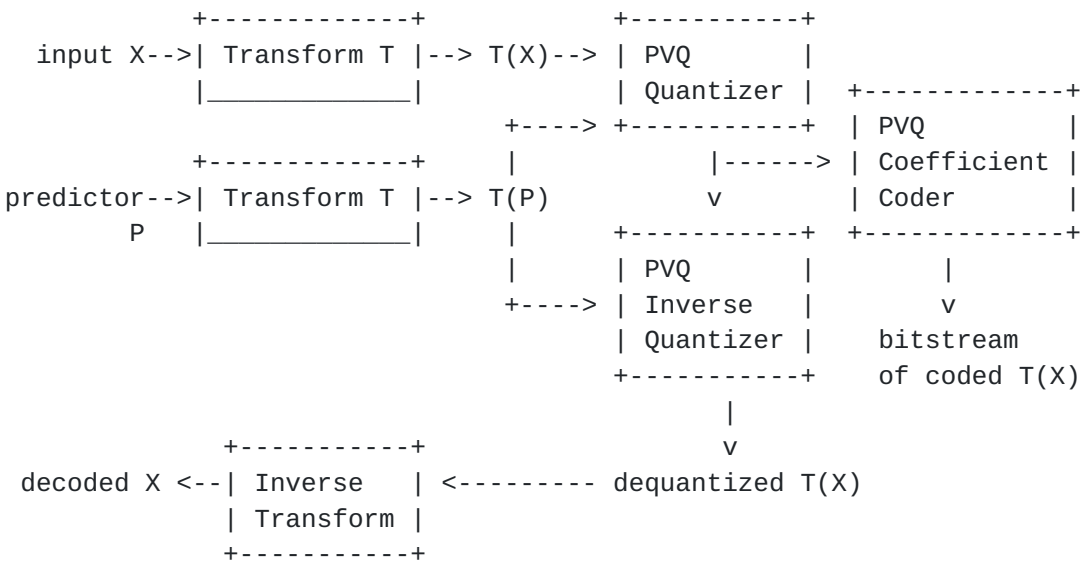


Figure 2: AV1 with PVQ

2.1. Signaling Skip for Paritition and Transform Block

In AV1, a skip flag for a partition block is true if all of quauntized coefficients in the partition are zeros. The signaling for the prediction mode in a partition cannot be skipped. If the skip flag is true with PVQ, the predicted pixels are the final decoded pixels (plus frame wise in-loop filtering such as deblocking) as in AV1 then a forward transform of a predictor is not required.

While AV1 currently defines only one 'skip' flag for each 'partition' (a unit where prediction is done), PVQ introduces another kind of 'skip' flag, called 'ac_dc_coded', which is defined for each transform block (and thus for each Y'CbCr plane as well). AV1 allows

that a transform size can be smaller than a partition size which leads to partitions that can have multiple transform blocks. The ac_dc_coded flag signals whether DC and/or whole AC coefficients are coded by PVQ or not (PVQ does not quantize DC itself though).

2.2. Issues

- PVQ has its own rate-distortion optimization (RDO) that differs from that of traditional scalar quantization. This leads the balance of quality between luma and chroma to be different from that of scalar quantization. When scalar quantization of AV1 is done for a block of coefficients, RDO, such as trellis coding, can be optionally performed. The second pass of 2-pass encoding in AV1 currently uses trellis coding. When doing so it appears a different scaling factor is applied for each of Y'CbCr channels.

- In AV1, to optmize speed, there are inverse transforms that can skip applying certan 1D basis functions based on the distribution of quantized coefficients. However, this is mostly not possible with PVQ since the inverse transform is applied directly to a dequantized input, instead of a dequantized difference (i.e. input source - predictor) as in traditional video codec. This is true for both encoder and decoder.

- PVQ was originally designed for the 2D DCT, while AV1 also uses a hybrid 2D transform consisting of a 1D DCT and a 1D ADST. This requires PVQ to have new coefficient scanning orders for the two new 2D transforms, DCT-ADST and ADST-DCT (ADST-ADST uses the same scan order as for DCT-DCT). Those new scan orders has been produced based on that of AV1, for each PVQ-defined-band of new 2D transforms.

3. Performance of PVQ in AV1

3.1. Coding Gain

With the encoding options specified by both NETVC ([2]) and AOM testing for high latency case, PVQ gives similar coding efficiency to that of AV1, which is measured in PSNR BD-rate. Again, PVQ's activity masking is not turned on for this testing. Also, scalar quantization has matured over decades, while video coding with PVQ is much more recent.

We compare the coding efficiency for one of IETF test sequence set "objective-1-fast" defined in [3], which consists of sixteen of 1080p, seven of 720p, and seven of 640x360 sequences of various types of content, including slow/high motion of people and objects, animation, computer games and screen casting. The encoding is done

for the first 30 frames of each sequence. The encodig options used is : "-end-usage=q -cq-level=x --passes=2 --good --cpu-used=0 --auto-alt-ref=2 --lag-in-frames=25 --limit=30", which is official test condition of IETF and AOM for high latency encoding except limiting 30 frames only.

For comparison reasons, some of the lambda values used in RDO are adjusted to match the balance of luma and chroma quality of the PVQ-enabled AV1 to that of current AV1.

- Use half the value of lambda during intra prediction for the chroma channels.

- Scale PVQ's lambda by 0.8 for the chroma channels.

- Do not do RDO of DC for the chroma channels.

The result are shown in Table 1, which is the BD-Rate change for several image quality metrics. (The encoders used to generate this result are available from author's git repository [4] and AOM's repositiony [5].)

Metric	AV1 --> AV1 + PVQ
PSNR	0.10%
PSNR-HVS	0.53%
SSIM	1.27%
MS-SSIM	0.42%
CIEDE2000	-0.94%

Table 1: Coding Gain by PVQ in AV1

3.2. Speed

Total encoding time increases roughly 20 times or more when intensive RDO options, such as "--passes=2 --good --cpu-used=0 --auto-alt-ref=2 --lag-in-frames=25", are turned on. The biggest reason for significant increase in encoding time is due to the increased computation by the PVQ. The PVQ tries to find asymptotically-optimal codepoints (in RD optimization sense) on a hypershpere with a greedy search, which has the time complexity close to O(n*n) for n coefficients. Meanwhile, scalar quantization has the time complexity of O(n).

Comparing to Daala, the search space for a RDO decision in AV1 is far larger because AV1 considers ten intra prediction modes and four

different transforms (for the transform block sizes 4x4, 8x8, and 16x16 only), and the transform block size can be smaller than the prediction block size. Since the largest transform and the prediction sizes are currently 32x32 and 64x64 in AV1, PVQ can be called approximately 5,160 times more in AV1 than in Daala. Also, AV1 uses transform and quantization for each candidate of RDO.

As an example, AV1 calls PVQ function 632,520 times to encode the grandma_qcif (176x144) in intra frame mode (for a actual quantizer used for quantization being 38), while Daala calls 3843 times only. So, PVQ was called 165 times more in AV1 than Daala. Table 2 shows the frequency of PVQ function calls in AV1 at each speed level (mode = good). The first column indicates speed level, the second column shows the number of calls to PVQ's search for each band (function pvq_search_rdo_double() in [6]), and the third column shows the number of calls to PVQ's encoding of whole transfrom block (function od_pvq_encode() in [7]). Smaller speed level gives slower encoding but better quality for the same rate by doing more RDO optimizations. The major difference from speed level 4 to 3 is enabling that a transform block size can be smaller than a prediction (i.e. partition) block size.

Speed Level	# of calls to PVQ search for a band	# of calls to PVQ encode
5	365,913	26,786
4	472,222	56,980
3	3,680,366	564,724
2	3,680,366	564,724
1	3,990,327	580,566
0	4,109,113	632,520

Table 2: Number of Calls to PVQ in AV1

4. Future Work

Possible future works include:

- Enable activity masking, which also needs HVS-tuned quantiztion matrix (bandwise QP scalers).
- Adjust, probably perceptualy driven, the balance between luma and chroma qualities.
- Optimize the speed of the PVQ codes, adding SIMD.

- RDO with more model-driven decision making, instead of full transform + quantization.

5. Development Repository

The ongoing work of integrating PVQ into AV1 video codec is located at the git repository [8].

6. Acknowledgements

Thanks to Tim Terriberry for his proof reading and valuable comments. Also thanks to Guillaume Matres for his contibutions to intergrating PVQ into AV1 during his intership at Mozilla and Thomas Daede for providing and maintaining testing infrastructure by way of www.arewecompressedyet.com (AWCY) web site [9].

7. IANA Considerations

This memo includes no request to IANA.

8. Informative References

[I-D.valin-netvc-pvq]
Valin, J., "Pyramid Vector Quantization for Video Coding", draft-valin-netvc-pvq-00 (work in progress), June 2015.

[PVQ-demo]
Valin, JM., "Daala: Perceptual Vector Quantization (PVQ)", November 2014, <https://people.xiph.org/~jm/daala/pvq_demo/>.

[Perceptual-VQ]
Valin, JM. and TB. Terriberry, "Perceptual Vector Quantization for Video Coding", Proceedings of SPIE Visual Information Processing and Communication , February 2015, <https://arxiv.org/pdf/1602.05209v1.pdf>.

[Terriberry_16]
Terriberry, TB., "Perceptually-Driven Video Coding with the Daala Video Codec", Proceedings SPIE Volume 9971, Applications of Digital Image Processing XXXIX , September 2016, <https://arxiv.org/pdf/1610.02488.pdf>.

Author's Address

Yushin Cho
Mozilla Corporation
331 E. Evelyn Avenue
Mountain View, CA 94041
USA

Phone: +1 650 903 0800
Email: ycho@mozilla.com