SVQ: A PROPOSAL FOR STILL IMAGE CODING IN MPEG 4 - SNHC

L. Torres D. Gimeno D. García*

Universitat Politècnica de Catalunya *DGG Software
{luis,albroto}@gps.tsc.upc.es dgg@grn

ABSTRACT

A technique for efficient coding of homogeneous textures is presented here. The technique is based on the use of Stochastic Vector Quantization and provides very high compression with graceful degradation. To encode the image, a linear prediction filter is computed. Then, the prediction error is encoded using a Stochastic Vector Quantization approach. To decode the image, the prediction error is decoded first and then filtered as a whole using the prediction filter, thus avoiding the block effect found in conventional VQ. The approach has been proposed as a still image coding technique in MPEG 4 SNHC. Comparisons with the Video VM of MPEG 4 are also presented.

1. INTRODUCTION

Vector quantization (VQ) has been extensively used as an effective image coding technique. One of the most important steps in the whole process is the design of the codebook. The codebook is generally designed using the LBG algorithm which uses a large training set of empirical data that is statistically representative of the images to be encoded. Stochastic vector quantization (SVQ) provides an alternative way for the generation of the codebook [1]. The main difference of the SVQ with respect to the conventional vector quantizer is the design of the codebook. In the SVQ the codewords are generated by stochastic techniques instead of being generated by a training set representative of the expected input image. This means that the codebook is generated using a random number generator. In the original SVQ approach white-gaussian noise images of the same size as the subimages to be encoded, are passed through some shaping filter $H(z_1,z_2)$ whose output follows the selected model. The scheme has been modified to cope with homogeneous textures and high compression.

This work has been partially supported by the VIDAS ACTS project of the European Union and by TIC 95-1022-C05-05 of the Spanish Government

The objective of this paper is to provide details of the SVQ approach for homogeneous textures in the context of the proposal made to MPEG 4 SNHC.

2. STOCHASTIC VQ

The Stochastic Vector Quantization approach is based on concepts related to Linear Prediction. To have a self-contained paper some explanations are given here.

2D Linear Prediction

Given a grayscale digital image $u[x,y]$, a *linear predictor* can be defined by

$$\hat{u}[x,y] = \sum_k a_k u[x-\alpha_k, y-\beta_k]$$

The pixels are assumed to be ordered by rows. In order for the 2D filter to be causal, we have to consider only the pixels previous to the one being predicted. Figure 1 shows the terms used by a 2D causal linear predictor of order K; all the terms labeled with a number less than or equal to K are used. The number of predictor coefficients is $P = 2K(K+1)$.

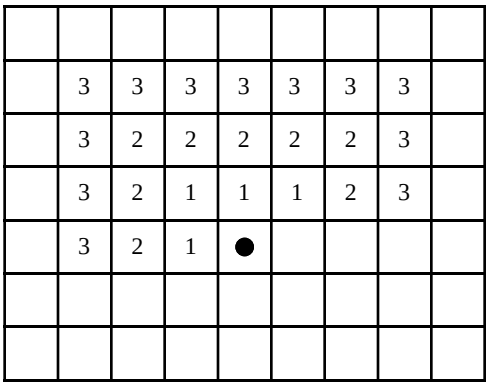


Fig. 1 terms used by a 2D causal linear predictor

The predictor coefficients a_k are chosen such as to minimize the *mean square estimation error*

(MSEE), considering $\mathbf{u}[x,y]$ a random variable being estimated in terms of the RVs $\mathbf{u}[x-\alpha_k, y-\beta_k]$.

The *prediction error* $e[x,y]$ is the difference between $u[x,y]$ and its prediction:

$$e[x,y] = u[x,y] - \hat{u}[x,y]$$

The *prediction filter (LPF)* allows the original image to be reconstructed from its prediction error:

$$u[x,y] = \hat{u}[x,y] + e[x,y] = \sum_k a_k u[x-\alpha_k, y-\beta_k] + e[x,y]$$

SYSTEM DESCRIPTION

In what follows, the image size will be assumed to be $N \times N$, although the system is well suited for non-square images as well. The *order of the filter* is K, the *codebook length* (number of codewords) is L and the *codeword size* is $M \times M$. The number of filter coefficients is $P = 2K(K+1)$.

Encoder Operation

The current implementation of the encoder is for grayscale images only. To encode a color image, each of its components (*YUV*) is encoded separately, using the following steps:

1. Mean extraction. Compute the mean of the image and remove it. *Complexity:* $2N^2$ additions.

2. Filter computation. Compute the predictor coefficients that minimize the mean square estimation error for the image. Only filters of order 1 and 2 are used (4 and 12 coefficients, respectively). For some images and a given filter order, some of the filter coefficients may be greater than 1 in magnitude and the filter becomes unstable. The problem can be solved changing the order of the filter.
Complexity: $\left[\frac{P(P+1)}{2}+1\right]N^2$ multiplication's-additions. Also, a system of equations of order P must be solved. The complexity of this step is due to multiple computations of the self-correlation of the image to get the coefficients for the system of equations. It can be reduced by ignoring some of the samples (256×256 samples are usually enough).

3. Prediction error variance computation. Compute the variance σ^2 of the prediction error. Complexity: None. It can be done as part of the prediction filter computation and its complexity has already been included there.

$$\sigma_e^2 = E\{\mathbf{e}^2\} = R[0,0] - \sum_k a_k R[\alpha_k, \beta_k]$$

4. Codebook generation. Generate a stochastic codebook following the prediction error model (a gaussian PDF). All that is needed for that is to fill the codewords using a gaussian random number generator. For efficiency reasons, a second codebook can also be generated by feeding the prediction filter with the codewords of the prediction error codebook. Complexity: LM^2 gaussian random numbers for the prediction error codebook, PLM^2 multiplication's-additions for the filtered codebook.

5. Block coding. Encode the prediction error using stochastic vector quantization methods (see below). *Complexity:* $(3P+L)N^2$ multiplication's, $(3P+3L)N^2$ additions.

3. SVQ OF THE PREDICTION ERROR

For every block in the prediction error image, the encoder chooses a codeword from the codebook following its stochastic model. The prediction error itself needs not be computed. To encode a block, the SVQ encoder computes the output that each codeword would produce at the decoder after filtering and it is that output distortion which is minimized. Since the codebook is stochastically generated, the decoder can use exactly the same codewords as the encoder without the need of transmitting them, just using the same seed for the random number generator. Only the codeword indices need to be encoded (together with the mean of the image, the variance of the prediction error and the filter coefficients).

Prediction error encoding hints

It is very important to keep in mind two points. First, although we are using SVQ to encode the prediction error, we are not interested in the prediction error itself, but the original image instead. Thus, the codewords should not be chosen to minimize the distortion in reproducing the prediction error. Rather, they should be chosen to minimize the distortion in reproducing the original image after filtering the prediction error with the prediction filter in the decoder.

Second, to improve image reproduction, the whole prediction error image is decoded first and then filtered as a whole, rather than using a block-by-block procedure. That means that to encode a particular block, surrounding blocks must be taken into account to evaluate the resulting image after filtering.

Thus, the encoder needs to know what will be the output at the decoder if a particular codeword is chosen to encode a given block. The problem is that, due to the autoregressive nature of the

prediction filter, given two contiguous blocks to be encoded the best choice for anyone of them depends on what was the choice for the other. That is, assuming that blocks are encoded in sequence (by rows), to choose the best codeword for a particular block we need to know the prediction filter output at pixels located in blocks that have not been encoded yet. The problem arises from the fact that the order in which pixels shall be filtered in the decoder (by rows) is not the order in which they are encoded (by blocks) –unless the blocks are 1-pixel rows.

While using 1-pixel rows for the blocks could be a solution, a more flexible approach has been devised to allow using blocks of arbitrary dimensions, thus increasing the flexibility of the system. The following considerations are of great help:

1. Actually, we don't need to know exactly the output values produced by encoding a given block with a particular codeword. All we need to know is if that codeword gives better results than any other in the codebook.

2. The influence of a prediction error block in the output image is more relevant for the pixels located in that block. Hence, the pixels located in the block being encoded is where different codewords will produce greater differences in output distortion, so we do not need to evaluate output distortion outside that block.

3. In principle, since the PF is recursive, to compute the output of the prediction filter at a particular pixel, all previous output pixels should have already been computed. If the reproduction quality is good enough, however, the unknown (because the block they are in has not been encoded yet) but needed pixels can be estimated from the original image. Such pixels are those located to the right of the block being encoded.

4. All the output pixels surrounding the block being encoded can thus be estimated, either because their blocks have already been encoded or because we use the original image values. This and the fact that we only evaluate the output distortion at the block being encoded, can dramatically reduce the filtering needed to choose the best codeword for a given block. Rather than filtering all the prediction error already encoded, we consider the output in that block as the result of the superposition of two signals at the input of the prediction filter: (1) the codeword itself (with zero surrounding pixels) and (2) all the surrounding pixels (with zero values at the location of the block). Thus, we need to filter every codeword Wi only once before encoding any block. The result of such pre-filtering is stored in what can be viewed as a second codebook, {Vi}, reproducing locally the output image statistics.

5. For a given block, the encoder chooses the codeword Wi such that Vi plus the effect of filtering an empty input block with the non-zero estimated output surrounding pixels minimizes output MSEE for pixels in that block.

6. After encoding a block, it is convenient to re-filter it, together with one or more previous blocks, to avoid error propagation in output estimations for pixels surrounding the next block to be encoded.

The goal of the procedure outlined above is to reproduce, as closely as possible, the decoding process at the encoder so that the codewords are chosen taking into account the way they will be used.

Overall encoder complexity

Given the parameters of the system, the overall encoder complexity is $O(N^2)$. For a given image size and large codebooks, it is $O(L)$. So the complexity of the encoder increases linearly with the image size and the codebook length.

Decoder Operation

The current implementation of the decoder is for grayscale images only. To decode a color image, each of its components (*YUV*) is decoded separately, using the following steps:

1. Codebook generation. Generate the same stochastic codebook for the prediction error as the encoder. Complexity: LM^2 gaussian random numbers.

2. Block decoding. Decode the prediction error. Complexity: None (N^2 memory moves).

3. Image filtering. Filter the decoded prediction error with the prediction filter.

3. Complexity: PN^2 multiplication's-additions.

4. Mean restoration. Restore the mean to the image. Complexity: N^2 additions.

Overall decoder complexity

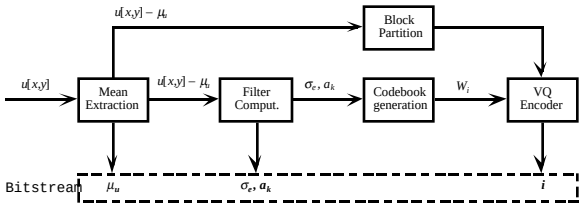
The overall decoder complexity is $O(N^2)$. For large codebooks, the encoder complexity can be written as $O(LN^2)$, which means that the decoder is much simpler than the encoder.

4. SYSTEM SYNTAX

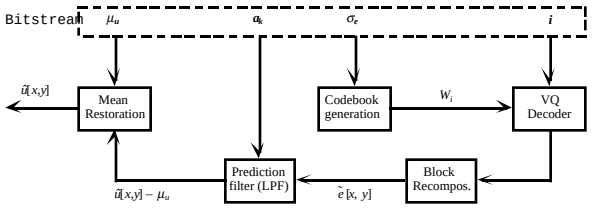
The system syntax for grayscale and color images follows. The current system syntax for color (*YUV*, 4:2:0) images is basically the concatenation

of the grayscale syntax applied to each component. This syntax may change in the future due to system tuning for color images. The following diagrams show the actual encoder and decoder.

SVQ Encoder



SVQ Decoder



5. RESULTS AND CONCLUSIONS

The SVQ scheme has been applied to a variety of images with similar results. We have selected here the Fabric image from the MIT data base used in Core Experiments X1 and Z1 of MPEG 4 SNHC [2].. The Video VM 5.0.1 of MPEG 4 was not able to go further than a compression ratio of 14 for this image. As a conclusion it can be said that the SVQ approach provides coding results for homogeneous textures in the range of 10 - 200. For the high compression range, the SVQ always outperforms the Video VM developed in MPEG 4 video.

6. REFERENCES

[1] D. Gimeno, L. Torres, J.R. Casas, "A New Approach to Texture Coding Using Stochastic Vector Quantization", ICIP, Vol. 1, pp. 119 - 123, Austin, Texas, USA, November 1994.

[2] Results of Core Experiment X1 and Z1". M1706 - 1900 ISO/IEC JTC1/SC29/WG11.

Original image

SVQ Compression 10

VM 5.0.1 Compression 14

SVQ Compression 50

SVQ Compression 101

SVQ Compression 10