# ILLINOIS INSTITUTE OF TECHNOLOGY
# ECE 508 SIGNAL AND DATA COMPRESSION (Fall 1999)
# PROJECT #2
# Performance Comparison Between Vector and JPEG Quantization of Images

**Due Date:** December 6, 1999

## Tasks

The major tasks of this computer simulation project are:

1. Design vector quantization codebooks for coding image pixels.
2. Measure the rate-distortion performance of the codebooks on test images.
3. Compare the rate-distortion performance and (optional) subjective visual quality, between the vector quantized and JPEG quantized test images.

## Ground Rules

All the ground rules laid down for Project 1 also apply to this Project.

## Image File Format

The most compact way (other than storing it as a compressed bit stream, using for instance JPEG) of storing an 8-bit grey scale (no color components) image in a file is to use one byte to represent the amplitude of each pixel. The byte is interpreted as an unsigned integer with numeric range from 0 to 255, with 0 for the darkest pixel and 255 for the brightest pixel. The pixels of the image are ordered in the file in a raster sequence: from the top row to the bottom row, each row from left to right. When the pixels are stored in a "raw" (headerless) file format, the file size in number of bytes would equal to the number of pixels in the image.

You can display a raw image file using the Matlab program display_im.m, supplied to you on the course Web site. If you had not worked with images in Matlab before, you might want to study the display_im.m program just to learn the basic setup in Matlab. You are of course welcome to use other software tools, e.g. xv and ImageMagick. Viewing images is an optional exercise in this project. However, viewing the quantized images does offer you an additional means of checking the correctness of your work.

There are numerous standard file formats for storing image and video data. Some formats are "lossless," in the sense that if you convert a raw file into that format, you can recover from the formatted file the original raw file, with every bit unchanged. Other formats are "lossy," so you have to be careful with choosing the format. The images you use for this project are supplied to you as raw files and also in TIFF format. Matlab's image toolbox can handle the TIFF format. There is no "loss" in storing a raw image in TIFF, though a TIFF file takes more disk space. If you are unfamiliar with standard image file formats, perhaps the most hassle-minimizing strategy is just work with raw files.

## The Generalized Lloyd Algorithm

You can choose between using the GLA program you wrote for Project 1, or the sample GLA program posted on the course Web site. The sample program is offered to you as a contingency, in case your own GLA program does not function properly. Do not assume that the sample GLA program necessarily works better than yours. You can try running both programs and compare them on the basis of computation efficiency and correctness of results.

If the GLA program you use does not already measure the empirical probabilities of the code vectors in the codebook, you will need to modify the program to include this feature. In this project, we assume that the code vectors in the VQ codebook are coded using a variable-length code (VLC). In order to measure the bit rate, we need to know the lengths of the VLC codewords. Since we do not need to generate the bit stream that constitutes the output of the VQ encoder, we do not need to know values of the VLC codewords.

In this project, we will not design any VLC. We would simply use $l_i = \mathrm{int}(-\log_2(p_i))$ as an estimate of the length of the VLC codeword for the $i$-th code vector, where $p_i$ is its empirical probability measured during the codebook training phase, and $\mathrm{int}(\cdot)$ is the smallest integer equal to or greater than the argument. In the derivation of the "loose" upper bound on the average length of a uniquely decodable code, we saw that the Kraft inequality guarantees that a uniquely decodable code with codeword lengths $l_i, i = 1, \ldots, N$ exist. You may compare the resultant average length $L = \sum_{i=1}^{N} p_i l_i$ with the entropy lower bound $H(a) = -\sum_{i=1}^{N} p_i \log_2 p_i$. An actual Huffman code you design will give an average length greater than or equal to $H(a)$ and less than $L$; hence, $L$ is a pessimistic estimate.

Note that the code vector probabilities should be obtained for the final VQ codebook produced by the GLA, not for any of the intermediate codebooks that are evolving while GLA is still converging.

If you plan to store the quantized image in 8-bit raw file format, you might find it convenient to simply round the values in the final VQ codebook to an integer between 0 and 255 inclusive. (This does not imply that GLA should use integer variables for computation.) Having the code vector values already in the permissible integer range enables you to dispense with the need to remap the quantized pixel amplitudes before writing them out to an 8-bit raw image file.

In the GLA, the centroids and other intermediate variables related to pixel amplitudes should be kept as floating point numbers; variables that are used to accumulate a large number of values should be in double precision. The integer conversion applies only to the final VQ codebook.

## Performance Measures

We assume that the VQ code vectors are variable length coded. JPEG uses zero-run-length and variable-length coding of DCT coefficients. Thus, it is reasonable to measure the coding rate for either scheme in *average number of bits per pixel.* This measure can be calculated as the total number of bits used by the encoder to describe the image, divided by the total number of pixels in the image.

In image and video coding, it is common to measure the distortion in *peak signal-to-noise* *ratio* (PSNR) in dB. For 8-bit gray-scale images, PSNR (dB) is defined as $10\log_{10}(255^2/D)$ where $D = E\{(I - \hat{I})^2\}$ is the MSE incurred by approximating the pixel random variable $I$ with the quantized pixel $\hat{I}$. If the image is vector quantized, then $D$ can alternately be written as $D = E\{\|\mathbf{X} - \hat{\mathbf{X}}\|^2\}/k$, where $\hat{\mathbf{X}} = \mathbf{Q}(\mathbf{X})$.

We define *operational rate-distortion performance* (ORDP) as PSNR in dB as a function of the coding rate in average number of bits per pixel, produced by applying a particular coding scheme to a particular image or a set of images. ORDP will be used as our *objective* characterization of coding performance. A coding scheme is said to offer better objective performance if the ORDP graph of the scheme lies above that of another coding scheme.

Our goal is to compare the ORDP of VQ and JPEG over a range of coding rates between roughly 0.25 bit/pixel and 1 bit/pixel. A ORDP graph for a test image can be obtained by plotting a few ORDP data points. JPEG quantized test images, at average coding rates of roughly 1/4, 1/2, 3/4, and 1 bit/pixel, are available on the course Web site. The exact values of the coding rates can be found in a README file. To generate an ORDP plot for a test image, you need to calculate the PSNR at each coding rate.

OPTIONAL: For audio-visual signals, the subjective or *perceptual* quality of the quantized signal is even more important. Thus, you are encouraged to view the quantized images and compare them on the basis of perceptual quality and discernable quantization artifacts. Usually, we compare the quantized images produced by different coding schemes at roughly the same coding rate. Typical artifacts to look for are distortion of edges and roughening of smooth surfaces.

A high quality report should include images that exhibit discernable differences. So you might want to generate images from the image viewing software, to include in your report.

Note that viewing images that are rendered on a screen, with the proper setting of screen contrast, image size, and viewing distance, can unveil more image details and coding artifacts, than viewing images that are printed on paper using an ordinary laser printer.

## Organizing Image Pixels for Quantization

For the purpose of quantization, the pixels in an image are organized into blocks. The pixels in each block are scanned into a vector. The most common approach is to use contiguous and non-overlapping square blocks of a fixed size $M \times M$. The images used in this project have a size $W = 176$ pixels per row and $H = 144$ rows per image. You can scan the pixels in each block into a vector either by row or by column. However, you need to apply your scanning scheme consistenty, as you will need to organize the pixels into vectors for the purpose of quantization, and organize the quantized pixels to create a quantized image. The rule is: you put a quantized pixel into the same location of the quantized image as the location of the original unquantized pixel in the unquantized image.

Generally, VQ gives lower distortion as the vector dimension $k = M \times M$ grows. However, aside from complexity consideration, we may be limited by the size of the training set. One method to obtain a larger training set from a given set of images is to create training vectors from overlapping blocks. That is, instead of shifting by $M$ pixels in either dimension to read out the next training vector from an image, we shift by an amount $L < M$ pixels. This way, we obtain $W \times H/L^2$ training vectors instead of $W \times H/M^2$ training vectors from each image. This method of enlarging the training set has its drawback. Ideally, for memoryless

VQ coding, we prefer the training vectors to be independent realizations drawn from the same probability distribution. In practice, the vectors are correlated because adjacent pixel blocks are correlated. Making the blocks overlap increases the correlation. Generally, the more correlated is the training data, the greater the chance of occurence of empty cells during training.

In this project, maintain a training ratio no smaller than 15. Use the two images "akiyo" and "foreman" to test your VQ codebooks. Use all the other images to populate the training set. Note that the blocks obtained from a test image for quantization should not overlap.

## Performance of VQ

In most image and video coding standards, where transform coding rather than VQ is used, the transform block size is typically $8 \times 8$. Generally, for best VQ performance, we would like to make the block size $M \times M$ as large as possible. Using $M = 8$ for VQ is not very practical. Even $M = 4$ is pushing the complexity envelope, if the target bit rate is high enough. Thus, in this project, use a $3 \times 3$ block size. Since $H/M = 176/3 = 58\frac{2}{3}$, we are left with a partial block at the end of each row of blocks in an image. The remedy is to discard the right most two columns of pixels. Do not code them or include them in your ORDP calculations. If you are interested, you may also design VQ codebook for $4 \times 4$ blocks. You should definitely present results for $M = 3$. Results you present for other values of $M$ would be treated as "extra mile" effort.

We have chosen to measure the bit rate using $R$, the average number of bits per pixel. When a VLC is used to identify the code vectors, there is no obvious relationship between $N$, the number of code vectors in a VQ codebook, and $R$, other than that $R$ is likely to be less than $\log_2(N)/k$. For a quantized test image, $R$ is calculated as the total number of bits needed by the encoder to identify the code vectors, divided by the total number of pixels actually quantized. Thus, $R$ depends on the statistics of the test image, which may differ from the statistics of the images in the training set. This should not cause any difficulty towards generating an ORDP graph for each test image. Once you have several ORDP points, obtained using VQ codebooks of different sizes to quantize the test image, you can plot the graph. However, if you elect to compare the visual quality of VQ quantized and JPEG quantized images, you would need to generate VQ quantized images with $R$ values that are pretty close to those of the JPEG quantized images.

In your report, you should try to explain the performance difference observed between VQ and JPEG, based on ORDP and (optionally) subjective characterization. Note any change in their relative performance as a function of the bit rate.