Lab 5
Lab report due by 2:00 PM on Monday, April 11, 2016

This is a MATLAB only lab, and therefore each student needs to turn in her/his own lab report and own programs.

---

1 Introduction

In this lab you will implement the JPEG algorithm for image compression.

1.1 Why compressing images

Images are increasingly being acquired in a digital form. Analog images, such a camera pictures, or a chest X-Rays are more and more often digitized. Images exist not only in their two-dimensional (2-D) form, but also as three-dimensional (3-D) volumes (e.g. medical images), or spatio-temporal sequences (movies). The availability of images in a digital form offers a very large number of advantages:

- images can be stored, and transmitted very efficiently
- images can be processed. Examples of post processing include: enhancement, denoising, zooming, etc.
- Computer assisted diagnosis can be performed on digital images. These higher level processing can involve feature extraction, decision making, classification, etc.
- Images can be stored in large data bases, where they can be indexed, and very efficiently searched for.

However the very large size of images, and movies are basic hurdles to any attempt at storing, transmitting, or searching for images. Table 1 show samples of average sizes of image files or movie files. For each application, we calculated the compression ratio required for a realistic usage under normal conditions. It is clear from this table that the ability to decrease the amount of bits necessary to represent an image, or a sequence of images, will have a dramatic impact in a wide variety of domains, including but not limited to: multimedia technology, medical imaging, geophysical exploration, satellite imagery, etc.

Applications	horiz x vert	frame/s	size	ratio
Digital camera	768 x 512		375kB	10
Fax	1728 x 1100		240 kB	20
CD	352 x 240	30	7.6 Mb/s	50
HDTV	1920 x 1080	30	186 Mb/s	80
VideoPhone	176 x 144	15	1.1 Mb/s	300

Table 1: Typical size of images or sequence of images, and the compression ratio required for a realistic usage under normal conditions.

2 Structure of a compression system

A compression system can be decomposed into three components: (1) an encoder, (2) a decoder, and (3) and a channel. The channel is that part of the communication system that is out of the designer's control. Examples of channels are: wireless (radio), satellite, ethernet, CD, modem, optic fiber, ISDN, computer memory, etc.

In this chapter we will assume that the channel provides a reliable communication of the compressed data, and we will ignore the channel coding errors. It is possible to combat channel errors, by using error protection codes, but we will not explore these here.

2.1 Encoder

The encoder can be decomposed in a series of elementary stages:

1. Pre-processing. A number of operations can be performed on the image prior the effective compression. The goal of pre-processing is to prepare the image in order to minimize the coding complexity, and increase the compression efficiency. Examples of pre-possessing operations are:
   - filtering,
   - padding with zeros,
   - symmetric extension on the boundaries,
   - tiling (cutting in smaller blocks) to enable the coding of images of arbitrarily large size,
2. Transform. The goal of the transform is to reduce the correlation among image pixels, and describe with a smaller set of significant coefficients, most of the image content.
3. Quantization. The output of the transform are real-valued coefficients. In order to decrease the amount of information needed to describe these coefficients, one represents the coefficients with a much smaller set of values. Quantization is a non reversible process that permits to balance the reconstructed image quality with the amount of bits available to code the image.
4. Entropy coding. After quantization, the image is approximated with a small set of codewords. Entropy coding permits to describe in the most efficient way the sequences of symbols from this alphabet with strings of "0", and "1".

2.2 Decoder

The decoder reconstructs the image by applying in a reverse order the inverse of each of the processes 4), 3), and 2) described in the encoder. While entropy coding is a reversible process, quantization is not invertible. A final post-processing is often performed on the reconstructed image in order to conceal coding artifacts, and improve the visual appearance.

2.3 Important parameters of a compression system

Some of the fundamental parameters of a coding system are:

- Compression efficiency: it measures the compression ratio (often quoted in bits per pixel: bbp),
- Fidelity: this is the distortion due to the coding. While everybody agrees that the PSNR (Peak Signal to Noise Ratio) is a global parameter, often unrelated to the visual quality, there is currently no definition of a quantitative measure of visual distortion. For this reason visual inspection of reconstructed images remains one criterion for the evaluation of a compression method.
- Complexity: this is a key parameter for a number of applications, where real-time compression is required.
- Memory. Some applications will require an encoder that can work with a limited memory.
- Coding delay: real-time applications such as video-phone cannot tolerate a coding delay that is longer than a few frames.
- Robustness. This is a key feature in areas such as wireless communication where channel coding errors can have a catastrophic effect on the reconstructed image.

3 Transform coding

Transform coding consists in applying a linear transformation from the image space to a transformed space. The image in the transformed space is represented by a new vector (of same dimension as the original image) of coefficients. The principle of transform coding is that the transformed coefficients should be less correlated than the original samples. Another way to express the same idea is that a small number of transformed coefficients should carry most of the energy of the image, and the rest of the coefficients can be quantized to zero. Most of the transform operate in the frequency domain. This can be justified by a theoretical argument : if the images are realizations of a wide sense stationary process, then the Fourier transform is the Karhunen-Loève transform that diagonalizes the correlation matrix. As we will see in this section, the KLT is the transform that permits to achieve the smallest distortion of the quantized coefficients for a given rate.

From a practical point of view, because the human visual system is sensitive to artifacts in the frequency domain, it is easier to work directly in this domain in order to control the distortion, and even incorporate cues about the human visual system sensitivity.

In the remainder of this section we demonstrate that an advantage can be gained by applying a linear transform before quantization. Precisely, we will prove this result when the linear transform is the Karhunen-Loève transform (KLT), which is the transform that *decorrelates* the input data.

While this transform depends on the knowledge of the correlation matrix of the input, and is therefore rarely used in practice, this result provides a theoretical justification for the performance of transform coding. For certain inputs the Discrete Cosine Transform provides an approximation to the data-dependent KLT. We will present some application of the DCT to transform coding. Finally, we will conclude with some very recent results on wavelet-based image compression algorithm. The wavelet transform is an orthonormal transform that provides a very efficient decorrelation of images, and can provide under certain circumstances an approximation to the KLT.

3.1 Discrete Cosine based coders

There exists four different versions of the discrete cosine transform [3]. The importance of the DCT comes from several different reasons. First, the DCT provides a good approximation to the KLT if the image has a Markov behavior. In practice, the DCT provides excellent energy compaction for highly correlated data. From a practical point of view, the DCT can be computed using an FFT, but unlike the FFT, the DCT coefficients are real. In 1988, a DCT based algorithm won in blind subjective assessments, for the definition of a new image compression standard for continuous-tone images (Joint Photographic Experts Group - JPEG).

4 Reading, writing, and displaying images

The following MATLAB commands can be used to read, write and display an image.

```
>> f = imread ('clown.pgm');
>> figure,imagesc(f, [0 255]),colormap gray,axis square, axis off;
>> g = f;
>> imwrite (g,'clown2.pgm', 'pgm');
>>
```

4.1 Test images

In this lab, we will be using standard test images available on D2L.

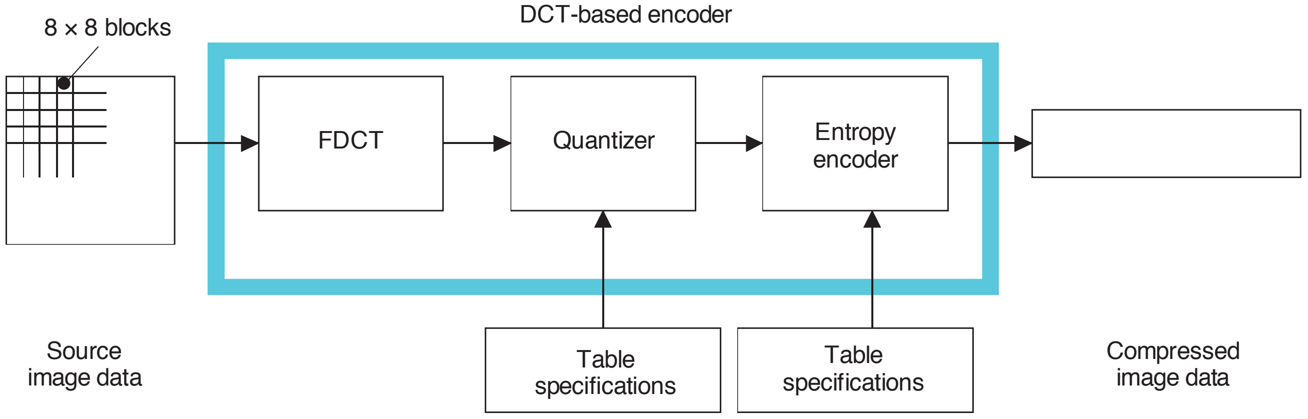

Figure 1: The different stages in the JPEG compression algorithm

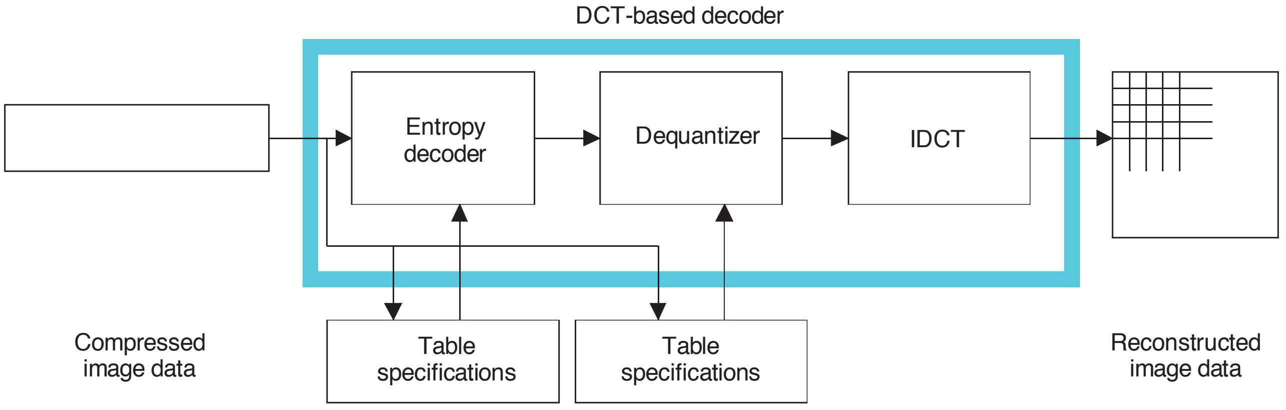

Figure 2: The different stages in the JPEG decoding algorithm

5 The JPEG compression algorithm

Figure 1 shows the three main steps of the JPEG compression algorithm. We will implement the three stages: forward and inverse DCT, forward and inverse quantizer, and entropy coder and decoder. Figure 2 shows the three main steps of the JPEG decoding algorithm. Each step implements the inverse of the corresponding operation performed by the encoder.

5.1 The forward and inverse discrete cosine transform.

The original image is divided into non overlapping blocks of size 8×8. The blocks are scanned horizontally, and then vertically. Each block is transformed using a two-dimensional discrete cosine transform,

$$F_{u,v} = \frac{1}{4} C_u C_v \sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y) \cos\frac{(2x+1)u\pi}{16} \cos\frac{(2y+1)v\pi}{16}, \quad u,v = 0,\ldots 7 \tag{1}$$

where

$$C_u = \begin{cases} 1 & \text{if } u \neq 0, \\ \frac{1}{\sqrt{2}} & \text{if } u = 0. \end{cases} \tag{2}$$

The coefficients $F_{u,v}$ are coarsely approximated over a small set of possible values (quantization), and replaced by $\widetilde{F}_{u,v}$. The *quantization* process cannot be inverted and contributes to the loss in image quality.

Given the coefficients $\widetilde{F}_{u,v}$, one can use the inverse Discrete cosine transform to reconstruct an estimate of the block using

$$\widetilde{f}(x,y) = \frac{1}{4}\sum_{u=0}^{7}\sum_{v=0}^{7} C_u\, C_v\, \widetilde{F}_{u,v} \cos\frac{(2x+1)u\pi}{16} \cos\frac{(2y+1)v\pi}{16}, \quad x,y = 0,\ldots 7 \tag{3}$$

where

$$C_u = \begin{cases} 1 & \text{if } u \neq 0, \\ \frac{1}{\sqrt{2}} & \text{if } u = 0. \end{cases} \tag{4}$$

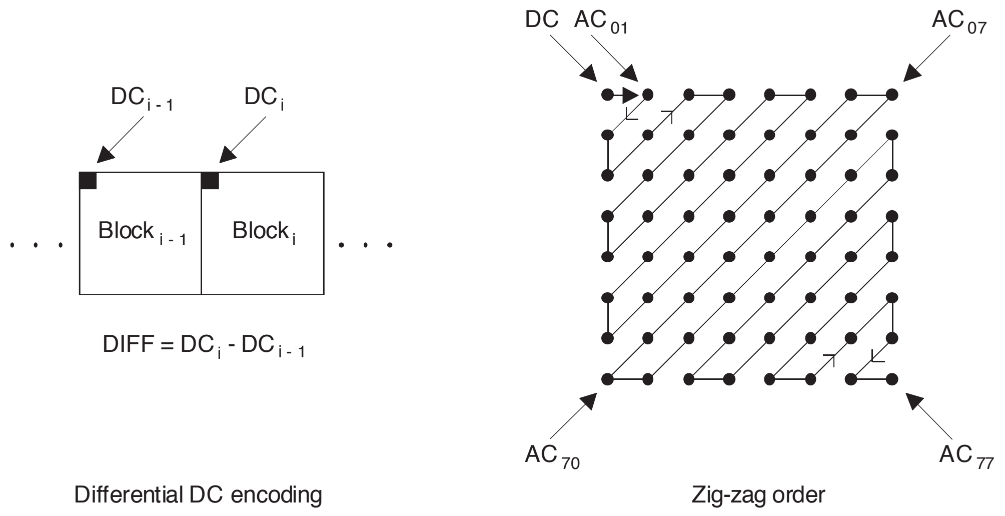

Figure 3: Zig-zag order for mapping the two-dimensional frequencies to a one dimensional array.

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Figure 4: The quantization table Q

Assignment

1. Write the MATLAB function `dctmgr` that implements the following functionality:

   (a) it takes an luminance (gray-level) image as an input, divides it into non overlapping 8×8 blocks, and DCT transform each block according to (1).

   (b) The DCT coefficients for the entire image are returned into a matrix `coeff` of size $64 \times N_{\text{blocks}}$, where N_{blocks} is the number of 8×8 blocks inside the image.

   (c) For a given block `b`, the coefficients of in the column `coeff(:,b)` are organized according to the zig-zag order shown in Fig. 3. That is,

   `coeff(2,b) = F(1,2), coeff(3,b) = F(2,1), coeff(4,b) = F(3,1),...`

   (d) The zero-frequency coefficients, $F(1,1)$, for each block is encoded using differential encoding,

   `coeff(1,1)` $= F_1(1,1)$,
   `coeff(1,2)` $= F_2(1,1) - F_1(1,1)$,
   `coeff(1,3)` $= F_3(1,1) - F_2(1,1)$,
   ...
   where $F_i(1,1)$ is the zero frequency DCT coefficient of block i.

2. Write the MATLAB function `idctmgr` that implements the following functionality:

   (a) It takes a matrix `coeff` of size $64 \times N_{\text{blocks}}$, where N_{blocks} is the number of 8×8 blocks inside the image and reconstruct a luminance image.

   (b) For each given block `b`, the coefficients in the column `coeff(:,b)` are used to reconstruct the block according to (3).

5.2 Quantization and inverse quantization

The effect of the DCT is too create many small coefficients that are close to zero, and a small number of large coefficients. We can think of the DCT as a rotation of the space $\mathbb{R}^{64}$ that aligns the coordinate system along the direction associated with the largest variance in the distribution of the coefficients. In order to benefit from this redistribution of the energy, we need to simplify the representation of the floating numbers that are used to describe the DCT coefficients. This task is achieved by the quantization step.

The goal of quantization is to represent a continuum of values with a finite (and preferably) small set of symbols. In theory, a scalar quantizer can be defined as a map from the set of real numbers to the set of integers, or any subset thereof. In practice, real numbers are represented in a computer as floating point numbers and have only a fixed number of digits, and thus a fixed precision. In this context a scalar quantizer is a map from a large set of discrete values, to a much smaller set of codewords. Quantization permits to control the performance of the compression by performing a reduction of the set of codewords that are used to represent the initial samples. In principle quantization is the only mechanism in a compression system where irrecoverable loss is introduced.

In the JPEG compression standard, the quantization of the DCT coefficients is performed as follows

$$F^q_{u,v} = \text{round}\left(\frac{F_{u,v}}{\texttt{loss-factor} \times Q_{u,v}}\right). \tag{5}$$

$Q_{u,v}$ is the entry (u, v) of the standard quantization table shown in Fig. 4, and the scaling factor `loss-factor` is a user supplied parameter that measures the quality of the compressed image. For instance, the choice of `loss-factor` $= 2$ should yield an image that is visually identical to the original image, while `loss-factor = 10` produces an image of lower quality.

The inverse quantization of the DCT coefficients is defined by the map

$$\widetilde{F}_{u,v} = F^q_{u,v} \times \texttt{loss-factor} \times Q_{u,v}. \tag{6}$$

We note that the compression becomes lossy during the quantization step performed in (5) that creates many zero coefficients as `loss-factor` becomes larger. The quantization step $Q_{u,v}$ varies as a function of the frequency. As a result, it becomes possible to spend fewer bits for the high frequency coefficients than for the low frequency coefficients. Several quantization tables have been proposed. In this lab we use the table Q defined in Fig. 4.

Assignment

3. Implement the quantizer and inverse quantizer, as defined by (5) and (6), and integrate it with the DCT and IDCT transforms.

4. For the six test images available on D2L, display reconstructed images for `loss-factor = 1`, `loss-factor = 10`, and `loss-factor = 20`. Display also the reconstruction error.

5. For the six test images, compute the Peak Signal to Noise Ratio (PSNR) between the original image, and the reconstructed image,

$$\text{PSNR} = 10 \log_{10} \left(\frac{255^2}{\frac{1}{N^2} \sum_{i,j=0}^{N-1} |f(i,j) - \widetilde{f}(i,j)|^2} \right), \tag{7}$$

for the values of `loss-factor` in $\{1, \dots, 50\}$. You will plot the values of PSNR as a function of `loss-factor`.

5.3 Variable length and runlength coding

In this section we are interested in the coding of the quantized DCT coefficients associated with frequencies that are not zero, that is for $(u, v) \neq (0, 0)$.

After zig-zag ordering and quantization, we expect that the sequence of quantized coefficients within each block contains mostly zeros, except maybe for the first few coefficients.

If the original image is encoded using 8 bits per pixel (the intensity is in the range $[0, 2^8 - 1]$), then we expect that each quantized DCT coefficient be in the range $[-2^{10}, 2^{10} - 1]$. However, we expect most quantized coefficients to be much smaller.

The first idea is then to encode each quantized coefficient with the strict minimum number of bits, and encode as well this number of bits.

When the quantized coefficient is a zero (and this should happen often for `loss-factor` ≤ 50), the coefficient is not coded. Instead, we count the number of zeros separating two non zero coefficients. For each non zero coefficient, we add a symbol that encodes the number of zeros between this coefficient and the previous non zero coefficient.

Finally, JPEG uses an End Of Bock (EOB) symbol to indicate that all coefficients after the current coefficient are zero.

In summary, the encoding of the quantized coefficient $F^q_{u,v}$, $(u, v) \neq (0, 0)$ is performed as follows,

$$\begin{bmatrix}\texttt{nZeros} & \texttt{nBits} & \texttt{value}\end{bmatrix} \tag{8}$$

where

1. `nZeros` is the number of zeros skipped since the last non zero coefficient. We will modify the JPEG standard, and use 6 bits to encode `nZeros`,

$$\texttt{nZeros} \in [0, 63] \tag{9}$$

2. `nBits` is the number of bits needed to encode the value of the coefficient using two's complement representation,
$$\texttt{value} \in [-2^{\texttt{nBits-1}}, 2^{\texttt{nBits -1}} - 1]. \tag{10}$$
Because the original image value $f(x, y)$ is encoded using 8 bits, we only need `nBits` $= 11$ to encode `value`.

   The $(0, 0)$ frequency (DC) coefficient is encoded differentially (see Fig. 3). Because each DC coefficient can be as large as $2^{11} - 2^3$, each difference is in the range $[-2^{11}, 2^{11} - 1]$, and thus we need as many as `nBits` =12 bits to encode `value`.

3. `value` is the actual value of the coefficient encoded using two's complement representation in the range
$$\texttt{value} \in [-2^{\texttt{nBits -1}}, 2^{\texttt{nBits -1}} - 1] \tag{11}$$
where `nBits` =12 for the DC coefficient, and `nBits` =11 otherwise.

4. If there will be no more non zero coefficients, then `nZeros` = 0, and `nBits` = `0`, and there is no `value`. This symbol is the EOB symbol.

Assignment

6. Implement the variable length coder and decoder that is used to encode the quantized coefficients. The input parameter should be an array $\tilde{q}$uant of 64 quantized coefficients. The output should be an array of signed 16 bit integers, `symb`, and you will store sequentially the values `nZeros`, `nBits` and `value` for each non zero coefficient. The last two entries should be `00` followed by `00`. The size of `symb` should be variable, but cannot exceed $192 = 3 \times 64$.

5.4 Entropy coding

The aim of entropy coding is to provide a representation of a set of symbols with a description whose average length is minimal. This can be achieved by using a variable length coding mechanism: the most frequent outcomes of the data is encoded with the shortest description, and longer descriptions are kept for less frequent realizations.

Two entropy coding algorithms are proposed in the JPEG standard: Hufmann coding, and arithmetic coding. Arithmetic coder is more efficient, because it can adapt itself to the statistics of the quantized coefficients. In this lab we will use existing MATLAB functions to generate the stream of bits (0s and 1s) that a Huffman , or an arithmetic coder would generate. A note is in order here, while MATLAB can generate the bistream, it actually stores the sequence of 0 and 1 into an array of 64-bit floats. In other words, the final stage of bit packing is not performed. This is not a real problem.

In the following, we describe how to use the existing MATLAB functions to encode, and decode a stream of integers using arithmetic coding and Huffman coding respectively.

5.4.1 Arithmetic coding

MATLAB provides two functions that encode and decode a stream of integers using an adaptive arithmetic coder. The following example describes the encoding and decodings stages:

```
%
%
>> centers=[0: max(symb(:))];  % compute the count for each symbol
>> pdf=hist(s,centers);         % we need to add one so that no symbol as a zero
>> pdf = pdf +1;                % count

>> bistream = arithenco (s,pdf); % actual encoding: bitstream is an
                                 % array of 0s and 1s.
>> length (bistream)             % this is the length of the code
```

The decoding needs three parameters: the code itself, the probability distribution of the symbols, as well as the length of the original sequence of integers

```
>> symbol = arithdeco (bitstream,pdf,l); %l is the length of the array symb
```

MATLAB arithmetic encoder does not like negative integers, and therefore you will have to interpret `value` as an unsigned integer encoded using `nBit` bits. This can be done by adding $2^{\texttt{nBit} -1}$ to `value`. In MATLAB you will do

```
>> value = value + 1024;
```

if `value` is encoding a non DC coefficient, or

```
>> value = value + 2048;
```

if `value` is encoding a differentially encoded DC coefficient. You will need to shift back these values as you reconstruct by adding the corresponding offset (1024, or 2048).

Assignment

7. Implement the arithmetic coder to encode the sequence of variable length integers that is stored in `symb`. The input to your function should be the array `symb`, and the output should be a array `bitcode`. Each entry of `bitcode` is a 0 or a 1, and the actual size of the code is simply given by `length(bitcode)`.

8. Implement the arithmetic decoder to decode the sequence of 0s and 1s and reconstruct the sequence of variable length integers. The input to your function should be the `bitcode` and the length of `symb`. The output should be an array `symb`.

9. You will now evaluate the quality of the coder. For the six test images, you will compute the PSNR between the reconstructed image and the original image, for every value of the parameter `loss-factor =1,..100`. You will also compute the true number of bits needed to compress the image. This will lead a compression ratio, given by

$$\text{compression ratio} = \frac{8 \times \text{number of rows} \times \text{number of columns}}{\text{total number of bits}} \tag{12}$$

   You will plot a curve that gives the PSNR as a function of the compression ratio.

A Appendix: technical supplements

In the following sections, we provide some brief theoretical descriptions of the three main components of the JPEG algorithm.

A.1 Discrete Cosine Transforms

One can construct four different cosine and sine transforms, that are all based on a Fourier transform. As opposed to the discrete Fourier transform that assumes a periodic signal, the cosine/sine transform does not require a periodic signal. The principle of the different constructions is to extend the input signal in a periodic manner, using odd or even extensions. Let us examine the construction of the DCT II for instance. We consider a series of samples $x_n, n = 0, \ldots N-1$. We extend the samples for $n = -N, \ldots -1$ by mirror symmetry across the half integer $-1/2$:

$$x_n = x_{-n-1} \quad , \quad n = -N, \ldots, -1 \tag{13}$$

Fig 5 demonstrates the extension. We obtain then a periodic sequence of size $2N$, and we can calculate its discrete Fourier transform: $\hat{x_k} \quad , \quad k = -N, \ldots N-1$. Note that because the extended sequence is symmetric around -1/2, we know that the Fourier coefficients will be real (up to a phase shift, that would not exist if we were to calculate the symmetry with respect to 0), and even. We need only to compute the coefficients for $k = 0, \ldots, N-1$.

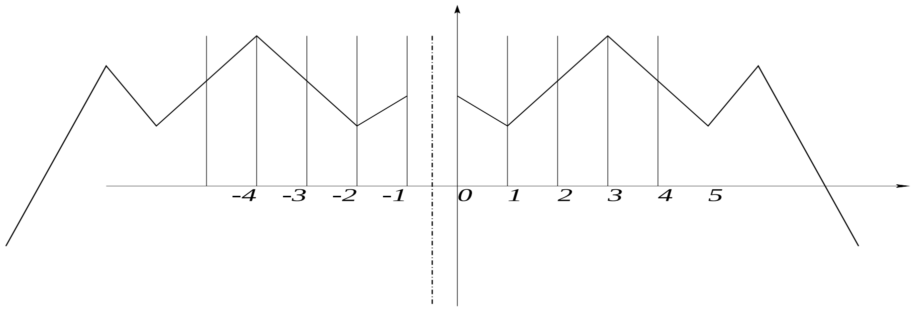

Figure 5: The original signal is extended with an even parity

$$\hat{x}_k = \sum_{n=-N}^{N} x_n e^{-\frac{2\pi\, i\, kn}{2N}} \tag{14}$$

We cut the sum into two parts:

$$\hat{x}_k = \sum_{n=-N}^{-1} x_n e^{-\frac{2\pi\, i\, kn}{2N}} + \sum_{n=0}^{N} x_n e^{-\frac{2\pi\, i\, kn}{2N}} \tag{15}$$

We can rewrite the first sum, using the mirror symmetry:

$$\sum_{n=-N}^{-1} x_n e^{-\frac{2\pi\, i\, kn}{2N}} = \sum_{n=1}^{N} x_{-n} e^{\frac{2\pi\, i\, kn}{2N}} = \sum_{n=0}^{N-1} x_n e^{\frac{2\pi\, i\, k(n+1)}{2N}} = e^{\frac{\pi\, i\, k}{2N}} \sum_{n=0}^{N-1} x_n e^{\frac{2\pi\, i\, k(n+1/2)}{2N}} \tag{16}$$

Now we can rewrite the second sum as follows:

$$\sum_{n=0}^{N} x_n e^{-\frac{2\pi\, i\, kn}{2N}} = e^{\frac{\pi\, i\, k}{2N}} \sum_{n=0}^{N} x_n e^{-\frac{2\pi\, i\, k(n+1/2)}{2N}} \tag{17}$$

And if we put the two pieces back together, we get:

$$\begin{aligned}\hat{x}_k &= e^{\frac{\pi\, i\, k}{2N}} \sum_{n=0}^{N-1} x_n e^{\frac{2\pi\, i\, k(n+1/2)}{2N}} + e^{\frac{\pi\, i\, k}{2N}} \sum_{n=0}^{N} x_n e^{-\frac{2\pi\, i\, k(n+1/2)}{2N}} \\ &= e^{\frac{\pi\, i\, k}{2N}} \sum_{n=0}^{N-1} x_n \cos\frac{\pi\ k(n+1/2)}{N}\end{aligned} \tag{18}$$

We then define the discrete cosine transform II of a vector x_n, $n = 0, \ldots,$ N-1 as follows :

$$y_k = \sum_{n=0}^{N-1} x_n \cos\frac{\pi\ k(n+1/2)}{N} \qquad k = 0, \ldots, N-1 \tag{19}$$

Obviously we could have extended the initial signal using different polarity another example is provided in Fig 6 where the original signal is extended with an even parity on the right end side, and then with odd parity on the left end side.

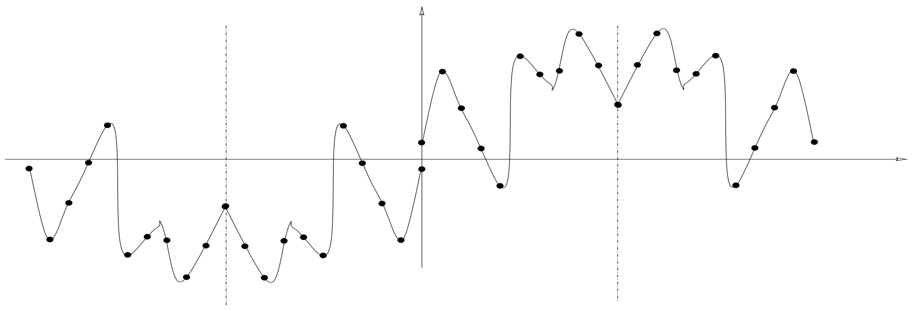

Figure 6: The original signal is extended with an even parity on the right end side, and then with odd parity on the left end side.

Definition 1 *One defines four discrete cosine transforms $\mathbf{C}^I$,$\mathbf{C}^{II}$,$\mathbf{C}^{III}$,$\mathbf{C}^{IV}$. The coefficients of these matrices are given given by :*

- *DCT-I*

$$\begin{aligned} &\mathbb{R}^{N+1} \to \mathbb{R}^{N+1} \\ C^I_{N+1}(k,n) =& b(k)b(n)\sqrt{\frac{2}{N}}\cos\frac{\pi kn}{N} \end{aligned} \tag{20}$$

- *DCT-II*

$$\begin{aligned} &\mathbb{R}^{N} \to \mathbb{R}^{N} \\ C^{II}_{N}(k,n) =& b(k)\sqrt{\frac{2}{N}}\cos\frac{\pi k(n+1/2)}{N} \end{aligned} \tag{21}$$

- *DCT-III*

$$\begin{aligned} &\mathbb{R}^{N} \to \mathbb{R}^{N} \\ C^{III}_{N}(k,n) =& b(n)\sqrt{\frac{2}{N}}\cos\frac{\pi (k+1/2)n}{N} \end{aligned} \tag{22}$$

- *DCT-IV*

$$\begin{aligned} &\mathbb{R}^{N} \to \mathbb{R}^{N} \\ C^{IV}_{N}(k,n) =& \sqrt{\frac{2}{N}}\cos\frac{\pi (k+1/2)(n+1/2)}{N} \end{aligned} \tag{23}$$

where the normalization constant $b(n)$ is defined by

$$b(n) = \begin{cases} 0 & \textit{if} \quad n < 0 \quad \textit{or} \quad n > N \\ 1/\sqrt{2} & \textit{if} \quad n = 0 \quad \textit{or} \quad n = N \\ 1 & \textit{if} \quad 1 \le n \le N-1 \end{cases} \tag{24}$$

In this definition, k plays the rôle of the frequency index (row index in the matrix), whereas n plays the rôle of the time index (column index in the matrix).

The construction just described can be performed in the continuous domain. Instead of working with discrete samples, we calculate Fourier series of properly extended functions, and expand them using the family $\{e^{i\,\pi\,kx}\ ,\ k = 0, 1, \ldots\}$. We obtain the following four orthonormal bases.

Theorem 1 *Each of the four sets constitutes an orthonormal basis of $L^2[0,1]$:*

$$\begin{aligned} &\left\{\sqrt{2}\ \sin(k+1/2)\pi x\ ,\ k = 0,1,2,\ldots\right\} \\ &\left\{\sqrt{2}\ \sin k\pi x\ ,\ k = 1,2,\ldots\right\} \\ &\left\{\sqrt{2}\ \cos(k+1/2)\pi x\ ,\ k = 0,1,2,\ldots\right\} \\ &\left\{1, \sqrt{2}\ \cos k\pi x\ ,\ k = 1,2,\ldots\right\} \end{aligned} \tag{25}$$

Figure 7 displays the first three basis functions of the last set.

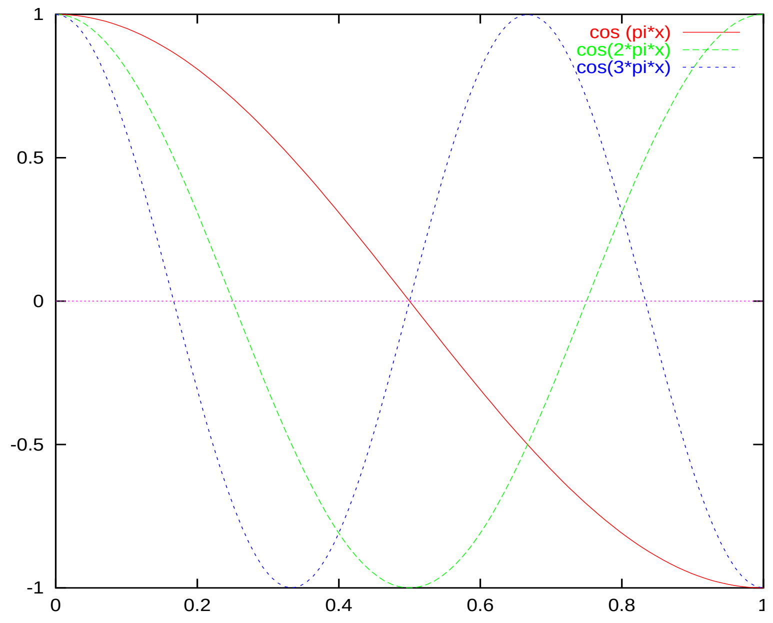

Figure 7: First three basis functions of the DCT-II

We can sample the four sets using integers or half integers, and we obtain four different discrete cosine transforms, and four discrete sine transforms.

A.2 Fast computation of the discrete cosine transform

There exists a fast algorithm to calculate DCT transforms. The algorithm is based on a formula that expresses the DCT as an FFT of half length, up to a phase shift. For instance, the DCT-IV coefficients, $\hat{x}(j), \quad j = 0, \dots, N-1$ of the sequence $x(n), \quad n = 0, \dots, N-1$ are given by:

$$\begin{aligned}\hat{x}(j) &= \text{Re}\left(e^{-\dfrac{ij\pi}{2N}} \sum_{n=0}^{N/2-1} y(n) e^{-\dfrac{2i\pi jn}{N}}\right) \\ \hat{x}(N-j-1) &= -\text{Im}\left(e^{-\dfrac{ij\pi}{2N}} \sum_{n=0}^{N/2-1} y(n) e^{-\dfrac{2i\pi jn}{N}}\right)\end{aligned} \tag{26}$$

with

$$y(n) = (x(2n) + i\, x(N-2n-1))\, e^{-\dfrac{i(n+1/4)\pi}{N}} \tag{27}$$

B Quantization

For a given budget (measured in number of bits, or compression ratio), one wishes to design the optimal quantizer in order to obtain the best reconstructed image. A very large body of work has been expended on the problem of the optimal design of quantizer [2]. We will only give a succinct overview of the subject in the following section.

B.1 Rate distortion

One consider the following problem: let X be a random variable that is being approximated by $Q(X)$. $\hat{X}$ takes only a finite number $N = 2^R$ of values, and thus can be represented using R bits. The question is then : – what is the optimal set of values that Q should take in order to minimize the error between X and $Q(X)$? For instance, let X be Gaussian distributed, $X \sim N(0, \sigma^2)$, and let us assume that $R = 1$. If one has only one bit to approximate X, and if one measures the error of quantization using the expected squared error, $E(X - Q(X))^2$, then $Q(X)$ should be defined as follows :

$$Q(x) = \begin{cases} \sqrt{\frac{2}{\pi}}, & \text{if} \quad x \geq 0 \\ -\sqrt{\frac{2}{\pi}}, & \text{if} \quad x < 0 \end{cases} \tag{28}$$

Indeed, X is approximated by the mean of the Gaussian distribution over the positive or negative values. If one has two bits, or 4 values to describe X, then the problem becomes more complicated.

In the following we present Shannon source coding theorem that gives a lower bound on the rate R, needed to encode a random variable, given a distortion D. In order to achieve this bound one needs to approximate an entire sequence of independent random variables of size n, by a vector in $\mathbb{R}^n$. In fact, the bound requires the size of the sequence to grow to infinity. Let us first introduce a few definitions.

One assumes that $X_1, X_2, \ldots, X_n$ are independent identically distributed (i.i.d.) random variables taking their values in $\mathcal{X}$. Let $p(x)$ be the probability mass function of X_i.

Definition 2 *The encoder is a function f_n that replaces the sequence $X_1, X_2, \ldots, X_n$ by an index q in $1, 2, \ldots, 2^{nR}$.*

$$f_n \ : \mathcal{X} \times \mathcal{X} \cdots \times \mathcal{X} \rightarrow \{1, 2, \ldots 2^{nR}\} \tag{29}$$

We note that if X_i is represented by 2^R discrete values, then $(X_1, \cdots, X_n)$ is represented by a vector in $2^R \times \cdots 2^R$. The previous definition assumes that $2^R \times \cdots 2^R$ is mapped to $\{1, 2, \cdots, 2^{nR}\}$.

Definition 3 *The decoder is a function g_n that reconstructs an estimate of $X_1, X_2, \ldots, X_n$ from the index q in $\{1, 2, \ldots, 2^{nR}\}$.*

$$g_n \ : \{1, 2, \ldots 2^{nR}\} \rightarrow \mathcal{X} \times \mathcal{X} \cdots \times \mathcal{X} \tag{30}$$

The set $\{g_n(1), \ldots, g_n(2^{nR}\}$ is called the codebook.

In order to be able to assess the fidelity of the reconstructed image, one needs to define a measure of the distortion introduced during the quantization process. In principle, a good measure should be easy to compute, and should bear some connection to visual perception. In practice the most commonly used measure is the Mean Square Error (MSE).

Definition 4 *A distortion function is a non negative function d :*

$$\mathcal{X} \times \mathcal{X} \rightarrow \mathcal{R}+ \, . \tag{31}$$

In the following we will consider the squared error distortion :

$$d(x,y) = (x-y)^2 \tag{32}$$

Definition 5 *The distortion between two sequences* $\mathbf{X} = (X_1, X_2, \ldots, X_n)$ *and* $\mathbf{Y} = (Y_1, Y_2, \ldots, Y_n)$ *is the average distortion :*

$$d(\mathbf{X}, \mathbf{Y}) = \frac{1}{n}\sum_{j=1}^{n} d(X_j, Y_j) \tag{33}$$

Definition 6 *A rate distortion code is composed of an encoding function :*

$$f_n \ : \mathcal{X} \times \mathcal{X} \cdots \times \mathcal{X} \to \{1, 2, \ldots 2^{nR}\} \tag{34}$$

and a decoding function

$$g_n \ : \{1, 2, \ldots 2^{nR}\} \to \mathcal{X} \times \mathcal{X} \cdots \times \mathcal{X} \tag{35}$$

The distortion associated with the code is the expected distortion

$$\begin{aligned} D &= E[d((X_1, X_2, \ldots, X_n), g_n(f_n(X_1, \cdots, X_n)) \\ &= \sum_{x_1, \cdots, x_n} p(X_1 = x_1, \cdots X_n = x_n) d((X_1, \cdots, X_n), g_n(f_n(X_1, \cdots, X_n)) \end{aligned} \tag{36}$$

Definition 7 *A rate distortion pair* $(D, R) \in \mathbb{R} \times \mathbb{N}$ *is said to be achievable if there exists a sequence of rate distortion codes* $\{f_n, g_n\}$*, such that*

$$f_n \ : \mathcal{X} \times \mathcal{X} \cdots \times \mathcal{X} \to \{1, 2, \ldots, 2^{nR}\} \tag{37}$$

and

$$\lim_{n \to \infty} E[d((X_1, \cdots, X_n), g_n(f_n(X_1, \cdots, X_n)))] \leq D \tag{38}$$

In other words, the distortion can be achieved by coding block increasing size n, while keeping the same number of bits R to describe the X_i.

Definition 8 *The rate distortion region for a given source that generates* $(X_1, X_2, \ldots, X_n)$ *is the closure of the set of achievable rate distortion pairs* (R, D)*.*

Definition 9 *For a given distortion* D*, the rate distortion function* $R(D)$ *is the infimum of rates* R *such that the pair* (R, D) *is in the rate distortion region of the source (see Figure 8).*

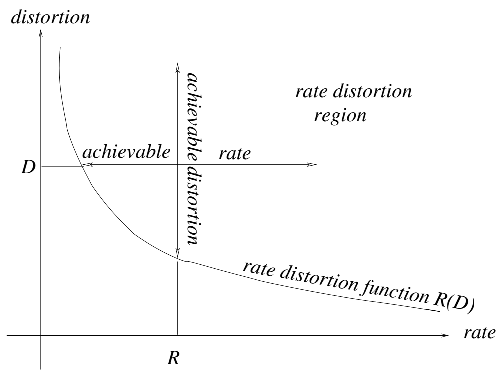

Figure 8: Rate distortion function $R(D)$

Example 1 *Let X be a Gaussian source, then the rate distortion is given by :*

$$R(D) = \begin{cases} \frac{1}{2}\log_2 \frac{\sigma^2}{D} & \textit{if} \quad 0 \le D \le \sigma^2 \\ 0 & \textit{otherwise} \end{cases} \tag{39}$$

In an equivalent form we have :

$$D(R) = \sigma^2 2^{-2R} \tag{40}$$

An informal proof of this result is provided by a sphere packing argument. One considers the n-dimensional vector formed by sequence $\mathbf{X} = (X_1, X_2, \ldots, X_n)$. Because the X_i are i.i.d $\sim N(0, \sigma_2)$, the vector $\mathbf{X}$ is on the average inside a sphere of radius $\sqrt{n\sigma^2}$ centered around the origin. Indeed, the square of the norm of $\frac{1}{\sigma}\mathbf{X}$ is distributed with a chi-square density with n degrees of freedom :

$$\frac{1}{\sigma^2}\sum_{j=1}^{n} X_i^2 \sim \chi^2(n). \tag{41}$$

Thus the mean of $\|\mathbf{X}\|^2$ is $n\sigma^2$, and $\mathbf{X}$ lies on the average inside the sphere $\mathcal{B}(0, \sqrt{n\sigma^2})$ of radius $\sqrt{n\sigma^2}$. On the other hand, if the distortion is D, then one has :

$$E(\frac{1}{n}\|\mathbf{X} - \hat{\mathbf{X}}\|^2) \le D \tag{42}$$

where $\hat{\mathbf{X}} = g_n(f_n(\mathbf{X}))$ is the reconstructed estimate of $\mathbf{X}$ after quantization. Therefore :

$$E(\|\mathbf{X} - \hat{\mathbf{X}}\|) \le \sqrt{nD} \tag{43}$$

In other words, on the average, the vector $\mathbf{X}$ lies within the sphere $\mathcal{B}(\hat{\mathbf{X}}, \sqrt{nD})$ of radius $\sqrt{nD}$ centered around $\hat{\mathbf{X}}$. If one neglects the vectors $\mathbf{X}$ falling outside $\mathcal{B}(0, \sqrt{n\sigma^2})$, then the design of the quantizer consists in finding the optimal positions of the codewords $\hat{\mathbf{X}}$ inside $\mathcal{B}(0, \sqrt{n\sigma^2})$ in such a way that all the vectors $\mathbf{X}$ fall inside a sphere $\mathcal{B}(\hat{\mathbf{X}}, \sqrt{Dn})$. If one positions the spheres $\mathcal{B}(\hat{\mathbf{X}}, \sqrt{nD})$ uniformly inside $\mathcal{B}(0, \sqrt{n\sigma^2})$, then the number of small spheres $\mathcal{B}(\hat{\mathbf{X}}, \sqrt{nD})$ needed to fill the large sphere $\mathcal{B}(0, \sqrt{n\sigma^2})$ is roughly the ratio of the volume of the large sphere to the volume of a small sphere. In dimension n, the volume of a sphere of radius r is of the form Kr^n, where K is a constant that depends on n. Therefore the number N of spheres needed to fill $\mathcal{B}(0, \sqrt{n\sigma^2})$ is given by the ratio :

$$N = \frac{K(\sqrt{n\sigma^2})^n}{K(\sqrt{nD})^n} = \left(\frac{\sigma^2}{D}\right)^{n/2} \tag{44}$$

But the size of the codebook is $N = 2^{nR}$, and therefore

$$R = \frac{1}{2}\log_2 \frac{\sigma^2}{D} \tag{45}$$

In the case where $D > \sigma^2$, then one can use $\hat{\mathbf{X}} = 0$. For all $\mathbf{X}$, the sphere $\mathcal{B}(0, \sqrt{n\sigma^2})$ is included inside $\mathcal{B}(0, \sqrt{nD})$, and the distortion constraint is automatically satisfied.

This qualitative argument gives us exactly the same rate distortion function that could be obtained if one were to use the information rate distortion function $R^I(D)$, and compute the optimal codebook that minimizes the mutual information $I(X, \hat{X})$.

It is instructive to compare this result to the scalar quantization that we described in section B.1. With 1 bit per sample, one had $D_{scalar} = \frac{\pi-2}{\pi}\sigma^2$. The optimal rate distortion computed previously gives us $D = \sigma^2/4$. This is better than the scalar case, but one should keep in mind that this is an ideal distortion that can only be achieved by encoding arbitrary long sequences.

B.2 Scalar quantization

A scalar quantizer will process each sample individually. Let X be a random variable that takes its values in an interval $[a, b]$. Let $f(x)$ be the probability density function of X. Let $Q(X)$ be the reconstructed value after scalar quantization. We measure the distortion with the mean squared error :

$$D = E[\|X - Q(X)\|^2] = \int (x - Q(x))^2 dx \tag{46}$$

A scalar quantizer is a piece-wise constant function. The regions where the functions is constant are called the bins or quantizer cells (see Fig 9). One divides $[a, b]$ into N quantizers bins, that need not have the same size. We have :

$$Q(x) = Q_j \quad \text{if} \quad x \in [x_j, x_{j+1}) \tag{47}$$

Let $\Delta_j = x_{j+1} - x_j$ be the size of the cell j.

B.2.1 Uniform scalar quantization

The simplest form is a uniform scalar quantizer, where all bins have the same size. Figure 9 shows an example of a uniform scalar quantizer, where the central bin, called the dead zone (where all values are quantized to zero) has a size twice as large as the size of the other bins.

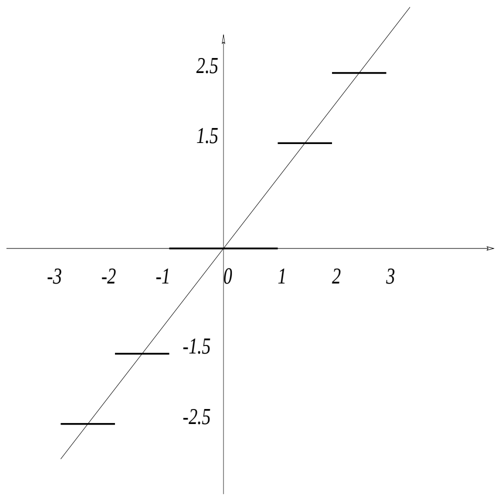

Figure 9: Uniform scalar quantizer, with a dead zone twice as large as the size of the other bins.

B.2.2 High resolution scalar quantization

In the following we examine the properties of the scalar quantizers under the following conditions :

1. the probability density function $f(x)$ of the source is smooth,
2. the number of bins N is large,
3. the size of the quantizers bins, Δ_j, is very small,

4. the p.d.f $f(x)$ in a cell is approximately constant. In particular, one has

$$f(x) = \frac{p_j}{\Delta_j} \quad \text{if} \quad x \in [x_j, x_{j+1}) \tag{48}$$

where $p_j = P(X \in [x_j, x_{j+1}))$.

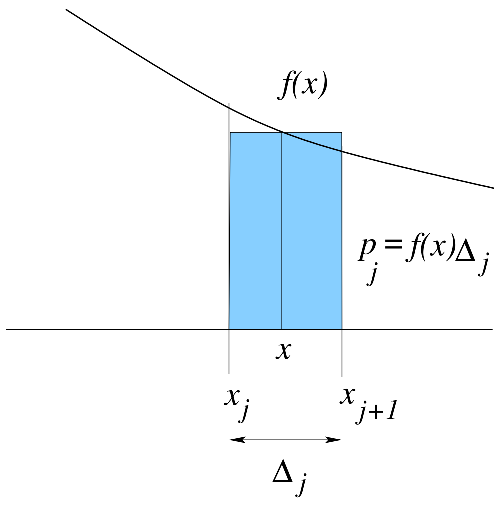

Figure 10: High resolution quantization

If these assumptions are true, we will say that the quantizer operates in the "high resolution" mode. Under these circumstances, one can compute the distortion D as a function of p_j and Δ_j.

$$D = \sum_{j=0}^{N-1} \int_{x_j}^{x_{j+1}} (x - Q(x))^2 f(x) dx \tag{49}$$

but on $[x_j, x_{j+1}]$, $f(x) = p_j/\Delta_j$, and $Q(x) = Q_j$ so one gets :

$$D = \sum_{j=0}^{N-1} \frac{p_j}{\Delta_j} \int_{x_j}^{x_{j+1}} (x - Q_j)^2 dx \tag{50}$$

Clearly, the integral $\int_{x_j}^{x_{j+1}} (x - Q_j)^2 dx$ is minimum if one chooses $Q_j = x_j + \Delta_j/2$. Then the integral is simply $\Delta_j^3/12$. Therefore the distortion becomes :

$$D = \frac{1}{12} \sum_{j=0}^{N-1} p_j \Delta_j^2 \tag{51}$$

In the case where all quantization cells are equal to Δ, the quantizer is uniform and one gets :

$$D = \frac{1}{12} \Delta^2 \sum_{j=0}^{N-1} p_j = \frac{1}{12} \Delta^2 \tag{52}$$

Let us normalize the source X, and define $Y = X/\sigma$. Y takes its values in $[a/\sigma, b/\sigma]$, and let us define

$$c = \frac{b-a}{\sigma} \tag{53}$$

c measures the size of the "normalized" range of X. We can then express D as a function of R.

$$D = \frac{\Delta^2}{12} = (\frac{b-a}{\sigma})^2 \sigma^2 \frac{1}{12N^2} \tag{54}$$

but $N = 2^R$, and thus

$$D = \frac{c^2}{12}\sigma^2 2^{-2R} \tag{55}$$

The distortion increases as σ increases, and it decays exponentially as a function of the rate R.

C Entropy coding

We very briefly describe in this section the principles of entropy coding. A very good reference on the subject is provided in [1].

The aim of entropy coding is to provide a representation of a set of symbols with a description whose average length is minimal. This can be achieved by using a variable length coding mechanism: the most frequent outcomes of the data is encoded with the shortest description, and longer descriptions are kept for less frequent realizations.

We consider a source vector $\mathbf{X} = (X_0, X_1, \dots, X_{N-1})$, where the X_i are random variables that take their values in an alphabet $\mathcal{A} = \{a_0, a_1, \dots, a_{M-1}\}$. We assume that the X_i are identically distributed with a probability distribution p.

An encoder α is defined as a map from the alphabet $\mathcal{A}$ to the set of binary sequences:

$$\alpha \ : \mathcal{A} \longrightarrow \{0,1\}^* \tag{56}$$

A decoder β is defined as a map from the the set of binary sequences back to the alphabet:

$$\beta \ : \{0,1\}^* \longrightarrow \mathcal{A} \tag{57}$$

A minimum property required for the decoder is that one should be able to recover the original data perfectly:

$$\beta(\alpha(a)) = a \tag{58}$$

We define the length $l(\alpha(a))$ of an encoded symbol $\alpha(a)$ as the number of bits (0, and 1) in $\alpha(a)$. The average length of the code over the alphabet is

$$\bar{l}(\alpha) = \sum_{a \in \mathcal{A}} p(a) l(\alpha(a)) \tag{59}$$

Encoders can generate fixed length sequences, or variable length sequences.

Example 2 *Let X be a random variables that takes its value in $\{0, 1, 2, 3\}$, with the probabilities :*

$$\begin{aligned} P(X = 0) &= 1/2 \\ P(X = 1) &= 1/4 \\ P(X = 2) &= 1/8 \\ P(X = 3) &= 1/8 \end{aligned}$$

We construct the following code α,

$$\begin{aligned} \alpha(0) &= 0 \\ \alpha(1) &= 10 \\ \alpha(2) &= 110 \\ \alpha(3) &= 111 \end{aligned} \tag{60}$$

The average code length is

$$\bar{l}(\alpha) = \frac{1}{2} + \frac{2}{4} + \frac{6}{8} = 1.75$$

We note that any sequence of bits that was generated by the coder by concatenating the codewords can be uniquely decoded to generate the original sequence of symbols. For instance, the sequence 110010101110 *is decoded as* 312241.

Example 3 *Let X be a random variables that takes its value in $\{0, 1, 2, 3\}$. We construct the following code α,*

$$\begin{aligned} \alpha(0) &= 0 \\ \alpha(1) &= 01 \\ \alpha(2) &= 010 \\ \alpha(3) &= 10 \end{aligned} \tag{61}$$

This code can generate sequences of bits that can be decoded in multiple ways. The code is not useful, because it creates ambiguity in the decoding process. For instance the sequence 01010 *can be interpreted as decoded in the following ways :*

$$\begin{aligned} 0, 10, 10 &\rightarrow 0, 3, 3 \\ 01, 0, 10 &\rightarrow 1, 0, 3 \\ 01, 010 &\rightarrow 1, 3 \\ 010, 10 &\rightarrow 3, 1 \end{aligned} \tag{62}$$

The ambiguity stems from the fact that the beginning (prefix) of some codewords coincide with already existing codewords.

Definition 10 *A code is called a prefix code, or an instantaneous code if no codeword is a prefix of any other codeword*

The code defined in example 2 is instantaneous, while the code defined in example 3 is not.

As shown in Fig. 11, the codewords of a prefix code can be organized in a binary tree.

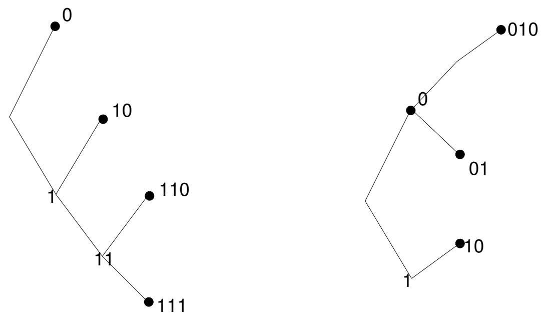

Figure 11: Left : prefix code of example 2. Right : the code of example 3 is clearly not a prefix code.

The fact that the code is instantaneous can be described by the fact that the codewords cannot have any children in the tree (see Fig. 11).

Definition 11 *We define a uniquely decodable code as a code such that, for any valid encoded sequence of finite length, there exists only one possible input symbol.*

The goal of entropy encoding is to build a uniquely decodable encoder, that minimizes the average code length. As one can expect, there is a lower bound on the average length of a uniquely decodable code. This lower bound is given by he Kraft inequality:

Lemma 1 (Kraft Inequality) *If a code is uniquely decodable, then*

$$\sum_{a \in \mathcal{A}} 2^{-l(\alpha(a))} \leq 1 \tag{63}$$

Example 4 *Let us assume that the each X_i can take L values with a uniform probability $p = 1/L$. If one codes each X_i with $\log_2 L = -\log_2 p$ bits, then*

$$\sum_{l=1}^{L} 2^{-\log_2 L} = \sum_{l=1}^{L} \frac{1}{L} = 1 \tag{64}$$

and the Kraft inequality is satisfied. In fact , one can get a more precise estimate of the average length of a code as a function of the complexity of the source (measured with the entropy).

Definition 12 *Let X be a random variable that takes its values in the alphabet $\mathcal{A}$, and let p be the probability distribution of X. We define the entropy of X as:*

$$H(X) = -\sum_{a \in \mathcal{A}} p(a) \log_2 p(a) \tag{65}$$

where $\log_2$ is the logarithm in base 2.

Example 5 *Let*

$$X = \begin{cases} 1 & \text{with probability} \quad p \\ 0 & \text{with probability} \quad 1-p \end{cases} \tag{66}$$

Then

$$H(X) = -p\log_2 p - (1-p)log_2(1-p) \tag{67}$$

We can then state Shannon's lossless coding theorem.

Lemma 2 (Shannon 1948) *For any uniquely decodable scalar lossless code α we have*

$$\bar{l}(\alpha) \geq H(X) \tag{68}$$

and there exists a code such that

$$\bar{l}(\alpha) < H(X) + 1 \tag{69}$$

We will not give the complete proof of the theorem, but we will show that $\bar{l}(\alpha) \geq H(x)$. From Kraft inequality we have

$$\frac{1}{\sum_{b \in \mathcal{A}} 2^{-l(\alpha(b))}} \geq 1 \tag{70}$$

and therefore

$$\frac{2^{-l(\alpha(a))}}{\sum_{b \in \mathcal{A}} 2^{-l(\alpha(b))}} \geq 2^{-l(\alpha(a))}. \tag{71}$$

Therefore

$$\bar{l}(\alpha) = \sum_{a \in \mathcal{A}} p(a) l(a) = -\sum_{a \in \mathcal{A}} p(a) \log_2 2^{-l(\alpha(a))} \geq -\sum_{a \in \mathcal{A}} p(a) \log_2 \frac{2^{-l(\alpha(a))}}{\sum_{b \in \mathcal{A}} 2^{-l(\alpha(b))}} \tag{72}$$

The quantity

$$q(a) = \frac{2^{-l(\alpha(a))}}{\sum_{b \in \mathcal{A}} 2^{-l(\alpha(b))}} \tag{73}$$

can be interpreted as a probability distribution over the alphabet $\mathcal{A}$. We can use the inequality

$$\sum_{a \in \mathcal{A}} p(a) \log_2 \frac{1}{q(a)} \geq \sum_{a \in \mathcal{A}} p(a) \log_2 \frac{1}{p(a)} \tag{74}$$

to conclude that

$$\bar{l}(\alpha) \geq -\sum_{a \in \mathcal{A}} p(a) \log_2 p(a) = H(X). \tag{75}$$

Example 6 *For the code of example 2, we have $H(X) = -(\frac{1}{2}\log_2 \frac{1}{2} + \frac{1}{4}\log_2 \frac{1}{4} + \frac{1}{8}\log_2 \frac{1}{8} + \frac{1}{8}\log_2 \frac{1}{8}) = 1.75$. Therefore we have $\bar{l}(\alpha) = H(X)$.*

If one is coding a block of stationary random variables $\mathbf{X} = (X_0, \ldots, X_{N-1})$ then the entropy rate is defined as

$$\overline{H} = \min_{N} \frac{H(\mathbf{X})}{N} = \lim_{N \to \infty} \frac{H(\mathbf{X})}{N} \tag{76}$$

and there exists a prefix code for which the average length per symbol is bounded by

$$\overline{H}(\mathbf{X}) \leq \frac{1}{N}\bar{l}(\alpha) < \overline{H}(\mathbf{X}) + \frac{1}{N} \tag{77}$$

The performance of the coder can in principle be improved by coding blocks of larger size. But the increase in block size result in a major increase in complexity.

References

[1] T. Cover and J. Thomas, *Elements of information theory*, John Wiley, 1991.

[2] A. Gersho and R.M. Gray, *Vector quantization and signal compression*, Kluwer, 1992.

[3] KR Rao and P. Yip, *Discrete cosine transform: algorithms, advantages, applications*, Academic Press Professional, Inc. San Diego, CA, USA, 1990.