Tutorial 3: Sampling and Quantising

When we want to manipulate an image in a computer, it is necessary that this image be digitised both in its spatial and intensity dimensions. Digitisation in the spatial coordinates *(x,y)* is called *image sampling* and amplitude digitisation is called *grey-level quantisation*.

Sampling is the process of selecting regularly spaced points in time in which to record the level of a signal. Quantisation is the process of turning a continuous or analogue signal into one, which has discrete numerical values at each point in (for instance) time. Voice was sampled for telephone at 8,000 samples/sec each sample quantised with 8 bits. In audio applications, a voltage is typically quantised into a 16-bit number, that can represent 65536 different levels of voltage. The numbers are usually proportional to the input voltages, but logarithmic relationships have also been used. It is important to note that quantisation is a separate process from sampling; after quantisation the amplitude of the signal is still represented continuously in time.

The previous sampling rates were selected because it allows us to reconstruct the original signal with adequate precision. Nyquist's theorem says that if we sample a signal in which the highest frequency we wish to reproduce correctly is ***f*** then we must sample the signal at a minimum frequency of ***2f***. For this sampling rate, the frequency ***f*** is known as the *Nyquist* frequency, f_N.

A signal that is sampled below this minimum rate is *undersampled*, and the effect of doing this is to generate spurious contributions of lower frequencies in the reconstructed signal. This process is known as *aliasing*.

Analogue-to-digital conversion is the overall process of taking an analogue signal, such as the fluctuating voltage from a microphone and turning it into a digital signal, consisting of a stream of numbers.

Let the original digital image $I= f(x,y)$ have dimensions for rows and columns N_r x N_c. Let L_c =1,2,...,N_c and L_r=1,2,...,N_r be the horizontal and vertical spatial domains, and G=1,2,...,N_g the set of grey tones. The image I can be represented then as a function that assigns a grey tone to each pair of co-ordinates:

$$L_r \times L_c; I : L_r \times L_c \rightarrow G$$

In many cases the dimensions of the image are integer powers of two:

$N_r = 2^{nr}$, $N_c = 2^{nc}$, $N_g = 2^{ng}$

The number of bits required to store the image become:

$b = N_r$ x N_c x Ng

Since f(x,y) is already an approximation of the original image $f_a(x,y)$, the resolution required for a "good" representation can vary according to the image itself and the quality desired (which defines what "good" means for every application). To sample an image in Matlab, you can use the index of the corresponding matrix to select a reduced number of entries of the matrix. First convert image into a matrix of doubles:

```
>>image2=double(image2);        %transform into doubles to manipulate
```

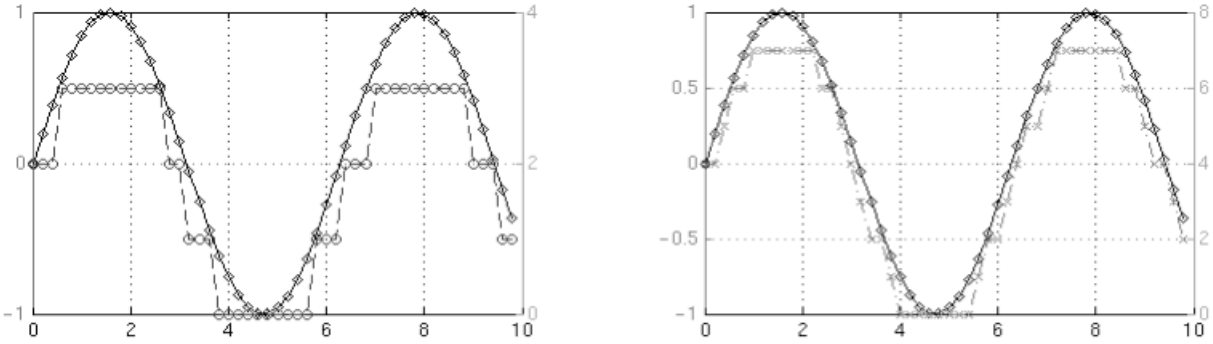
In the same way that the matrices were defined with the colon operator (*lower limit : increment : upper limit*) the elements can be selected. Try:

```
>> imagesc (image2(1:1:end,1:1:end))
>> imagesc (image2(1:2:end,1:2:end))
>> imagesc (image2(1:3:end,1:3:end))
>> imagesc (image2(1:4:end,1:4:end))
```

3.1 The previous instructions should display the moon image with different resolutions. Try this on other images.

In all cases the number of grey levels is not modified. To quantise any signal, the *uencode* function can be used. The following example quantises a sine function. Figure 2 shows the sine curve and its quantised version with (a) 2 bit = 4 levels and (b) 3 bits = 8 levels.

```
>> t=0:0.2:9.9;
>> x=sin(t);
>> y2=uencode(x,2);
>> y3=uencode(x,3);
>> [ax,h1,h2]=plotyy(t,x,t,y2);
>> [ax,h3,h4]=plotyy(t,x,t,y3);
```



(a) (b)

Figure 2

3.2 Find out more of the functions *uencode* and *plotyy*. The *plotyy* returns the handles of the two axes created in *ax*, and the handles of the graphics objects from each plot in *h1* and *h2*. *ax*(1) is the left axes and *ax*(2) is the right axes.

3.3 Use *help* to study about handles and their properties, they are quite useful!

3.4 Use *uencode* to quantise any of the previous images, change the number of quantising bits and display the quantised images. What can you observe?