

SIGNAL PROCESSING IN PROCESSING: SAMPLING AND QUANTIZATION*

Davide Rocchesso
Pietro Polotti

Based on *Signal Processing in Processing: Sampling and Quantization*[†] by
Davide Rocchesso
Pietro Polotti

This work is produced by The Connexions Project and licensed under the
Creative Commons Attribution License [‡]

Abstract

Fundamentals of sampling, reconstruction, and quantization of 1D (sounds) and 2D (images) signals, especially oriented at the Processing language.

1 Sampling

Both sounds and images can be considered as signals, in one or two dimensions, respectively. Sound can be described as a fluctuation of the acoustic pressure in time, while images are spatial distributions of values of luminance or color, the latter being described in its RGB or HSB components. Any signal, in order to be processed by numerical computing devices, have to be reduced to a sequence of discrete **samples**, and each sample must be represented using a finite number of bits. The first operation is called **sampling**, and the second operation is called **quantization** of the domain of real numbers.

1.1 1-D: Sounds

Sampling is, for one-dimensional signals, the operation that transforms a continuous-time signal (such as, for instance, the air pressure fluctuation at the entrance of the ear canal) into a discrete-time signal, that is a sequence of numbers. The discrete-time signal gives the values of the continuous-time signal read at intervals of T seconds. The reciprocal of the sampling interval is called **sampling rate** $F_s = \frac{1}{T}$. In this module we do not explain the theory of sampling, but we rather describe its manifestations. For a a more extensive yet accessible treatment, we point to the *Introduction to Sound Processing* [1]. For our purposes, the process of sampling a 1-D signal can be reduced to three facts and a theorem.

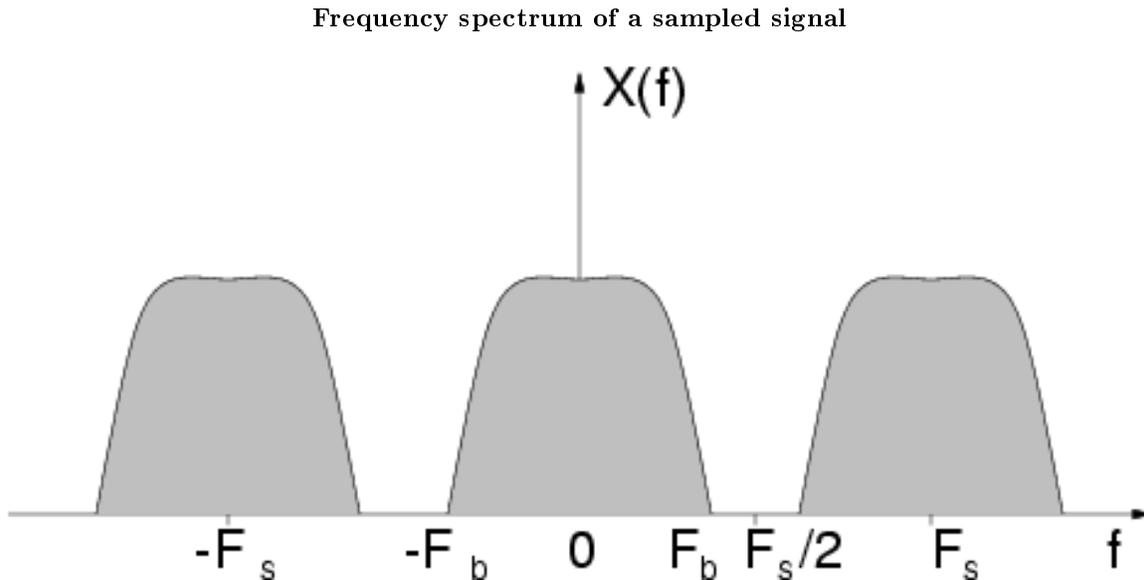
*Version 1.6: Dec 9, 2011 2:33 pm -0600

[†]<http://cnx.org/content/m12751/1.7/>

[‡]<http://creativecommons.org/licenses/by/3.0/>

- The Fourier Transform¹ of a discrete-time signal is a function (called **spectrum**) of the continuous variable ω , and it is periodic with period 2π . Given a value of ω , the Fourier transform gives back a complex number that can be interpreted as magnitude and phase (translation in time) of the sinusoidal component at that frequency.
- Sampling the continuous-time signal $x(t)$ with interval T we get the discrete-time signal $x(n) = x(nT)$, which is a function of the discrete variable n .
- Sampling a continuous-time signal with sampling rate F_s produces a discrete-time signal whose frequency spectrum is the periodic replication of the original signal, and the replication period is F_s . The Fourier variable ω for functions of discrete variable is converted into the frequency variable f (in Hertz) by means of $f = \frac{\omega}{2\pi T}$.

The Figure 1 (Frequency spectrum of a sampled signal) shows an example of frequency spectrum of a signal sampled with sampling rate F_s . In the example, the continuous-time signal had all and only the frequency components between $-F_b$ and F_b . The replicas of the original spectrum are sometimes called **images**.



Given the facts (p. 1), we can have an intuitive understanding of the Sampling Theorem, historically attributed to the scientists Nyquist and Shannon.

Theorem 1: Sampling Theorem

A continuous-time signal $x(t)$, whose spectral content is limited to frequencies smaller than F_b (i.e., it is band-limited to F_b) can be recovered from its sampled version $x(n)$ if the sampling rate is larger than twice the bandwidth (i.e., if $F_s > 2F_b$)

The reconstruction can only occur by means of a filter that cancels out all spectral images except for the one directly coming from the original continuous-time signal. In other words, the canceled images are those

¹"Derivation of the Fourier Transform" <<http://cnx.org/content/m0046/latest/>>

having frequency components higher than the **Nyquist frequency** defined as $\frac{F_s}{2}$. The condition required by the sampling theorem (Theorem 1, Sampling Theorem, p. 2) is equivalent to saying that no overlaps between spectral images are allowed. If such superimpositions were present, it wouldn't be possible to design a filter that eliminates the copies of the original spectrum. In case of overlapping, a filter that eliminates all frequency components higher than the Nyquist frequency would produce a signal that is affected by **aliasing**. The concept of aliasing is well illustrated in the Aliasing Applet², where a continuous-time sinusoid is subject to sampling. If the frequency of the sinusoid is too high as compared to the sampling rate, we see that the waveform that is reconstructed from samples is not the original sinusoid, as it has a much lower frequency. We all have familiarity with aliasing as it shows up in moving images, for instance when the wagon wheels in western movies start spinning backward. In that case, the sampling rate is given by the **frame rate**, or number of pictures per second, and has to be related with the spinning velocity of the wheels. This is one of several stroboscopic³ phenomena.

In the case of sound, in order to become aware of the consequences of the 2π periodicity of discrete-time signal spectra (see Figure 1 (Frequency spectrum of a sampled signal)) and of violations of the condition of the sampling theorem, we examine a simple case. Let us consider a sound that is generated by a sum of sinusoids that are harmonics (i.e., integer multiples) of a fundamental. The spectrum of such sound would display peaks corresponding to the fundamental frequency and to its integer multiples. Just to give a concrete example, imagine working at the sampling rate of 44100 Hz and summing 10 sinusoids. From the sampling theorem we know that, in our case, we can represent without aliasing all frequency components up to 22050 Hz. So, in order to avoid aliasing, the fundamental frequency should be lower than 2205 Hz. The Processing (with Beads library) code reported in table Table 1 implements a generator of sounds formed by 10 harmonic sinusoids. To produce such sounds it is necessary to click on a point of the display window. The x coordinate would vary with the fundamental frequency, and the window will show the spectral peaks corresponding to the generated harmonics. When we click on a point whose x coordinate is larger than $\frac{1}{10}$ of the window width, we still see ten spectral peaks. Otherwise, we violate the sampling theorem and aliasing will enter our representation.

²"Aliasing Applet" <<http://cnx.org/content/m11448/latest/>>

³http://www.michaelbach.de/ot/mot_strob/

Aliasing test:
Applet to
experience
the effect
of aliasing
on sounds
obtained by
summation of
10 sinusoids
in harmonic
ratio ⁴

```

import beads.*; // import the beads library
import beads.Buffer;
import beads.BufferFactory;

AudioContext ac;
PowerSpectrum ps;

WavePlayer wavetableSynthesizer;
Glide frequencyGlide;
Envelope gainEnvelope;
Gain synthGain;

int L = 16384; // buffer size
int H = 10; //number of harmonics
float freq = 10.00; // fundamental frequency [Hz]
Buffer dSB;

void setup() {
    size(1024,200);

    frameRate(20);

    ac = new AudioContext(); // initialize AudioContext and create buffer

    frequencyGlide = new Glide(ac, 200, 10); // initial freq, and transition time
    dSB = new DiscreteSummationBuffer().generateBuffer(L, H, 0.5);
    wavetableSynthesizer = new WavePlayer(ac, frequencyGlide, dSB);

    gainEnvelope = new Envelope(ac, 0.0); // standard gain control of AudioContext
    synthGain = new Gain(ac, 1, gainEnvelope);
    synthGain.addInput(wavetableSynthesizer);
    ac.out.addInput(synthGain);

    // Short-Time Fourier Analysis
    ShortFrameSegmenter sfs = new ShortFrameSegmenter(ac);
    sfs.addInput(ac.out);
    FFT fft = new FFT();
    sfs.addListener(fft);
    ps = new PowerSpectrum();
    fft.addListener(ps);
    ac.out.addDependent(sfs);

    ac.start(); // start audio processing
    gainEnvelope.addSegment(0.8, 50); // attack envelope
}

void mouseReleased(){
    println("mouseX = " + mouseX);
}

void draw()
{
    background(0);

    text("click and move the pointer", 800, 20);
    frequencyGlide.setValue(float(mouseX)/width*22050/10); // set the fundamental frequency

```

Table 1

1.2 2-D: Images

Let us assume we have a continuous distribution, on a plane, of values of luminance or, more simply stated, an image. In order to process it using a computer we have to reduce it to a sequence of numbers by means of sampling. There are several ways to sample an image, or read its values of luminance at discrete points. The simplest way is to use a regular grid, with spatial steps X e Y . Similarly to what we did for sounds, we define the spatial sampling rates $F_X = \frac{1}{X}$ and $F_Y = \frac{1}{Y}$. As in the one-dimensional case, also for two-dimensional signals, or images, sampling can be described by three facts and a theorem.

- The Fourier Transform of a discrete-space signal is a function (called **spectrum**) of two continuous variables ω_X and ω_Y , and it is periodic in two dimensions with periods 2π . Given a couple of values ω_X and ω_Y , the Fourier transform gives back a complex number that can be interpreted as magnitude and phase (translation in space) of the sinusoidal component at such spatial frequencies.
- Sampling the continuous-space signal $s(x, y)$ with the regular grid of steps X, Y , gives a discrete-space signal $s(m, n) = s(mX, nY)$, which is a function of the discrete variables m and n .
- Sampling a continuous-space signal with spatial frequencies F_X and F_Y gives a discrete-space signal whose spectrum is the periodic replication along the grid of steps F_X and F_Y of the original signal spectrum. The Fourier variables ω_X and ω_Y correspond to the frequencies (in cycles per meter) represented by the variables $f_X = \frac{\omega_X}{2\pi X}$ and $f_Y = \frac{\omega_Y}{2\pi Y}$.

The Figure 2 (Spectrum of a sampled image) shows an example of spectrum of a two-dimensional sampled signal. There, the continuous-space signal had all and only the frequency components included in the central hexagon. The hexagonal shape of the spectral support (region of non-null spectral energy) is merely illustrative. The replicas of the original spectrum are often called spectral **images**.

⁴See the file at <<http://cnx.org/content/m13045/latest/.index.html>>

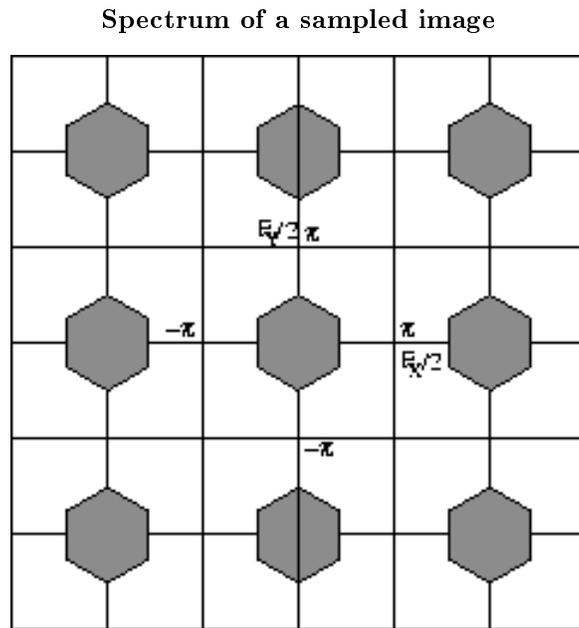


Figure 2

Given the above facts (p. 5), we can have an intuitive understanding of the Sampling Theorem.

Theorem 2: Sampling Theorem (in 2D)

A continuous-space signal $s(x, y)$, whose spectral content is limited to spatial frequencies belonging to the rectangle having semi-edges F_{bX} and F_{bY} (i.e., bandlimited) can be recovered from its sampled version $s(m, n)$ if the spatial sampling rates are larger than twice the respective bandwidths (i.e., if $F_X > 2F_{bX}$ and $F_Y > 2F_{bY}$)

In practice, the spatial sampling step can not be larger than the semi-period of the finest spatial frequency (or the finest detail) that is represented in the image. The reconstruction can only be done through a filter that eliminates all the spectral images but the one coming directly from the original continuous-space signal. In other words, the filter will cut all images whose frequency components are higher than the **Nyquist frequency** defined as $\frac{F_X}{2}$ and $\frac{F_Y}{2}$ along the two axes. The condition required by the sampling theorem (Theorem 2, Sampling Theorem (in 2D), p. 6) is equivalent to requiring that there are no overlaps between spectral images. If there were such overlaps, it wouldn't be possible to eliminate the copies of the original signal spectrum by means of filtering. In case of overlapping, a filter cutting all frequency components higher than the Nyquist frequency would give back a signal that is affected by aliasing.

We note how aliasing can be produced by down-sampling (or decimating) a sampled image. Starting from a discrete-space image, we can select only a subset of samples arranged in a regular grid. This will determine the periodic repetition of the spectral images, that will end up overlapping.

In order to explore the concepts of sampling, down-sampling, and aliasing, run the applet drawing ellipses⁵. With the keyboard arrow you can double or halve the horizontal and vertical sampling steps.

⁵See the file at <http://cnx.org/content/m13045/latest/resampling_ellipse.html>

A simple introduction to the first elements of image processing is found in Digital Image Processing Basics⁶.

2 Quantization

With the adjective "digital" we indicate those systems that work on signals that are represented by numbers, with the (finite) precision that computing systems allow. Up to now we have considered discrete-time and discrete-space signals as if they were collections of infinite-precision numbers, or real numbers. Unfortunately, computers only allow to represent finite subsets of rational numbers. This means that our signals are subject to quantization.

For our purposes, the most interesting quantization is the linear one, which is usually occurring in the process of conversion of an analog signal into the digital domain. If the memory word dedicated to storing a number is made of b bits, then the range of such number is discretized into 2^b quantization levels. Any value that is found between two quantization levels can be approximated by truncation or rounding to the closest value. The Figure 3 (Sampling and quantization of an analog signal) shows an example of quantization with representation on 3 bits in two's complement⁷.

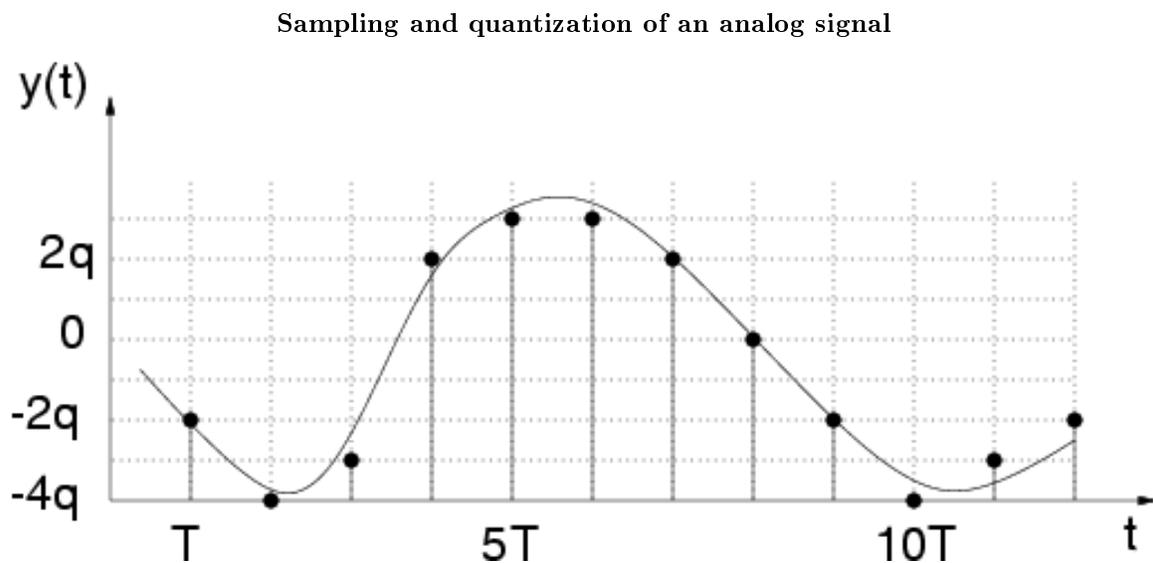


Figure 3

The approximation introduced by quantization manifests itself as a noise, called **quantization noise**. Often, for the analysis of sound-processing circuits, such noise is assumed to be white and de-correlated with the signal, but in reality it is perceptually tied to the signal itself, in such an extent that quantization can be perceived as an effect.

To have a visual and intuitive exploration of the phenomenon of quantization, consider the applet ⁸ that allows to vary between 1 and 8 the number of bits dedicated to the representation of each of the RGB

⁶"Digital Image Processing Basics" <<http://cnx.org/content/m10973/latest/>>

⁷"Two's Complement and Fractional Arithmetic for 16-bit Processors" <<http://cnx.org/content/m10808/latest/>>

⁸See the file at <<http://cnx.org/content/m13045/latest/quantagondoleBeads.html>>

channels representing color. The same number of bits is dedicated to the representation of an audio signal coupled to the image. The visual effect that is obtained by reducing the number of bits is similar to a **solarization**.

Exercise 1*(Solution on p. 9.)*

Extend the code of the applet Table 1 to add some interaction features:

- Make the fundamental frequency of the automatically-generated sound changing randomly at each frame (see `random()`⁹).
- Make the framerate (and the metronome for generating notes) dependent on the horizontal position of the mouse (`mouseX`).
- Make the number of harmonics of the sound (i.e., its brightness) dependent on the vertical position of the mouse (`mouseY`).
- Paint the window background in such a way that moving from left to right the tint changes from blue to red. In this way, a blue color will correspond to a slow tempo, and a red color to a fast tempo.
- Make the color saturation of the background dependent on the vertical position of the mouse. In this way a sound with a few harmonics (low brightness) will correspond to a grayish color, while a sound rich of harmonics (high brightness) will correspond to a saturated color.
- Add a control to stop the computation and display of the spectrum and create an effect of image freezing, while sound continues to be generated (for instance by keeping the mouse button pressed).
- Add a control to cancel the dependence of tempo, brightness, and color saturation on mouse position (for instance by pressing a key).
- Add a control that, in case of image freezing (mouse button pressed), will stop the generation of new notes while "freezing" the current note.
- Finally, add a control that, in case of image freezing, will stop the sound generation and make the application silent.

⁹http://www.processing.org/reference/random_.html

Solutions to Exercises in this Module

Solution to Exercise (p. 8)

The proposed extensions are implemented in the Processing code¹⁰.

References

- [1] Davide Rocchesso. *Introduction to Sound Processing*. Mondo Estremo, 2003. <http://www.mondo-estremo.com>.

¹⁰See the file at <<http://cnx.org/content/m13045/latest/./aliasingfermoDBeads.pde>>