Antialiased shifting and resizing

Dan Fain, Denis Zorin

Contents

1 Introduction

1.1 Aliasing

The goal of this handout is to describe the mathematical foundations of sampling and aliasing.

Aliasing is a potential problem whenever something continuous is represented by a set of discrete samples. In computer graphics, aliasing is most pervasive when displaying idealized objects with sharp lines and edges. The resulting staircase-like artifacts are known as *jaggies*. Aliasing also occurs when images are scaled, rotated or even translated. Elimination of aliasing is particularly important for computer-generated animations, where it leads to a number of highly objectionable perceptual artefacts.

1.2 Quantization

Aliasing is often confused with *quantization error*. Since aliasing is caused by sampling, it is related to *resolution*, which for images measured in number of pixels or pixel density. Quantization is the roudning off of the color of each pixel into a finite set of values. It is related to the number of *bits of color*. Computer video modes are generally specified in terms of these parameters (e.g. 1024×768 resolution with 24-bit color).

In this handout we will not consider quantization error. We will treat computer arithmetic as if it operated with infinite precision. If the techniques described in this handout are implemented using 32-bit floating point calculations on 8-bit-per-channel source images, then roundoff error may be neglected.

2 Background

2.1 Fourier transform

Frequency domain. In this section, we will look at how a signal, such as an audio recording or an image, may be broken down into different frequency components. This change in representation is accomplished by the *Fourier tranform*. The

Spatial frequency. When we discuss audio signals we typically talk about time frequency, that is, how fast a signal oscillates in time. In the context of antialiased image manipulation, we almost always use "frequency" to mean *spatial frequency*, rather than, for example, the frequency of light waves. Just as the simple harmonic oscillator is a simple case of something with a well-defined frequency of motion, a *sine wave grating* (Figure 1) is an image with a single, well-defined spatial frequency.

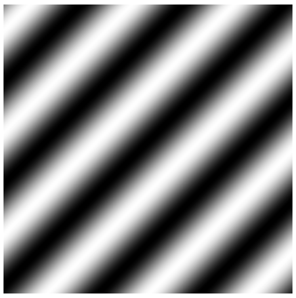


Figure 1: Sine-wave grating. The intensity of the image varies as a sine wave.

Angular frequency units. The unit of frequency that we use in this handout is the *radian per unit time or space*. For example, if a weel rotates at the rate 1 full rotation per second, the frequency is 2π radians per second. If a sine wave has period T seconds, the frequency of this wave is $2\pi/T$. When we consider images, the period T is spacial, not temporal, and is measured in suitable units, such as inches or pixels.

One-dimensional signals We will mostly consider one-dimensional signals–functions of one variable, such as monophonic audio recordings, or a single row of an image. Everything can be easily generalized to two-dimensional signals, such as images.

Fourier transform The Fourier transform $\mathcal{F}[x]$ of a function $x(t)$ is defined as:

$$\mathcal{F}[x](\omega) = \int_{-\infty}^{\infty} x(t)e^{-i\omega t}dt$$

For a given a frequency ω, the Fourier transform $\mathcal{F}[x](\omega)$ is a complex number c. The magnitude of c indicates how much of the frequency is present, and the ω indicates the phase of that frequency component relative to the origin.

Inverse Fourier transform The inverse Fourier transform recovers the original signal from its Fourier transform, and is defined as:

$$\mathcal{F}^{-1}[X] = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(\omega)e^{i\omega t}d\omega$$

Note that the forward and inverse FTs are almost the same operation. This will be quite convenient when analyzing sampling and filtering.

Examples The Fourier transforms of the following simple signals will be useful for understanding sampling, shifting and resizing.

Constant function A constant function k consists of only one frequency–zero. Therefore its Fourier transform $K = \mathcal{F}[k]$ must be zero everywhere except at $\omega = 0$.

$$\text{for all } \omega \neq 0 \ , \quad K(\omega) = 0$$

The value at zero frequency should be selected such that the inverse Fourier transform yields k:

$$k = \frac{1}{2\pi} \int_{-\infty}^{\infty} K(\omega)e^{i\omega t}d\omega$$

To satisfy these two conditions simulteously we need to introduce an unconventional function, the δ-function, which is defined as:

$$\text{for all } x \neq 0 \ , \quad \delta(x) = 0$$

$$\text{for all } \epsilon \ , \quad \int_{-\epsilon}^{+\epsilon} \delta(x)dx = 1$$

Despite its name, it is not a function at all; it is a more general object called *distribution*. The δ-function can be thought of as an operator that cancels integration: when multiplied by a function appearing inside an integral, it causes the value of the integral to be equal to the value of the function evaluated at a point.

So the Fourier transform $K(\omega)$ of a constant function k is a delta-function at the origin:

$$K(\omega) = 2\pi k\delta(\omega)$$

Box The *box function* is defined by:

$$b(t) = \begin{cases} 1 & -1/2 \leq t \leq 1/2 \\ 0 & \text{otherwise} \end{cases}$$

Its Fourier transform, known as the *sinc*, is given by:

$$\begin{aligned} B = \mathcal{F}[b] &= \int_{-\infty}^{\infty} b(t) e^{-i\omega t} dt \\ &= \int_{-1/2}^{1/2} e^{-i\omega t} dt \\ &= \frac{\sin(\omega/2)}{(\omega/2)} = sinc\left(\frac{\omega}{2\pi}\right) \end{aligned}$$

The definition of *sinc* is

$$sinc(x) = \frac{\sin \pi x}{\pi x}$$

Comb The definition of the *comb function* is:

$$comb = \sum_{n=-\infty}^{\infty} \delta(t - nT)$$

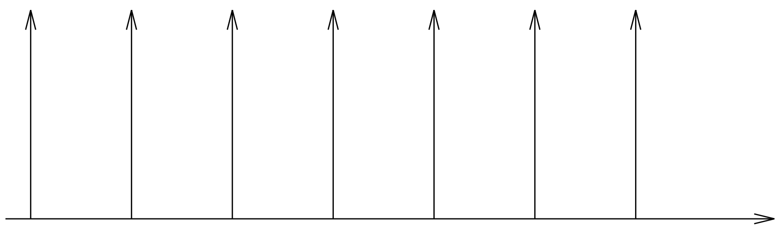


Figure 2: The *comb* function.

Since this function is periodic, it may be expressed as a Fourier series, and all its frequency content are in a series of harmonics. In fact, its transform is another comb function. Since the period of the original comb was T, the fundamental frequency of its transform will be $2\pi/T$.

$$\mathcal{F}[comb] = \frac{2\pi}{T} \sum_{n=-\infty}^{\infty} \delta(\omega - (2\pi n/T))$$

The derivation of this fact is relatively complicated and we omit it here. Section 2.3 describes a generalization of the comb function known as the *impulse train*.

2.2 Filtering and convolution

Later in this handout we will describe circumstances in which we want to remove high or low frequencies from images. The frequency domain is a natural place to think of doing this; ideally, we could multiply all the desired frequencies by one and all the unwanted ones by zero. Although strictly speaking such an operation is not possible, we will use it as an idealization throughout this handout.

Useful properties Figure 3 summarizes a number of identities involving the Fourier transforms of special types of signals.

Name of the property	signals	Fourier transforms
Linearity	$ax(t) + by(t)$	$a\mathcal{F}[x](\omega) + b\mathcal{F}[y](\omega)$
Time shifting (Section 3)	$x(t - t_0)$	$e^{-i\omega t_0}\mathcal{F}[x](\omega)$
Zooming (Section 4.1)	$x(t/a)$	$a\mathcal{F}[x](a\omega)$
Differentiation	$dx(t)/dt$	$i\omega\mathcal{F}[\omega]$
Convolution (Section 2.2)	$x(t) * y(t)$	$\mathcal{F}[x](\omega)\mathcal{F}[y](\omega)$

Figure 3: Properties of Fourier transform

Linearity and shift invariance Operations on signals are referred to as *filters*. A simplified type of filter–linear, shift invariant–turns out to be adequate for our antialiasing purposes. A filter $\mathcal{L}[s]$ of this class (operating on a signal s) has the following properties:

- Linearity: for signals x, y, and real-valued α:

$$\mathcal{L}[x + y] = \mathcal{L}[x] + \mathcal{L}[y]$$

$$\mathcal{L}[\alpha x] = \alpha\mathcal{L}[x]$$

- Shift invariance: if $\hat{x}(t) = x(t - \tau)$ and $\hat{\mathcal{L}}(t) = \mathcal{L}[t - \tau]$, then the following relation holds.

$$\mathcal{L}[\hat{x}] = \hat{\mathcal{L}}[x]$$

  i.e. the result of application of a filter to a shifted signal is the shifted version of the result that we get if we apply the same filter to the original signal.

For the rest of this handout we will use "filter" only to refer to linear and shift-invariant filters.

Band-pass filtering example For an example of filtering, consider the analog telephone, which is supposed to transmit audio frequencies in the range of 300 to 3000 Hz. We could represent the sound of a voice on the telephone as:

$$T(\omega) = H(\omega)V(\omega)$$

Where V is the Fourier transform of the original voice, T is the Fourier transform of the voice on the telephone, and H is the following *frequency response*:

$$H(\omega) = \begin{cases} 1 & \text{if } 300 \cdot 2\pi \text{ Hz} \leq |\omega| \leq 3000 \cdot 2\pi \text{ Hz} \\ 0 & \text{otherwise} \end{cases}$$

Frequency response; impulse response A filter, like a signal, may be represented either in the frequency domain, in terms of its *frequency response*, or in the time domain, by its *impulse response*. As with a signal, the forward and inverse Fourier transforms convert between representations.

We can express filtering in the time domain rather than the frequency domain as an operation known as *convolution*. The expression for the convolution on two signals $x(t)$ and $y(t)$ is

$$u(t) = \int_{s=-\infty}^{+\infty} x(s)y(t - s)ds$$

It is usually written $u = x * y$.

In the frequency domain, filtering is simply multiplication by a function. In the time domain, it becomes convolution. Conversely, multiplication by a function in the time domain becomes convolution in the frequency domain.

2.3 Discrete signals

So far we were discussing only continuous signals. If we are going to process our signal digitally, we will have to deal with *discrete signals*, that is, signals defined only at isolated points. We will consider signals defined at uniformly spaced points of the real line nT, where n is an integer, T is the sampling period. We will use the notation $x[n]$ for such signals.

An important question is how to represent the sampled signal in such a way that we can use the machinery of the Fourier transform. In order to use the continuous Fourier Transform, we need a signal defined on the real line, not on a set of discrete points. The naive approach doesn't work: if we define the signal $x_{sampled}(t) = x[n]$ at points nT and zero elsewhere, the Fourier integral of this signal will be zero.

It turns out that we can obtain a convenient representation of a discrete signal if we use the *comb* function. We represent the discrete signal $x[n]$ with

$$\sum_n x[n]\delta(t - nT)$$

Then sampling a continuous function $x(t)$ is equivalent to multiplying $x(t)$ by the *comb* function $x[n] = x(nT)$ is represented by

$$x(t)\sum_n \delta(t - nT) = \sum_n x(t)\delta(t - nT) = \sum_n x(nT)\delta(t - nT)$$

We will call a signal represented in this manner an *impulse train.*

Intuitively, the impulse train representation can be interpreted as follows: an approximation to a δ-function is a spike of a small width a and the height $\frac{1}{a}$, with area 1; the area under the plot of a spike can be interpreted as the *energy* of the spike. If we multiply the spike by a number c, the energy becomes c. Thus, we can think of the impulse trains as idealizations of uniform sequences of spikes with varying energy.

2.4 Fourier transform of discrete signals

A reason for using the impulse train representation of discrete signals is that now we can apply the CFT to the signal:

$$\mathcal{F}[\sum x[n]\delta(t - nT)] = 2\pi/T \int \sum x[n]\delta(t - nT)e^{-i\omega t}dt = \sum x[n]e^{-i\omega nT} \qquad (1)$$

Alternatively, we could define the FT of a discrete signal to be the sum above. The impulse train allows us to establish a connection between these two types of Fourier transforms.

Note that the FT in Equation 1 is periodic in ω with period $\frac{2\pi}{T}$, because $e^{-i\omega nT} = e^{-i(\omega + \frac{2\pi}{T})T}$. That means that we have to consider FT of discrete signals only on the interval $[-\pi/T, \pi/T]$. The number $\frac{2\pi}{T} = \Omega_s$ is called *sampling frequency.*

We can derive a number of properties of the FT of discrete signals from the properties of the CFT. A list of such properties is shown in the table below.

2.5 Shannon sampling theorem

An interesting question is when the FT of a continuous signal $x(t)$ coincides with the FT of the discrete signal on $[-\Omega_s/2, \Omega_s/2]$. FT has all the information about the signal, and if we can recover the FT of the continuous signal from the FT of the sampled version, we can recover the continuous signal from the samples exactly.

We can answer this question using the convolution theorem. The sampled signal $x_{sampled}(t)$ is just the product of the *comb* function with the continuous function $x(t)$; therefore,

Name of the property	signals	Fourier transforms
Linearity	$ax[n] + by[n]$	$a\mathcal{F}[x](\omega) + b\mathcal{F}[y](\omega)$
Time shifting	$x[n - n_0]$	$e^{-i\omega n_0 T}\mathcal{F}[x](\omega)$
Frequency shifting	$e^{-i\omega_0 nT}x[n - n_0]$	$\mathcal{F}[x](\omega - \omega_0)$
Frequency-domain differentiation	$nx[n]$	$i\frac{d\mathcal{F}[x]}{d\omega}(\omega)$
Convolution	$x[n] * y[n]$	$\mathcal{F}[x](\omega)\mathcal{F}[y](\omega)$

Figure 4: Properties of the Fourier transform of discrete signals

$$\mathcal{F}[x_{sampled}] = \mathcal{F}[x\ comb] = \frac{1}{2\pi}\mathcal{F}[x] * \mathcal{F}[comb] =$$
$$\frac{1}{2\pi}\mathcal{F}[x] * \Omega_s comb(\omega) = \frac{1}{2\pi}\mathcal{F}[x] * \Omega_s \sum_{k=-\infty}^{+\infty} \delta(\omega - k\Omega_s) =$$
$$\frac{1}{T}\sum_{k=-\infty}^{+\infty} \mathcal{F}[x] * \delta(\omega - k\Omega_s)$$

As the convolution with $\delta(\omega - k\Omega_s)$ is just a shift by Ω_s, we can see that the Fourier transform of the sampled signal is just a sum of shifted copies of Fourier transforms of the continuous signals:

$$\mathcal{F}[x_{sampled}] = \frac{1}{T}\sum_{n=-\infty}^{+\infty} \mathcal{F}[x](\omega - n\Omega_s)$$

Equation 1 gives a different representation:

$$\mathcal{F}[x_{sampled}] = \sum_{n=-\infty}^{+\infty} x[n]e^{-i\omega nT}$$

It is quite remarkable that an infinite sum of shifted copies of the FT of a continuous function turns out to be just a sum of exponents multiplied by discrete samples of the signal.

This fact has some important consequences:

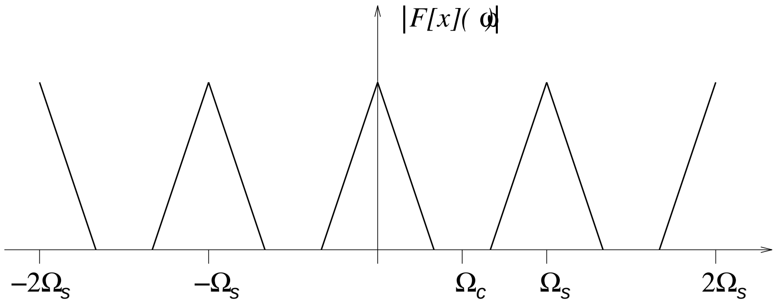
We can immediately see when the FT of the sampled signal coincides with FT of the original signal on $[-\Omega_s/2, \Omega_s/2]$: this happens only when there is no overlap between the shifted copies of $\mathcal{F}[x]$. In particular, there is no overlap if the continuous signal $x(t)$ is *band-limited*, that is, the Fourier transform of the signal is 0 for all frequencies above some frequency Ω_c. Figure 5 shows that there is no overlap if $\Omega_c < \Omega_s - \Omega_c$, that is, if $\Omega_c < \Omega_s/2$. The frequency $\Omega_s/2$ is called the *Nyquist frequency.*

If a signal $x(t)$ is band-limited below the Nyquist frequency, then the Fourier transform of the sampled signal consists of copies of $\mathcal{F}[x]$ centered at $k\Omega_s$; we can use a low-pass filter with cutoff frequency $\Omega_s/2$ to get rid of extra copies. If we use the ideal low pass filter *sinc*, we will recover the original signal exactly. This result is called *Shannon theorem* or *Nyquist theorem*:

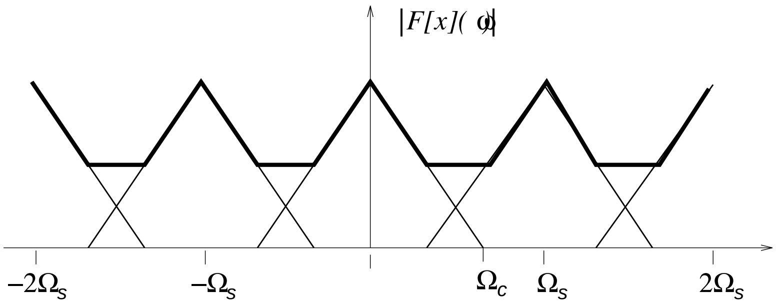
Theorem 1 *(Shannon,Nyquist) Let $x(t)$ be a band-limited signal with*

$$\mathcal{F}[x](\omega) = 0 \quad \text{for } |\omega| > \Omega_c = \Omega_s/2$$

Then $x(t)$ is determined exactly by its samples $x[n] = x(nT)$, where $T = \frac{2\pi}{\Omega_s}$.



Spectrum of a sampled signal with $\Omega_c < \Omega_s/2$



Spectrum of a sampled signal with $\Omega_c > \Omega_s/2$

Figure 5: Sampling

We can derive an explicit formula for reconstructing the signal by convolving appropriate *sinc* function with the sampled function $x_s(t) = x(t)comb(t)$. The signal $x(t)$ can be reconstructed from samples $x[n]$ using

$$x(t) = \sum_{n=-\infty}^{+\infty} x[n] \frac{\sin \pi(t - nT)/T}{\pi(t - nT)/T}$$

2.6 Discrete convolution

If we convolve an impulse train $\sum x[n]\delta(t - nT)$ with another impulse train $\sum y[n]\delta(t - nT)$, the result will be

$$\sum_{n,m} x[m]y[n]\delta(t - mT) * \delta(t - nT) = \sum_{n,m} x[m]y[n]\delta(t - (m + n)T) \quad =$$
$$\sum_{k}(\sum_{n+m=k} x[m]y[n])\delta(t - kT) = \sum_{k}(\sum_{n} x[k - n]y[n])\delta(t - kT)$$

We get a new impulse train which corresponds to a discrete signal

$$(x * y)[k] = \sum_{n} x[k - n]y[n] \qquad (2)$$

which is called the *discrete convolution of x and y* Given that the signals $x[n]$ and $y[n]$ have finite length, this is an operation that we can perform on a computer. As it is just the regular convolution applied to impulse trains, the Convolution Theorem holds:

Theorem 2 *If $x[n]$, $y[n]$ are discrete signals and $(x * y)[n]$ is their convolution,*

$$\mathcal{F}[x * y] = \mathcal{F}[x]\mathcal{F}[y]$$

Now we have all tools in place for the applications that we are interested in: resampling of a discrete signal at a different set of points and filtering the signal in order to modify its properties.

2.7 Sampling and reconstruction pipelines

Sampling The following pipeline summarizes the process of sampling. Given a sampling rate, Shannon's theorem gives the cutoff frequency of the prefilter h.

$$\begin{array}{cl} \text{Original signal} & \\ x(t) & \\ \downarrow & \quad\text{Prefiltering} \\ \text{Band-limited signal} & \\ x(t) * h & \\ \downarrow & \quad\text{Sampling} \\ \text{Impulse train} & \\ \sum_i x_i \delta(t-i) & \\ \downarrow & \quad\text{Discretization} \\ \text{Sequence} & \\ x[n] & \end{array}$$

Figure 6: Sampling

As described above, if the original signal has been passed through an ideal low-pass filter during sampling, then an *identical filter* h may be used to reconstruct the signal.

$$\begin{array}{cl} x[n] & \\ \downarrow & \quad\text{Inverse discretization} \\ \text{Impulse train} & \\ \hat{x}(t) = \sum_i x_i \delta(t-i) & \\ \downarrow & \quad\text{Postfiltering} \\ \hat{x} * h & \end{array}$$

Figure 7: Reconstruction

3 Shifting images

The first image manipulation we consider is shifting an image by a non-integer number of pixels. We have to choose an interpolation scheme for filling in pixels in the target image when they do not line up exactly with source-image pixels. The scheme described here is based on sampling theory, and in an ideal implementation would completely eliminate aliasing due to shifting.

The aliasing we have looked at up to now has all been due to *sampling*. How does sampling affect the *shifting* of images? If we had access to the original continuous function from which an image was generated, we could shift that function by an arbitrary amount, and *resample* it. This would be an ideal shift operation. However, we often have access only to an already discretized image. In that case, we could reconstruct the continuous function as well as possible, shift it, and resample it.

3.1 Combination of filtering operations

This illustrates the basic idea of antialiased shifting, but doesn't indicate how it may be done efficiently on a digital computer. First, note that a bandlimited signal, after shifting, is still bandlimited. Therefore the prefiltering stage in resampling is unnecessary. This is because shifting is a linear, time-invariant operation. If the sampling and

$$
\begin{array}{cl}
\text{Source sequence} & \\
x[n] & \\
\downarrow & \text{Reconstruction (Figure 7)} \\
\hat{x} * h & \\
\downarrow & \textit{Shifting} \\
y(t) = \hat{x} * h(t - \tau) & \\
\downarrow & \text{Resampling (Figure 6)} \\
\text{Target sequence} & \\
y[n] &
\end{array}
$$

Figure 8: Reconstruction, shifting, and resampling

reconstruction filters are given by H, and the shift operator is given by S, the continuous part of the pipeline in Figure 8 may be expressed as:

$$Y(\omega) = H(\omega)S(\omega)H(\omega)\hat{X}(\omega) = S(\omega)H^2(\omega)\hat{X}(\omega)$$

If H is an ideal low-pass filter, then it is everywhere zero or one, so that $H^2 = H$.

$$Y(\omega) = S(\omega)H(\omega)\hat{X}(\omega)$$

Although, as discussed in Section 5.1 below, practical implementations use non-ideal low-pass filters, they still leave out the second low-pass filtering stage.

$$y(t) = (\hat{x} * h)(t - \tau)$$

3.2 Discrete implementation

By applying discrete convolution techniques (Section 2.6) to the problem, we obtain the following expression in terms of pixels and continuous filter kernels:

$$y[n] = \sum_{m=-\infty}^{\infty} x[m]h(T(n - m) - \tau)$$

4 Enlargement and reduction

As in the case of shifting, the way to avoid aliasing during enlargement and reduction is to simulate an operation on a continuous image. First, we will look at what happens to the Fourier transform of a continuous image when it is enlarged or reduced.

4.1 Fourier zoom property

Consider the zoom operator Z_a, in which a is a magnification factor.

$$Z_a[x](t) = x(\frac{t}{a})$$

When $a > 1$, the resulting image is larger by a factor of a. When $0 < a < 1$, it is smaller by a factor of a^{-1}. A larger image will contain lower frequencies, and vice versa. We can see why this is true by evaluating the Fourier transform of $Z_a[x]$:

$$\mathcal{F}[Z_a[x]] = \int_{-\infty}^{+\infty} Z_a[x](t)e^{-i\omega t}dt$$

$$= \int_{-\infty}^{+\infty} x(\frac{t}{a}) e^{-i\omega t} dt$$

With a change of variables $\tau = t/a$, $d\tau = dt/a$,

$$= \int_{-\infty}^{+\infty} x(\tau) e^{-i(a\omega)\tau} d\tau a$$

$$= a Z_{1/a}[\mathcal{F}[x]]$$

4.2 Reconstruction, zooming, and resampling

The ideal antialiased zoom operation on a sampled image would involve reconstruction of the continuous signal, continuous zooming, and resampling.

$$\begin{array}{cl} \text{Source sequence} & \\ x[n] & \\ \downarrow & \text{Reconstruction (Figure 7)} \\ \hat{x} * h & \\ \downarrow & \textit{Zooming} \\ y(t) = \hat{x} * h(\frac{t}{a}) & \\ \downarrow & \text{Resampling (Figure 6)} \\ \text{Target sequence} & \\ y[n] & \end{array}$$

Figure 9: Reconstruction, zooming, and resampling

Combination of filtering operations As in the case of image shifting, the two filtering operations–postfiltering for reconstruction, and prefiltering for resampling–can be combined into one. However, the zooming operation changes the frequency content of the signal, so the cutoff frequencies of the two filters will not match. The operations in Figure 9 may be expressed as:

$$Y(\omega) = aH(\omega) Z_{1/a}[H\hat{X}](\omega) = aH(\omega)H(a\omega)\hat{X}(a\omega)$$

By looking at the product of the two filters we can see why the pair will have a joint cutoff of $\min(\omega_c/a, \omega_c)$:

$$H(a\omega) = \begin{cases} 1 & \text{if } |\omega| \leq \omega_c/a \\ 0 & \text{otherwise} \end{cases}$$

$$H(\omega)H(a\omega) = \begin{cases} 1 & \text{if } |\omega| \leq \omega_c/a \text{ and } |\omega| \leq \omega_c \\ 0 & \text{otherwise} \end{cases}$$

The reason for the min operation is that the zoom process is more constrained by whichever image has lower resolution. When $0 < a < 1$, the image is being shrunk, and the target image resolution is the limiting factor. Because the target image has fewer pixels than the source, there are fewer degrees of freedom in which to encode the signal. Therefore unless we do something very tricky, information will be lost. The source image resolution is limiting when $a > 1$.

$$Y(\omega) = aH(\max(\omega, a\omega))\hat{X}(a\omega)$$

4.3 Discrete implementation

The following equations, obtained by applicaton of discrete convolution (Section 2.6), describe how antialiased zooming may be performed directly on discrete images.

$$A = \min(a, 1)$$

$$y[n] = A \sum_{m=-\infty}^{\infty} x[m] h(AT(\frac{n}{a} - m))$$

where T is the sampling interval of $x[n]$.

5 Implementation considerations

5.1 Choice of low-pass filters

In the previous sections we have ignored the problem of implementing a filter with an infinite length. In fact, in most cases it is not possible to implement an infinite filter unless it has a very special form. In particular, it is not possible to implement the ideal low-pass filter. All we can do is to make approximations to it, choosing finite filters with a similar frequency response. Fortunately, our eye is not that sensitive to small amounts of aliasing and we can get away with rather simple approximations for images. However, we should be careful when we choose a filter: some of them produce better results than others. For example, it turns out that the obvious choice–truncated *sinc* filter–generates excessive ringing. A number of filters produce adequate results. A classical example is the Lanczos filter

$$h(t) = \begin{cases} sinc(t) sinc(t/3) & \text{if } |t| < 3 \\ 0 & \text{otherwise} \end{cases}$$

Another example is a cubic filter proposed by D. Mitchell:

$$h(t) = \begin{cases} 7|t|^3 - 12|t|^2 + \frac{16}{3} & \text{if } |t| < 1 \\ -\frac{7}{3}|t|^3 + 12|t|^2 - 20|t| + \frac{32}{3} & \text{if } 1 < |t| < 2 \\ 0 & \text{otherwise} \end{cases}$$

The frequency response of the first filter is somewhat better, but it is more expensive to compute than the second one.

Note that when you use a filter, it is desirable to scale it in such a way that when you apply it to a constant, you get the same constant. This is achieved by a scaling factor equal to the reciprocal of the integral of the filter. In the case of Mitchell's filter the scaling factor should be 1/6.

6 Two-dimensional convolution

All the theory that was developed above for one-dimensional signals can be generalized to two dimensions and applied to images. In most cases, we will treat each dimension separately: if we have a two-dimensional signal $f(x, y)$ we can first apply FT for the variable x keeping y fixed, then for y and obtain a function of two frequency variables. Unfortunately, not all intersting things that can be done in two dimensions can be decomposed into a sequence of two one-dimensional operations.

Two-dimensional discrete convolution can be defined as

$$z[n1, n2] = \sum_{k_1=-\infty}^{+\infty} \sum_{k_2=-\infty}^{+\infty} x(k_1, k_2) y(n_1 - k_1, n_2 - k_2)$$

Two-dimensional convolution can be represented as a sequence of two one-dimensional convolutions only if one of the signals is *separable*, that is if $x[n_1, n_2] = x_1[n_1] x_2[n_2]$ for some one-dimensional signals $x_1[n_1]$ $x_2[n_2]$.

7 Edge detection

We can use filters to approximate various operations on continuous functions. For example, a possible approximation for a derivative at a point nT for a function $f(x)$ is $\frac{1}{T}(f[n+1] - f[n-1])$ Note that this is the convolution of $f[n]$ with $\frac{1}{T}h[n]$ which is defined by

$$h[-1] = 1, \quad h[1] = -1, \quad h[n] = 0 \text{ for all other n} \tag{3}$$

Approximations of the derivative can be used to detect edges in a picture.

We can define an edge point as a point where the function $f(x)$ (intensity in the case of images) changes faster than at the points next to it. That means that the derivative $f'(x)$ has a maximum at this point. (We have assumed that the function is differentiable.) Thus we can find the edges of a function by looking at the local maxima of the derivative.

In two dimensions, the analog of the derivative is the gradient vector $(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y})$. The edges can be defined as points where the magnitude of the gradient is above some threshold.

We approximate the directional derivatives $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ in a way similar to (3). To make the method less sensitive to fluctuations in the images, we can average the derivative in perpendicular direction. This leads to the following filter for the horizontal derivative:

$$\begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix}$$

In the matrix above, the center corresponds to $h[0,0]$.

A similar filter can be used for the vertical derivative:

$$\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

To find the edges in an image, we compute the convolutions with each of these filters and estimate the magnitude of the gradient at each point

$$\sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

.

and compare it to he chosen threshold. The choice of threshold depends on the image.