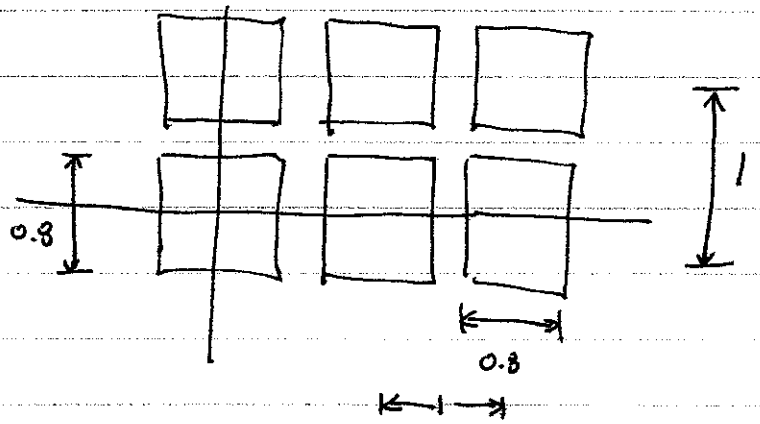
Undersampling

Our pictorial - transform view of sampling also allows us to understand what happens in the case of undersampled data. We were able to isolate the correct set of (u,v) data in the previous example because the bandwidth of the signal was less than the replication offset of the multiple islands in the transform domain. More exactly, since the islands were spaced at integer increments, and each island was 0.8 units across, it was easy to separate the islands:



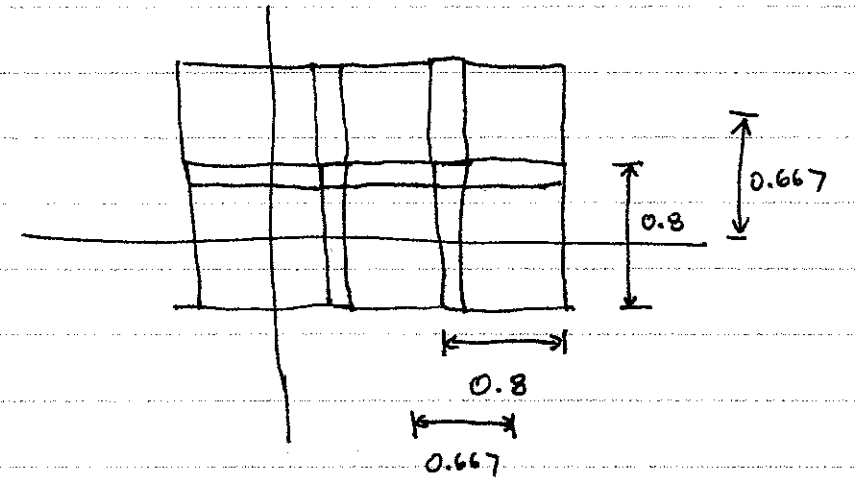
Suppose instead of sampling by ${}^2\Pi\Pi(x,y)$, we sampled every 1.5 units instead of every integer value. Then the sampled function $s(x,y)$ would be:

$$s(x,y) = {}^2\Pi\Pi\left(\frac{x}{1.5}, \frac{y}{1.5}\right) f(x,y) \frac{1}{1.5^2}$$

Then the spectrum $S(u,v)$

$$S(u,v) = F(u,v) ** {}^2\Pi\Pi(1.5u, 1.5v)$$

Here the islands are separated by $\frac{2}{3}$ of a sample, and overlap occurs!



So only the values in the central part of the islands can be recovered exactly. If we tried to invert the entire island the coefficients near the edges will be incorrect.

However, one thing we could do would be to exclude the outer, overlapped region and inverse transform only the unambiguous area. So some portion of $f(x,y)$ could be recovered. Excluding the outer region may, for example, be the high-frequency information, so the reconstruction would be of the lower-frequency part. The result would then be in some sense a "smoothed" version of the original function.

Aliasing

Let's say we have an undersampled signal but we use it as if it were not undersampled. We would have a set of overlapping islands, multiplied by rect( ), and inverse transformed. We can visualize what is happening with this picture:

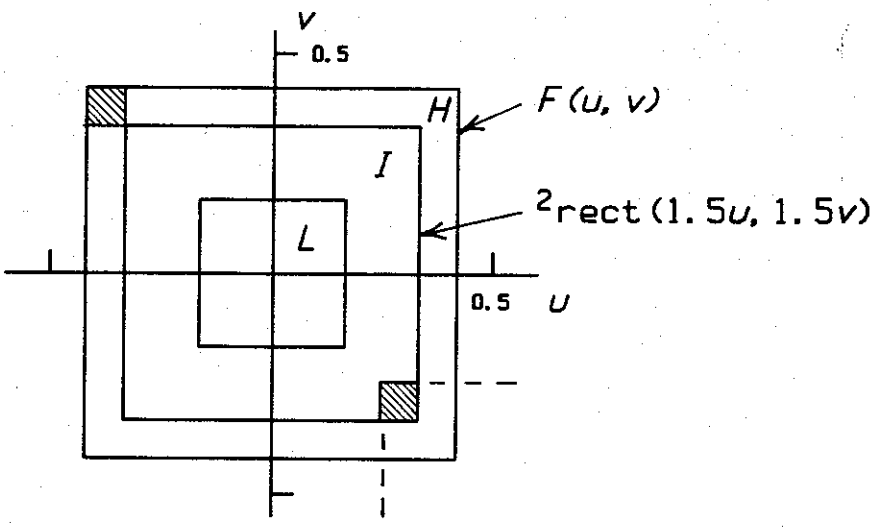


Figure 7-12 The band limit of $F(u, v)$ (outer square) contains an area L of low frequencies that have been preserved, an area H of high frequencies rejected by filtration, and an area I of intermediate frequencies contaminated by contributions from overlapping islands such as the one indicated in broken outline.

In this image, the low frequencies (L) are preserved, high frequencies (H) are eliminated, and intermediate frequencies (I) are contaminated. The high frequencies are not lost because of the replicated nature of the aliasing in the frequency domain, but appear in the incorrect location in the intermediate band. Hence high frequency components are shifted to lower frequency and appear in the output image.

Prefiltering

Since aliasing results mainly in corruption of higher frequency terms of a signal, sometimes we can prefilter the input so that only the low-frequency parts are sampled and transmitted by the system. This results is, again, a low-frequency smoothed version of the original that is, however, transmitted accurately.

Example: Telephone Signal, 3 Khz bandwidth.

Circular Cutoffs

Changing sampling patterns can allow us to pack more signals in a given channel if we understand the nature of the cutoffs. Suppose, for example, that we use an image with a circular passband rect(r) rather than the square used previously. We might view two ways of packing the signals such as:

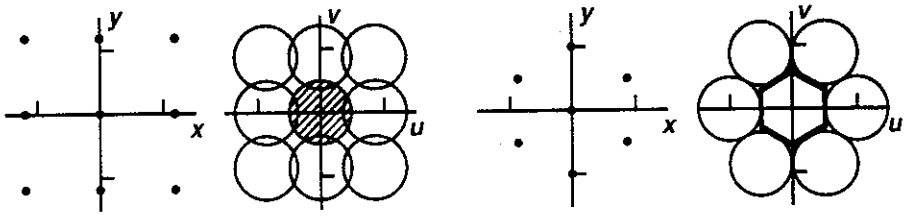


Figure 7-13 A square sampling pattern (left) with spacing 1.128 applied to a function with circular cutoff at spatial frequency 0.5 (grey circle) and the corresponding overlapping islands of spacing 0.886 in the (u, v)-plane. A hexagonal sampling pattern exists (right) that involves no overlap.

The sampling pattern in frequency we have shown to be simply the transform of the sampling pattern in space, which we relate through the convolution theorem. So by choosing the correct spatial sampling function we can pack the circular passbands in a hexagonal pattern, and avoid undersampling. Equivalently, we can increase available bandwidth.

Double-rectangle passband

Another non-square passband we encounter from time to time is the double-rectangle pattern, found for example in telescope interferometers. We could pack these side by side for insurance against aliasing, but a cleverer sampling scheme might be to do something like the following:

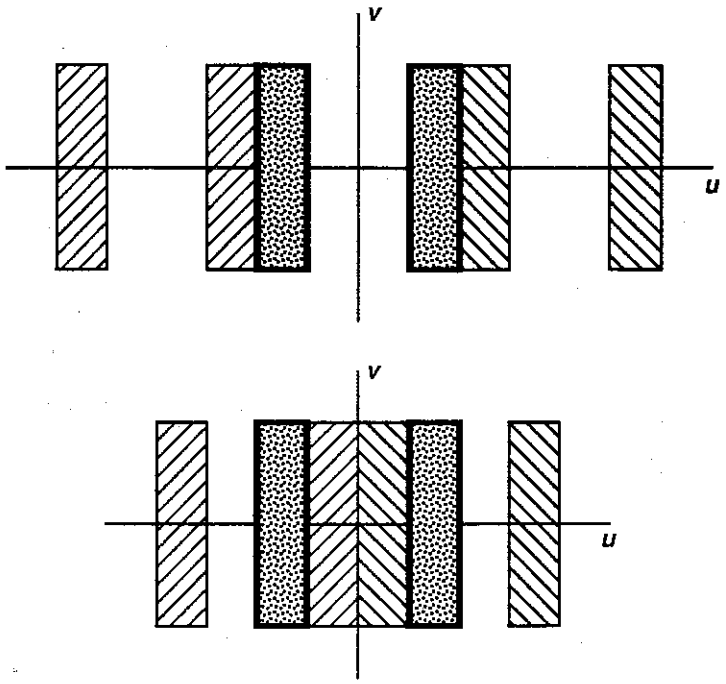
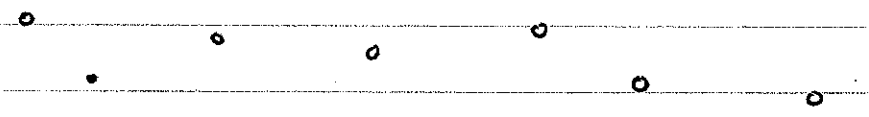


Figure 7-14 A region (shown stippled) lying entirely within unit square.

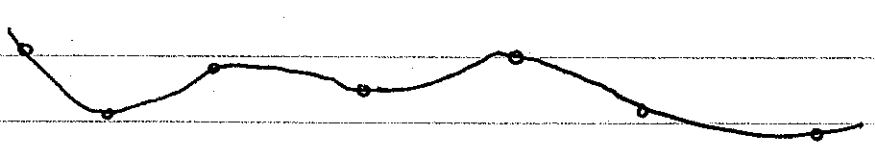
Interpolation between samples

We have shown already it is possible to reconstruct a signal from discrete samples, provided the samples are dense enough. How do we actually implement interpolation? That is, how do we estimate the value of a function between known points.

Say we have the following set of points:



We can draw a smooth curve between them:



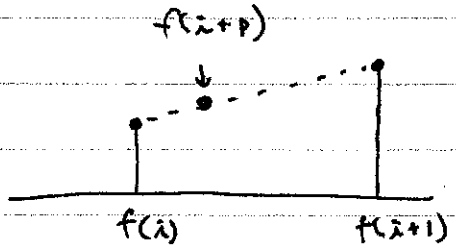
But how do we know the answer is right? And how would we do this in two dimensions? Many algorithms exist.

Nearest-neighbor algorithm

Suppose we have a sequence $f(i)$ and want to find the value at $f(i+p)$, $0 < p < 1$. The simple, "natural" thing to do might be to interpolate linearly:

$$f(i+p) = (1-p)f(i) + p f(i+1)$$

Graphically,



Sometimes we need a faster algorithm which still retains the correct value at the sample points. This might be done with the nearest-neighbor algorithm:

$$\text{if } p \le \tfrac{1}{2}, \quad f(i+p) = f(i)$$

$$\text{if } p > \tfrac{1}{2}, \quad f(i+p) = f(i+1)$$

This algorithm uses only logical operations, not computations, and is much faster than linear interpolation. A quicker method of writing the algorithm might be:

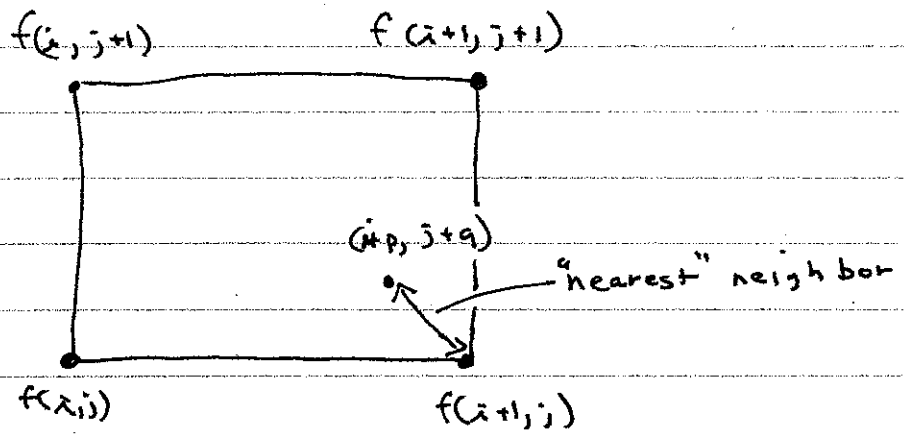
$$f(i+p) = f(\text{nint}(i+p))$$

where nint() is the nearest integer value.

The extension to 2-D is straightforward for this algorithm:

$$f(i+p, j+q) = f(\text{nint}(i+p), \text{nint}(j+q))$$

We can picture this as follows:



Twisted-plane (sometimes called bilinear) interpolation

So what do we do if we need more accuracy than nearest-neighbor interpolation allows? In one-D, we had a simple linear interpolation scheme. Its 2-D analog is:

$$f(i+p, j+q) = (1-p)(1-q)\,f(i,j) + p(1-q)\,f(i+1, j) + q(1-p)\,f(i, j+1) + pq\,f(i+1, j+1)$$

We could visualize linear interpolation in 1-D as a string stretched between points on the line, in 2-D the model is a sheet tacked at the four corners:

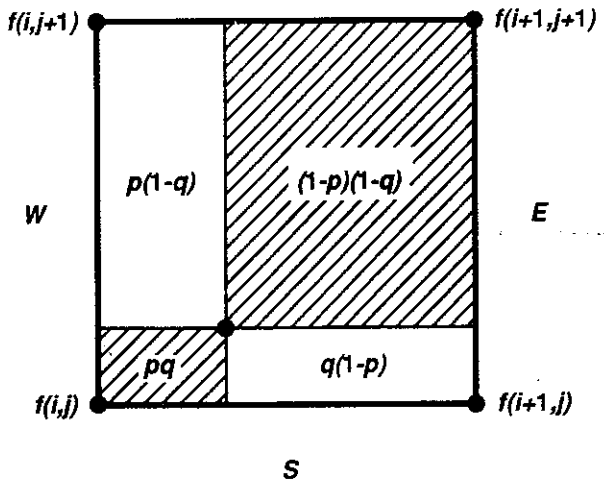


Figure 7-15 The "twisted plane" through the four corner samples has a height at $(i + p, j + q)$ that is a weighted mean of the corner heights; the weight for any corner is equal to the area opposite that corner.

We can also think of bilinear interpolation as follows: First implement 2 one-d linear interpolations of $f(i,j)$ and $f(i+1,j)$ to get $f(i+p,j)$. Similarly obtain $f(i+p,j+1)$. Then do an orthogonal linear interpolation between $f(i+p,j)$ and $f(i+p,j+1)$ to get $f(i+p,j+q)$.

Twisted plane in 2-D is actually more accurate than simple 1-D linear interpolation because the slopes are measured at two locations in each direction, rather than just one.

Six-point formula

If curvature is significant, we need to consider the second derivatives $\frac{\partial^2}{\partial x^2}$ and $\frac{\partial^2}{\partial y^2}$, or, equivalently, the second differences. A common approach is to add the two points located at $(i-1,j)$ and $(i,j-1)$. The resulting 6-point formula is

~~$f(i+p,j+q) =$~~

$$f(i+p,j+q) = \tfrac{1}{2}q(q-1)f(i,j-1) + \tfrac{1}{2}p(p-1)f(i-1,j) + (1+pq-p^2-q^2)f(i,j) + \tfrac{1}{2}p(p-2q+1)f(i+1,j) + \tfrac{1}{2}q(q-2p+1)f(i,j+1) + pq\,f(i+1,j+1)$$

Cubic-splines

When it is more important to have a smooth curve than it is to have absolute accuracy, we may choose ~~to~~ spline interpolation to fit a smooth curve between points. In two-D this means less "noise" in the high-frequency part of the spectrum. A popular smoothing polynomial is of order 3, it uses data at four points to interpolate between the innermost 2 points:

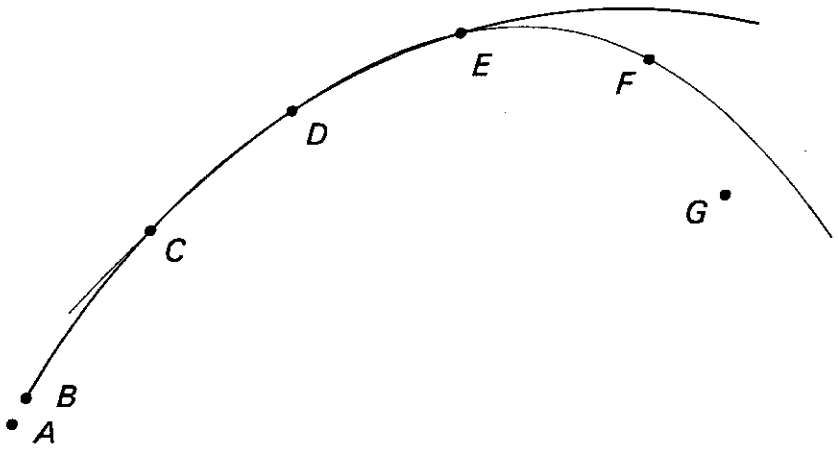


Figure 7-16 Cubic spline interpolation.

This algorithm has the advantage of smoothness, reasonable accuracy, and relatively few computations.

Midpoint Interpolation

Often we can formulate an interpolation requirement in terms of obtaining the midpoints in a sequences - clearly this can be extended to permit multiple passes aimed at obtaining solutions spaced by $\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \ldots$. This allows us, say, to zoom an image by a factor of 2, or permits us to faithfully reproduce the spectrum of the product of two bandlimited signals.

In two dimensions, sampling along midpoints in one dimension does not depend on behavior in the other dimension. This follows from the notion that if a two-D function is bandlimited, any one-D cut through it will also be bandlimited. If the samples are adequately spaced along the cut, any intermediate value may be determined from those samples alone. This may be seen by considering the image as a superposition of corrugations - clearly each corrugation may be interpolated along a cut only considering value along the cut. Doing this for each corrugation

frequency and direction means we can do the same to the total function.

We can represent linear midpoint interpolation by the following notation:

$$\{\ 0.5 \ \uparrow \ 0.5\ \}$$

which means the value at the arrow is equal to the sum of the points on either side divided by 2. Looking at higher moments and differences, if we need more accuracy we could use

$$\frac{1}{16}\{-1 \quad 9 \uparrow 9 \quad -1\}$$

or

$$\frac{1}{256}\{3 \quad -25 \quad 150 \uparrow 150 \quad -25 \quad 3\}$$

These sequences have unit sum, and the 1st, 2nd, and 3rd order moments are zero, respectively, as well as lower moments (other than 0, of course).

Sinc interpolation

For band-limited functions, sinc interpolation is very common. It is defined according to:

$$f(x,y) = \sum_i \sum_j s(i,j)\ \text{sinc}(x-i, y-j)$$

Here we can choose the range of i, j in the sum depending on our requirements for accuracy. For approximate work, perhaps only four points are needed in each dimension, for greater accuracy more points may be taken. Since sinc() falls off

as $1/x$, The power associated with each coefficient falls as $\frac{1}{x^2}$ so increasing coefficients rapidly increases accuracy.