Jim Blinn's Corner

Return of the Jaggy

James F. Blinn, Caltech

In our last episode the evil high frequencies from the Aliasing Empire were sneaking into our image and transforming themselves into deadly Jaggies. Our research labs had come up with a theoretical weapon to defeat the Jaggies. Unfortunately, the purely theoretical solution cannot be used in practice. This time I will discuss why this is so, and give some ideas about how close we can come.

To best understand this process you need some facility for thinking in the frequency domain as well as the spatial domain. So, like last month's column, this one will largely be a picture show to help you build your intuition.

Simple filters

Let's remember the punch line of the theory from last time:

> Before sampling an image, you must first filter out the high frequencies. To view (reconstruct) the sampled image, you must interpolate between the samples (i.e., filter them).

Filtering is the same as taking a weighted average of the intensity function. This weighting function is sometimes called an "impulse response" or a "point-spread function." You center the PSF at a pixel, multiply by the continuous-intensity function, and integrate the result. This number becomes the filtered value to display at that pixel.

Let's look at some functions commonly in use and their effects. You can judge the effectiveness of a particular PSF by examining its Fourier transform: In Figure 1 a PSF is on the left and its Fourier transform is on the right. Notice that each PSF is scaled so the total integral under it equals 1. This is done so that filtering a constant-intensity image gives back the same intensity.

0272-1716/89/0300-0082$01.00 1989 IEEE

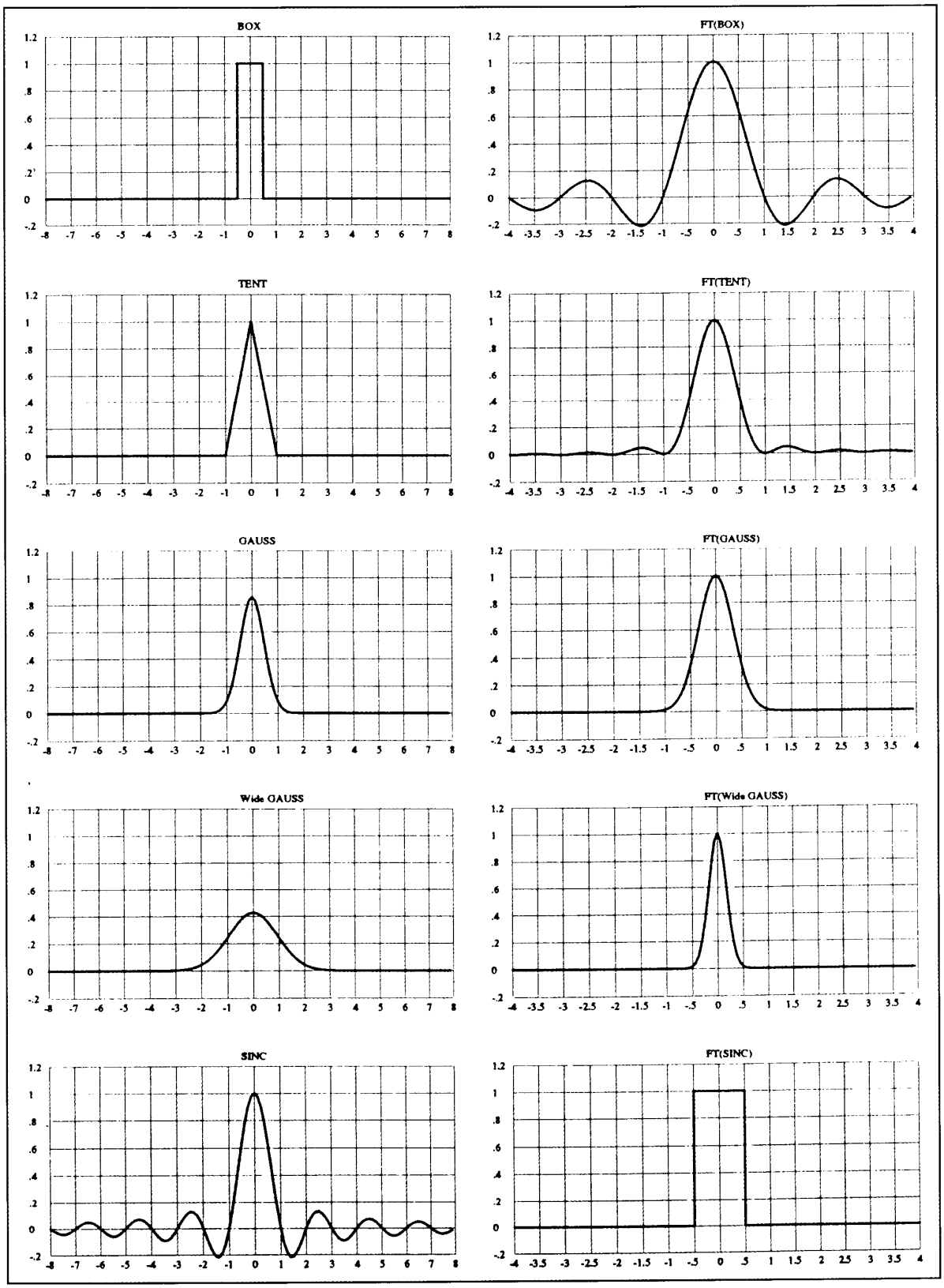

Figure 1. Filters.

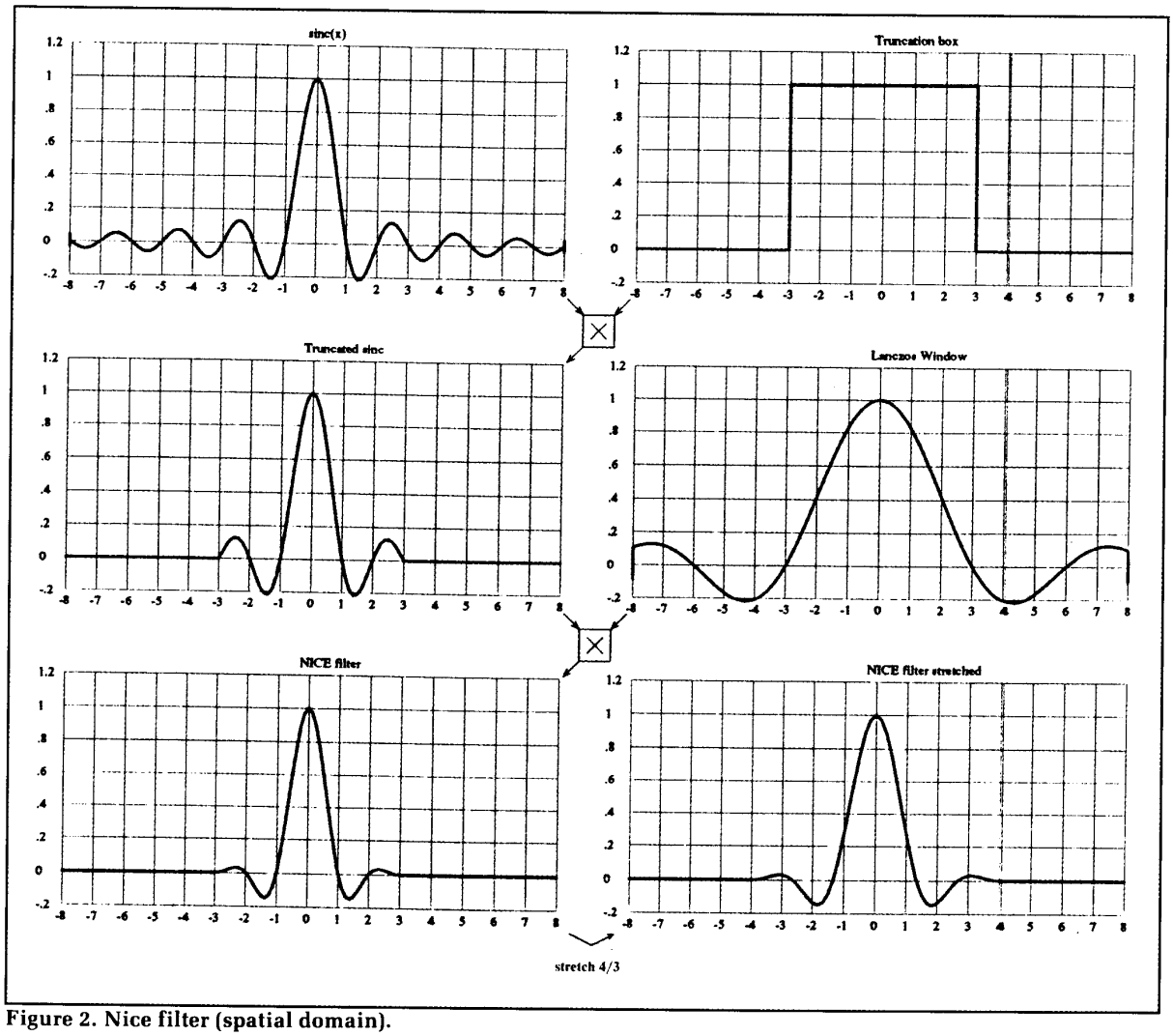

Figure 2. Nice filter (spatial domain).

The box filter

The box filter is the most common function and the easiest to implement; just average the intensities over the area from −.5 to +.5 of the pixel center. When textbooks discuss finding the amount of a pixel covered by an edge, this is the filter they are using. Frank Crow came up with a clever algorithm for implementing this filter.[1] The problem is that the filter is a pretty crummy approximation to what we really want. Its Fourier transform shows that many high frequencies can slip through relatively unscathed. The FT even hits zero twice as far from the origin as we want (at ±1 instead of ±.5). It also attenuates some nice friendly frequencies in the range .3 to .5, producing the visual effect of excessive blurring.

It may be bad (purely coincidentally, it's just the Fourier transform of what we want), but it's better than doing nothing. High frequencies are reduced; low frequencies are kept. I've gotten away with using it for years.

The triangle or tent filter

The triangle or tent filter is the next easiest to implement. Paul Heckbert[2] and Ken Perlin[3] have described clever ways to expand on Crow's technique to implement both this and the following filter.

It does introduce a new pragmatic difficulty. Notice that a copy of the PSF centered at one pixel will overlap a copy centered at another. The copy applied to a pixel at the edge of the screen sticks out beyond the screen area. You have to be able to calculate intensities a smidgen outside the screen area to calculate the edge pixels properly.

But how good a filter is it? Use your imagination a bit and you can see that the tent is just the convolution of a box with another box. So in the frequency domain the

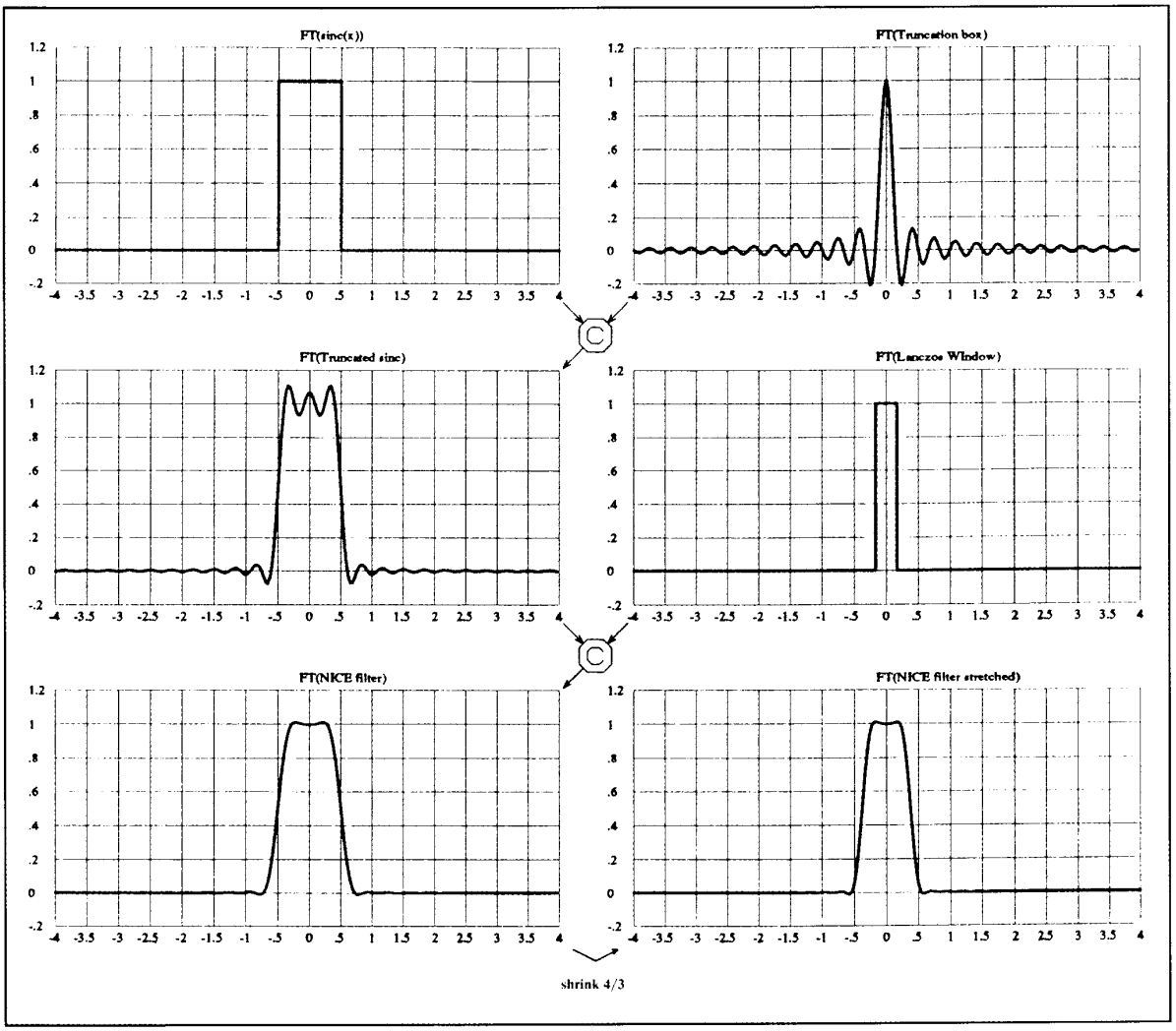

Figure 3. Nice filter (frequency domain).

FT is just the square of the FT of the box. (That's why the convolution business is interesting. It allows us to find some non-obvious FT of a function by building it from simpler functions.) The FT of the tent is better about getting rid of high frequencies than the box, but it still hits zero at too high a frequency and attenuates the nice low frequencies.

Gaussian and similar shapes

If you convolve a tent with another box, you get a smooth bump made of three parabolic segments. Convolve with another box and it gets still smoother. Keep doing it and you approximate a Gaussian normal distribution function. How good a filter is it? The FT of a Gaussian is another Gaussian function. The wider the Gaussian for the PSF, the narrower its FT. A pretty wide function gets rid of high frequencies real well, but it also gets rid of nice low ones.

Ideal filter

The theoretically ideal PSF for both filtering and reconstruction is at the bottom of Figure 1. It has the somewhat strange name "sinc" and is defined as

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

So why don't we just use this in the first place? Because it's infinitely wide. And you can't just keep the middle part and slam the rest down to zero either. The next section shows why and what we can do about it.

Building a nice filter

For the following, refer to Figure 2 (spatial domain) and Figure 3 (frequency domain).

Suppose we did just take the middle part out of the sinc function as our PSF. This is the same as multiplying the sinc with a box stretching, say, ±3 pixels around it. In the frequency domain this implies convolving the ideal box frequency response with sinc(3*f*). The middle plot in the left column of Figure 3 shows the result. It's all ripply (believe it or not).

To get rid of the ripples in the frequency response, we can do some sort of area averaging (convolution) to the FT. For example, suppose we convolved the ripply FT with a box function that is the width of one cycle of the ripple. This flattens out the ripples fairly nicely. Meanwhile, back in the spatial domain we have effectively multiplied the original weighting function by sinc(x/3), giving a nice usable PSF. This process is called "windowing." People have used a whole mess of functions for this purpose—functions with names like Bartlett, Hamming, and Hanning. The one we have used is called a Lanczos window. They all represent trade-offs between eliminating ripples and rounding off the corners of the desired frequency response.

Our filter keeps low frequencies and rejects high frequencies better than any (achievable) filter we've seen so far. This is largely because of the negative region or "lobe" in the PSF. People who advocate this type of filter are called negative lobists. (Ask someone at your next cocktail party whether they are a negative lobist and see what happens.)

The FT of this filter has almost a trapezoid shape, with the sloping sides stretching from frequency 1/3 to 2/3. Depending on how paranoid you are about aliasing (see below), you might want to contract it to bring the bottom in to 1/2. This requires shrinking the FT by 4/3, which is the same as stretching the impulse response by 4/3, making it cover 8 pixels. More arithmetic.

Subsampling

It's a real pain to convolve continuous filters with a continuous-intensity function. One of the easiest dodges in the antialiasing game is subsampling. You sample the picture at two or four times the pixel spacing, and then average the subsamples together. Our Fourier transform technique lets us see how well this works.

In the top three rows of Figure 4 we subsample the picture two times (the calculated samples are a half pixel apart) and take weighted averages. The three weighting functions are just the box, tent, and nice filters. Their FTs are shown on the right side of the figure. Note that they keep repeating, with a copy of the basic FT at frequencies of 2, 4, 6, 8, etc. This means that even with the best shape only about half the alias-producing frequencies are eliminated. But we use a lot less arithmetic than with continuous integration.

There is another trade-off here between the extra arithmetic needed to use the nice function and the straight add-and-divide-by-two for the box PSF. Nevertheless, you are going to all the work of calculating subsamples, so you may as well make the best use of them by using a nicely shaped weighting function.

Now look at the bottom row of Figure 4. Here I have plotted the result for a subsampling rate of 4. The effect of higher subsampling rates is to move the copies of the basic FT farther apart. For subsampling at the rate of 4, they are at 4, 8, 12,... cycles per pixel. In general, the shape of the PSF determines the shape of the repeated bumps in the FT. The subsample rate determines the separation of the copies.

The effect of D/A converters

Just when we think we have it all figured out, something new comes in to foul things up. Remember what happens when we go to display our carefully crafted sequence of pixel values. In the theoretical ideal, we convolve the discrete samples with an ideal low-pass filter to turn them back into a continuous signal. You can think of the reconstruction filter as an interpolation technique in this sense.

In the real world we put the samples in a frame buffer and the hardware sends them to a D/A converter to make the analog signal to send to the display. The D/A converter holds a particular sample for 1 pixel time and then the next one for 1 pixel time. Thus it is effectively convolving the samples with a box function. This again is the FT of what we would really like.

For an extreme example of why this causes problems, look at Figures 5 and 6. We start with a picture of a sine wave of frequency 7/16 cycles per pixel. This is below the Nyquist rate, so just sampling produces no aliasing. Everything should be fine. But when we reconstruct with the D/A filter, something awful happens. This beat pattern appears out of nowhere. Blurring it with the Gaussian spot helps, but we still get some crud. Putting this pattern on your display is very amusing: Use a frequency of about .49 cycles per pixel to make it more obvious.

Here we've done all this work to get rid of aliases, and the hardware creates new artifacts out of what we thought were nice friendly frequencies. One solution is to filter out frequencies somewhat below the Nyquist limit. This is another reason for contracting our nice function by 4/3.

Another solution is to modify the display hardware. What should happen (and what does happen with high-quality frame buffers for the broadcast video market) is that a filter contained in the frame buffer corrects for the D/A function. The idea is to design a filter so that the net frequency response of the D/A converter plus filter is close to an ideal low-pass filter. This again means negative lobes. Because computer displays tend to have white 1-pixel-wide lines on black backgrounds, using such a filter might require the production of negative light at some places on the screen.

Summary

There is no such thing as full antialiasing. Anyone who tries to tell you differently is trying to sell you

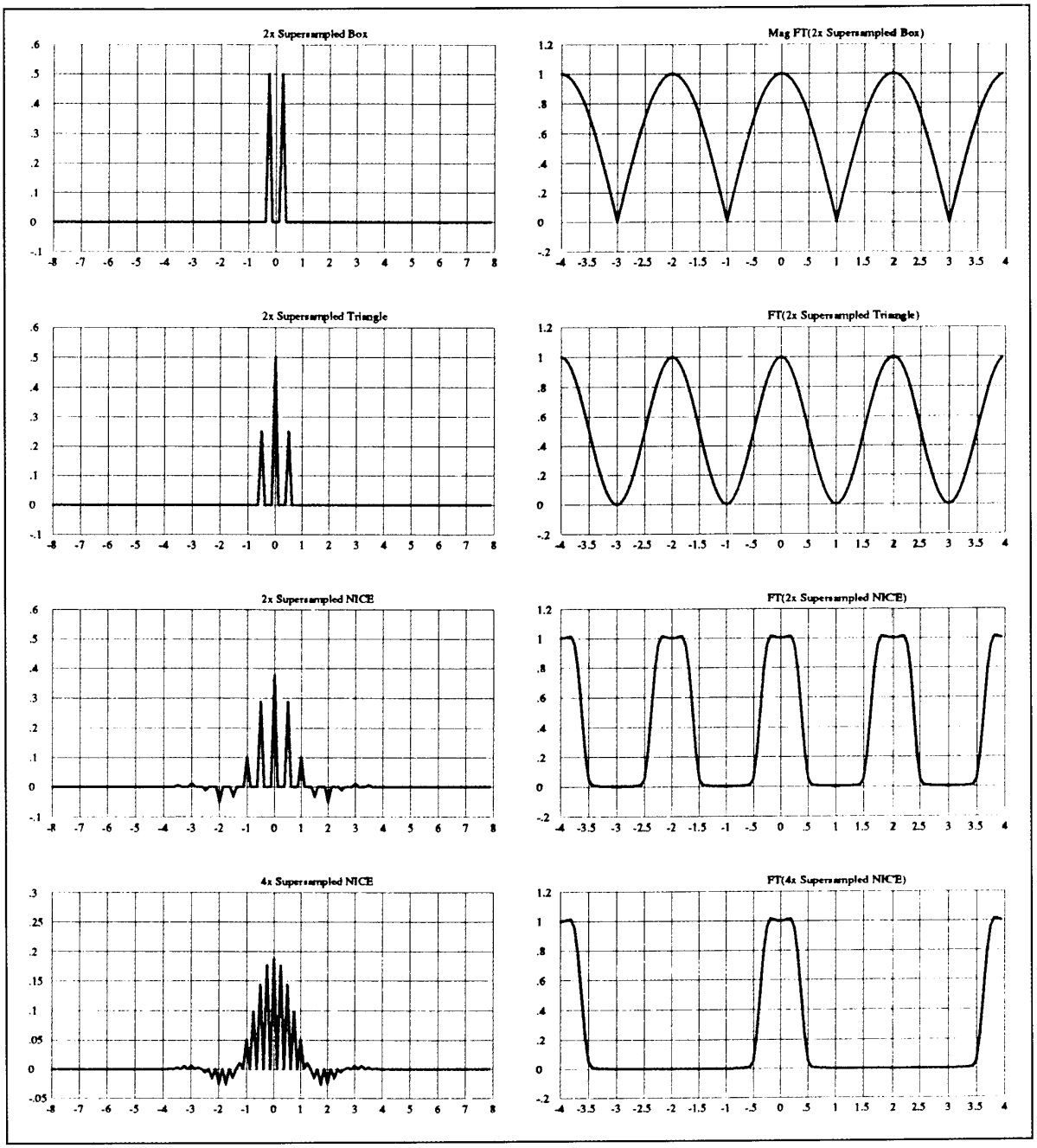

Figure 4. Subsampling.

something. Any real-world solution to the problem suffers from one or more of the following problems:

- An achievable filter is not a perfect low-pass filter.
- The steeper the sides of the FT, the wider the PSF needs to be, and the more arithmetic you have to do.
- There is no such thing as negative light. Any decent approximation to a low-pass filter for either sampling or reconstruction has negative lobes. This might result in a negative pixel value. Clamping them to zero is about all you can do, but it is not ideal.
- Antialiasing requires filtering a continuous function. Subsampling is only a crude approximation to this.
- You have to calculate outside the screen region if the

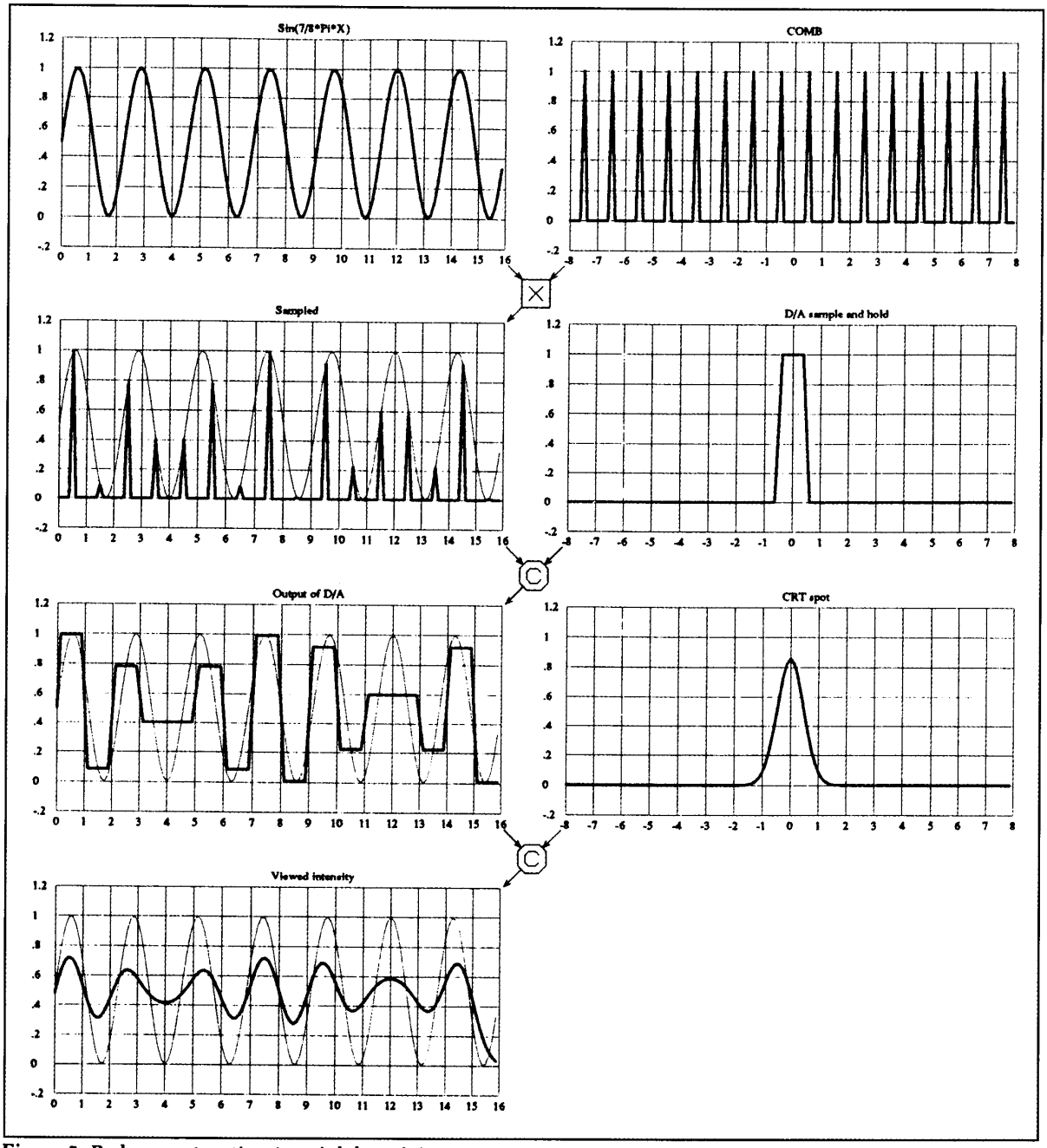

Figure 5. Bad reconstruction (spatial domain).

filter PSF is wider than ±.5.

- Gamma correction can mess things up. I won't go into this here.
- The reconstruction filter might mess you up anyway, unless you add special hardware filters to your frame buffer.

This is a big subject. I haven't even mentioned Cook's stochastic sampling or Kajiya's approximation sampling. Also, I haven't covered the 2D case (complete images) or the 3D case (time).

If you want to build intuition about this, I would recommend that you find or write a program that takes Fourier transforms and play with it. Also take a look at the *Atlas of Optical Transforms*, by Harburn, Taylor, and Welberry.[4] This is a whole book of 2D images and their FTs. ■

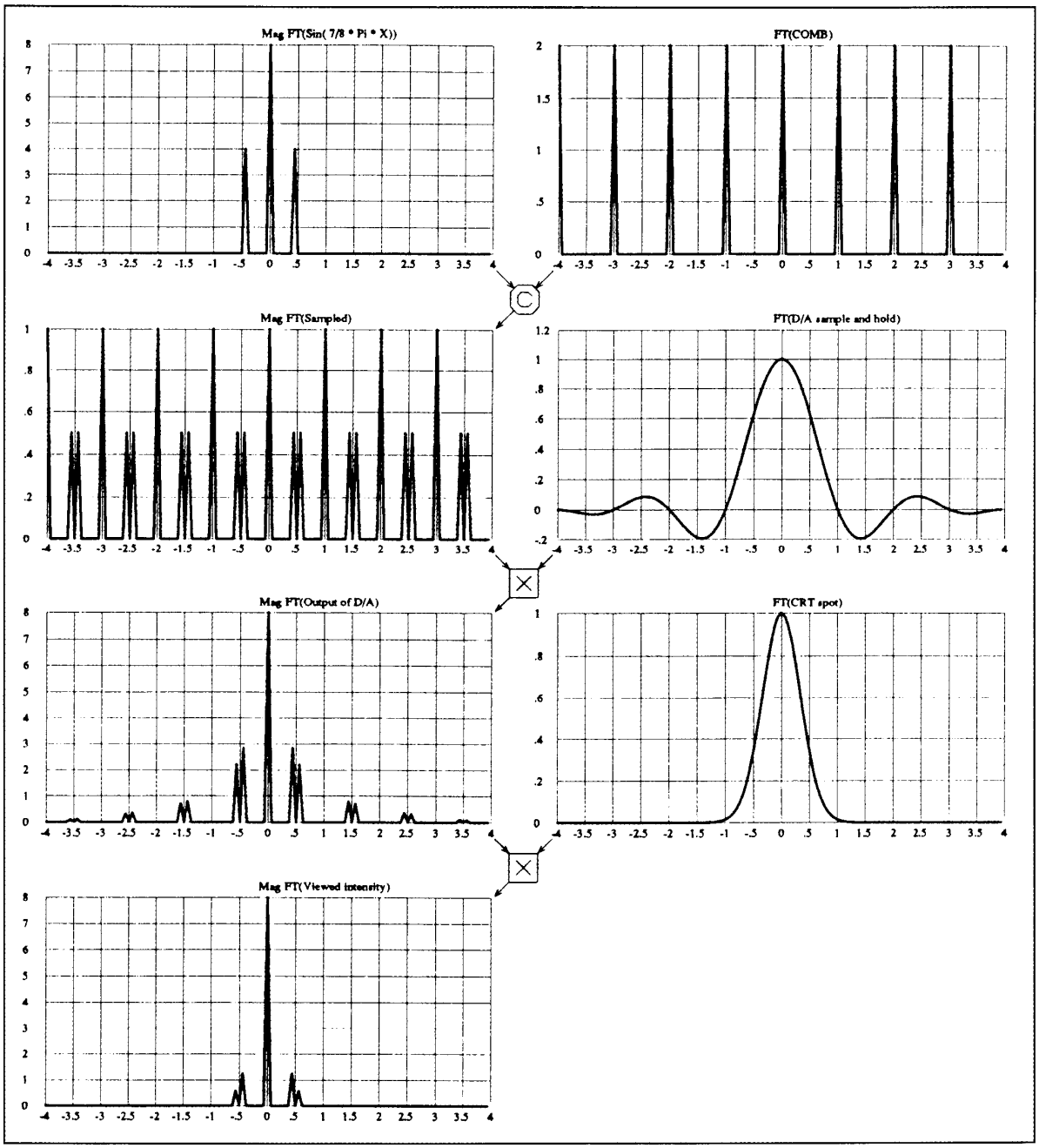

Figure 6. Bad reconstruction (frequency domain).

Acknowledgment

Thanks to Jim Kajiya and Steve Gabriel for keeping me from making too many stupid mistakes in this column.

References

1. F.C. Crow, "Summed-Area Tables for Texture Mapping," *Computer Graphics* (Proc. SIGGRAPH), Vol. 18, No. 3, July 1984, pp. 207-212.

2. P.S. Heckbert, "Filtering by Repeated Integration," *Computer Graphics* (Proc. SIGGRAPH), Vol. 20, No. 4, Aug. 1986, pp. 315-321.

3. K. Perlin, notes for SIGGRAPH course *State of the Art in Image Synthesis*, July 1985, ACM, New York, 1985.

4. G. Harburn, C.A. Taylor, and T.R. Welberry, *Atlas of Optical Transforms*, Cornell Univ. Press, Ithaca, N.Y., 1975.