A Pixel Is *Not* A Little Square, A Pixel Is *Not* A Little Square, A Pixel Is *Not* A Little Square! (And a Voxel is *Not* a Little Cube)[1]

Technical Memo 6

Alvy Ray Smith

July 17, 1995

Abstract

My purpose here is to, once and for all, rid the world of the misconception that a pixel is a little geometric square. This is not a religious issue. This is an issue that strikes right at the root of correct image (sprite) computing and the ability to correctly integrate (converge) the discrete and the continuous. The little square model is simply incorrect. It harms. It gets in the way. If you find yourself thinking that a pixel is a little square, please read this paper. I will have succeeded if you at least understand that you are using the model and why it is permissible in your case to do so (is it?).

Everything I say about little squares and pixels in the 2D case applies equally well to little cubes and voxels in 3D. The generalization is straightforward, so I won't mention it from hereon[1].

I discuss why the *little square model* continues to dominate our collective minds. I show why it is wrong in general. I show when it is appropriate to use a little square in the context of a pixel. I propose a discrete to continuous mapping—because this is where the problem arises—that always works and does not assume too much.

I presented some of this argument in Tech Memo 5 ([Smith95]) but have encountered a serious enough misuse of the little square model since I wrote that paper to make me believe a full frontal attack is necessary.

The Little Square Model

The little square model pretends to represents a pixel (picture element) as a geometric square[2]. Thus pixel (i, j) is assumed to correspond to the area of the plane bounded by the square $\{(x, y) \mid i\text{-}.5 \leq x \leq i+.5, j\text{-}.5 \leq y \leq j+.5\}$.

[1] Added November 11, 1996, after attending the Visible Human Project Conference 96 in Bethesda, MD.

[2] In general, a little rectangle, but I will normalize to the little square here. The little rectangle model is the same mistake.

I have already, with this simple definition, entered the territory of controversy—a misguided (or at least irrelevant) controversy as I will attempt to show. There is typically an argument about whether the pixel "center" lies on the integers or the half-integers. The "half-integerists" would have pixel (i, j) correspond instead to the area of the plane $\{(x, y) \mid i \le x \le i+1., j \le y \le j+1.\}$.

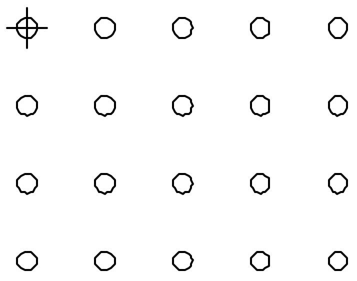
This model is hidden sometimes under terminology such as the following—the case that prompted this memo, in fact: The resolution-independent coordinate system for an image is $\{(x, y) \mid 0. \le x \le W/H, 0 . \le y \le 1.\}$, W and H are the width and height of the image. The resolution dependent coordinate system places the edges of the pixels on the integers, their centers on the edges plus one half, the upper left corner on (0., 0.), the upper right on (W., 0.), and the lower left on (0., H). See the little squares? They would have edges and centers by this formulation.

So What *Is* a Pixel?

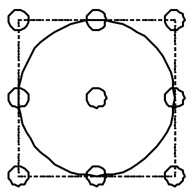
A pixel is a *point* sample. It exists only at a point. For a color picture, a pixel might actually contain three samples, one for each primary color contributing to the picture at the sampling point. We can still think of this as a point sample of a color. But we cannot think of a pixel as a square—or anything other than a point. There are cases where the *contributions* to a pixel can be modeled, in a low-order way, by a little square, but not ever the pixel itself.

An image is a rectilinear array of point samples (pixels). The marvelous Sampling Theorem tells us that we can reconstruct a continuous entity from such a discrete entity using an appropriate *reconstruction filter*[3]. Figure 1 illustrates how an image is reconstructed with a reconstruction filter into a continuous entity. The filter used here could be, for example, a truncated Gaussian. To simplify this image, I use only the *footprint* of the filter and of the reconstructed picture. The footprint is the area under the non-0 parts of the filter or picture. It is often convenient to draw the minimal enclosing rectangle for footprints. They are simply easier to draw than the footprint—Figure 1(d). I have drawn the minimal rectangles as dotted rectangles in Figure 1.

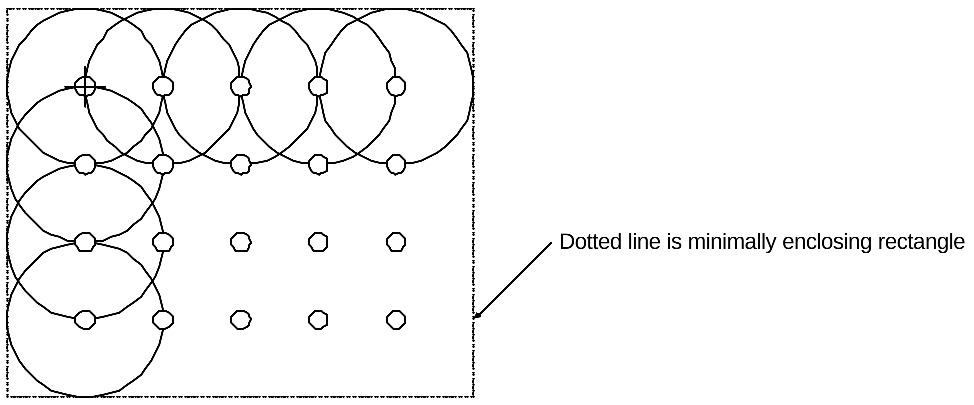
[3] And some assumptions about smoothness that we do not need to worry about here.



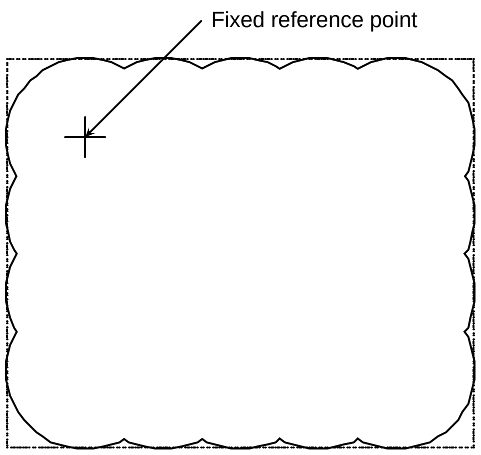
(a) A 5x4 image.



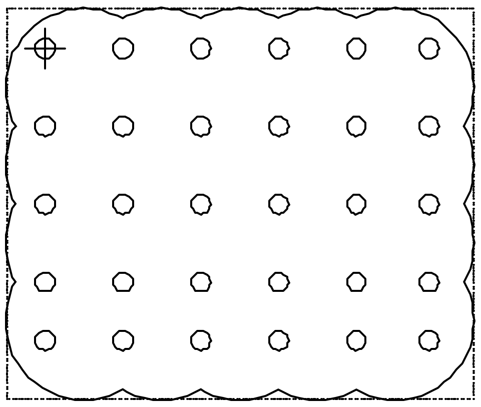
(b) The footprint of a reconstruction filter. A truncated Gaussian, for example.



(c) Footprint of image under reconstruction.

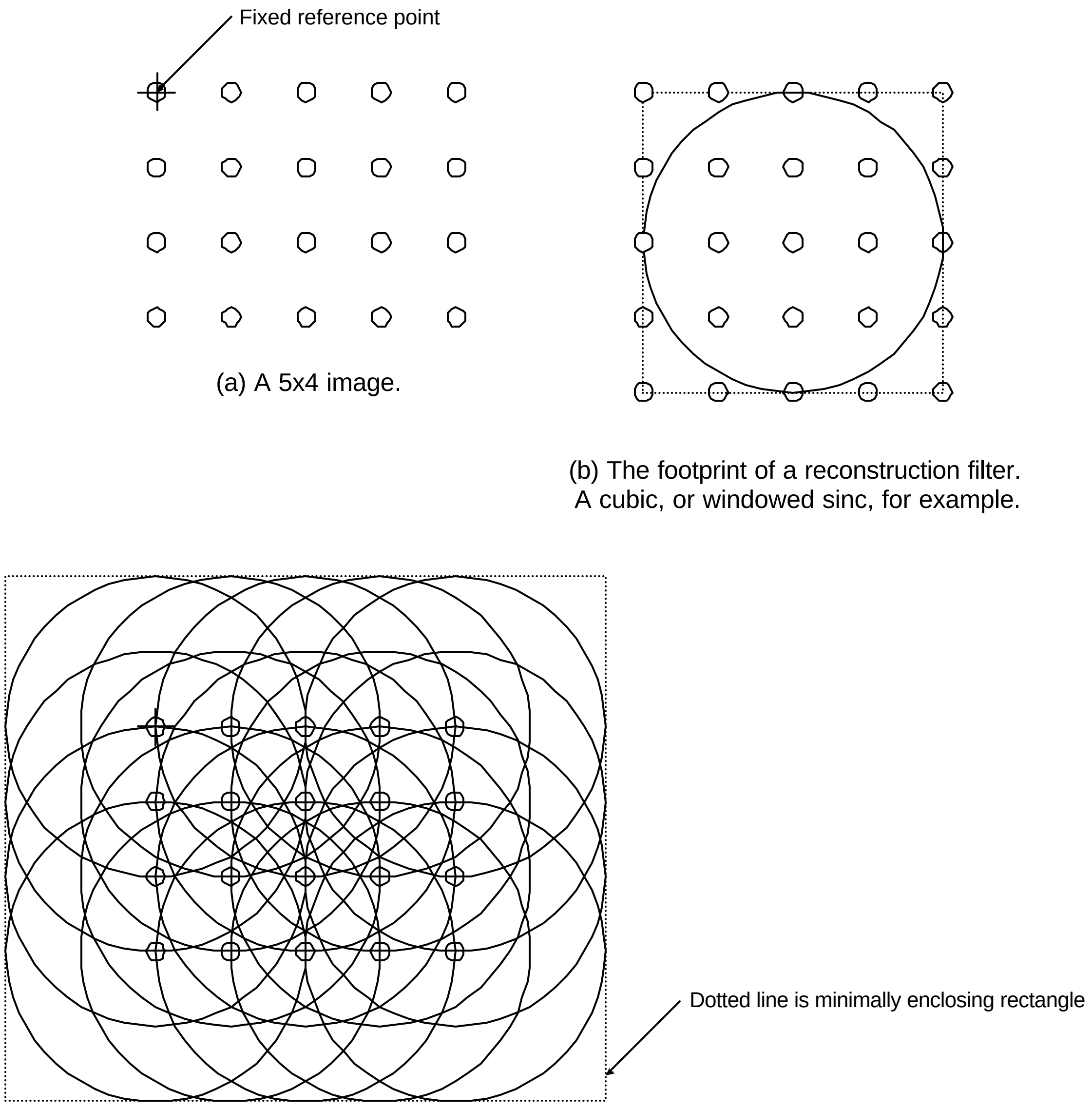


(d) Footprint of reconstructed image. Medium quality reconstruction.



(e) Reconstruction translated (.5,.5), then resampled into a 6x5 image.

FIGURE 1



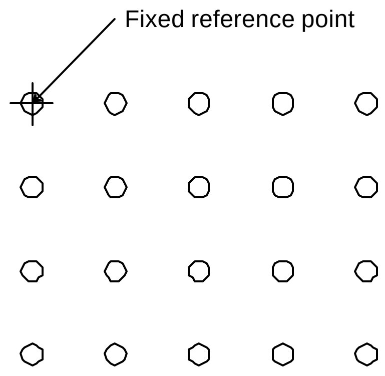
(a) A 5x4 image.

(b) The footprint of a reconstruction filter. A cubic, or windowed sinc, for example.

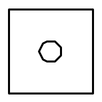
(c) Footprint of reconstructed image. Typical high quality reconstruction. Would be resampled into 7x6 image.

FIGURE 2

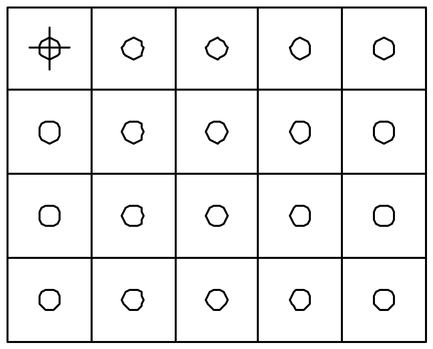
Figure 2 is the same image reconstructed with a better reconstruction filter—eg, a cubic filter or a windowed sinc function—and not an unusual one at all. Most quality imaging uses filters of this variety. The important point is that both of these figures illustrate valid image computations. In neither case is the footprint rectangular. In neither case is the pixel ever approximated by a little square. If a shape *were* to be associated with a pixel (and I am not arguing that it should), then the most natural thing would be the shape of the footprint of the reconstruction filter. As these two examples show, the filters typically overlap a great deal.



(a) A 5x4 image.



(b) The footprint of a reconstruction filter.
A simple box filter, for example.



(c) Footprint of reconstructed image.
The worst case: low quality reconstruction.

FIGURE 3

Figure 3 is the same image reconstructed with one of the poorest reconstruction filters—a box filter. The only thing poorer is no reconstruction at all—resulting in the abominable “jaggies” of the early days of computer graphics—and we will not even further consider this possibility. The Figure 3 reconstruction too is a valid image computation, even though it is lacking in quality. This lowest-quality case is the only one that suggests the little square model.

So it should be clear that the coordinate system definition given above is not suitable for anything but the lowest-quality image computing. The edges of a reconstructed image are most naturally defined to be its minimally enclosing rectangle. But these are dependent on the chosen reconstruction filter.

The only resolution independent coordinate system that one can safely map to an image requires a known reconstruction filter. Given an image, say that represented by Figure 2(a), and a known filter, say that of Figure 2(b), then I can determine exactly what the minimally enclosing rectangle is and map this to the normalized rectangle of the proposed definition above: $\{(x, y) \mid 0. \leq x \leq W/H, 0 .\leq y \leq 1.\}$. Then the pixel (point sample, remember) locations can be backed out of the mapping. Will they sit on the half-integers? In the three cases above, all of which are valid, only Figure 3 (the worst) will have the samples on the half-integers under this mapping. Will the left edge of the reconstructed image lie distance .5 left of the leftmost column of pixels? Again, only in the worst case, Figure 3.

I would suggest at this point that the only thing that is fixed, in general, are the samples. Doesn't it make sense that they be mapped to the integers since that is so simple to do? Then the edges of the reconstructed continuum float depending on the chosen filter. I believe that if you rid yourself of the little square model, it does not even occur to you to put the samples on the half-integers, an awkward position for all but the lowly box filter. The bicubic filter that I most often use has a footprint like the minimal enclosing rectangle of Figure 2(b). Half-integer locations for this filter are simply awkward. Certainly one can do it, but why the extra work?

I believe that the half-integer locations are attractive for "little-squarists" because the minpoint (upper left corner) of the reconstructed entity falls at (0, 0). But note that this only happens for—yes, again—the box filter. For my favorite filter, the minpoint would fall at (-1.5, -1.5). Is that more convenient, prettier, or faster than (-2., -2.)? No.

Why Is the Little square Model So Persistent?

I believe there are two principal reasons that the little square model hasn't simply gone away:

- Geometry-based computer graphics uses it.
- Video magnification of computer displays appears to show it.

Geometry-based computer graphics (3D synthesis, CGI, etc.) has solved some very difficult problems over the last two decades by assuming that the world they model could be divided into little squares. *Rendering* is the process of converting abstract geometry into viewable pixels that can be displayed on a computer screen or written to film or video for display. A modern computer graphics model can have millions of polygons contributing to a single image. How are all these millions of geometric things to be resolved into a regular array of pixels for display? Answer: Simplify the problem by assuming the rectangular viewport on the model is divided regularly into little squares, one per final pixel. Solve the often-intense hidden surface problem presented by this little square part of the viewport. Average the results into a color sample. This is, of course, exactly box filtering. And it works, even though it is low order filtering. We probably wouldn't be where we are today in computer graphics without this simplifying assumption. *But,* this is no reason to *identify* the model of geometric contributions to a pixel with the pixel. I often meet extremely intelligent and accomplished geometry-based computer graphicists who have leapt to the *identification* of the little square simplification with the pixel. This is not a plea for them to desist from use of the little square model. It is a plea for them to be aware of the simplification involved and to understand that the other half of computer picturing—the half that uses no geometry at all, the imaging half—tries to avoid this very simplification for quality reasons.

When one "magnifies" or "zooms in on" an image in most popular applications, each pixel appears to be a little square. The higher the magnification or the closer in the zoom, the bigger the little squares get. Since I am apparently magnifying the pixel, it must be a little square, right? No, this is a false perception. What is happening when you zoom in is this: Each point sample is being replicated MxM times, for magnification factor M. When you look at an image consisting of MxM pixels all of the same color, guess what you see: A square of that solid color! It is not an accurate picture of the pixel below. It is a bunch of pixels approximating what you would see if a reconstruction with a box filter were performed. To do a true zoom requires a resampling operation and is much slower than a video card can comfortably support in realtime today. So the plea here is to please disregard the squareness of zoomed in "pixels". You are really seeing an MxM array of point samples, not a single point sample rendered large.

How Does a Scanner Digitize a Picture?

Just to be sure that I eradicate the notion of little square everywhere, let's look at the scanning process. I want to be sure that nobody thinks that scanners work with little squares and that, therefore, it is all right to use the model.

A scanner works like this: A light source illuminates a piece of paper containing a colored picture. Light reflected from the paper is collected and measured by color sensitive devices. Is the picture on the paper divided up into little squares, each of which is measured and converted to a single pixel? Not at all. In fact, this would be very hard to accomplish physically. What happens instead is that the illuminating light source has a shape or the receiving device has an aperture that gives the incoming light a shape or both, and the device integrates across this shape. The shape is never a square. It is not necessarily accurate, but will give you the correct flavor, to think of the shape as a Gaussian. In general, overlapping shapes are averaged to get neighboring pixel samples.

So scanning should not contribute any weight to the little square model.

How Does a Printer Print a Digital Image?

Again, let's look at a familiar process to determine if it contributes to the little square model. The process this time is printing. This is a very large and complex subject, because there are many different ways to print an image to a medium.

Consider printing to film first. There are several ways to do this. In any process where a flying spot of light is used to expose the film, then the exposing beam has a shape—think of it as Gaussian. There are no little squares in this process.

Consider half-tone printing of ink on paper. The concept here is to convert a pixel with many values to a dot of opaque ink on paper such that the area of the dot relative to a little square of paper occupied by the dot is in the ratio of the intensity of the pixel to the maximum possible intensity. Sounds like the little square model, doesn't it? But for color printing, the different primaries are printed at an angle to each other so that they can “show through” one another. Therefore, although there are little squares in each color separation, there are none for the final result.

There is a new technique for printing ink-on-paper that uses stochastic patterns within each little square. Hence the separations do not have to be rotated relative one another in order to “show through”. The little square model is a decent model in this case.

There are sublimation dye printers and relatives now that print “continuous tone” images in ink on paper. I believe that these use little squares of transparent dyes to achieve continuous tone. In that case, they probably use the little square model.

The point here is that the use of the little square is a printing technology decision, not something inherent in the model of the image being imaged. In fact, the image being imaged is simply an array of point samples in all cases.

How Is an Image Displayed on a Monitor?

Many people are aware that a color monitor often has little triads of dots that cause the perception of color at normal viewing distances. This is often true, except for Sony Trinitrons that use little strips of rectangles rather than triads of dots. In neither case are there little squares on the display. I will assume triads for this discussion. It goes through for the Trinitron pattern too, however.

While we are at it, I would also like to dispel any notion that the triads are pixels. There is no fixed mapping between triads and pixels driving them. The easiest way to understand this is to consider your own graphics card. Most modern cards support a variety of different color resolutions—eg, 640x480, 800x600, 1024x768, etc. The number of triads on your display screen do not change as you change the number of pixels driving them.

Now back to the display of pixels on a screen. Here’s roughly what happens. The value of a pixel is converted, for each primary color, to a voltage level. This stepped voltage is passed through electronics which, by its very nature, rounds off the edges of the level steps. The shaped voltage modulates an electron beam that is being deflected in raster fashion across the face of your display. This beam has shape—again think of it as Gaussian (although it can get highly distorted toward the edges of the display). The shaped beam passes through a shadow mask that ensures that only the red gun will illuminate the red phosphors and so forth. Then the appropriate phosphors are excited and they emit patterns of light. Your eye then integrates the light pattern from a group of triads into a color. This is a complex process that I have presented only sketchily. But I think it is obvious that there are no little squares involved at any step of the process. There are, as usual in imaging, overlapping shapes that serve as natural reconstruction filters.

Another display issue that implies the little square model is the notion of displays with “non-square pixels”. Although it is becoming less common now, it used to be fairly common to have a video display driven by, say, a 512x480 image memory. This means that 512 samples are used to write a row to the display, and there are 480 rows. Video monitors have, if correctly adjusted, a 4:3 aspect ratio. So the *pixel spacing ratio* (*PSR*) for this case is 1.25 = (4/3)/512:1/480 = (4/3)*(480/512). So the correct terminology for this case is that the monitor has a "non-square pixel spacing ratio", not that it has "non-square pixels". Most modern computer displays, if correctly adjusted, have square PSR—ie, PSR = 1.

So we have no contributions from scanning or display processes for the little square model. We have a case or two for particular printing technologies that support a little square model, but they are not the general printing case. In summary, the processes used for image input and output are not sources for the little square model.

What Is a Discrete to Continuous Mapping That Works?

The only mapping that I have been able to come up with that doesn't assume too much is this: Assume the samples are mapped to the integers (it's so easy after all—just an offset to the array indices used to address them in an image computation). Then the outer extremes of the image are bounded by a rectangle whose left edge is (filterwidth/2) to the left of the leftmost column of integers, right edge is (filterwidth/2) to the right of the rightmost column of integers, top edge is (filterheight/2) above the topmost row of integers, and bottom edge is (filterheight/2) below the bottommost row of integers occupied by image samples).

This is still not quite complete, because it assumes symmetric filters. There are cases—for example, perspective transformations on (reconstructions of) images—when asymmetric filters are useful. So the mapping must be generalized to handle them. By the way, by symmetrical filter I mean that it is symmetrical about its horizontal axis, and it is symmetrical about its vertical axis. I do not mean that it is necessarily the same in both dimensions.

I do believe that it is important to have the vertical dimension increase downward. It is not required, but it is so natural that I find it hard to argue against. We display from top to bottom on millions of TVs and computer displays. We read from top to bottom. We compute on matrices—the natural container for images—from top to bottom. Nearly all popular image file formats store images from top to bottom[4], or if in tiled form, in rows of tiles from top to bottom.

Image Broadening

I mentioned earlier that I would return to a discussion of whether the minimal enclosing rectangle of a reconstructed image were the most natural representation of it. This will fall out of a discussion of image *broadening*.

[4] The Targa file format is the most flagrant violator of this rule; it also reverses RGB to BGR. And Windows has adapted the Targa format for its "bitmaps" (without documenting the color channel reversal, as I unhappily discovered as a developer).

By the very nature of the Sampling Theorem that underlies everything that we do, during a reconstruction an image suffers broadening due to the width of the reconstruction filter. Again, look at Figures 1-3 for examples. The amount of broadening is dependent on the filter used. If the filter is asymmetric, then so is the broadening.

I call the excess of a broadened image over itself to be its *margin*. In general then, an image margin is not symmetric about its image and is dependent on the reconstruction filter being used.

Let's be concrete: Consider scaling 640x480 image down by a factor of 2 in both dimensions. The original image is assumed to have its minpoint at (0,0) and its maxpoint (lower right corner) at (639, 479). Assume we are using a symmetric bicubic filter for good results. Its canonical footprint is $\{(x, y) \mid -2. \leq x \leq 2., -2. \leq y \leq 2.\}$. Then to do the scaling we ***reconstruct-transform-resample***[5]: When we reconstruct our image into a continuum, it occupies the space from -2. to 641. horizontally and from -2. to 481. vertically. Thus its minimal enclosing rectangle has minpoint (-2., -2.) and maxpoint (641., 481.). If we were to resample at this moment, before doing any transform on the reconstruction, we would have broadened the image by one pixel all around; we would have introduced a 1-pixel margin by the act of reconstruction. (The filter has 0 weight at its extremes so the samples along the minimal enclosing rectangle would be 0 and not figure into the support of the sampled image.)

This is a good point to pause and note that it is not necessarily the broadened footprint that is interesting. In this case, if we have only a 640x480 display then we would crop off the margin pixels for redisplay anyway. In my experience, I usually want to know about an image's extent and its margins, before and after transformations, in full detail so that I can decide exactly what I want to do. The important point is that this information be available, not how.

Back to the scaling example: Now let's minify the reconstruction by 2. This means that the reconstruction is scaled down about its center. This is easily modeled by scaling down the minimal enclosing rectangle that I will represent as $(-2., -2.) \rightarrow (641., 481.)$. The scaling happens about the center at (319.5, 239.5). There are many ways to proceed from here so I will not pursue the details. I believe that I have presented enough of the problem and technique of solution that it is clear that nothing is offered to it by a little square model for the pixels.

In fact, the little square model might have misled us into a compounding of low quality techniques: It is tempting to box filter box filters. Thus it is tempting to take the 640x480 example above and look at each set of 2x2 pixels to scale down by 2. It is another application of the box filter to simply average each 2x2

[5] Generally, there needs to be a low-pass filtering step too, before resampling. When scaling up, no high frequencies are introduced, but when scaling down, they are and must be low-pass filtered to satisfy the Sampling Theorem.

set of pixels into a single new pixel. This is generally a bad idea and not the path to take for good results.

Summary

I have presented a brief but inclusive analysis of sampling and filtering. It has been shown that the little square model does not arise naturally in sampling theory, the main underpinning of everything we do. It does not arise in scanning or display. It arises in printing only in restricted cases. The geometry world uses it a great deal because they have had to simplify in order to accomplish. Their simplified model of *contributions* to a pixel should not be confused with or identified with the pixel. Magnified screen pixels that look like little squares have been shown to be a quick and dirty trick (pixel replication) by graphics boards designers, but not the truth. In short, the little square model should be suspect whenever it is encountered. It should be used with great care, if at all, and certainly not offered to the world as "standard" in image computing.

In the process of this presentation, I have demonstrated at least one very natural mapping of the discrete samples of an image to a continuum reconstructed from it. The method used is straightforward and always works. I have also argued that the following details of such a reconstruction are interesting enough to track: the exact filter used, the image margins created by reconstruction, and minimal enclosing rectangles. I have also suggested that it is natural for the vertical coordinate to increase downwards, and that it is simple and natural for sample locations to be mapped to the integers. However, neither of these is required, so long as consistency reigns. I do argue, however, that placing samples at the half-integers seems to indicate a lurking reluctance to dismiss the little square model, or the box filter, the only two common situations where the half-integers are natural concepts.

Finally, I have pointed out two related misconceptions: (1) The triads on a display screen are not pixels; they do not even map one-to-one to pixels. (2) People who refer to displays with non-square pixels should refer instead to non-square, or non-uniform, pixel spacing.

References

[Smith95] Smith, Alvy Ray, *A Sprite Theory of Image Computing*, Tech Memo 5, Microsoft, Jul 1995.