ELECTRONOTES
1016 Hanshaw Rd
Ithaca, NY 14850

**APPLICATION NOTE NO. 376**

December 2011

# INTERPOLATION OF SAMPLES ON A SQUARE LATTICE

## 1. INTRODUCTION

In a simple compression scheme for images represented by samples (pixels) on a square lattice, we might consider downsampling an image, and then transmitting or storing only the retained samples. When we want to see a full-sized image, we interpolate the missing points if we can. There is no way to say, in general, just how successful this might turn out. It all depends on the actual image, its size, the downsampling factor, the interpolation method, and the particular application.

## 2. DOWNSAMPLING

We will be using here the most straightforward ways of downsampling (resampling) a rectangular lattice [see Application Note AN-369 “Two Dimensional Uniform Sampling”, Feb. 2007]. We resample in both directions by the same factor. This is fairly drastic since the smallest downsampling is by 2 in each direction, keeping only 1 of every 4 original pixels. Resampling by 3 in each direction retains only 1 of 9 original pixels, and by 4, we have only 1 of every 16 pixels retained, and so on. [A sampling that retains half the pixels is available through "quincunx" sampling. See AN-371 “A 2D Wavelet Lifting Example,” April 2007].

The first thing we note is that in our experiments, we will find it convenient to just resample with a lattice - multiplying each original pixels by a lattice of either ones or zeros. We will not discard the zeroed pixels, as very soon we would bring them back for interpolation. This means that the resampled image is the same size (in pixels) as the original, but surely, it will look very dark. Fig. 1 shows an original image that is 240x240 pixels, while Fig. 2 shows the case where it is downsampled by 3 in both directions, retaining the zeroed pixels. Fig. 2 is very dark. One way to get rid of the darkness is to discard the zeroed pixels, and this is shown in Fig. 3 where the size is smaller now (80x80). In Fig. 4, the resampled image is convolved with a 3x3 matrix of ones, causing each pixel to be repeated 8 additional times, restoring the brightness, but not adding any pixels. Indeed, Fig. 3 and Fig. 4 are the same thing in terms of pixels.

The images of Fig. 3 and Fig. 4 can be described as "pixelated" or "jaggy" in that we can easily see the pixels. Of course, as with most image processing studies, the viewing distance is also an issue. But one basic goal is to return the brightness, and of much more interest, to reduce the distraction of the pixelation. We expect that this may also cause some blurring of details.


**Fig. 1** **Original**

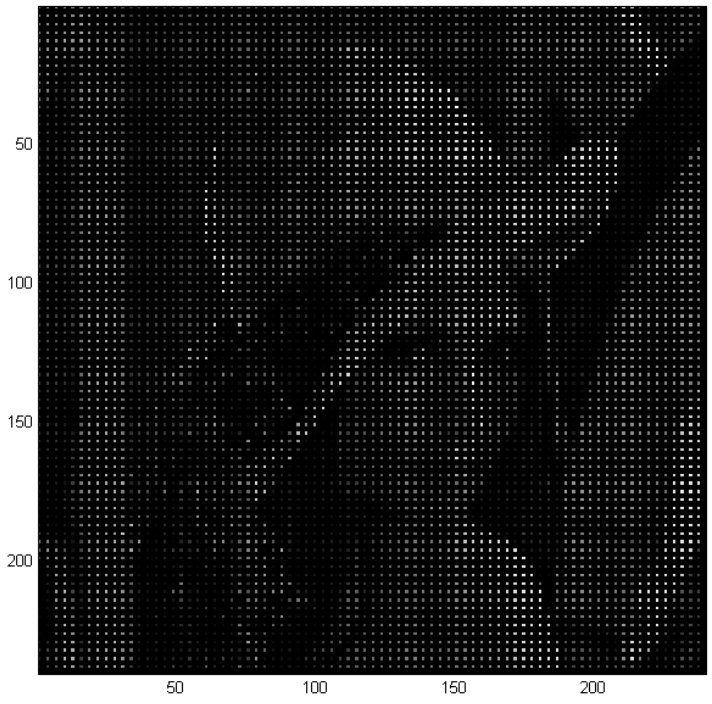



**Fig. 2** **Sampled by 3 in Each Direction (DARK!)**

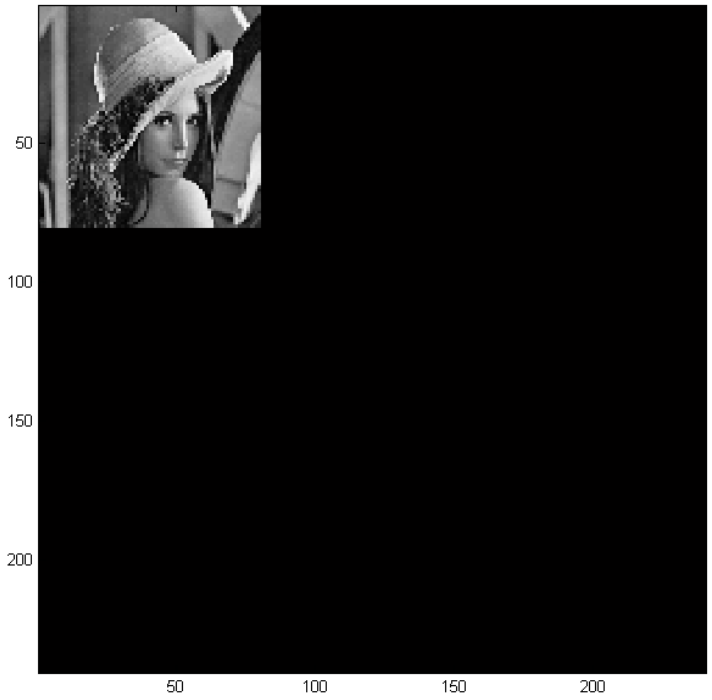



**Fig. 3 Zero Samples Removed (SMALL!)**


**Fig. 4 Sampled/Held (Pixelated)**

## 3. INTERPOLATION OF IMAGES IN GENERAL

Normally we think of interpolation in terms of filtering, and in turn, we think of filtering in terms of the frequency domain. That is, we suppose that we would interpolate an image by using a two-dimensional filter that has low-pass properties relative to some bandwidth? Actually, while bandlimiting is often of great importance in one dimensional signals (i.e., audio) it is often of less importance (at least, of less use) in images which are often not bandlimited. In audio, it is often the case that, for practical purposes, a signal is bandlimited or can easily be made so without significant loss of quality. For one thing, consider that acoustically produced sounds have features limited by mechanical inertia (by physics) so very sharp transitions (the highest frequencies) are already relatively suppressed. In our electronic generation of musical sounds, we likewise intentionally low-pass, in most cases.

An image on the other hand may not be bandlimited. Much as we can electronically generate a sharp edge (a step or pulse), we can easily arrange materials so that sharp transitions will be imaged. We can paint a house the whitest of whites and the window frames the blackest of blacks. Large contrasts with sharp transitions nearly as spectacular can occur naturally. Nothing in physics prevents it. Of course, it is also possible to have natural images of very low bandwidth. A blue sky, by itself, is pretty much blue. A sandy beach may involve slowly transitioning shades of grey.

And, while we often love to point out the spectacular parallels in aural and visual perception, there are probably just about as many differences.

As we shall see below, many interpolation notions for images are in the spatial domain (analogous to time domain) rather than in the spatial frequency domain.

## 4. PIECEWISE PLANAR USING TRIANGLES (ONE NOTION OF LINEAR INTERPOLATION)

Probably no notion of interpolation is as well known as that of linear interpolation, since it was so often used with math tables in the past before calculators and computers. That is, when we needed a value of a math function that was not a tabulated value, we calculated it assuming a straight line between tabulated values. Notice that we call it linear interpolation because of the line, a first-order polynomial. Other types of interpolation such as parabolic, cubic, and sync (associated with bandlimiting) may well be "linear" in the engineering sense of "linearity" (which we look for in superposition examples).

Because linear (straight line) interpolation is so fundamental in one dimension (1D) we might look for an equivalent notion in two dimensions (2D), and this would seem to be a planar rather than a line model. In fact, given a local group of three points, we could fit a plane to the three points exactly, in order to identify candidates for calculated interpolated points. In a lattice such as quincunx, such an approach is extremely intuitive.

In interpolating a rectangular lattice, we naturally think in terms of sampling lattices that are rectangular (most likely square in fact). Thus we begin with a set of four points at the corners of a square that we propose to work with. These squares can be divided along a diagonal (either choice of diagonal is possible) into two triangles, and each square cell is thus divided into two (generally non-coplanar) planar regions. We could then decide to calculate needed interpolated points to be on the appropriate triangle.

Fig. 5 shows a rectangular lattice that has been sample by two in each direction, and a typical sampling square has been divided into two triangles (one light green, one light blue). In this case, no points to be interpolated lie on the interior of the triangles, but only along the sides. The sides of the triangle are of course straight lines, and thus the interpolated points are all determined as the average of values at the ends of the lines.

A few more details make this figure more clear. What we should envision from Fig. 5 is a lattice with points at all integers, with the red points representing retained samples and the green points being discarded values replaced with zeros. Accordingly, at the red points, we envision pixel values extending like sticks out of the page to heights corresponding to the values. The tips of these three sticks (extending from the points marked "Original") support a triangle. We extend the "to Interpolated" points up to touch the triangle to obtain the interpolated values.

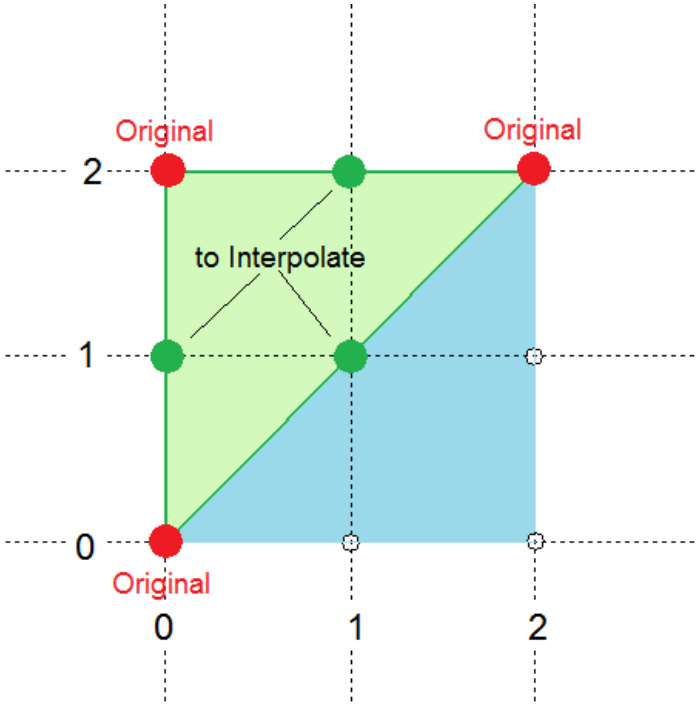



Fig. 5

Rectangular sampling (by 2) in both directions to be interpolated

Red samples are originals, green are to be interpolated as being on the light green triangular plane. The light blue plane will complete the lattice. Note diagonal boundary is common to both triangles.

Note that once we choose a diagonal, it does not matter which of the two triangles (green or blue) the center of the sampling cell [the point (1,1)] is on (the diagonal is a hypotenuse side to both triangles). If however we had taken the diagonal the other way [from the point (2,0) to the point (0,2)], we would in general have a different value for the center. So we might have a notion that the center, in this case, might better be the average of the two diagonal calculations, which is of course, the average of all four corners. The simplest cases often lead to multiple approaches and interpretations.

We now have a formula. The point (0,1) is the average of the points (0,0) and (0,2), while the point (1,2) is the average of the points (0,2) and (2,2). To be piecewise planar, the point on the diagonal (hypotenuse), the point (1,1) must be the average of the points (0,0) and (2,2). However, we have also considered the logic of making the point (1,1) the average of (2,0) and (0,2), or perhaps even as the four-way average of (0,0), (2,0), (0,2), and (2,2), although this is not in general on any of the triangles. This will come back for our consideration later.

This "formula" may be realized by a 2D convolution (See also Section 6) of the sampled image with a filtering matrix – the impulse response of a 2D filter. The 2D impulse response for piecewise planar here is:

$$\begin{matrix} 0 & 1/2 & 1/2 \\ 1/2 & 1 & 1/2 \\ 1/2 & 1/2 & 0 \end{matrix} \qquad (1)$$

For the modified case where the center is the average of the four corners, the impulse response would be:

$$\begin{matrix} 1/4 & 1/2 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1/4 & 1/2 & 1/4 \end{matrix} \qquad (2)$$

The impulse response value of 1 is the center of the convolution process. This means that the retained samples are returned exactly by the interpolation.

It is useful to remark on the fact that the interpolation here is a filtering or convolution. So an impulse response arranged as an array of numbers is thought of as sliding over an array of pixel values on the same grid spacing. The overlapped cells are multiplied point by point and summed. One excellent way to envision this process is to use two sheets of grid paper. One of these has the pixel values, and the other (best written on tracing paper or a transparency) has the impulse response values. This we slide up/down and left/right to do the convolution. (See Section 6 for a detailed example with 3x3 sampling). It is also a good way to "derive" the impulse response. One simply chooses one square of the upper sheet as the "center" and parks this above a point to be interpolated. Then in the empty grid squares above retained pixels, you enter the weights desired. This should fill the non-zero part of the impulse response grid.

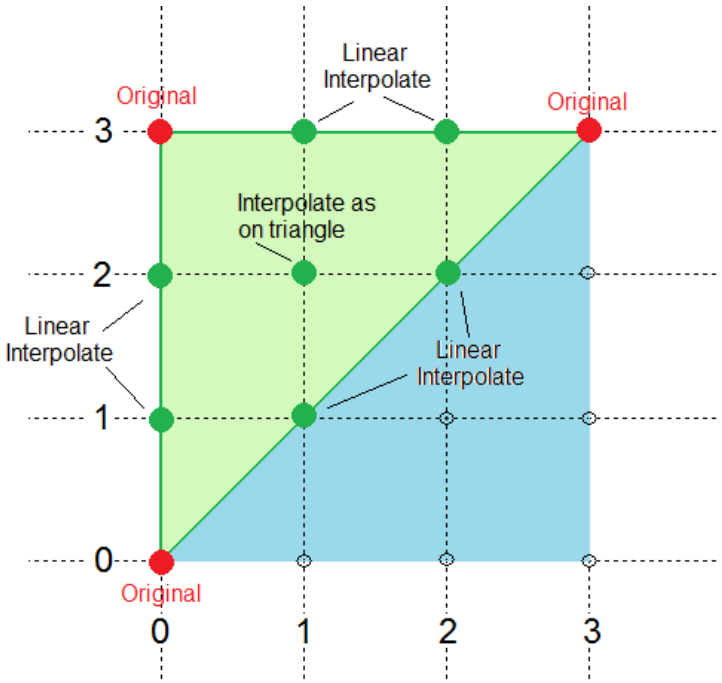



Fig. 6

rectangular sampling (by 3) in both directions to be interpolated

Red samples (3) are the originals and green samples (7) are to be interpolated as being on the green triangle. All but one of these (1,2) are on an edge and linear interpolation applies.

The blue triangle completes the lattice.

Fig. 6 shows an additional example – that of sampling by 3 in both directions, where once again, we assume a piecewise planar model, and have chosen a triangle as shown. This is in several ways different from the sampling by 2 case just above. For one thing, the linear interpolation on the edges are not just an average, but rather have weights. For example, the point (1,1) is obtained as 2/3 of (0,0) plus 1/3 of (3,3), and so on. (See also Section 5). Also here, these is no notion of finding a point as the average of the four corners, as there is no center point [no lattice point at (1.5, 1.5)].

And, we now have a point that is interior to the triangle, the point at (1,2). Perhaps the easiest way to find this value is to note that it lies on the triangle, on a line, midway between the points (0,1) and (2,3), so it is the average of these two points. The points (0,1) and (2,3) are obtained by linear interpolation (using weights 2/3 and 1/3 as appropriate) and then we average these two results for (1,2). Perhaps surprisingly here the result is that the point at (1,2) is the average of the three points (0,0), (0,3) and (3,3).

The impulse response of the piecewise planar interpolator for this case is:

```
  0      0      1/3      1/3**    1/3
  0     1/3*    2/3      2/3      1/3*
 1/3    2/3      1       2/3      1/3
1/3**   2/3     2/3      1/3**     0
 1/3    1/3*    1/3        0       0
```

**(3)**

Note that this matrix has as its third row, third column, and upward-going diagonal, the length-5 sequence (1/3 2/3 1 2/3 1/3) that corresponds to linear interpolation by 3, and are responsible for interpolation of the horizontal side, the vertical side, and the diagonal of the triangle of Fig. 6, respectively. This accounts for 13 of the 19 non-zero values in equation (3). Of the remaining six non-zero numbers of equation (3), the three matrix values of 1/3 that are marked as 1/3* are responsible for interpolating the point in the interior of the green triangle (1,2). The remaining three non-zero points, marked as 1/3** are perhaps more mysterious. They are required to get the interior point of the "other (blue) triangle", the point at (2,1).

## 5. AREA RATIO - (A SECOND NOTION OF LINEAR INTERPOLATION)

A second notion of linear interpolation relates to what we can call area ratio. That is, when finding the weight of a particular sample to be applied to a point to be interpolated, we look at the proportion of area behind the samples. Fig. 7 shows a simple case in 1D. Here we have two samples a and b, and we want to find the weighted sum of these two that give us the interpolated point c. The simple result is

$$c = aA + bB \qquad (4)$$

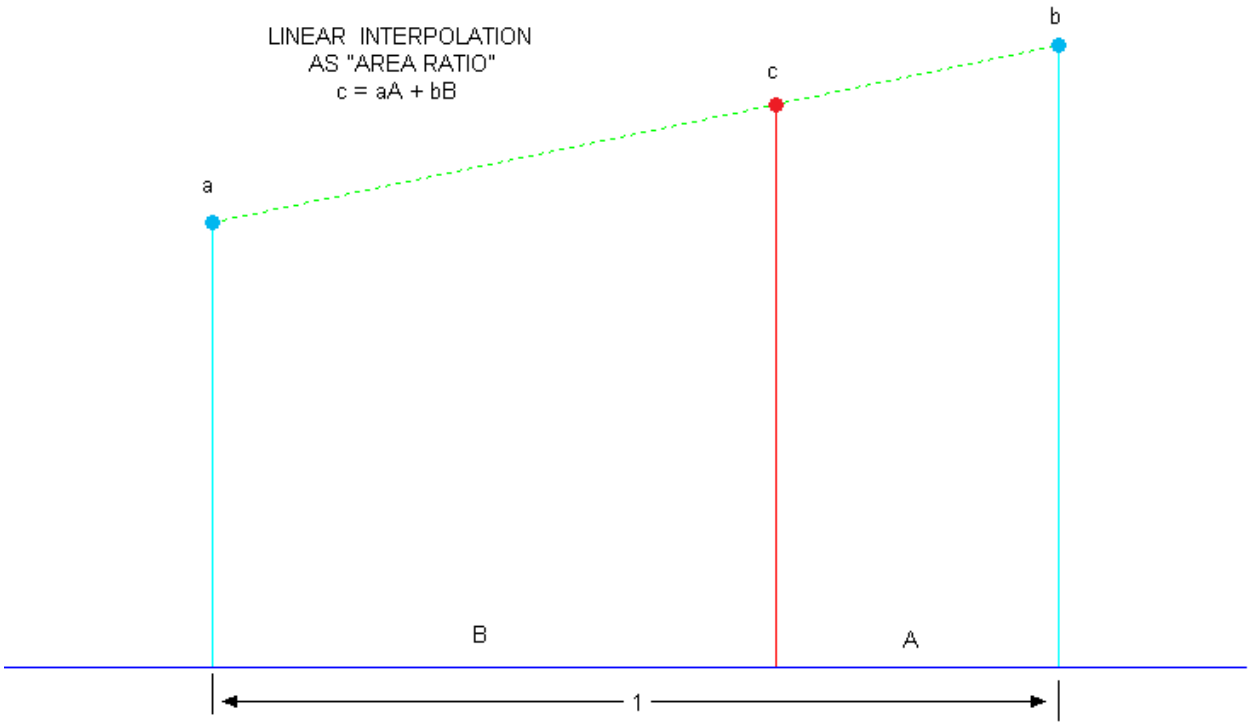



Fig. 7 Linear Interpolation in One Dimension

where the original samples are separated by a distance of A + B = 1. This is just simple 1D linear interpolation of course. The "area ratio" interpretation is that the weight given to an original point is the "area" that is, when viewed from the original point, "behind" the interpolation point, divided by the total area. Of course, here the notion of "area" is just that of length. Thus we have equation (4), which we have already used for the edges of triangles above.

The result of this interpretation depends on how we might choose to define "area behind". Fig. 8 shows 2D sampling by 3 in two directions, and here we are considering the square sampling cell. Interpolation of the points on the sides of the cell are just linear interpolation as before. For example, the point (1,3) is 2/3 of the point (0,3) plus 1/3 or the point (3,3), and so on.

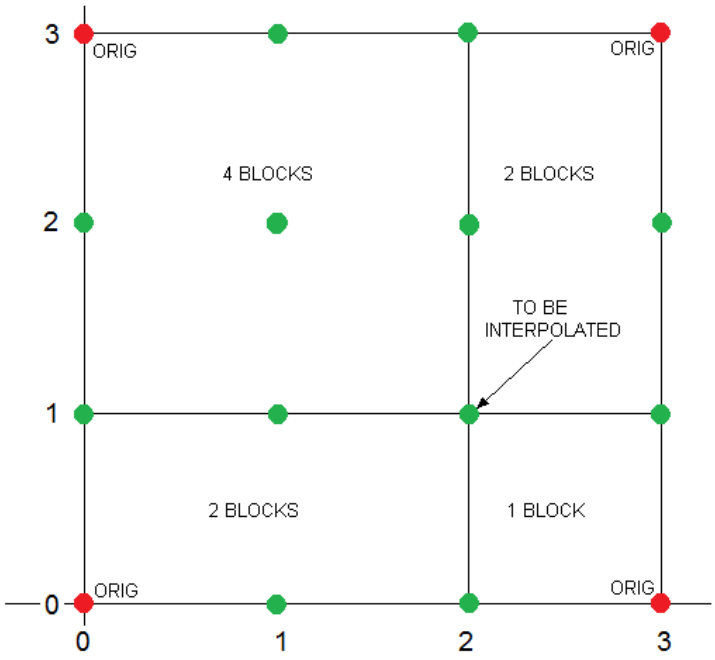



Fig. 8

Red points on corners are originals. The green points on the boundaries and in the interior are to be interpolated (one typical is on the crossing lines).

The edges of the square reduce to linear interpolation along the edges by the area ratio. The edges are common to neighboring squares which give the same linear interpolation on the edges.

The tricky points are the interior ones, such as the point (2,1) which is at the crossing point of the subdivision shown. Looking from the point (3,0), at the point (2,1), the area behind it is 4 blocks of 9 total. The ratio is 4/9. Looking at (2,1) from the point (0,3), the area behind it is 1 block of the 9 total, for a ratio of 1/9. While less well defined, the area behind the point (2,1) looking from the point (3,3) is 2 blocks. This is the rectangle that has corners (0,0), (0,1), (2,0), and (2,1). Thus the ratio is 2/9. Likewise, the weight for the point (0,0) is 2/9.

In the same manner (See Section 6 just below) in which we slid papers around to obtain the impulse responses for earlier examples, we obtain here the impulse response:

$$\begin{array}{ccccc} 1/9 & 2/9 & 3/9 & 2/9 & 1/9 \\ 2/9 & 4/9 & 6/9 & 4/9 & 2/9 \\ 3/9 & 6/9 & 9/9 & 6/9 & 3/9 \\ 2/9 & 4/9 & 6/9 & 4/9 & 2/9 \\ 1/9 & 2/9 & 3/9 & 2/9 & 1/9 \end{array} \qquad \textbf{(5)}$$

This is not only a neat result, but it is the outer product of the 1D impulse response, [ 1/3 2/3 3/3 2/3 1/3 ], which is 1D linear interpolation, with itself.

It is left to the reader to show that for sampling by 2 in both directions, this same area ratio gives the impulse response of equation (2), which was our modified version of piecewise planar. This is also the outer product of the linear impulse response [ 1/2 1 1/2 ] with itself.

## 6. CONVOLUTION EXAMPLE

What are we really doing here? The 5x5 matrix of equation (5) above will serve as our example here. This we choose as being the result of the area ratio method, and the outer product of 1D linear interpolation. A similar approach applies to other impulse responses.

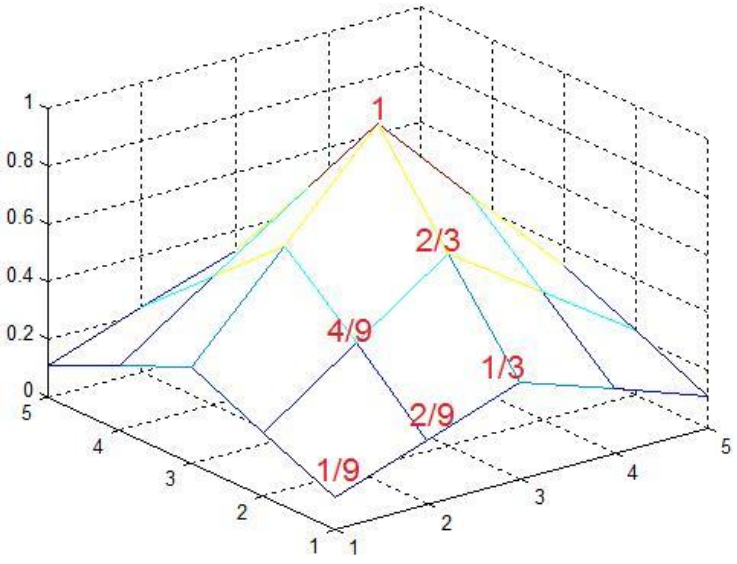



Fig. 9

The matrix of Equation (5) as a mesh plot. This is the 2D impulse response used to interpolate by 3, which was the result of several procedures.

This will be our illustration.

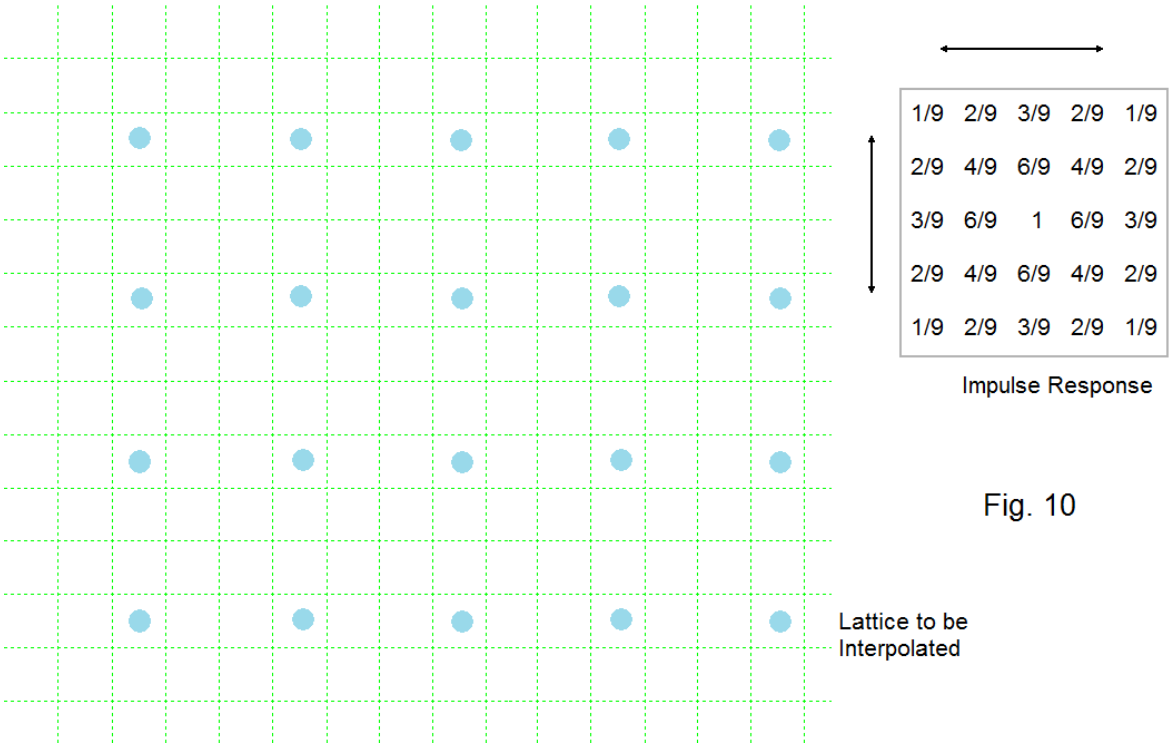



Fig. 10

To do the 2D convolution (filtering) we will of course use computer code (Matlab in this case). For hand examples, and to get the basic ideas, we can use the device of Fig. 10, which we have suggested above, not just as a means of doing a convolution, but as a way of obtaining the impulse response values in the first place.

In Fig. 10, the blue dots on the green grid represent pixels for which we have the given non-zero values. The empty pixels have value zero at the moment. Note that there are eight unknown (zeroed) pixels for every known one. This corresponds to the case of sampling by 3 in both directions (as in Fig. 2), or of upsampling by 3 in both directions in anticipation of interpolation (as in enlarging an image). In either view, our goal is to put acceptable numbers in the empty cells. When we place the 5x5 “impulse response” over the lattice (sized exactly as in Fig. 10), it covers either one, two, or four blue dots, with the remainder of the 25 being zero. But there are only six typical cases here, as shown in Fig. 11. More generally, we could say there are only three unique cases: (1) center of the impulse response on a blue dot, (2) the center horizontally or vertically displaced one unit from a blue dot, or (3) the center diagonally displaced one unit from the blue dot.

The top row of two cases in Fig. 11 shows examples where the impulse response is centered on positions that are horizontal to a given blue dot. The analogous situation is seen when we move vertically with regard to a blue dot. Thus we find 8 of the 25 cases of this type, each overlapping two blue dots. The middle row of two in Fig. 11 shows cases where the impulse response is centered diagonally with respect to a blue dot (another 8 cases), each involving four blue dots. In the third row, left, we show a case where the center is displaced from a blue dot over two and up one. Equivalently this is

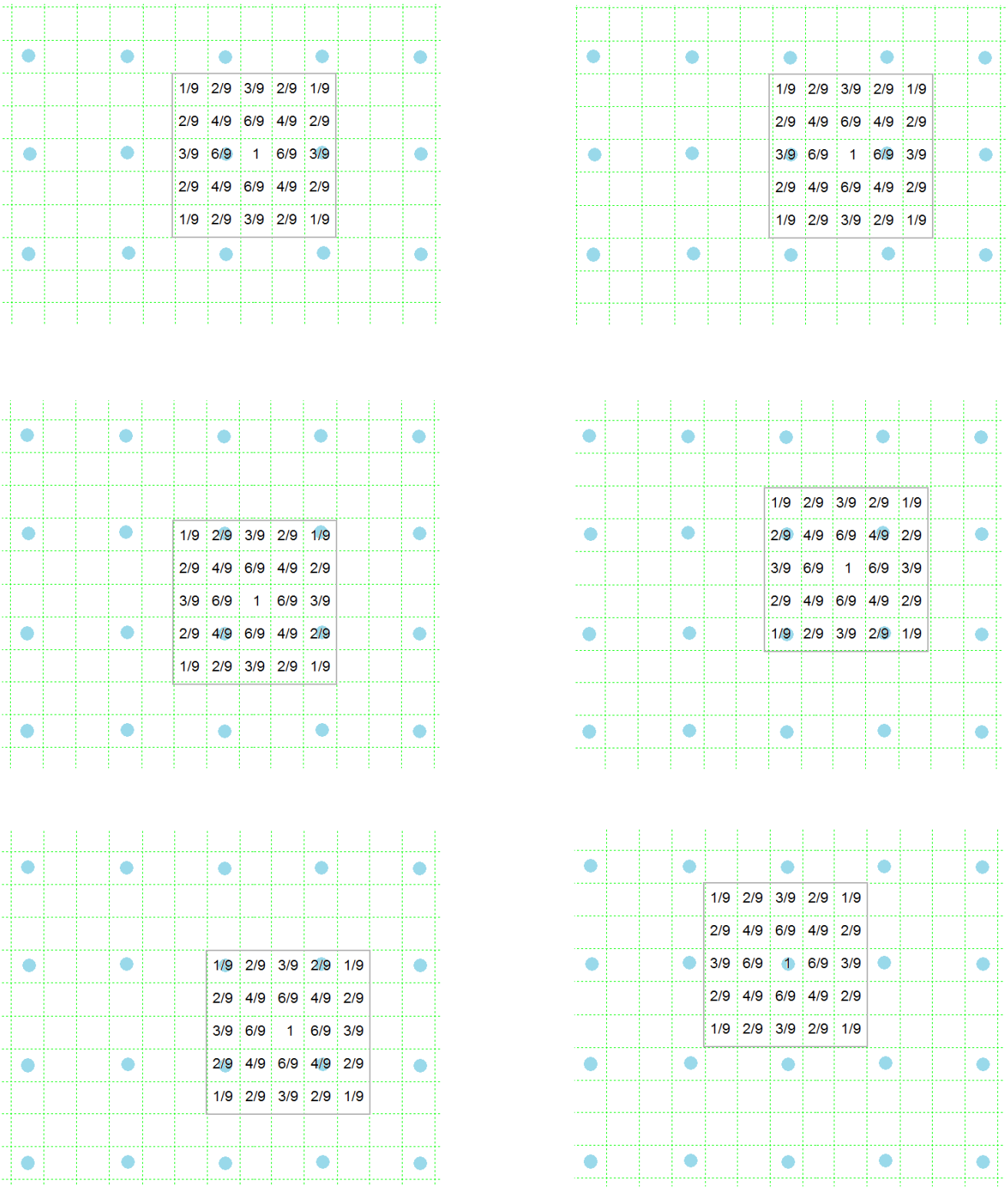



Fig. 11 Examples of the impulse response sliding over the lattice to be interpolated. Note that the weights (fractions over the blue dots) always sums to 1.

just diagonally displaced from the next blue dot over. There are also 8 of these cases which involve four blue dots. The final case, the bottom right, is the case of centering on a blue dot. There is just one such case. The weight being 1 in this case clearly means that the non-zero pixel is retained – a good test of any interpolator. That brings us to the 25 total. Note that in this case the sum of the weights for any position of the impulse response is 1, like the weight with the impulse response over a given pixel. We shall always need to look to see if this happens.

For any of these, what we do is compute the sum of the products of the pixel value times the overlying impulse response values. In this case, at least 21 of the 25 products are zero due to the zeroed pixel values. The sum becomes the value of the pixel over which the impulse response is centered. Sliding the impulse response matrix over each pixel in this way is what convolution is. (But it is best to think of the sum as assigned to a whole new lattice so that we don't suppose that interpolated values are subsequently included in the interpolation process when we shift the impulse response. Another point is that the new lattice will be larger than the original, since the impulse response overlaps the edges. These newly generated edges are often removed, reducing the image to its original size. We shall ignore edge effects here.)

Perhaps the most important thing to realize is that an interpolated pixel, as calculated by this process, is a weighted combination of the non-zero values, in the manner agreed upon for our interpolation procedure. In turn, this suggests how we can (and usually do) determine the impulse response matrix by this same procedure. We would start with a blank impulse response, overlay a particular pixel, and write in the weights we desire, shifting until the impulse response is fully populated, or until the pattern is evident.

## 7. TEST RUNS

For this section we want to run some test programs with the interpolators derived so far. Here we will be using a smaller version of the lena image. We will be downsampling (and slightly cropping) to 111x111 pixels. This means that we will in general not be looking to obtain excellent results even in the most favorable cases. One problem with many experiments with images is that often the results do all tend to be very similar, usually fairly good, and it is difficult to say that one is better than another. (Further, results as displayed on a computer screen and/or printed on paper by a printer, involving their own processing, can further confounded the study.) With our smaller image, we are looking to see if making a change in our interpolation procedures makes a visible difference in the result. This will be generally evident.

The starting image here is Fig. 12a, but keep in mind that we immediately discard $8/9^{th}$ of the information by sampling by 3 in both directions (Fig. 12b). Our goal is not to get back to something close to Fig. 12a, but rather, something that is better than the highly pixelated version, Fig. 12c, which was obtained from 12b by convolving with a "hold" (a 3x3 matrix of ones).

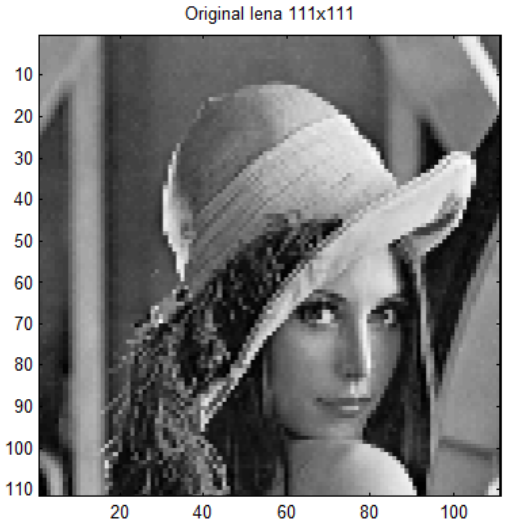



Fig. 12a

This figure corresponds to Fig. 1 except instead of 240x240 it is 111x111, and slightly cropped. If you are not familiar with the effect, try viewing this (and Fig. 1) from different distances. This image has 12,321 pixels while the one in Fig. 1 has 57,600 pixels.

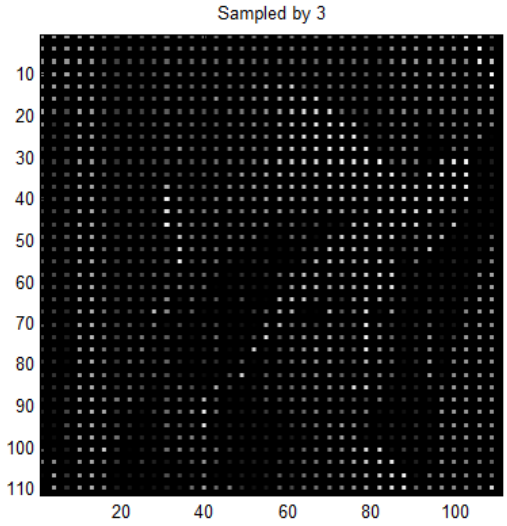



Fig. 12b

Here we have sampled by 3 in both directions, so there are 111/3=37 pixels in each direction, for 1369 pixels total. It is dark because eight of nine of the pixels from Fig. 12a which were not in general dark now are completely black. Note well that it is this image, not Fig. 12a, from which the images that follow are interpolated. Compare to Fig. 2.

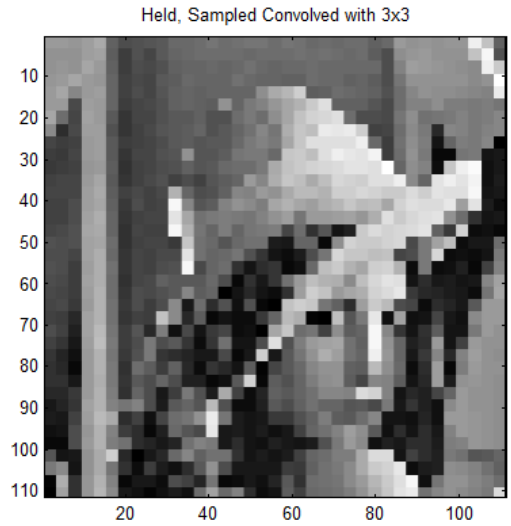



Fig. 12c

Here we have convolved Fig. 12b with a 3x3 matrix of ones. This means that the pixels of Fig. 12b are repeated nine times total, and the brightness is restored.

This is the most basic interpolation. No new data are added here however. Compare to Fig. 4.

We first try an interpolation of Fig. 12b using our area ratio method. Essentially we are just coding up the procedure discussed surrounding Fig. 9, Fig. 10, and Fig. 11. The result is shown in Fig. 13. We immediately recognize this as an improvement over Fig. 12c.

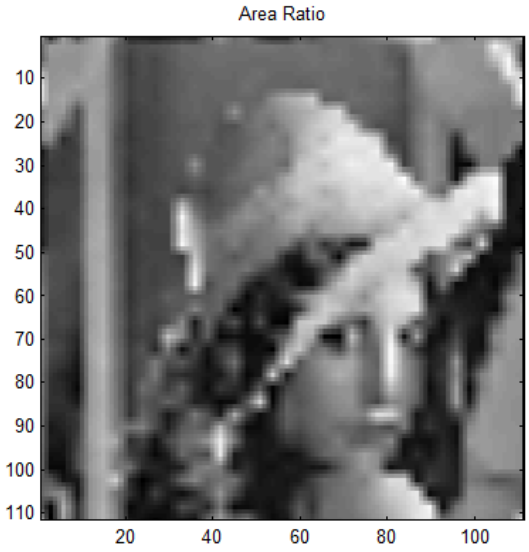
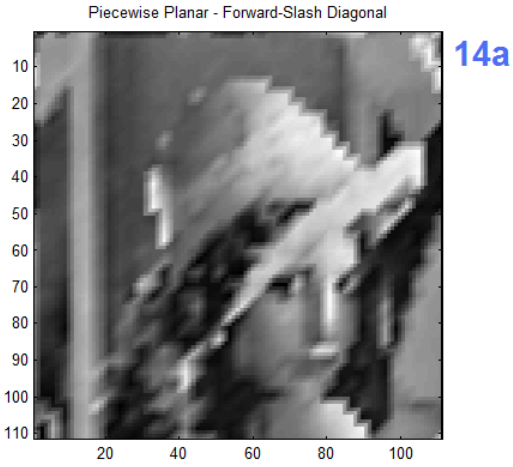


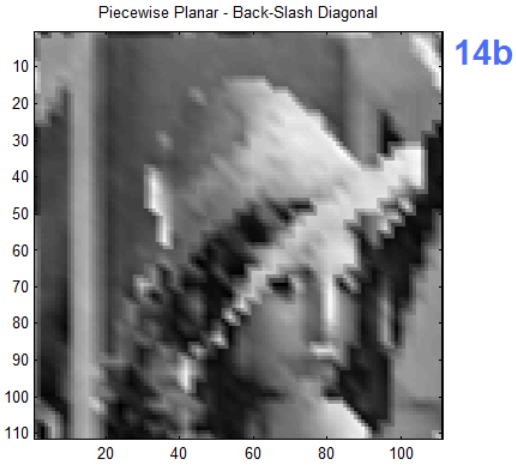

Fig. 13

Using the area ratio method Fig. 12b is interpolated giving this result. It is of course better than Fig. 12b (being brighter) but is also seen to be better than the held version, Fig. 12c.

We also looked at the method of piecewise planar. This we want to look at for two cases. Recall that for piecewise planar there was a choice of which of two diagonals of a square we used to divide the cell to be interpolated into two triangles. In presenting the two results from the two choices, we want to compare to the area ratio method of Fig. 13, and also to see if there is any difference between the choice of diagonal in piecewise planar. It might seem that any difference should perhaps be subtle, if any. Fig. 14 shows the results, using piecewise planar, on the image.

A very quick glance at Fig. 14a and Fig. 14b gives the correct impression that the piecewise planar is not as good as the area ratio (Fig. 13) at least on this particular evidence. (Keep in mind that we are using very small amounts of information to make all errors more apparent here.) A slightly closer look indicates that the two images are also different. In particular, compare the front of Lena's hat, and her shoulder, with the brim of the hat in the two cases. There is a drastic difference between these features in the two cases - for these items which are tilted something like, roughly, ±45°. There is little difference between lines that are horizontal or vertical. Keep in mind that the difference between the two is not which triangle is being used - green or blue in Fig. 6 - as indeed, both are used. Rather it is how the square was divided – which diagonal. We had the choice of what we can call a "forward-slash diagonal" or a "back-slash diagonal". The impulse responses are differently oriented in the different cases (See Fig. 14 c, d, e, and f). It would be unusual if we didn't suspect that this was the explanation.



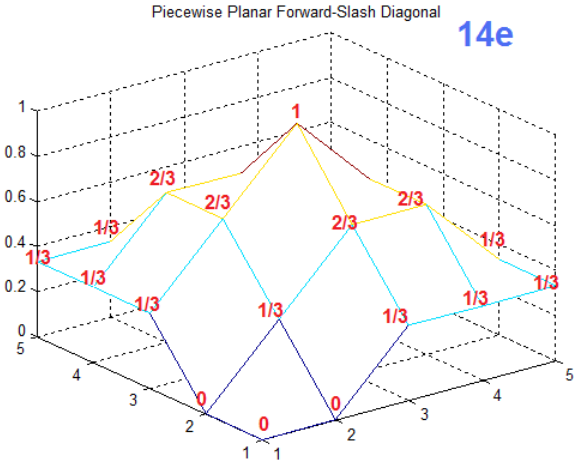




| | | | | |
|---|---|---|---|---|
| 0 | 0 | 1/3 | 1/3 | 1/3 |
| 0 | 1/3 | 2/3 | 2/3 | 1/3 |
| 1/3 | 2/3 | 1 | 2/3 | 1/3 |
| 1/3 | 2/3 | 2/3 | 1/3 | 0 |
| 1/3 | 1/3 | 1/3 | 0 | 0 |

14c

| | | | | |
|---|---|---|---|---|
| 1/3 | 1/3 | 1/3 | 0 | 0 |
| 1/3 | 2/3 | 2/3 | 1/3 | 0 |
| 1/3 | 2/3 | 1 | 2/3 | 1/3 |
| 0 | 1/3 | 2/3 | 2/3 | 1/3 |
| 0 | 0 | 1/3 | 1/3 | 1/3 |

14d



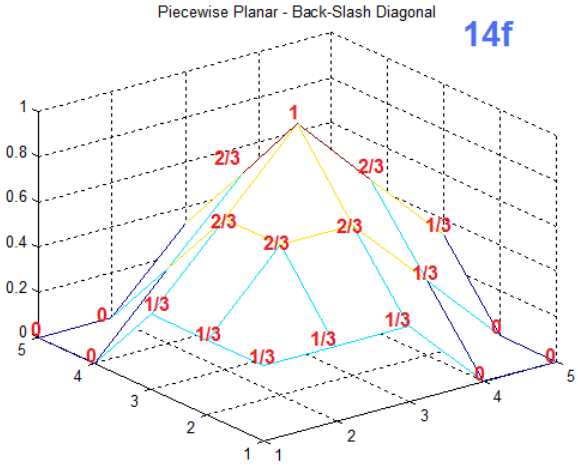
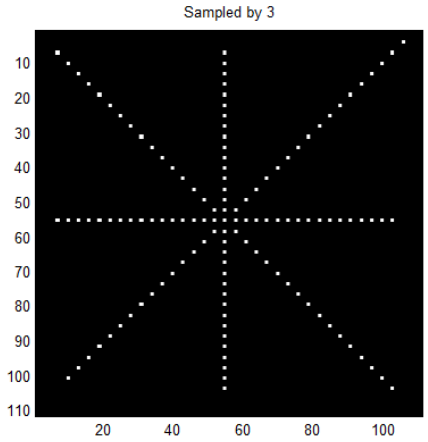



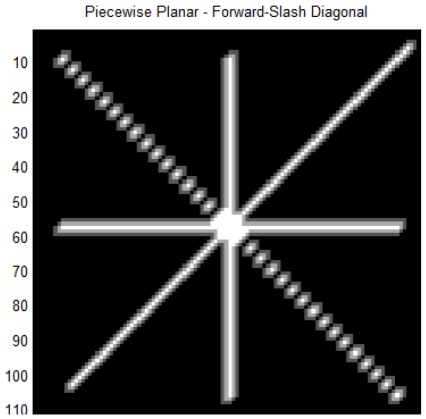

Fig. 14 Left Side: forward-slash diagonal. Right Side: back-slash diagonal. Note the slight elongation of the impulse response in the direction of the diagonal in the two different cases. Keep in mind that the axes start in the upper left corners for the images and for the matrix representations of the impulse responses, but as shown for the mesh plots of Fig. 14e and 14f.



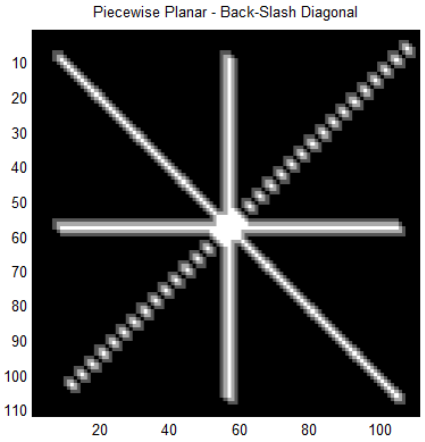

15a

Fig. 15

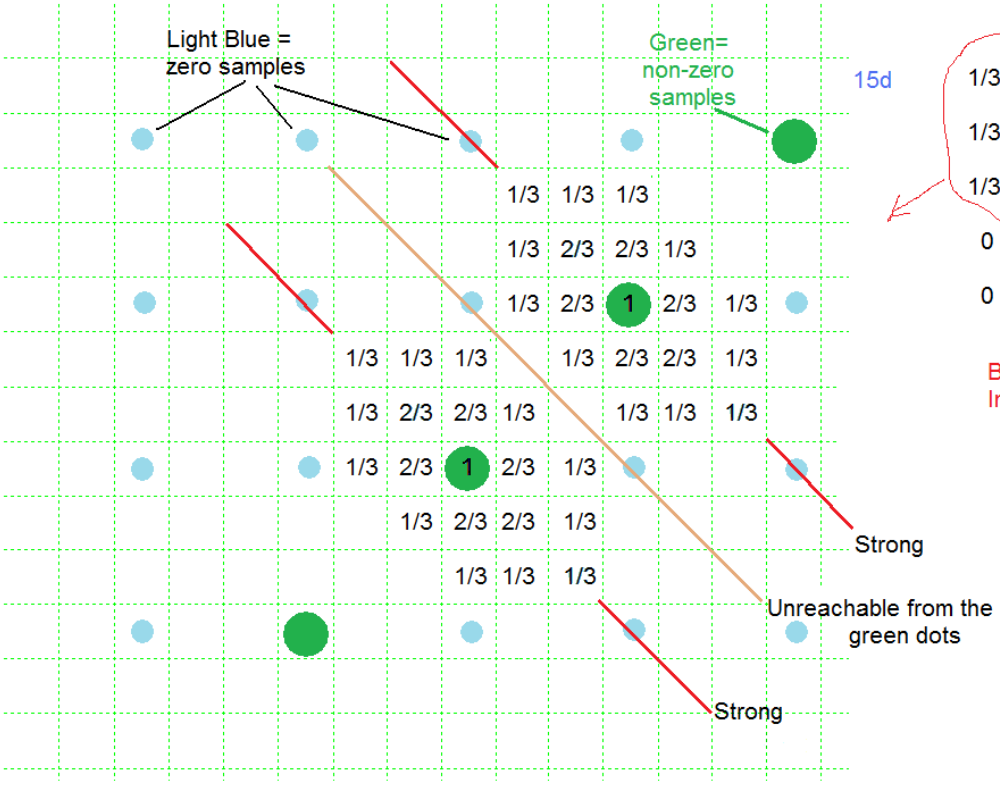
Here we have constructed an artificial "image" consisting of crossed lines at 45 degree angles. Fig. 15a shows the sampled version (sampled by 3 in both directions). Interpolating this with the forward-slash and the back-slash impulse responses (Fig. 15b and Fig. 15c) we see one of the 45 degree lines nicely interpolated while the other is segmented.



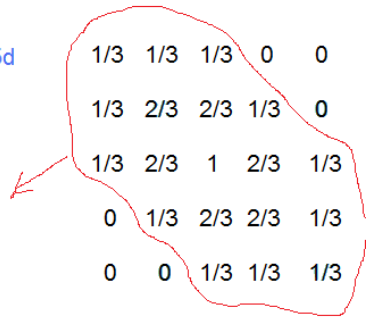

15b




15c




15d




Back-Slash
Impulse Response

Intuitively we understand that convolution “blurs” individual points (and lines) into larger areas, impressing the general shape of the impulse response on each such feature. It is helpful here to create an artificial image to test. (It was fortunate that the ubiquitous lena image already had near 45 degree angles). For Fig. 15, we made such an image, a crosshair with a superimposed crosshair rotated 45°. Fig. 15a is not this image, but rather the sampled by 3 version, which is the basis for interpolation. From Fig. 15b and Fig. 15c, it is clear that the two versions of the piecewise planar impulse response give different results, pretty much mirror images (indeed, the impulse response are mirror images). We see the expected blurring (interpolation) of the horizontal and vertical lines. The diagonal line that corresponds to the elongated direction of the impulse response is interpolated (and narrower than the horizontal and vertical lines), while the diagonal that is perpendicular to this elongation is broader, and segmented.

Fig. 15d shows clearly why this happens. Here we have our sampled lattice of blue dots, with the empty squares to be interpolated – however we assume that all the blue dots happen to be zero, except for those overlaid with green, which form our forward-slash line. We cut out the backward-slash impulse response (cropping out the zeros from the corners), and show this in two positions which reproduce two of the green dots. We thus achieve a strong support for the interpolation along the red lines. On the other hand, the empty pixels positions along the light brown line can not reach a green dot. Imagine the impulse response moving along this line. It can do so without ever overlapping a green dot (like driving a truck between two posts). Hence the segmentation.

This does indicate that we may need to pay attention to the general properties of an image to be interpolated when selecting a method, and perhaps adaptively choose different methods for different portions of the image. However at the moment we find ourselves considering if there is some way to combine the two cases of piecewise planar into one. The obvious choice would be the use an average of the two. This is just a matter of averaging impulse responses.

Fig. 16a shows this case where we have used an impulse response matrix (Fig. 16b) that is the average of Fig. 14c and Fig. 14d. Notice that this is now a matrix that has no zeros. The convolution procedure is no different from previous cases, and the image, Fig. 16a, is no more or less than the average of Fig. 14a and Fig. 14b. The general first impression is that it is better than either of the Fig. 14 cases, overall. It is true that both the good features and the bad features are averaged, so we have a trade-off of the type we often encounter.

We do not necessarily insist on comparing Fig. 16a to the Fig. 14 individual cases. Rather we are interested in how much better it is compared to the held case (Fig. 12c), and particularly, how does it compare to Fig. 13, the area ratio method. It is useful to put the two on the same page side-by-side, and this is shown in Fig. 17, Fig. 17a and Fig. 17b. The fact that they look very much alike is reflected in the fact that the two individual impulse response matrices (Fig. 17c and Fig. 17d) are very much alike and do not differ on any point by more than 1/18 (an absolute amount not a relative one)..

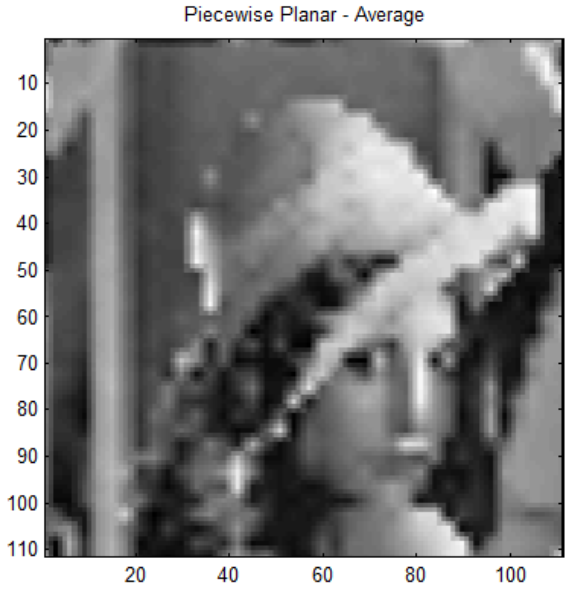



Fig. 16a

Fig. 16b

| | | | | |
|---|---|---|---|---|
| 1/6 | 1/6 | 1/3 | 1/6 | 1/6 |
| 1/6 | 1/2 | 2/3 | 1/2 | 1/6 |
| 1/3 | 2/3 | 1 | 2/3 | 1/3 |
| 1/6 | 1/2 | 2/3 | 1/2 | 1/6 |
| 1/6 | 1/6 | 1/3 | 1/6 | 1/6 |

**Impulse response is average of two choices of diagonal**

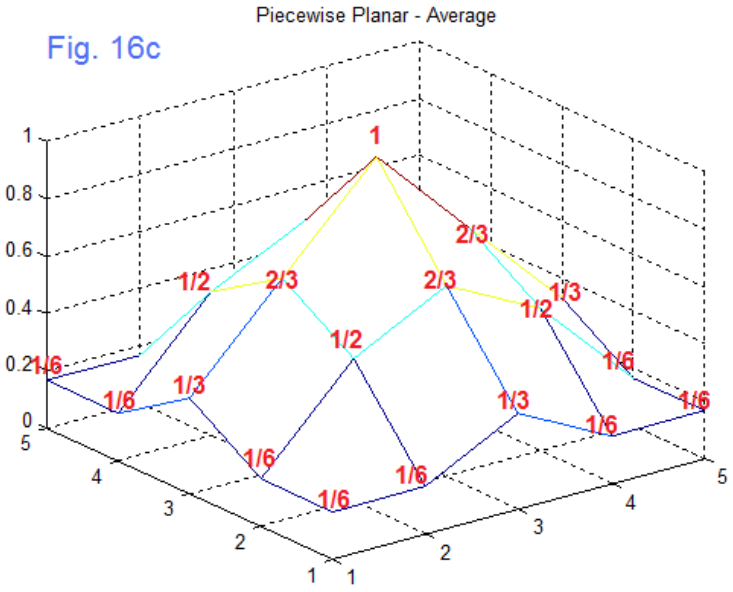
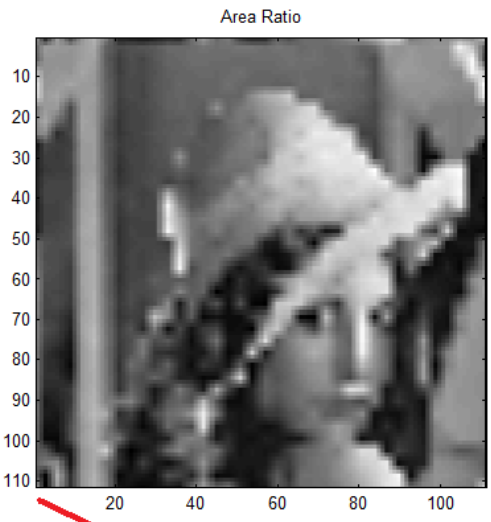



Fig. 16c

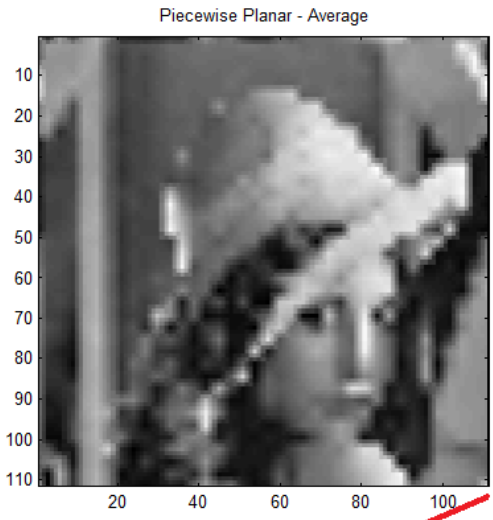
Fig. 16

Piecewise planar as the average of the two possible choices of diagonal that divides the square into two triangles

Further information on the similarity between the area ratio and the averaged piecewise planar is available if we actually compute the difference. This we could do by subtracting the images, or by subtracting the impulse responses and using this as a difference impulse response. Here we find the areas that the two methods tend to treat differently. Again, we learn that there are difficulties on the hat edges and the shoulder. It is not clear, however, how these differences are reflected in any features in the images.




17a




17b

| | | | | |
|---|---|---|---|---|
| 1/9 | 2/9 | 3/9 | 2/9 | 1/9 |
| 2/9 | 4/9 | 6/9 | 4/9 | 2/9 |
| 3/9 | 6/9 | 1 | 6/9 | 3/9 |
| 2/9 | 4/9 | 6/9 | 4/9 | 2/9 |
| 1/9 | 2/9 | 3/9 | 2/9 | 1/9 |

17d

17e

| | | | | |
|---|---|---|---|---|
| +1/18 | -1/18 | 0 | -1/18 | +1/18 |
| -1/18 | +1/18 | 0 | +1/18 | -1/18 |
| 0 | 0 | 0 | 0 | 0 |
| -1/18 | +1/18 | 0 | +1/18 | -1/18 |
| +1/18 | -1/18 | 0 | -1/18 | +1/18 |

| | | | | |
|---|---|---|---|---|
| 1/6 | 1/6 | 1/3 | 1/6 | 1/6 |
| 1/6 | 1/2 | 2/3 | 1/2 | 1/6 |
| 1/3 | 2/3 | 1 | 2/3 | 1/3 |
| 1/6 | 1/2 | 2/3 | 1/2 | 1/6 |
| 1/6 | 1/6 | 1/3 | 1/6 | 1/6 |

17f

17g

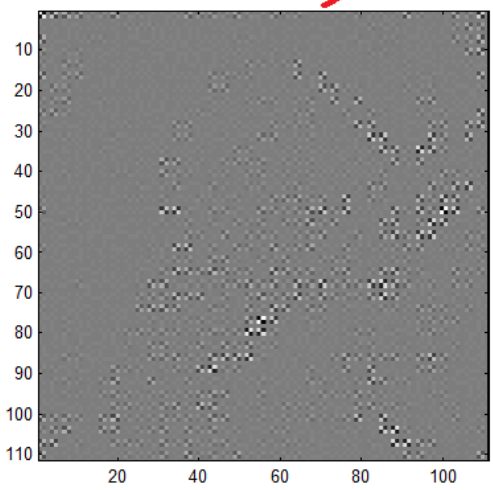



Fig. 17

Here we see that there is very little difference between area ratio and the average method of piecewise planar

# 8. PIECEWISE PLANAR WITH A LEAST-SQUARES RECTANGLE

For a rectangular lattice, the ideal "piecewise planar" object would seem to be a rectangle. The problem in fitting rectangles is that a plane is defined by three points, and not by four (in general). Hence we have used triangular interpolation surfaces. We can however fit a plane to four points in a manner which is approximate, by minimizing the squared error. This seems reasonable to try.

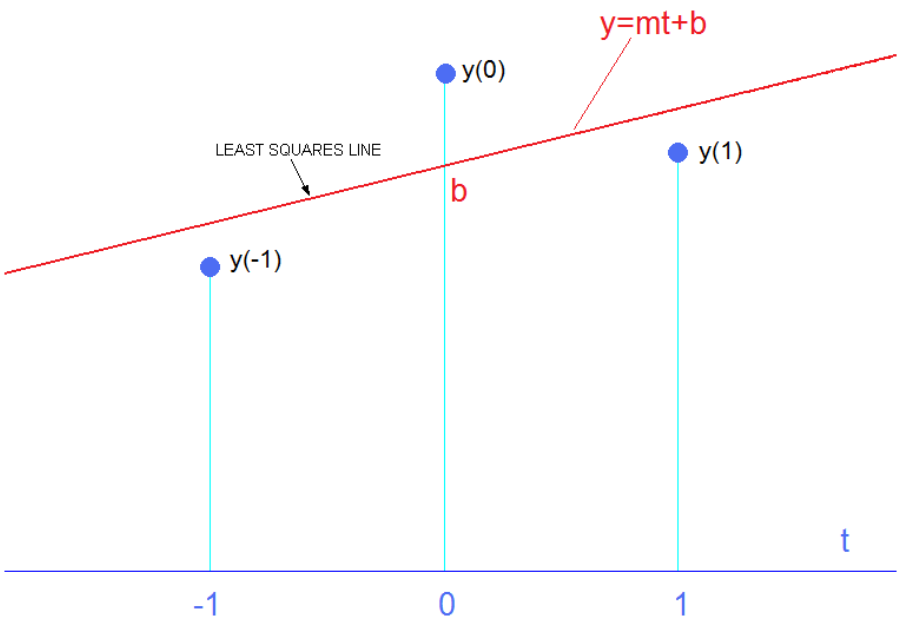



Fig. 18

A least squares line fit to three points

Recall the 1D case. This 1D case is a standard exercise [Application Note AN-318 "Time Domain Least Squares Low-Pass Filter," Feb. 1992] (Fig. 18) and the reader will recall that one assumes a form for the line (y=mt+b), computes the error on the three points, squares these errors and sums them, differentiates the sum with respect to m and b, setting the derivatives to zero, and solves the two equations for m and b. In fact, here,

$$m = [\, y(1) - y(-1) \,] / 2 \qquad (6)$$

and

$$b = [\, y(-1) + y(0) + y(1) \,] / 3 \qquad (7)$$

so the slope is determined by the outside points, and the intercept is the average of all three points. Note that the magnitude of the error is larger on the middle point. If we were to make this smaller (move the line up) it would make the error on both ends larger. The total squared error would be greater.

Fitting the plane is much the same. Fig. 19 shows four pixel samples at the points (0,0), (1,0), (0,1), and (1,1) in what we can call the $x_1$—$x_2$ plane. The values of these pixels are $y_{00}$, $y_{10}$, $y_{01}$, and $y_{11}$ respectively. We seek a plane that minimizes the squared error at the four corner points. That is, we know we can't place the plane exactly through more than three points in general, but we can make a least square fit to four (or more).

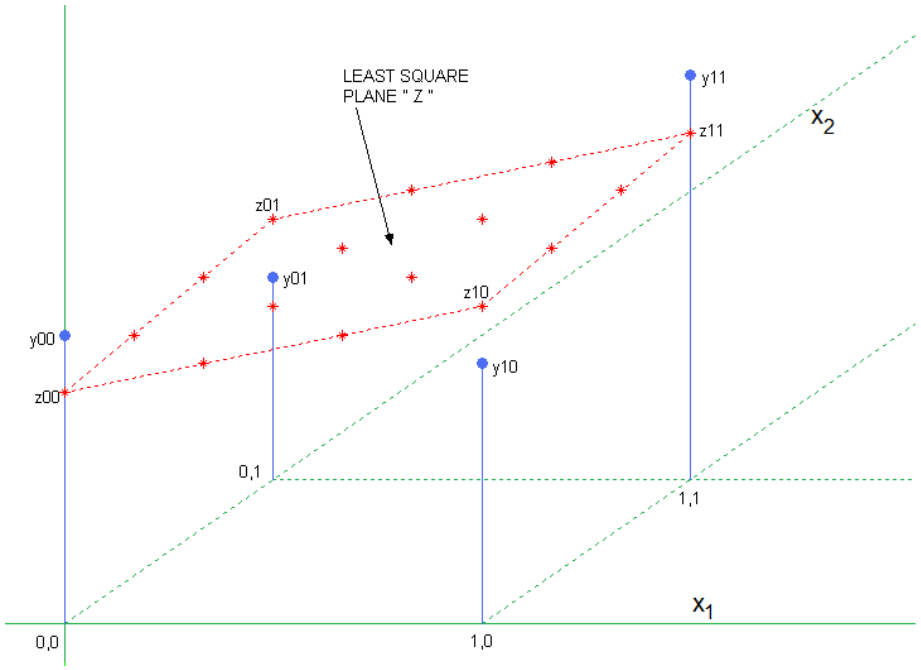



Fig. 19

Least squares fit of a plane to four points.

Note: The points y00, y10, y01, and y11 are not in general in a plane.

The points z00, z10, z01, and z11 are in the plane determined to minimize the squared error on all four corners.

The least squares plane is shown in terms of the rectangular region (red) above the square. The corners are the points $z_{00}$, $z_{10}$, $z_{01}$, and $z_{11}$ as shown. In addition, 12 other points on the plane are shown as stars. These 12 certainly represent interpolated points, although here even the corner points should be regarded as interpolated. We are not retaining any original points (in general).

The equation of the plane to be fit is:

$$z(x_1,x_2) = m_1x_1 + m_2x_2 + b \tag{8}$$

where $m_1$ and $m_2$ are slopes and b is the intercept at (0,0). There are errors at each of the four corners:

$$e_{00} = y_{00} - z(0,0) = y_{00} - b \tag{9a}$$

$$e_{10} = y_{10} - z(1,0) = y_{10} - m_1 - b \tag{9b}$$

$$e_{01} = y_{01} - z(0,1) = y_{01} - m_2 - b \tag{9c}$$

$$e_{11} = y_{11} - z(1,1) = y_{11} - m_1 - m_2 - b \tag{9d}$$

We next form

$$E^2 = e_{00}^2 + e_{10}^2 + e_{01}^2 + e_{11}^2$$

$$= y_{00}^2 + y_{10}^2 + y_{01}^2 + y_{11}^2 + 4b^2 + 2m_1^2 + 2m_2^2$$

$$- 2by_{00} - 2by_{10} - 2by_{01} - 2by_{11}$$

$$- 2m_1y_{10} - 2m_1y_{11} - 2m_2y_{01} - 2m_2y_{11}$$

$$+ 4m_1b + 4m_2b + 2m_1m_2 \qquad (10)$$

Differentiating with respect to the parameters $m_1$, $m_2$, and b

$$\partial E^2/\partial m_1 = 4m_1 - 2y_{10} - 2y_{11} + 4b + 2m_2 = 0 \qquad (11a)$$

$$\partial E^2/\partial m_2 = 4m_2 - 2y_{01} - 2y_{11} + 4b + 2m_1 = 0 \qquad (11b)$$

$$\partial E^2/\partial b = 8b - 2y_{00} - 2y_{10} - 2y_{01} - 2y_{11} + 4m_1 + 4m_2 = 0 \qquad (11c)$$

In matrix form

$$\begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & 2 \\ 2 & 2 & 4 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ b \end{bmatrix} = \begin{bmatrix} y_{10} + y_{11} \\ y_{01} + y_{11} \\ y_{00} + y_{01} + y_{10} + y_{11} \end{bmatrix} \qquad (12)$$

or symbolically

$$M\,a = Y \qquad (13)$$

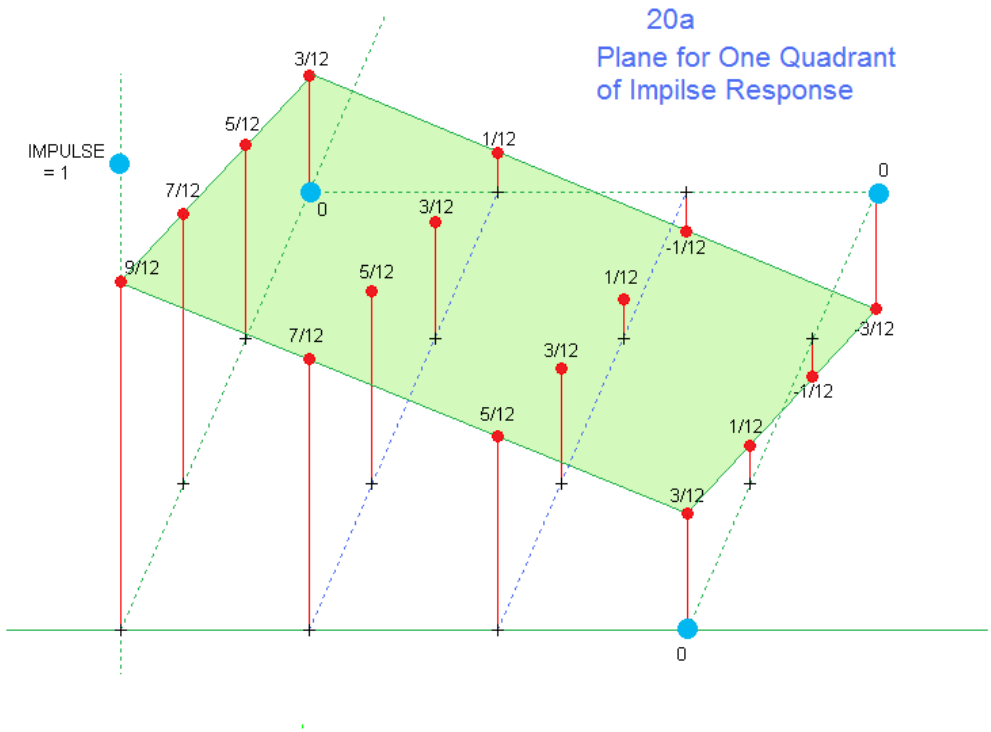
where a is the vector of parameters [$m_1$ $m_2$ b].

Hence $a = M^{-1}Y$, or

$$\begin{bmatrix} m_1 \\ m_2 \\ b \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1/2 \\ 0 & 1 & -1/2 \\ -1/2 & -1/2 & 3/4 \end{bmatrix} \begin{bmatrix} y_{10} + y_{11} \\ y_{01} + y_{11} \\ y_{00}+y_{01}+y_{10}+y_{11} \end{bmatrix} \qquad (14)$$

This is the answer. From the four given values of $y(x_1,x_2)$ on the corners, we obtain $m_1$, $m_2$, and b and we can calculate the values of points on the resulting plane, $z(x_1,x_2)$. This we can do for any point $(x_1,x_2)$. For example, the 16 points marked with a star in Fig. 19 are on the plane $z(x_1,x_2)$. Here not only points in the interior or the plane, but points on the edges, and indeed, at the original corners are calculated. Note that the corner points are not in general the given values.

Because we are looking to realize this least-squares fit as a filter (it is well-known that this works in the 1D case), we need to find the impulse response. If an impulse occurs at the origin (0,0) we need to know the four planes of which the point of the impulse (0,0) is the center. Fig. 20a shows one such quadrant. This is the same procedure that led to Fig. 19, except for the specific case of the impulse here. Once we have one quadrant, we can just write down the full impulse response, Fig. 20b. The next thing to consider is – just how much of this should we use? Fig. 21 shows the choices, a 7x7 case or a 5x5 case. Perhaps it is easiest to just try these, and the results are shown in Fig. 22.



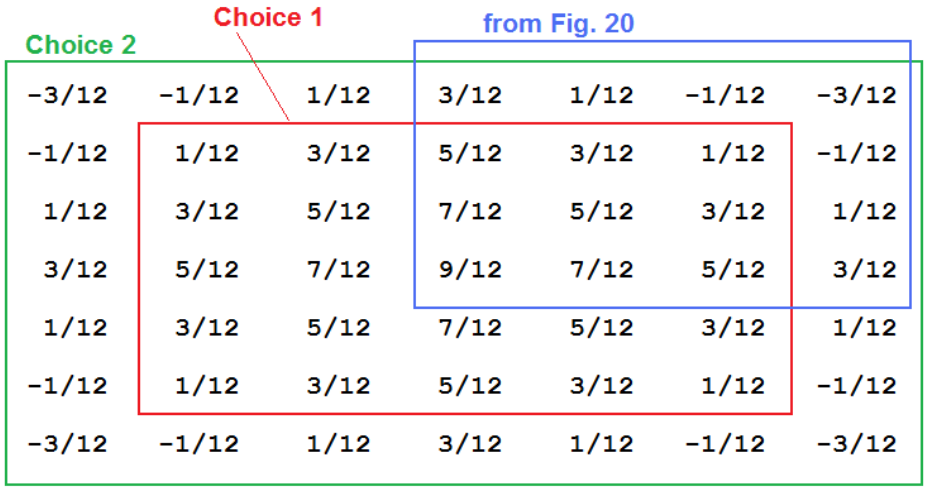

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| -3 | -1 | 1 | 3 | 1 | -1 | -3 | 20b |
| -1 | 1 | 3 | 5 | 3 | 1 | -1 | |
| 1 | 3 | 5 | 7 | 5 | 3 | 1 | |
| 3 | 5 | 7 | 9 | 7 | 5 | 3 | |
| 1 | 3 | 5 | 7 | 5 | 3 | 1 | |
| -1 | 1 | 3 | 5 | 3 | 1 | -1 | |
| -3 | -1 | 1 | 3 | 1 | -1 | -3 | |

Matrix of Full Impulse Reponse
(all to be divided by 12)

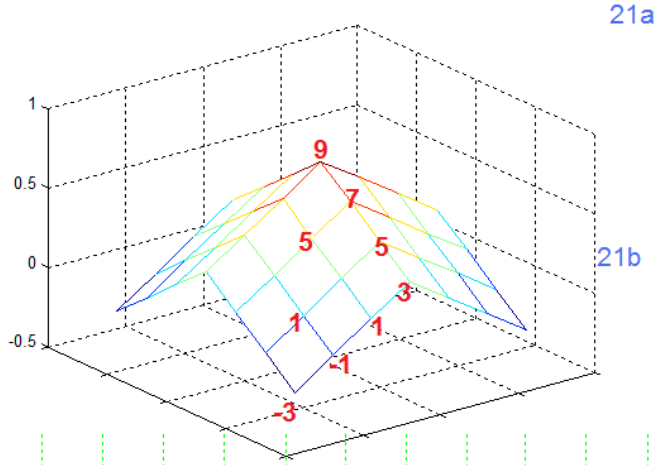
Fig. 20

We can fit a plane (light green) to an impulse (blue dots), as is shown for one quadrant.

The least squared error here is minimized, and in fact equalizes the error at the locations of all four blue points.




21a

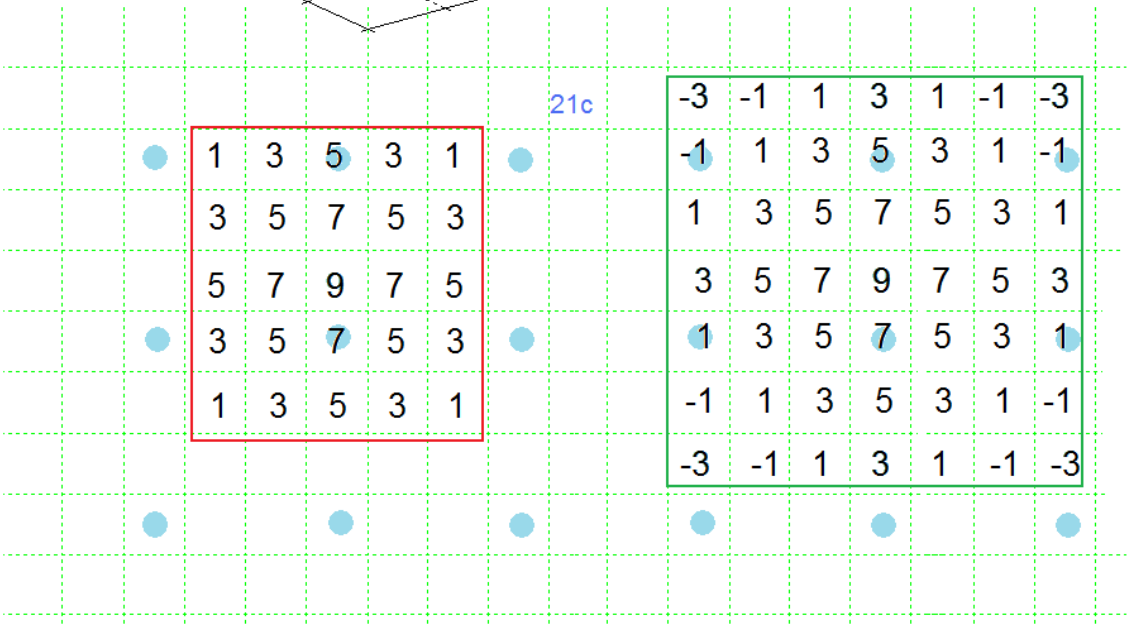



21b

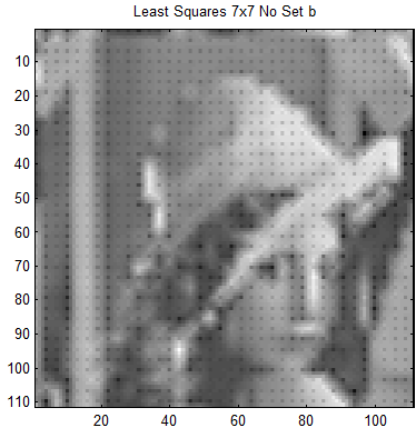
Fig. 21

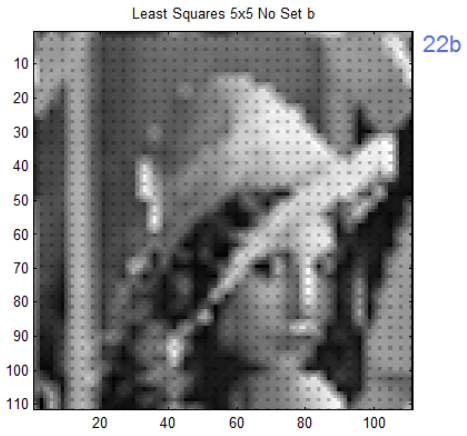
Choices of impulse response for least squares

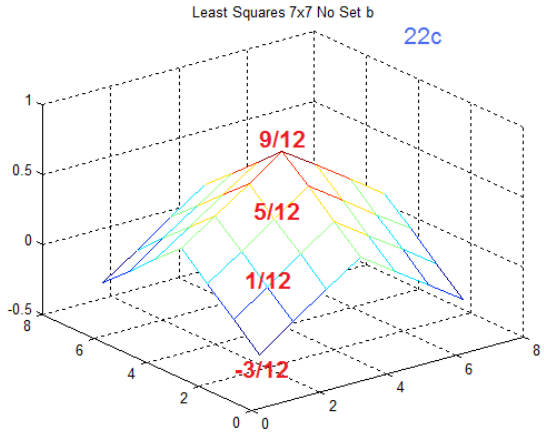
21c

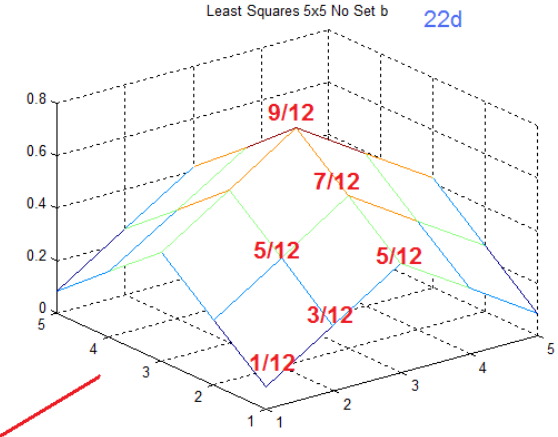



Neither the 7x7 nor the 5x5 case in Fig. 22 is satisfactory. Both have dark spots in the positions of the original pixels. We can attribute this to the fact that when we consider the weights the impulse response offers, the original pixels receive only 9/12 while the others receive 12/12 total (Fig. 22e).



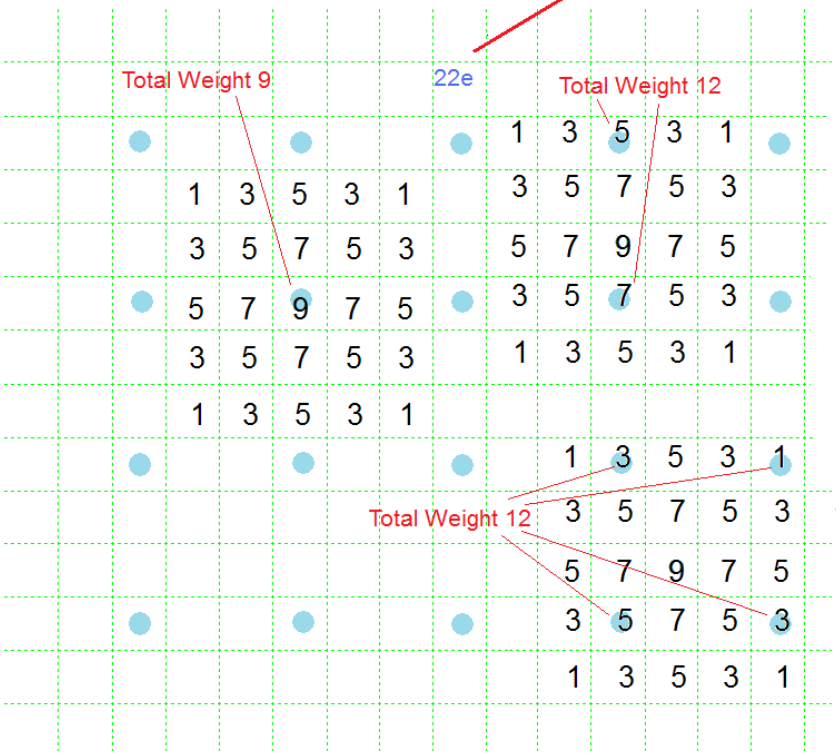




Fig. 22

First test shows darker dots in positions of original pixels.

Having seen that apparently our least squares procedure has given insufficient weight to the original pixels (total of 9 instead of 12) we might be able to fix this. After all, we argued above that one measure of a good interpolator is that it likely should give us back the originals unchanged. Just as clearly (for example Fig. 19), minimizing the squared error does not do this. So why not just change the weight from 9/12 to 12/12. Indeed – why not – this will turn out to be the right way to proceed. But, let's consider if we can make the math gives use the right answer. Why not solve the problem by constraining the solution to be 1 at the center?

This is a repeat of what we did in equations (8)-(14) except here we will simplify things as much as possible, and calculate only the impulse response quadrant. We have b=1 at (0,0), we recognize that the slope in both directions will be the same by symmetry, and the three remaining values to be fit [at (1,0), (0,1), and (1,1)] are all now zero. So equation (8) becomes:

$$z(x_1,x_2) = m(x_1 + x_2) + 1 \quad (15)$$

where m is the slope and b=1 is the intercept at (0,0). There are errors at each of the four corners:

$$e_{00} = 0 \quad (16a)$$

$$e_{10} = 0 - z(1,0) = -(m+1) \quad (16b)$$

$$e_{01} = 0 - z(0,1) = -(m+1) \quad (16c)$$

$$e_{11} = 0 - z(1,1) = -(2m+1) \quad (16d)$$

So we get:

$$E^2 = e_{00}^2 + e_{10}^2 + e_{01}^2 + e_{11}^2 = 6m^2 + 8m + 3 \quad (17)$$

Differentiating with respect to the parameters

$$\partial E^2/\partial m = 12m + 8 = 0 \quad (18a)$$

or

$$m = -2/3 \quad (18b)$$

Evaluating equation (15) at values of $x_1$ and $x_2$ from 0 to 3/3 at intervals of 1/3, we obtain the matrix values for the impulse response quadrant as:

$$\begin{matrix} 9/9 & 7/9 & 5/9 & 3/9 \\ 7/9 & 5/9 & 3/9 & 1/9 \\ 5/9 & 3/9 & 1/9 & -1/9 \\ 3/9 & 1/9 & -1/9 & -3/9 \end{matrix} \qquad (19)$$

In fact, comparing this to Fig. 20, we see that the numerators in the fractions in equation (19) are the same as the numerators in Fig. 20. In equation (19), the denominators are 9, while in Fig. 20, the denominators in Fig. 20 are 12. So, by choosing b=1,we did get the center to be 9/9=1. Do the dark spots go away? (Of course, we tried this immediately without thinking about it !) No they don't ! The result is the same as Fig. 22b. A minor factor in these being the same is that our image plotter (Matlab's ***imagesc***) scales images automatically. But the major things is that the impulse responses are identical except for a scale factor of 4/3.

It perhaps becomes clear that setting b to a particular value is only a scale. We had this one parameter to give. The relative values for the other three corners are the same. As we should have expected, by restricting one corner, the errors on the other three corners are larger (3/9 instead of 3/12). What remains to be done is to see if an *ad hoc* change of the middle value is successful, and if so, to attempt to say why this was the correct choice. So what happens if the 9 which appears in many places in Figures 20, 21, and 22 is replaced by 12, so that the center is weighted 12/12=1?

Fig. 23 shows the results. In Fig. 23a and Fig. 23c, we look at the 7x7 case. Fig. 23b, Fig. 23d, and Fig. 23e are the 5x5 case, which is a better result, and is therefore more completely considered. In both cases, the dark spots are gone, but the 7x7 case has less contrast and is subjectively less appealing. Not unlikely the much larger values on the corners in the 7x7 case (3/12) are working against us. Accordingly, we drop the 7x7 case.

We note that the 5x5 case is not only cured of its dark spots, but is in general, quite similar in performance to the previous methods. Fig. 23e in fact shows that we now have the same total weighting for the three typical cases.

We justify this approach, first of all, because it seems to work. But here is the possible point to make. When there is only the center pixel involved, should any other point be involved? Is not the plane with the least squared error for this case the plane that goes through the pixel value?

Fig. 24 gives us a summary (on a single page) of the methods so far, starting at the upper right with the held example, we move counter clockwise through the left side, and find the least squares cases to be very comparable to the piecewise planar (averaged case) and the area ratio. The bottom right shows us the unsampled case. One test is to get back, perhaps 15 feet from the page, and see if you can see any differences. I think you can still see that the held version is the poorest, but the three interpolated cases seem very similar at this distance.

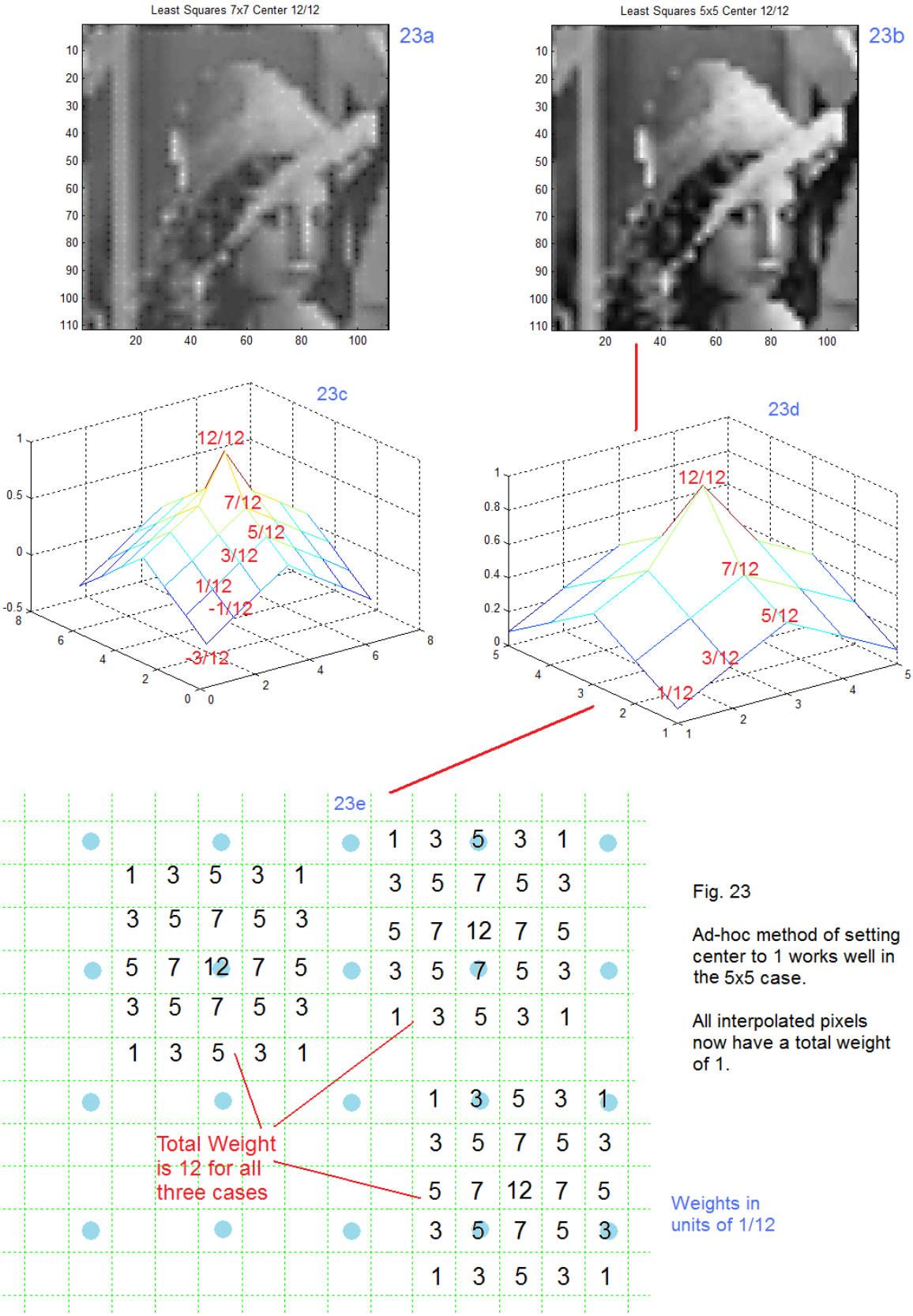
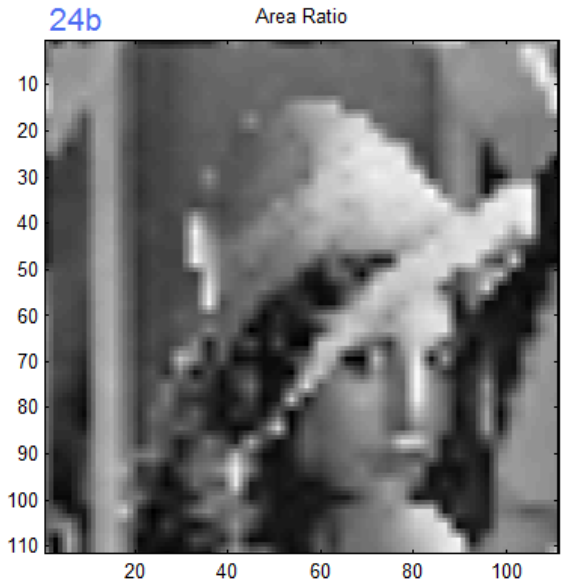
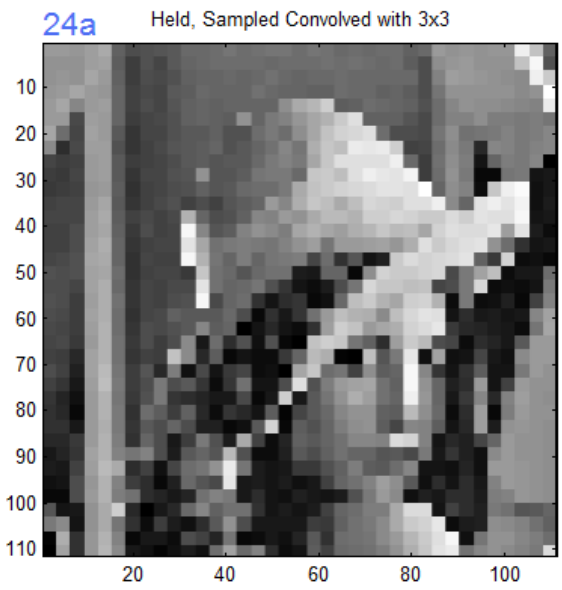



Fig. 23

Ad-hoc method of setting center to 1 works well in the 5x5 case.

All interpolated pixels now have a total weight of 1.



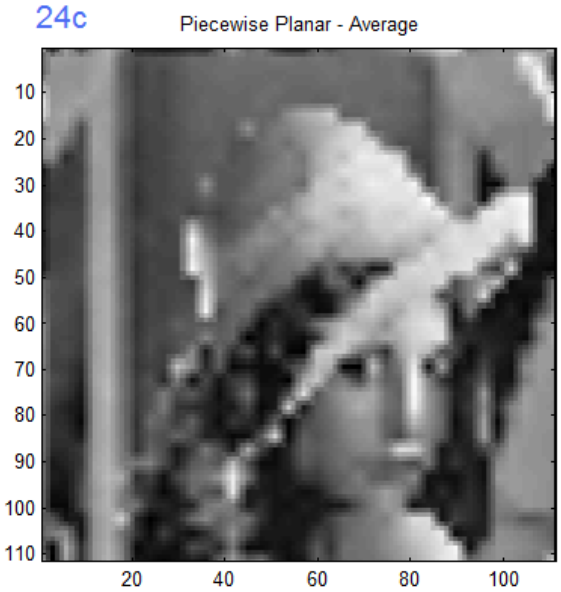
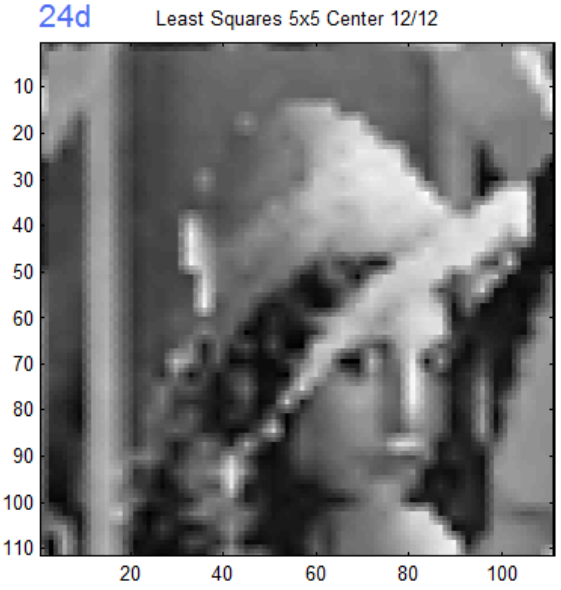




Fig. 24

Fig. 24e is the original (best) while Fig. 24a is the sample/held version (worst). Here the least square approach seems to be very much comparable to the other two interpolation methods we have investigated (all three on the left side)



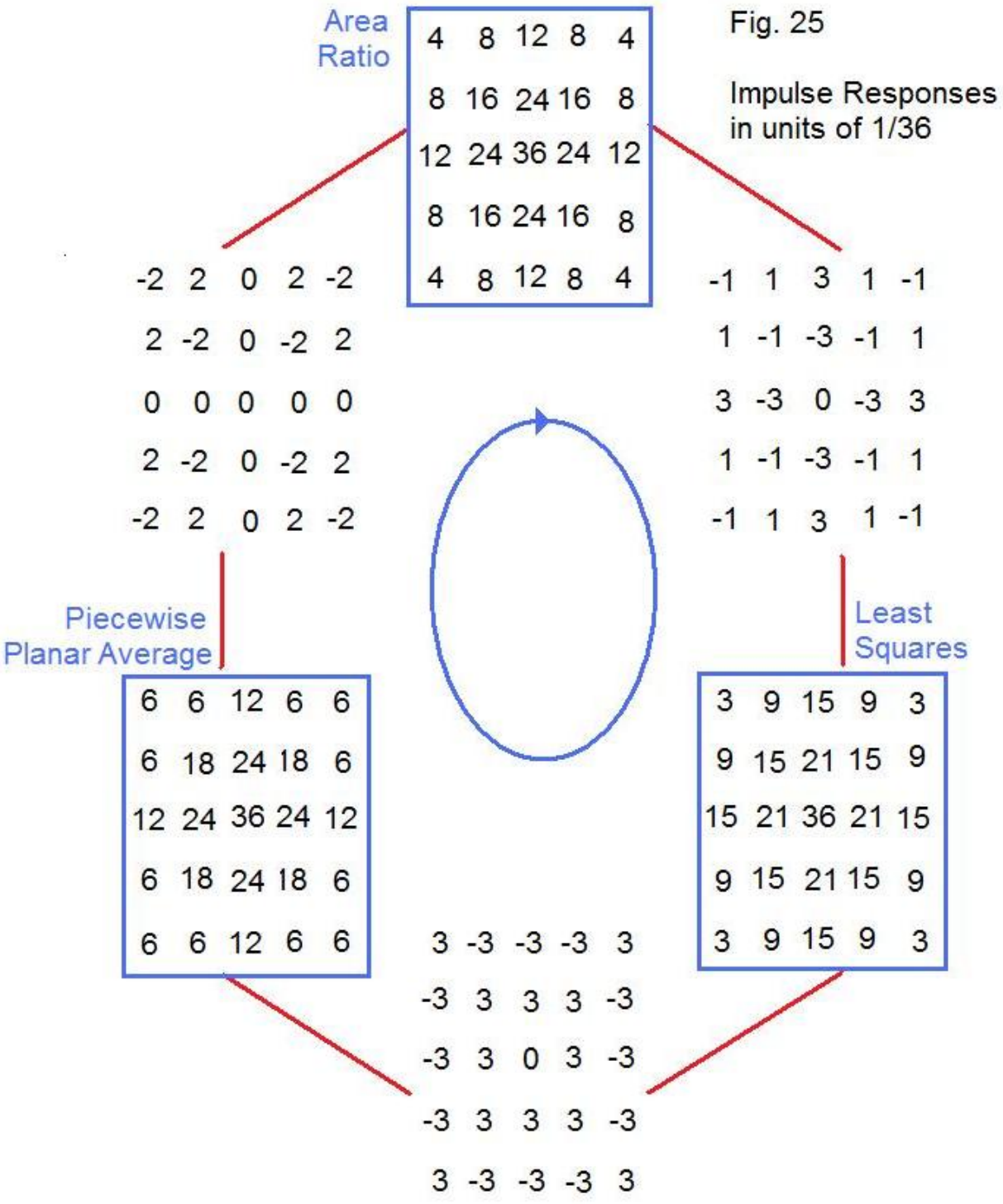




Fig. 25 shows that of the three methods, the matrices differ relatively little. None of these methods is getting close to a full recovery (of course, we do not expect that anyway) and we get similar results from all.

# 9. FREQUENCY DOMAIN APPROACHES

Working above in the spatial domain has taught us a lot about image interpolation. We had some good approaches, but two of them did need some modifications. Further we have a fairly good feeling for what a useful impulse response might look like – kind of tent-like surfaces. All this helps us think about frequency domain approaches.

Here are some things we consider:

(1) We need frequency-domain low-pass filters. Where should the cutoff frequency be? Roughly speaking, if the original spacing is 1, the sampling by 3 increases this spacing to 3. Thus the sampling frequency (in either direction) is 1/3, and a low-pass interpolator would need a cutoff below half this, or just below 1/6 (0.17).

(2) Our area ratio impulse response was (or could have been derived) from a one-dimensional linear interpolator by using the outer product. That is, the length 5 linear interpolator (1/3 2/3 3/3 2/3 1/3) was converted to the 2D case, equation (5). We know a lot about designing 1D filters.

(3) All three of our useful spatial methods resulted in a 5x5 impulse response for times 3 sampling.

(4) We have a strong preference for having the total weights being the same for all positioning of the impulse response over the sampled lattice, so as not to have periodic variations of intensity.

(5) What can we learn from the FFT's of the useful impulse responses?

(6) What can we learn from experiments?

Our first step will be to examine the frequency responses of one of the interpolators studied so far (the area ratio). This we will obtain using the 2D FFT of the 5x5 impulse response matrix, zero padded up to a 111x111 size. Fig. 26 shows the result. Because this is such a small filter, with such a slow roll-off, this is just a rough view and measurement. But, our result is consistent with a starting cutoff value of 0.17. (We won't argue if you prefer something else between 0.1 to 0.25.)

Our second step is to go experimental. Can we find a case that works? First we design a 1D length 5 filter. Here we try the popular Matlab ***firls*** function. The actual Matlab code line is:

```
h=firls(N,[0 cutoff cutoff+.02 .5]*2,[1 1 0 0])
```

we then form the 2D impulse response matrix as:

```
mm=h'*h;
mm=mm*(1/mm(3,3));
```

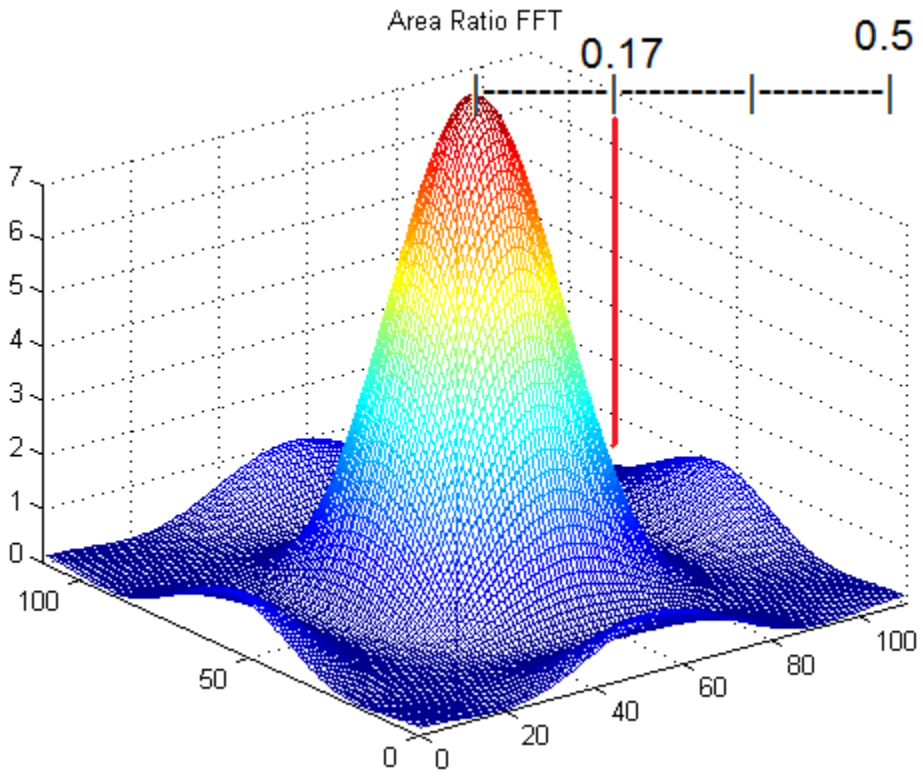



Fig. 26

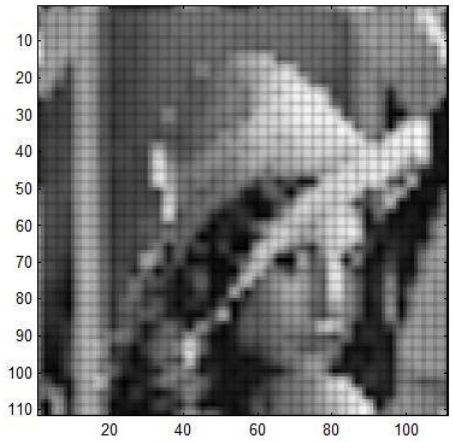
Very roughly speaking, the interpolators we have found to be successful that a frequency response going to about 0.17

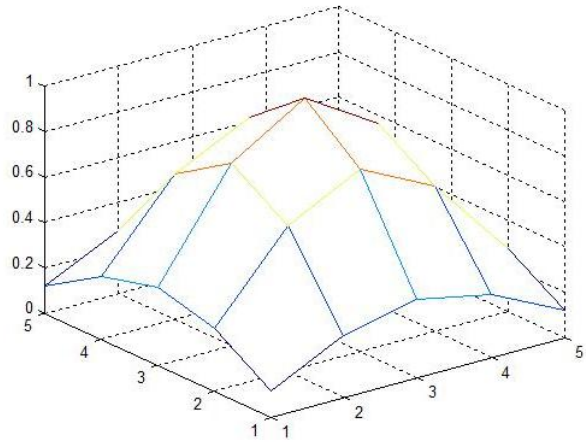
which forms and scales the impulse so that the center of the 5x5 is 1. With these simple code lines, we choose a cutoff, and we get our test impulse response. So what happens if we choose a cutoff=0.17? This is shown in Fig. 27a, and is what we can term to be “typical”. It does not work, and gives us this annoying checkerboard pattern.

However, putting together the arguments as listed above, it is likely that if we played around a bit, we could find something in the vicinity. In fact, this is surprisingly simple to do – a couple of minutes of trial and error. The result in Fig. 27b (cutoff = 0.18974) shows the checker boarding virtually gone, and the interpolation is very comparable to our previous methods. We actually got a bit of help however: we did not have to actually subjectively compare all images as we try different cutoffs. This help we got by calculating the sum of the weights over non-zero pixels as we have been doing to see if they come out near 1. Recall that there were three uniquely different alignment situations when overlaying a 5x5 impulse response on the 3x3 sampled lattice. We wanted these three to have the same total weights, else dark spots or check patterns could result.

In the case of Fig. 27b, we show these sums, and they worked out perfectly for this case. (That is, we were able to choose a cutoff such that they came out virtually perfectly, and this corresponded to the visual elimination of the checkerboarding (and for a range around that cutoff value). The green box (sum = 1) is the same as the sum of the two pink boxes and the same as the sum of the four blue boxes.

Fig. 27a Cutoff =0.17



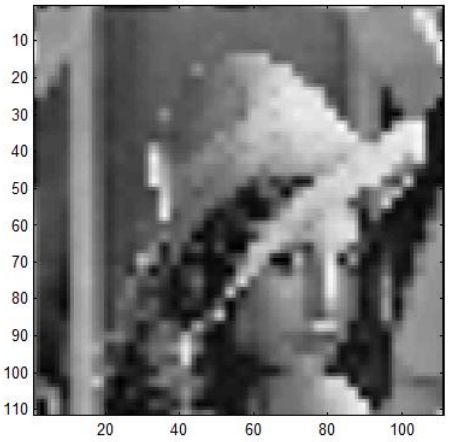

h = 0.1231 0.2866 0.3578 0.2866 0.1231

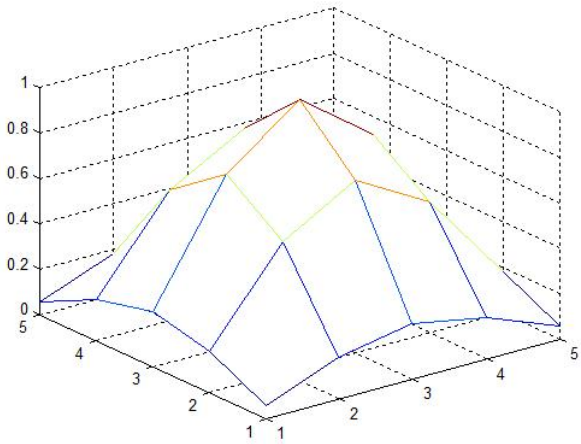
mm =

| | | | | |
|---|---|---|---|---|
| 0.1184 | 0.2756 | 0.3441 | 0.2756 | 0.1184 |
| 0.2756 | 0.6414 | 0.8009 | 0.6414 | 0.2756 |
| 0.3441 | 0.8009 | 1.0000 | 0.8009 | 0.3441 |
| 0.2756 | 0.6414 | 0.8009 | 0.6414 | 0.2756 |
| 0.1184 | 0.2756 | 0.3441 | 0.2756 | 0.1184 |

Sum 1 = 1.3110 Sum 2= 1.1450

Fig. 27b Cutoff = 0.18974




h = 0.0955 0.3011 0.3967 0.3011 0.0955

mm =

| | | | | |
|---|---|---|---|---|
| 0.0580 | 0.1828 | 0.2409 | 0.1828 | 0.0580 |
| 0.1828 | 0.5763 | 0.7592 | 0.5763 | 0.1828 |
| 0.2409 | 0.7592 | 1.0000 | 0.7592 | 0.2409 |
| 0.1828 | 0.5763 | 0.7592 | 0.5763 | 0.1828 |
| 0.0580 | 0.1828 | 0.2409 | 0.1828 | 0.0580 |

Sum 1 = 1.0000 Sum 2 = 1.0000

A very quick study suggests that this result (weight sums going to one with checkerboarding going away) can be found in general. In the ***firls*** call, the transition region was set to a width of 0.02, and this can be changed. In trying different transition regions, we found the sums to go to zero for the following cutoffs:

| Transition Region | Best Cutoff | Magic Number |
|---|---|---|
| 0 | 0.19875 | 0.0929 |
| 0.02 | 0.18974 | 0.0925 |
| 0.05 | 0.17530 | 0.0904 |

It appears that not only do we get a favorable result, but indeed both the sums (blue and pink in Fig. 27b go to 1 at the same cutoff. In addition, another interesting fact is that these ***firls***-generated matrices seem to differ from the area ratio matrix by amounts of the same magnitude on the horizontal and vertical axes. That is, the difference between the area ratio [ equation (5) ] and the ***firls*** result shown at the bottom of Fig. 27b, is:

diff =

| | | | | |
|---|---|---|---|---|
| 0.0531 | 0.0394 | 0.0925 | 0.0394 | 0.0531 |
| 0.0394 | -0.1319 | -0.0925 | -0.1319 | 0.0394 |
| 0.0925 | -0.0925 | 0 | -0.0925 | 0.0925 |
| 0.0394 | -0.1319 | -0.0925 | -0.1319 | 0.0394 |
| 0.0531 | 0.0394 | 0.0925 | 0.0394 | 0.0531 |

The numbers in red, here of magnitude 0.0925, are the same. Why do these two numbers match in magnitude? This is what we have tabulated as the "magic number" above. Likely there are many other things to discover and attempt to explain if we looked further.

We did substitute the "***remez***" (Parks-McClellan method) design for ***firls***, and it does not seem to work. We can easily get equal sums of weights. When we do this the result is that the interpolation looks virtually identical to the held version (Fig. 12c). If we look at the impulse response we see why – there is a dominant 3x3 in the center, approaching 1, with a very weak ring (approaching 0) beyond in the 5x5. This region of flatness in the impulse response is characteristic of the high-ripple of the Parks-McClellan method.

There is probably much more that can be studied.

In Fig. 28, we choose to put, all on one page, the four major results we found.

# SUMMARY

This note is likely tedious to read and study, although likely not what we would call difficult. The purpose here was to put the basic materials together and to illustrate the necessity of keeping an open mind about the practical differences between 1D interpolation (usually audio) and 2D. Largely we have found physical, spatial-domain models (like surfaces) to be more useful that frequency domain models (like bandlimiting). In addition, it was usually necessary to coax out (patch up initial notions) to get a useful result.

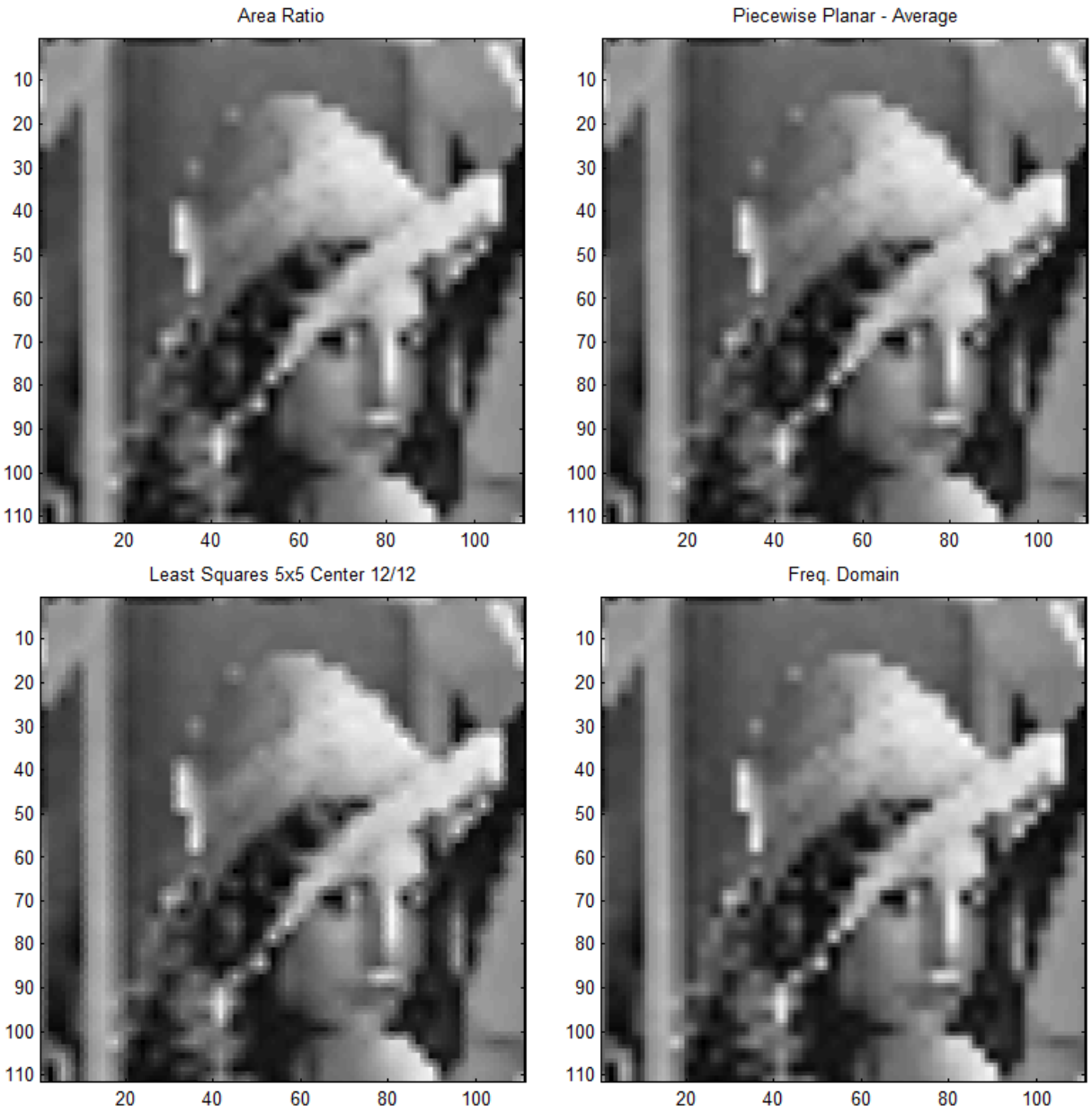



Fig. 28 Four Major Methods Examined Here