Technical memo number 1

January 2004, revised & enlarged August 2004

OUR ADOPTED SCHEME FOR SUBPIXEL MODELING

Kris Davidson, University of Minnesota

(If you use information or advice from this memo, please acknowledge it and the net site http://etacar.umn.edu in any resulting publications; thanks.)

1. Introduction: A special type of interpolation problem

These notes concern STIS observations, but can also be applied to other forms of data where the pixels are somewhat larger than the optimal size. In the past, spatial resolution better than 0.2″ has been difficult to attain with the STIS/CCD, mainly because the available software wasn't well adapted to the instrument's pixel size, which is too large for good sampling at most wavelengths. Using techniques described below, we obtain a genuine FWHM spatial resolution of about 2 pixels $\approx$ 0.1″, a substantial improvement over the standard software. This is nearly the best that one can do with procedures that are general. Better resolution may be possible based on prior knowledge of the target object's structure, but that requires special case-by-case treatment.

Here we have essentially a non-textbook interpolation problem. Image data, including the two-dimensional spectral images that STIS produces, require geometric transformations for distortion corrections, rotation, rectification of rows and columns, scale calibration, and pixel resizing. Such transformations involve *subpixel modeling*, i.e., the original pixel values $F(\text{column, row})$ must be converted into a continuous function $g(x, y)$, either explicitly or implicitly in the reduction process. Standard linear and cubic-spline interpolation techniques give poor results if the data pixels are wider than about half the FWHM of the relevant point-spread function (p.s.f.); this informal criterion is more demanding than the famous sampling theorem, see Section 3 below. Moreover, the bad effects of an unsatisfactory transformation are irreversible. HST imaging programs often use spatial "dithering" to improve the effective sample spacing, but such techniques have not been feasible for most STIS spectroscopy.

Therefore *we need a consistent method for sub-pixel modeling in data where the pixel size is comparable to the fundamental p.s.f.'s FWHM.*

2. Four preliminary remarks

(a) For simplicity, in these notes we discuss the 1-dimensional case. For a 2-dimensional image, apply the same reasoning to both dimensions, rows and columns separately.

(b) Cubic spline interpolation is not a panacea, and usually isn't even appropriate. Splines are best if we wish to interpolate a precisely-defined mathematical function for some purpose that requires smoothness -- quite different from working with observational data perturbed by noise. Moreover, a 5-to-7-point non-spline interpolation formula, with coefficients "tuned" to a well-chosen Fourier frequency, can represent high-S/N data with smaller errors than a spline, albeit less smoothly. Don't assume that spline fitting is best merely because textbooks feature it!

(c) Remember to distinguish between *pixel values* and *function samples.* Let's denote the underlying continuous function, which we hope to estimate from the data, by $f(x)$, where x is measured in pixels (either rows or columns). Then the *pixel value* function $F(x)$ is an average of $f(x)$ between $x - 0.5\,\Delta x$ and $x + 0.5\,\Delta x$, where usually we assume that Δx = 1 pixel. [Δx can be less than unity if there's "dead space" between adjacent pixel active areas; and, more generally, the local average might include some x-dependent weighting function.] If pixel centers are located at integer values of x, we can regard $F(x)$ as a continuous function that has been sampled only at $x = 1, 2, 3 \ldots$ As Rayleigh noted more than 130 years ago [*Phil. Mag.* XLII, 441 (1871)], a good approximation for f in terms of F is:

$$f(x) \approx F(x) - (\Delta x)^2 F''(x)/24 , \qquad (1)$$

where F'' denotes the local second derivative numerically estimated from the three or four closest data values $F(n)$. We can employ this formula at integer values of x and then estimate $f(x)$ elsewhere by interpolation. For our purposes the Rayleigh correction turns out to be relatively minor; but it shouldn't be ignored.

(d) Shape of the point spread function (p.s.f.): For a point source, $f(x)$ represents the overall resolution of the optical system. Then, with STIS/CCD data at blue wavelengths, experiments show that both $f(x)$ and $F(x)$ in the spatial direction (perpendicular to dispersion) can be approximated well by functions of the form

$$\alpha / \{ \alpha + \beta (x - x_0)^2 + \gamma (x - x_0)^4 \} ,$$

or often

$$a^4 / \{ a^2 + (x - x_0)^2 \}^2 .$$

The true p.s.f.'s are noticeably asymmetric, but these expressions can be used for realistic assessments of subpixel modeling methods, and can be applied to the high-x side which is steeper and thus more difficult to model. The minimum FWHM of $F(x)$ in data that

we're concerned with is close to 1.7 pixels, implying that $f(x)$ has a minimum FWHM of 1.3 to 1.5 pixels.

3. A practical limitation on general-purpose resolution

How seriously does pixel size limit the effective resolution? The sampling theorem basically requires (shortest usable Fourier wavelength) $\geq$ ($2 \times$ pixel size). Since the FWHM of a p.s.f. tends to be about half the dominant or critical Fourier wavelength, one might hope to achieve a p.s.f. whose FWHM is roughly one pixel. In practice, unfortunately, this criterion is too optimistic. *When seeking an accurate mode of real data that were obtained without "dithering" the observations, the attainable p.s.f. FWHM is almost two pixels, not one.* *

* Why is this true? Fundamentally, at the famous sampling limit half the Fourier information is missing, because $\sin(k_{\max} x + \phi_0)$ with a particular phase ϕ_0 gives zero signal. Close to the limit, Fourier components with disadvantageous phases can be measured only by sampling a large interval of many pixels. Noise and irregularities make this impractical in real data.

As an example, consider a specific pixel-value function:

$$F(x) \approx 3.0276 / \{ 1.74 + (x - x_0)^2 \}^2 , \tag{2}$$

which has FWHM $\approx$ 1.7 pixels. The underlying function $f(x)$ has FWHM $\approx$ 1.4 pixels.

Figure 1 shows a case where x_0 is an integer, so the profile is centered at the middle of a pixel. Then a spline fit, for instance, models the shape so accurately that the errors would be hard to distinguish in a figure of this size.

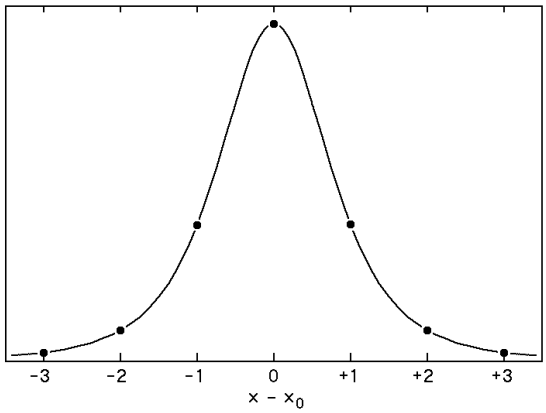

Figure 1. Typical $F(x)$ p.s.f. with FWHM = 1.7 pixels, with x_0 at a pixel center. A spline fit based on the discrete data points (dots) coincides with the true curve so well that we don't attempt to show it separately here.

In Figure 2 on the next page, however, x_0 is a half-integer, so the peak occurs at a boundary between two pixels. *Now the spline fit is not so good;* it has a lower peak, a broader FWHM, and an incipient kink around $x - x_0 \approx \pm 2.5$.

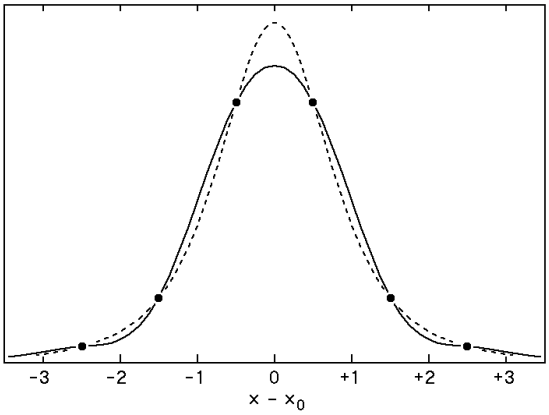

Figure 2. Same underlying function as in Fig. 1, but here the data points (dots) occur at half-integer values of $x - x_0$. The solid curve is a spline fit to the data points while the true pixel-value function $F(x)$ is shown by a dashed curve.

Evidently *the effective resolution depends on an object's precise location on the detector,* which doesn't surprise anyone accustomed to half-pixel "dithering" of HST images. Spectroscopists, alas, usually can't afford to dither.

The worsened resolution and lower peak in Figure 2 are not, *per se,* our main worry. Rather, the trouble is in the *variations* depending on precise location of an object on the detector, the difference between Figures 1 and 2. For instance, suppose we observe a point source and extract a spectrum only two or three STIS/CCD rows wide, seeking good spatial resolution. Since the spectrum isn't exactly parallel to the rows (see Fig. 3), at some columns (wavelengths) it coincides with row centers but for some other columns it's centered at boundaries between rows -- it continuously shifts between the cases shown in Figs. 1 and 2. One result: In a spectrum extraction narrower than four detector rows, the continuum flux level appears wavy or "scalloped" as a function of wavelength. Most STIS users avoid this difficulty by integrating over 5 or more rows, thereby sacrificing the instrument's spatial resolution. Can we do better?

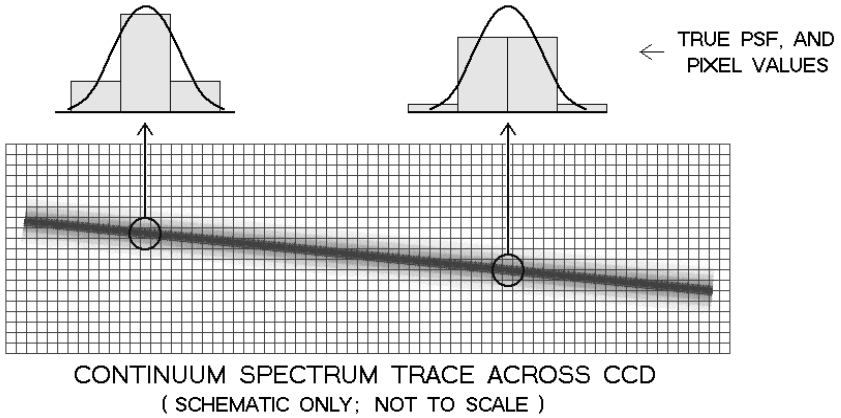

Figure 3. A cut across the the spectrum of any particular pointlike object matches Fig. 1 at some wavelengths and Fig. 2 at others.

In most cases, the spectral waviness or scalloping is not alarmingly bad in an extracted spectrum of a single star, centered on the star image. But it can become ruinously large in an offset extraction, e.g. if one is trying to get the spectrum of another object or location less than 5 pixels away. The same difficulty occurs in the dispersion (wavelength) direction, though it isn't as easy to notice that way; the apparent width and profile of a narrow spectral feature depends on its precise location on the detector. Since a general data-reduction program cannot know where objects or spectral features are located on the detector, at first sight the pixel-sampling problem may appear insoluble. Fortunately, however, an acceptable technique is available.

4. A general method that works fairly well

Our goal is to derive from the data points $F(n)$ a continuous function $g(x)$, whose purpose is to approximate $f(x)$ as well as possible under the circumstances. "As well as possible" means as *consistently* as possible: The deduced shape of $g(x)$ should not perceptibly depend on the exact fractional-pixel location of $f(x)$ and $F(x)$. The following procedure makes this feasible; numerical details will be given in Section 5.

(1) Based on nearby data points $F(n)$, we employ a suitable interpolation formula to calculate $g(x)$ at half-integer values of x, i.e., at the boundaries between pixels.

(2) Then, choose another formula which gives us $g(x)$ at integer values of x, the pixel centers. Our goal here is somewhat subtle. It's easy to develop a formula which gives a very accurate approximation to $f(n)$ in terms of $F(n)$. But that's not what we need for this problem! Instead, *we adapt our expression for $g(n)$ to be as consistent as possible with the formula used in step 1 above.*

What do we mean by "consistent"? Imagine two pixel-value functions $F_1(x)$ and $F_2(x)$ which have the same shape but different locations: $F_2(x) = F_1(x + 0.5)$. Then let's estimate values $g_1(n + 0.5)$ from data points $F_1(n)$ with the formula adopted in step 1, and separately estimate $g_2(n)$ from data points $F_2(n)$ using the formula chosen for step 2. *The two formulae are mutually "consistent" if the resulting values $g_2(n)$ are nearly the same as the corresponding $g_1(n + 0.5)$.* This goal differs from the standard interpolation recipes found in textbooks and reference books.

> Note that in step 1 we start with the half-integer x values rather than the integer x's. This is because the function values estimated at x = integers are normally less blurred than those at x = half-integers; compare Figs. 1 and 2. If we were to calculate the x = integer points first, then any attempt to find correspondingly good values at half-integer x's would amount to de-blurring or deconvolution, with consequent "ringing" and other artifacts of the process.

(Step 3) Finally, we estimate the continuous function $g(x)$ by interpolating between the points calculated in steps 1 and 2. Local cubic interpolation is best but quadratic interpolation may suffice, since the new data points are separated by only half the original detector pixel width and are somewhat blurred.

5. Specific formulae and coefficients

There's no need for spline techniques here; they would require careful non-routine definitions, and only the nearest few data points matter anyway. For modeling $g(x)$ within pixel n, five to seven nearby data values $F(n \pm m)$ provide as much accuracy as we can hope to attain.

For Section 4's "step 1", we use a symmetric interpolation formula:

$$(3) \qquad g(n+0.5) = P\cdot\{F(n) + F(n+1)\} + Q\cdot\{F(n-1) + F(n+2)\} + R\cdot\{F(n-2) + F(n+3)\},$$

where coefficient R should be relatively small. The constant coefficients P, Q, R have two definite constraints. First we require $g = f$ if $f(x) =$ constant, or, equivalently, "counts must be conserved". Therefore $P + Q + R = 0.5$. Second, the formula should reproduce $f(x)$ if it varies only slowly with x. By symmetry of terms eqn. (3) automatically does this for $f(x) = x$, so we get our second constraint by requiring the formula to work also for $f(x) = x^2$. This corresponds to pixel value function $F(x) = x^2 + (\Delta x)^2/12$ where Δx is the sensitive width of a pixel, see Section 2 above. The resulting constraint is

$$3P + 27Q + 75R = -(P+Q+R)(\Delta x)^2 = -0.5(\Delta x)^2.$$

If both constraints on P, Q, R are satisfied, then $g(n+0.5)$ calculated from formula (3) will be equal to $f(n+0.5)$ for any cubic* polynomial $f(x)$. [* Not just quadratic; symmetry in the formula takes care of x^3.] Thus we can represent low Fourier frequencies quite well.

For any given value of R (presumably small), the two constraints determine P and Q:

$$(4a) \qquad P = +\{27 + (\Delta x)^2\}/48 + 2R,$$

$$(4b) \qquad Q = -\{3 + (\Delta x)^2\}/48 - 3R.$$

$R \approx +0.028$ gives the best average fit if $\Delta x = 1$ and if the p.s.f. has FWHM $\sim$ 2 pixels. However, numerical experiments show that this choice entails complicated behavior two or three pixels away from the p.s.f. center: the tail of $g(x)$ calculated for the p.s.f. develops a slight wiggle. If instead we choose simply $R = 0$, then this effect vanishes while the central peak is only mildly affected. Thus we adopt a simple compromise based on many numerical experiments.

(next page)

Adopted coefficients for step 1:

(5a) $P = +0.58,$

(5b) $Q = -0.08,$

(5c) $R = 0.$

These rounded-off values are consistent with $(\Delta x)^2 = 0.84$, $\Delta x \approx 0.92$, a detail that only mildly affects the results. [$\Delta x = 0$ would amount to ignoring Rayleigh's distinction between $f(x)$ and $F(x)$. Because self-consistency requires our effective p.s.f. to be substantially blurred, this distinction plays only a minor role after all.]

Next, "step 2" in Section 4, we need an appropriate symmetric formula for a pixel center, $g(\text{integer})$:

(6) $$g(n) = A \cdot F(n) + B \cdot \{ F(n-1) + F(n+1) \} + C \cdot \{ F(n-2) + F(n+2) \} .$$

Coefficients $A = +1.123$, $B = -0.068$, $C = +0.0065$ provide an excellent fit to the STIS/CCD p.s.f.. However, as explained in Section 4 above, that is not our goal here. Instead we aim to get results that work well in combination with formula (3) with the chosen coefficients (5abc). Two constraints on $A, B, C,$ based on the same reasoning as those used for P, Q, R, are $A + 2B + 2C = 1$ and $24B + 96C = -(\Delta x)^2$. For a given value of C these imply

(7a) $A = 1 + (\Delta x)^2/12 + 6C,$

(7b) $B = -(\Delta x)^2/24 - 4C.$

We can use various criteria to choose the "best" value of C: Minimum mean-square difference between the $g_1(n + 0.5)$ and $g_2(n)$ described in Section 4, smoothest 1-row or 2-row extractions, etc. Numerical experiments show that for all obvious criteria, the optimum C for consistency with coefficients (5abc) is between -0.06 and -0.04. Therefore we choose $C = -0.05$. Assuming $(\Delta x)^2 = 0.84$ consistent with our P, Q, R values, we adopt

(8a) $A = +0.77,$

(8b) $B = +0.165,$

(8c) $C = -0.05.$

[It's obvious from the $+ + -$ signs that we're blurring the p.s.f. If these coefficients were designed for the best possible fit to $f(n)$, then their signs would alternate, $+ - +$.]

Incidentally, the best choices for coefficients (5abc) and (8abc) do not depend strongly on the assumed width of the $f(x)$ p.s.f. The adopted values are OK for FWHM > 1.3 pixels.

An afterthought: For a reason concerning statistical noise levels, noted in Section 8 below, it may be advantageous to require $A^2 + 2B^2 + 2C^2 = 2P^2 + 2Q^2$. Given the adopted values of P and Q in eqns. (5ab), this additional constraint leads to $A = +0.800$, B = +0.145, C = –0.045 (rounded to 3 places). But we adopted the coefficients (8abc) for the STIS / CCD processing, because, frankly, we didn't think of this extra consideration early enough. In practice the differences are slight.

6. Interpolation formulae for general x

Using the above formulae, we can calculate a set of values $g(x)$ at integer and half-integer values of x. For other values of x, we interpolate as follows:

- Identify the integer n which is closest to x.
- Define $s = x - n$, in the range –0.5 to +0.5. ($s = 0$ at the center of pixel n.)
- Based on the values $g(n-0.5)$, $g(n)$, and $g(n+0.5)$, employ local quadratic interpolation:

$$(9) \qquad g(x) = a + bs + cs^2,$$

where

$$(10a) \qquad a = g(n),$$

$$(10b) \qquad b = g(n+0.5) - g(n-0.5),$$

$$(10c) \qquad c = 2g(n+0.5) - 4g(n) + 2g(n-0.5).$$

- Since $g(n-0.5)$, $g(n)$, and $g(n+0.5)$ are linear combinations of five pixel values $F(n-2)$ **...** $F(n+2)$, we can write formulae for a, b, c in terms of these data points. Results:

$$(11a) \qquad a = -0.050F(n-2) + 0.165F(n-1) + 0.77F(n) + 0.165F(n+1) - 0.050F(n+2),$$

$$(11b) \qquad b = +0.08F(n-2) - 0.66F(n-1) + 0.66F(n+1) - 0.08F(n+2),$$

$$(11c) \qquad c = +0.04F(n-2) + 0.34F(n-1) - 0.76F(n) + 0.34F(n+1) + 0.04F(n+2).$$

The following rules allow us to check the coefficients in (11abc) for errors:

- The terms have obvious symmetries centered on n,
- Sum of coefficients = 1.00 for a, 0.00 for b and c,
- Sum of (offset × coefficient) in (11b) is 1.00,
- Sum of (offset^2 × coefficient) in (11c) is 1.00.

In each case, be careful with the signs.

…By the way, it would not be very difficult to work out a local cubic interpolation scheme, which would be smoother than the quadratic one and not much more complicated. Quadratic interpolation seems adequate for the STIS / CCD data.

7. A comparison of results using three techniques

As explained in Section 3 above, the shape of $g(x)$ estimated by any method depends on the precise fractional-pixel location x_0 of the underlying function $f(x)$, because this determines which points of the function are sampled by the data pixel values $F(n)$. Imagine shifting x_0 progressively. Then we get a set of functions $g(x)$, whose shapes all lie between a lower and an upper envelope.

Figure 4 shows an example using a naïve interpolation method. Here the underlying function $f(x)$ has the same shape as formula (2) but its FWHM is slightly less than 1.5 pixels. In this case $g(x)$ is estimated by linear interpolation between the $F(n)$ values, a method which is said to be used in some existing data reduction software, possibly including "pipeline" reduction of STIS data. The range of variation is disagreeably large.

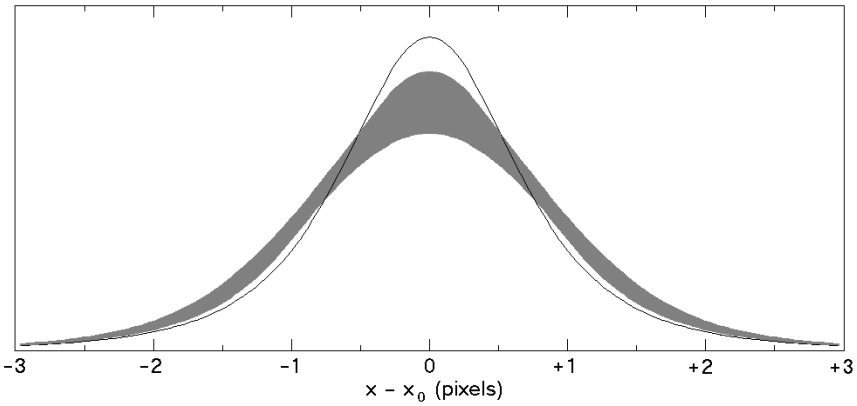

Fig. 4 . Range of $g(x)$ estimated by linear interpolation in $F(x)$. The narrow curve is the underlying function $f(x)$, and the shaded region shows the range of estimated $g(x)$ functions, whose shapes depend on the central location x_0.

----- continued on next page -----

Next, Figure 5 shows the same data modeled by cubic spline interpolation. It's much better than Fig. 4 but is still too fuzzy to be satisfying.

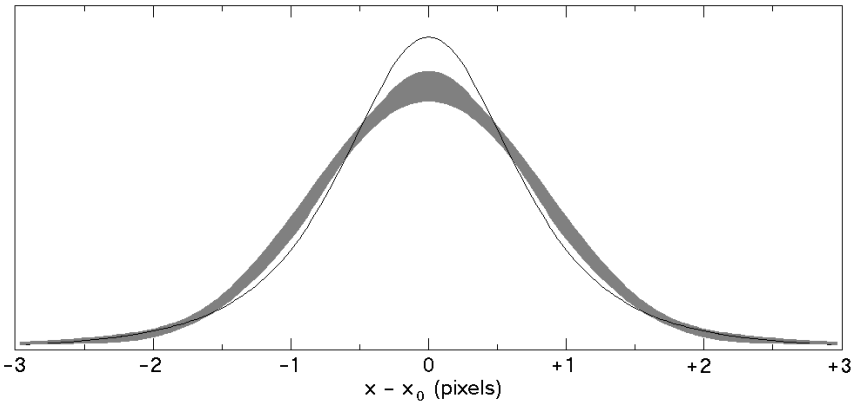

Fig. 5. Similar to Fig. 4, but here we have used cubic spline interpolation between $F(n)$ values.

Finally, Figure 6 shows what we get by using our adopted procedure and coefficients described in Sections 4--6 above. The improvement in consistency is obvious even though this method uses fewer points than spline fitting.

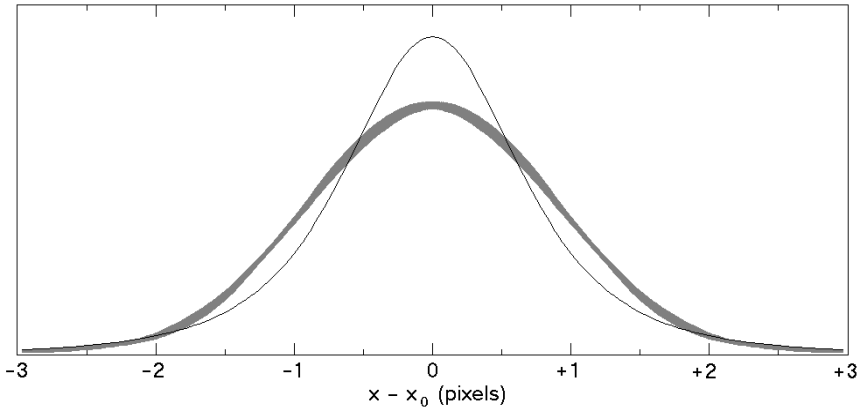

Fig. 6. Similar to Figs. 4 and 5, but using our adopted technique.

Now the FWHM of $g(x)$ is about 10% worse than in Fig. 5 (2.1 vs. 1.9 pixels); in effect, the original $f(x)$ has been convolved with a blurring function whose FWHM is about 1.5 pixels. An appreciably better solution to our overall problem would be hard to find. Using STIS/CCD data with 0.05″ pixels, evidently we can attain a general, fairly robust FWHM resolution of about 0.1″ or maybe 0.08″. This is considerably better than that allowed by previously existing "pipeline" software, and at long wavelengths it is close to the HST's fundamental resolution.

Clarification: The greatest advantage of our technique occurs when dealing with spatially complex objects. A "one-dimensional spectrum extraction" is, essentially, the sum of several CCD rows, which may behave fairly well in data processed by earlier techniques *for an isolated point source,* i.e., for a single star. In Fig. 4, the function $g(x)$ strongly depends on the precise subpixel location of the star, but the integral of $g(x)$ across an interval $\Delta x > 2$ does not, provided the integral is centered near the peak. In other words, the cases x_0 = integer and x_0 = half-integer then give fairly consistent extracted spectra. Small errors in centering the extraction don't affect this statement very much. Therefore our method gives only modestly improved spectra of isolated stars. Suppose, however, one is interested in two objects A and B which are located less than 0.4″ apart. Then, in terms of spatial location x, each spectrum is located on the shoulder of the other. In that case any extraction of spectrum B is obviously contaminated by object A. Since the contribution by A is very asymmetric in the spatial direction, it may be *severely* "scalloped" for the reasons implied on page 4, if an inferior subpixel modeling technique is used. If B is considerably fainter than A, then the artificial *irregularities* in the contaminating spectrum, introduced mainly by the processing method, can make the spectrum of B almost unusable. (Subtraction of a separately-extracted spectrum of A does not remove such irregularities -- if this isn't obvious, think carefully about the situation.) The same effects become even more serious for a spatially complex object like η Carinae. In such cases, the techniques described above are far better than previous methods.

The moral of this story, sadly familiar to thoughtful HST users: A satisfying instrument should have (effective pixel size) $\leq 0.5 \times \{$ FWHM of p.s.f. in fundamental $f(x)\}$. This is appreciably smaller than the size suggested by a naïve interpretation of the sampling theorem.

8. Noise statistics in the processed data

When we rebin pixels using any interpolation scheme, neighboring "output" pixel values are not independent of each other. This fact can greatly alter the results of statistical tests, and it alters the precise meaning of any quoted "r.m.s. noise" value. Such values of σ are often misinterpreted and misapplied, sometimes in the initial-processing software. (In particular, beware of the "error" files produced by standard pipeline processing of HST / STIS / CCD data.)

Let's examine r.m.s. noise levels in data files produced by the subpixel-modeling procedure described above. For simplicity, first consider one-dimensional data. Here we'll focus on a special case wherein each output pixel is half as wide as an input pixel, and each output pixel center coincides either with the center of an input pixel or with the boundary between two input pixels.

Original ("input") data: $F(n)$, where n = pixel number = 1, 2, 3, …

Processed ("output") data with pixel width = 0.5: $g(x)$, where x = 1, 1.5, 2, 2.5, …

The r.m.s. noise level

In the following notes m, n, and k denote integers. Our subpixel-interpolation formulae, explained in Sections 4 and 5 above, are

$$(12)\quad g(n) = -0.05\,F(n-2) + 0.165\,F(n-1) + 0.77\,F(n) + 0.165\,F(n+1) - 0.05\,F(n+2),$$

$$(13)\quad g(n+0.5) = -0.08\,F(n-1) + 0.58\,F(n) + 0.58\,F(n+1) - 0.08\,F(n+2).$$

Suppose the r.m.s. noise per pixel in F is a constant, σ_F. Then the corresponding noise parameters for g are

$$(14)\quad \sigma_g(n) = \sqrt{\{(0.05\,\sigma_F)^2 + (0.165\,\sigma_F)^2 + (0.77\,\sigma_F)^2 + (0.165\,\sigma_F)^2 + (0.05\,\sigma_F)^2\}} = 0.80768\,\sigma_F,$$

$$(15)\quad \sigma_g(n+0.5) = \sqrt{\{(0.08\,\sigma_F)^2 + (0.58\,\sigma_F)^2 + (0.58\,\sigma_F)^2 + (0.08\,\sigma_F)^2\}} = 0.82801\,\sigma_F.$$

These are smaller than σ_F because our procedure implicitly smooths or blurs the data. The values in (14) and (15) are close to each other because formulae (12) and (13) were adapted to give nearly consistent results regardless of the fractional-pixel location of a data feature. (As noted on p. 8, we could have adjusted the interpolation coefficients to make (14) and (15) exactly equal.) To sufficient accuracy we can adopt the quadratic average,

$$(16)\quad \sigma_g(\text{1-dim. av}) \approx 0.81791\,\sigma_F.$$

In this sense, each output pixel value g is statistically equivalent to an average of approximately 1.5 input pixels -- because $0.81791 \approx 1/\sqrt{1.5}$.

Formula (16) refers to the one-dimensional case; for a 2-dimensional image we find, by similar reasoning,

$$(17a)\quad \sigma_g(m, n) = 0.65235\,\sigma_F,$$

$$(17bc)\quad \sigma_g(m+0.5, n) = \sigma_g(m, n+0.5) = 0.66877\,\sigma_F,$$

$$(17d)\quad \sigma_g(m+0.5, n+0.5) = 0.68560\,\sigma_F.$$

These give a quadratic average

$$(18)\quad \sigma_g(\text{2-dim. av}) \approx 0.66898\,\sigma_F.$$

Of course this is smaller than (16) because the data have been smoothed or blurred along columns as well as rows. Note that σ_g (2-dim. av) / σ_F is the square of σ_g (1-dim. av) / σ_F .

It is essential to clearly recognize the precise meanings of σ_F *and* σ_g . The first of them is straightforward: It is the r.m.s. error in an area of one original data pixel, assuming that the noise in each pixel is independent of every other pixel. The noise parameter σ_g , however, is more subtle and less familiar. It is the r.m.s. error in the value of g calculated for a rebinned pixel, and in most cases this is *considerably smaller than one would naively guess from the number of counts in the area of either type of pixel.* In effect, σ_g represents statistical noise for a sample width of about 1.5 original pixels in the one-dimensional case, or a sample area of about 2.2 pixels in the two-dimensional case.

Tests to determine or verify the noise level from local characteristics of the data

Consider again the one-dimensional case. Given a set of truly independent data values $F(n)$, the simplest way to estimate σ_F is to calculate differences of many adjoining pixels:

(19) r.m.s. difference $[\, F(n+1) - F(n) \,] \quad = \quad \sqrt{2}\ \sigma_F$.

If, however, $F(n)$ is not a constant, then the difference used in (8) is affected by the local slope. A better simple estimator, valid to second order, is

(20) r.m.s. difference $[\, F(n) - 0.5\,\{\, F(n-0.5) + F(n+0.5) \,\}\,] \quad = \quad \sqrt{1.5}\ \sigma_F$.

But this formula must not be applied to our processed ("output") data, where adjoining pixels are strongly correlated. If we use nearest-neighbor pixels as in eqn. (9), the r.m.s. difference $[\, g(n) - 0.5\,\{\, g(n-0.5) + g(n+0.5) \,\}\,]$ turns out to be only about $0.3\ \sigma_g$, not $1.4\ \sigma_g$.

Expected r.m.s. differences $[\, g(n) - 0.5\,\{\, g(n-0.5k) + g(n+0.5k) \,\}\,]$ are listed on the next page for several values of the offset k .

----- continued on next page -----

Root Mean Square differences

$$\Delta = [\, g(n) - 0.5\,\{\, g(n-0.5k) + g(n+0.5k)\,\}\,]$$

(one-dimensional case)

Offset k (output pixels)	(r.m.s. Δ) / σ_F	(r.m.s. Δ) / σ_g
1	0.2399	0.2933
2	0.6976	0.8529
3	0.9954	1.2170
4	1.0708	1.3092
5	1.0430	1.2752
6	1.0068	1.2309
7	0.9977	1.2199
8	1.0005	1.2232
9 or more	1.0017	1.2247

… Perhaps a better empirical way to estimate noise in the data, especially for two-dimensional data, is to compare each pixel value with the average or the median of many surrounding values. Remember, though, that data points separated by less than about 3 pixels are noticeably correlated as the Table shows. (This statement refers to the rebinned processed pixels, not the original data pixels.)