1. Introduction: A special type of interpolation problem

These notes particularly concern STIS observations, but can also be applied to other forms of data. 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; the pixels are 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. In order to do better than this, one usually needs to invoke prior knowledge of the target object’s structure.

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 sampling rate, 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 that contain 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 \( \Delta x = 1 \) pixel. [\( \Delta x \) 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)], the simplest realistic approximation for \( f \) in terms of \( F \) is:

\[
(1) \quad f(x) \approx F(x) - (\Delta x)^2 F''(x)/24 ,
\]

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.): 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

\[
a/\{ a^2 + (x - x_0)^2 \}^2 .
\]

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) \( \geq (2 \times \text{pixel size}) \). Since the FWHM of a p.s.f. tends to be roughly 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; for accurate modeling the attainable p.s.f. FWHM is about two pixels, not one. The following example illustrates the situation.

Consider a specific pixel-value function:

\[
F(x) \approx \frac{3.0276}{1.74 + (x - x_0)^2},
\]

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

Figure 1 shows a case where the profile given by formula 2 is centered at the middle of a pixel, so \( x_0 \) is an integer. Then a spline fit, for instance, models the shape so accurately that we can’t show the errors very clearly at the scale of this figure.

Figure 2 on the next page, however, shows the same curve with its peak located at a boundary between two pixels, so the data points (integer values of \( x \)) occur at half-integer values of \( x - x_0 \). 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 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.
The same difficulty occurs in the dispersion (wavelength) direction, though it isn’t as conspicuous 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 are located on the detector, at first sight the pixel-sampling problem appears insoluble. Fortunately, however, an acceptable technique is available.

4. A general method that works fairly well

Here’s the basic plan; details will be given in Section 5.

(1) Let’s use the data points \( F(n) \) to estimate a continuous function \( g(x) \), whose purpose is to approximate \( f(x) \) as well as possible under the circumstances. “As well as possible” really 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 \( F(x) \).

(2) 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.

(3) Next, 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 \( F(n) \) [recall eqn. (1)]. 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 for intermediate points used in step 2 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 2, and separately estimate \( g_2(n) \) from data points \( F_2(n) \) using the formula chosen for step 3. 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.

[ There’s a reason why we adapt the step-3 formula to fit the step-2 expression, rather than vice-versa. Function values estimated at \( x = \) integers are inevitably less blurred than those at \( x = \) half-integers; compare Figs. 1 and 2. If we were to calculate the former 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. ]
Finally, we estimate the continuous function $g(x)$ by interpolating between the points calculated in steps 2 and 3. Local cubic interpolation is best but quadratic interpolation will usually suffice, since the new data points are separated by only half the original detector pixel width and are somewhat blurred anyway.

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 2”, we use a symmetric interpolation formula:

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

where coefficient $R$ should be relatively small. The constant coefficients $P, Q, R$ have two definite constraints. First we require $g = f$ if $f(x) = \text{constant}$, 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 $\Delta x$ 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 $f(x)$; if this isn’t obvious, think about why.] Thus we can represent low Fourier frequencies quite well.

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

\[
(4a) \quad P = + \{ 27 + (\Delta x)^2 \}/48 + 2R,
\]

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

$R \approx +0.028$ 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 )
\\{(5a)\quad P = +0.58,\\(5b)\quad Q = -0.08,\\(5c)\quad R = 0.\\\\}

These rounded-off coefficients are consistent with \((\Delta x)^2 = 0.84\), \(\Delta x \approx 0.92\), a detail that has only a minor effect on results in general. [\(\Delta x = 0\) would amount to ignoring the Rayleigh 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 3” in Section 4, we need an appropriate symmetric formula for \(g\) (pixel center):

\[
\begin{align*}
g(n) &= A \cdot F(n) + B \cdot \{F(n-1) + F(n+1)\} \\
&\quad + C \cdot \{F(n-2) + F(n+2)\}.
\end{align*}
\]

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, this is not our goal here. Instead we aim to get results that work well with formula (3) and the chosen coefficients in (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

\[
\begin{align*}
(7a) \quad A &= 1 + (\Delta x)^2/12 + 6C, \\
(7b) \quad B &= - (\Delta x)^2/24 - 4C.
\end{align*}
\]

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

\[
\begin{align*}
(8a) \quad A &= +0.77, \\
(8b) \quad B &= +0.165, \\
(8c) \quad C &= -0.05.
\end{align*}
\]

[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.
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 \(x\), 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 \(g(n + 0.5),\) we employ local quadratic interpolation:

\[
g(x) = a + bs + cs^2,
\]

where

\[
\begin{align*}
(10a) \quad a &= g(n), \\
(10b) \quad b &= g(n + 0.5) - g(n - 0.5), \\
(10c) \quad c &= 2g(n + 0.5) - 4g(n) + 2g(n - 0.5).
\end{align*}
\]

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

\[
\begin{align*}
(11a) \quad a &= -0.050 F(n - 2) + 0.165 F(n - 1) + 0.77 F(n) \\
&\quad + 0.165 F(n + 1) - 0.050 F(n + 2), \\
(11b) \quad b &= +0.08 F(n - 2) - 0.66 F(n - 1) \\
&\quad + 0.66 F(n + 1) - 0.08 F(n + 2), \\
(11c) \quad c &= +0.04 F(n - 2) + 0.34 F(n - 1) - 0.76 F(n) \\
&\quad + 0.34 F(n + 1) + 0.04 F(n + 2).
\end{align*}
\]

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 \((\text{offset} \times \text{coefficient})\) in \((11b)\) is 1.00,
- Sum of \((\text{offset}^2 \times \text{coefficient})\) 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 slightly better than this quadratic one.)

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.

![Figure 4](image)

**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 $x_0$.

Next, Figure 5 shows the same data modeled by cubic spline interpolation. It’s a considerable improvement over Fig. 4 but is still too fuzzy to be satisfying.

![Figure 5](image)

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

---- concluded on next page ----
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 requires less calculative effort than spline fitting. 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. This outcome is perhaps a little disappointing, but 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, reasonably robust FWHM resolution of about 0.1″.

![Figure 6](image.png)

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

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