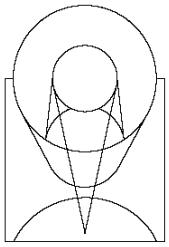

# TECHNICAL REPORT



SPACE TELESCOPE SCIENCE INSTITUTE
Operated for NASA by AURA

| | |
|---|---|
| Title:<br>NIRCam Dithering Strategies I: A Least-Squares Approach to Image Combination | Doc #: JWST-STScI-002199, SM-12<br>Date: October 9, 2010<br>Rev: - |
| Authors: Jay Anderson Phone: 410-338-4982 | Release Date: 10 May 2011 |

## 1 Abstract

This is the first of two reports that will help NIRCam users plan how many secondary dithers they will need in order to achieve adequate sub-pixel sampling in their images. This first report will present a new approach to constructing a composite "stacked" image from a series of dithered observations. Traditionally, this procedure has been done for HST images with the `Drizzle` software package. While `Drizzle` accomplishes its main objective of rigorously preserving flux, it does not allow users to assess the quality of the sampling of the output pixels and the correlations among them. The new algorithm outlined here represents a simple least-squares approach to the problem, and as such it is able to rigorously constrain the sampling to be regular and is also able to evaluate explicitly the covariances among pixels. We validate the approach on a portion of the UDF. A subsequent report will make use of the algorithm presented here in order to evaluate the quality of reconstructed images in terms of the sharpness of the PSF and the number and quality of dithers. The algorithm presented here may have broad application for HST science, as well.

## 2 Introduction

Planning observations with NIRCam will involve several steps. First, a target will be chosen, along with the desired S/N to achieve in each filter. The ETC (Exposure-Time Calculator) will then be used to determine how much total exposure time is required. Finally this total time must be divided among some number of exposures. One obvious reason for this is that the maximum exposure time on NIRCam will be 4000 s, but the main reason to take multiple exposures is that it allows users to identify and remove systematic artifacts from their data.

Once the number of exposures has been determined, NIRCam users will have to choose an appropriate dither pattern to optimize their science. There are two kinds of dithers: primary and secondary. The primary dithers involve large offsets, and the secondary dithers are small offsets executed at each of the primary-dither pointings. This gives a total of $N_P \times N_S$ exposures for a particular dither pattern.

**Operated by the Association of Universities for Research in Astronomy, Inc., for the National Aeronautics and Space Administration under Contract NAS5-03127**

The user must decide how to divide up the dithers among primary and secondary. The more secondary dithers are taken, the better the sub-pixel sampling will be. The more primary dithers are taken, the more each source will get moved around the detector thus mitigating L-flat errors, image defects, and the impact of detector gaps[1]. Secondary dithers will involve steps of a few pixels, so they will mitigate image defects somewhat, but they will not accomplish everything that large dithers can. In general, the small secondary dithers will be about 10 pixels (0.32″) and the large primary dithers will be anywhere from 300 pixels (10″) to 1500 pixels (50″).

A general rule of thumb will be to take only as many secondary dithers as are necessary to adequately sample the scene, since the more large dithers we take, the better we will mitigate *all* sources of systematic error. Thus, it will clearly be important to determine how many secondary dithers we need to provide adequate sampling.

This report and its companion will investigate the quality of image reconstruction as a function of dither pattern. We will use realistic model PSFs for several NIRCam filters to simulate observations and will combine simulated images taken with various dither strategies in order to assess the quality of the sampling. The actual tests will be done in the follow-up report. The present report gives an overview of image reconstruction strategies and develops a tool that can be used to combine images and evaluate the output.

In general, there is no unique way to combine multiple dithered observations of the same undersampled scene into a single composite image. If the images are well-sampled, then a simple shift-and-add-type approach can produce a near-optimal stack. If there is no dither, or if there are only whole-pixel dithers, then the same shift-and-add solution works as well, even in the presence of undersampling. However, when the detector is undersampled and there are sub-pixel dithers present, it means that there is fundamentally different information in the different exposures, and we must be careful how we combine them if we want to preserve that information.

We the report begin by defining the high-level challenge of image reconstruction in terms of what information each individual exposure contains about the scene and what we might hope to gain by combining them. To this end, we present the concept of the "effective" scene, and how this scene is sampled by the detector pixels in an individual exposure. We then discuss various standard ways of reconstructing this scene. We find that all the available algorithms have drawbacks, so we propose a new method for scene reconstruction that has the benefit of being least-squares. As such, it naturally provides valuable information about variances and covariances among pixels in the scene. We spend several pages motivating the new approach and deriving the equations that will allow us to go from a set of dithered images of a scene to a single composite representation. Finally, we demonstrate this new tool by reconstructing a galaxy from the UDF (Ultra-Deep Field) images taken with HST's Advanced Camera for Surveys.

While this new approach can be used for scientific-image analysis, the immediate goal of this pair of reports is to provide a mechanism for evaluating the quality of particular dither patterns in terms of their ability to reconstruct a scene.

[1] The issue of detector gaps will be non-trivial for NIRCam; a minimum of 3 large dithers are necessary to minimally cover all gaps in the field.

## 3 The Effective Scene

In an epic tutorial entitled *Combining Undersampled Dithered Images*, Lauer (1999) makes the distinction between image-reconstruction procedures that involve deconvolution and those that do not. One of his main points, which was under-appreciated at the time, is that considerable image reconstruction can be accomplished without recourse to deconvolution procedures. Deconvolution is well known to amplify noise and is extremely sensitive to errors in the PSF (point-spread function), so it should be avoided, if possible.

Lauer begins by noting that there are several scenes of interest. There is the raw astronomical scene that arrives at the telescope and can have structure at all spatial scales, including the finest scales imaginable. The raw scene then goes through the telescope's optics, which essentially convolves it with the instrumental PSF. This new scene has structure that is only as fine as can be resolved by the PSF. Intrinsic structure in the astronomical scene that is finer than this gets blurred out by the PSF. This "optical" scene is collected in the detector pixels and read out for analysis.

There are two equally valid ways to think of this last operation. We can think of it as an integration of the flux across the face of each pixel in the two-dimensional detector array. This is the normal way of thinking about how images are made. But, we can equivalently think of the pixelization operation as an additional convolution of the scene with the pixel-response function. The pixel values would then represent point samplings of this doubly convolved scene. Since convolution is commutative, we can treat the double convolution as a single convolution with a composite PSF. We will call this the "effective" PSF. It is simply the instrumental PSF convolved with a pixel:

$$\psi_E = \psi_I \otimes \Pi,$$

where $\Pi$ represents the response function of a pixel. It usually resembles a top-hat function, but it can have very different shapes. It can be smaller than an actual pixel, in the case where the pixel edges are insensitive; or it can be larger than a pixel, in the case where there is charge diffusion or inter-pixel capacitance. In practice, all of these effects can be present at the same time. The edges of pixels can sometimes be less sensitive because they contain electrodes to keep the electrons confined to the pixels; this leads to pixel edges that are less sensitive. Charge diffusion happens when a photon that lands at the edge of one pixel ends up generating an electron with enough kinetic energy to kick it into an adjacent pixel. Inter-pixel capacitance arises when the charge measured in one pixels can be affected by charge stored in adjacent pixels. The function $\Pi$ can have structure on very fine spatial scales, since it can have edges as sharp as the edge of a pixel. Therefore the structure in $\psi_E$ will be largely limited by the smoothness of $\psi_I$, the instrumental PSF from the telescope's optics.

Lauer encourages us to think of the detector image in the second way. Each detector pixel samples the "effective" scene at one point, and the array of pixels in an image sample the effective scene at an array of points. The effective scene is simply:

$$S_E = S_I \otimes \psi_E,$$

where $S_I(x,y)$ is the infinite-resolution scene delivered to the telescope. Essentially, the smooth, 2-D function $S_E(x,y)$ tells us how much flux an image pixel would receive if it were centered at location $(x,y)$ in the reference frame.

In conceptualizing the meaning of $S_E(x,y)$, it helps to imagine a thought experiment where you have a pixel that can be moved arbitrarily across the astronomical scene that is delivered to the face of the detector. For each location of the pixel, there is a particular number of electrons that would be recorded if it were placed there and exposed for the integration time. It is easy to see that the number recorded in this pixel would vary smoothly and continuously as the pixel is moved across the scene. If the scene that impinges on the detector is undersampled and has sharp features, then it is easy to see that the 2-D effective image will have similarly sharp features, as the features go in and out of the pixel-sized window.

The 2-D function $S_E$ is the product of a convolution, so it can have only as much structure as $\psi_E$, which itself is limited by the instrumental PSF, $\psi_I$. An observed image can then be thought of as an array of point-samplings of this smooth, two-dimensional function. These point-samplings are naturally spaced by the distance between pixel centers. To the extent that these pixels are spaced finely enough to sample all the structure in $S_E$, then — apart from detector artifacts and noise-related issues — a single image can be considered a full representation of the scene. However, if the pixels are spaced farther apart than the resolution scale in $S_E$, then the pixels in a single image will not adequately sample the effective scene, and as a result we will have imperfect knowledge of it, even in the absence of bad pixels or noise.

It is well known that dithering can mitigate the impact of bad pixels or cosmic rays. But dithering can also mitigate the effects of undersampling. If we shift the sample grid by half a pixel in the x direction, then combine the data in both exposures, we will end up with twice the frequency of samples along that axis. Structure that was too fine to be observed with the 1-pixel-spaced sampling, might now be detectable with this 0.5-pixel-spaced sampling.

Figure 1 illustrates these concepts. Panel (a) shows the astronomical scene, and the "instrumental" scene ($S_I$) that results when the infinite-resolution truth is convolved with an undersampled optical PSF. Panel (b) shows the "effective" scene ($S_E$) that results when this is subsequently convolved with the profile of a pixel. Panel (c) shows how this effective scene is sampled by real-image pixels. Depending on where the pixels are

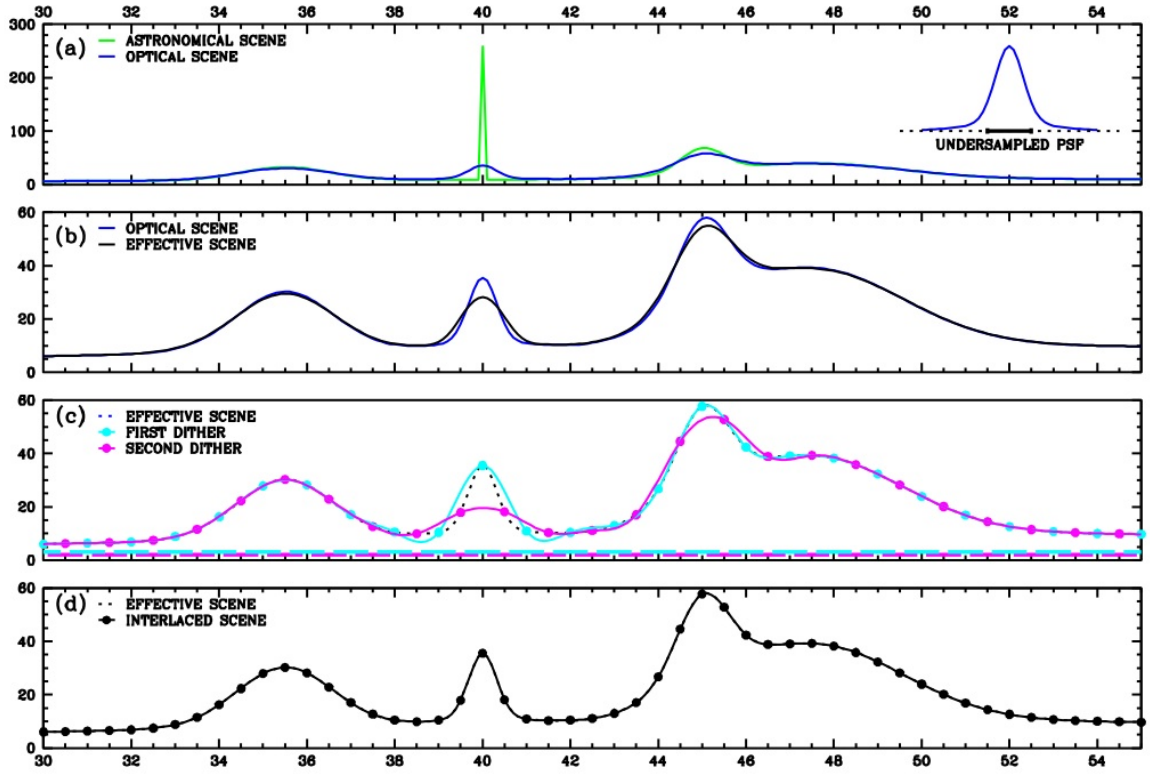



**Figure 1: (a) The raw astronomical scene and the optical/instrumental scene ($S_I$) delivered to the detector. PSF shown at left, with the size of a pixel marked immediately below it. (b) The optical and effective scenes ($S_E$). (c) How two different dithers would observe the scene, and how we would interpolate the image using each dither independently. The pixel boundaries for each are shown along the bottom of the panel. (d) How a combined version of the two images would sample and interpolate the scene.**

centered relative to the objects, a very different scene can be perceived. The filled dots correspond to the locations in the field where the two different exposures sample the scene. The colored lines are splines that connect these observed pixels. Without more information than a single image contains, this is the best estimate we have of what the scene is doing between the pixels. It is clear that a single image is not able to recover all the structure in the scene and two differently sampled observations imply very different scenes.

Panel (d) shows how well we can reconstruct the scene if we have the benefit of both dithers. Here, we have "interlaced" the dithers to double our sampling of the scene. The interpolation of the scene is now extremely robust: we have recovered all the structure in the scene.

Lauer's main thesis is that observations with different dithers can be combined in this manner without ever leaving the purely observational domain. The effective scene is a purely observational function that is sampled regularly, though sometimes inadequately, by the image pixels. A complete representation of the effective scene can be reconstructed by simply combining multiple dithered observations. This procedure is direct and involves no deconvolution. As such, it should reduce noise, rather than

amplifying it. Furthermore, the procedure depends on no assumptions about the PSF, except that it is the same for all exposures.

Lauer advises that if, in the end, information in the intrinsic scene $S_I$ is desired on a finer scale than can be discerned in this effective scene, then the best thing to do would be to first construct this effective scene, and only then to apply a deconvolution algorithm to it. A properly reconstructed effective scene $S_E$ will be well-sampled and should be more amenable to such an deconvolution operation[2]. Performing a deconvolution on poorly sampled individual exposures involves many unknowns at the same time: the pixel-response function, the instrumental PSF and furthermore it can recover no more structure than one resolution element per image pixel.

While deconvolution was quite popular in the early days of HST, particularly when a spherically aberrated PSF was still the best the telescope could produce, it became clear over time that deconvolution rarely produced unique, definitive results. So, most current image-analysis techniques focus on determining the effective scene (or similar constructs) or on parametric forward-modeling-type procedures.

## 4 Reconstruction Of The Effective Scene

The above discussion presented the concept of the effective scene. We showed that this scene is directly observable in the pixels of individual exposures, even though single observations cannot always perceive all of its structure. There are many approaches to reconstructing a super-sampled image of the scene from multiple independent dithers. In this section, I will give an overview of common image-reconstruction approaches that aim to recover a well-sampled scene from under-sampled images.

It is worth noting at the outset that while there is in fact a single effective image that can be considered to be the parent for a given set of dithered observations, there is not always enough information in the achieved dithers to arrive at this optimal product. Given this frequent fundamental limitation to typical data sets, many image-combination algorithms must be robust for these sub-optimal cases, and as such they have improving the sampling as but one of their many goals. `Drizzle` is one such algorithm (see Fruchter & Hook 2002).

### 4.1 Interlacing

A super-sampled composite image can most easily be constructed from a set of images that has been dithered in a regular square pattern. The simplest regular pattern is a 4-

[2] We note that such a deconvolution would involve $\psi_E$, and not $\psi_I$ and $\Pi$ individually; in practice we rarely need to know $\psi_I$ and $\Pi$ separately, since we only encounter $\psi_I$ after it has been integrated over pixels and we only encounter $\Pi$ in the context of the point-spread function (see Anderson & King 1999 for a discussion). If we could move pin-point laser sources around the detector, we could recover $\Pi$ without $\psi_I$, but we do not generally have such data. Thankfully, we do not require it.

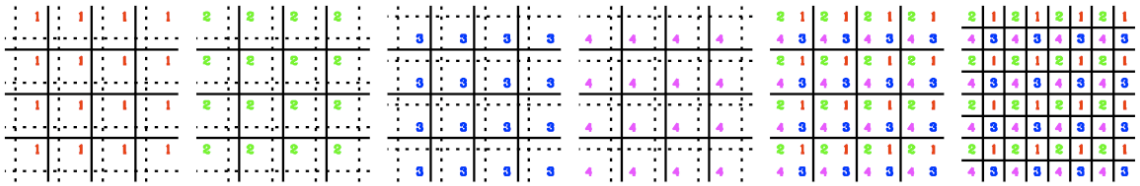


**Figure 2: The first four panels show how the pixels from four dithered images map onto the pixels of the reference frame (solid pixel frame). The pixel centers are marked. The fifth panel combines the mappings, but keeps the original reference-frame pixel scale. The final column shows that if we re-bin the reference-frame pixels, we can interlace the samplings to directly supply values for the finer pixels.**

point box dither: (0.0, 0.0), (0.0, 0.5), (0.5, 0.0) and (0.5, 0.5), and it can be constructed by simple interlacing. Often the individual dithers are offset by whole-integer pixels as well, in order to mitigate the impact of single bad pixels. So long as the shifts are small, the sub-pixel dither will remain coherent across the detector, but if the shifts are too large, then distortion will cause them to become non-optimal at the edges.

Figure 2 shows how the pixels in the above pattern map onto the reference frame. The four panels on the left show in dotted lines the boundaries of the pixels in each exposure. This figure shows the pixels in the sense of integrating over the instrumental scene $S_I$ that is delivered by the telescope optics to the detector before pixelization.

If we think of each pixel in the second sense discussed above, then we can map the center of each pixel into the master frame and consider the pixel value to be a point-sampling of the continuous 2-D effective scene, $S_E$. Recall that the effective scene is simply the 2-D function that tells us how much flux would be recorded by a pixel that is centered on the sampled location in the scene. We can construct a 2× super-sampled version of $S_E$ by simply interlacing the pixels of the four dithers as shown in the rightmost panel. This new image has twice the sampling of the original image, and is able to sample 2× finer variations. **It is important to keep in mind what this image represents. It is a super-sampled version of an original exposure. It *does not* represent the instrumental scene $S_I$ integrated over smaller pixels. It *does* represent $S_I$ integrated over the regular "large" image pixels, but sampled more finely.**

Figure 3 provides an example of an interlaced 2-D reconstruction. The image on the left shows a finely sampled image of a simulated effective scene that contains an equal-brightness binary with a separation of 1.25 image pixels in WFC3/IR through filter F110W. This scene is imaged at four different dithers, and the 5×5-pixel cut-outs from each dither are shown in the upper right. Several of these images appears to be slightly extended in the horizontal direction, but otherwise it looks like it could be one source. If

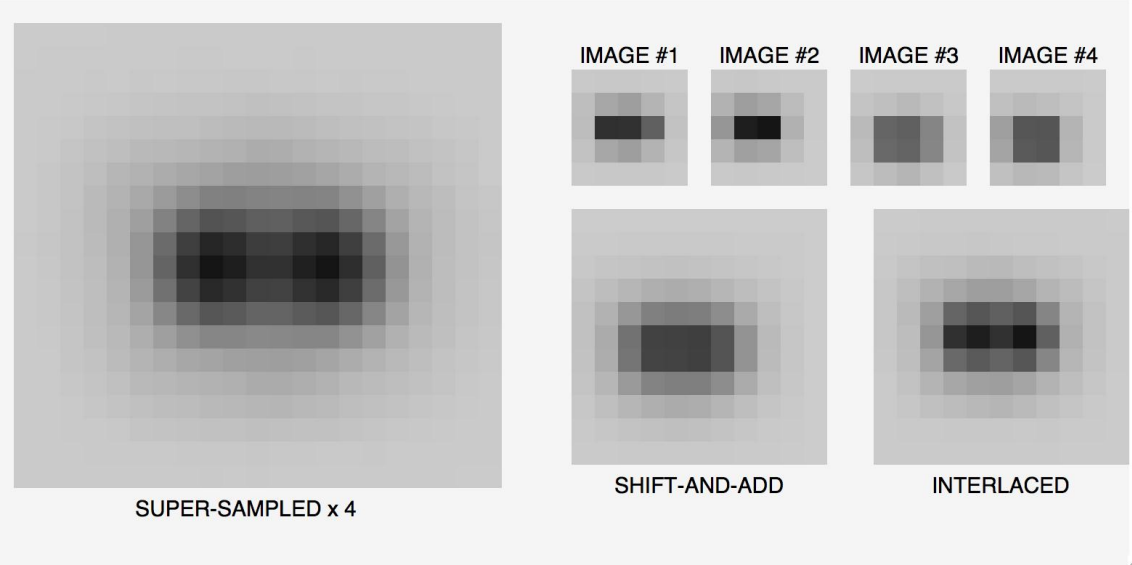



**Figure 3: (Left) The simulated effective scene at ×4 the pixel sampling. (Top right) the scene as observed in four different box-dithered images. (Bottom right) The shift-and-add ×2 reconstruction of the scene and the interlaced reconstruction.**

we do a simple shift-and-add combination we get the ×2 sampled image in the bottom middle panel, which is clearly blurrier than the true effective scene. The source shows a clear extension, but it is not clear whether this is two sources or a resolved extended source. The interlaced image is shown on the bottom right. Each pixel in this image comes from one of the pixels of the four source images. The binary nature of the source is now clear. Note that the interlacing did not improve the *resolution* of the scene, which was limited by the structure present in the effective image on the left, but it did improve the *sampling* of the output scene. Shifting and adding actually degraded the resolution.

In summary, interlacing is the ideal way to improve the sampling of a scene. It retains all of the information in the individual exposures and provides the best possible sampling for a given number of dithers. Unfortunately it requires a perfect dither. To the extent that the dithering is not perfect, the image sampling will be irregular, and the pixel values in the output image will not reflect the value of the effective scene $S_E$ at the centers of the output pixels. In general, distortion in the optics will cause the pixel-phases of dithers to drift relative to each other from the center of the detector to the edge, so it is impossible difficult to ensure a good dither everywhere. Moreover, it is anticipated that the FGS on JWST will be able to dither with an accuracy of 7 mas, which is about 20% of a NIRCam short-wave pixel, so our dithers will be limited to that accuracy. This is all the more reason to come up with algorithms that work in non-ideal situations.

### 4.2 Drizzle

`Drizzle` is by far the most common tool used to combine multiple exposures. It is designed to accomplish several goals at once. Among its many goals are: (1) correction of geometric distortion, (2) rigorous conservation of flux, (3) the ability to robustly reject bad data, and finally (4) the improvement of pixel sampling where possible. The pipeline version of `Drizzle` must do well on all these fronts, even for data sets that do not have adequate dithers for a proper sub-pixel restoration.

One of the main reasons that Lauer wrote his 1999 treatise was his concern about the sampling properties of the `Drizzle` algorithm, which was still under development at the time. He contended that a non-zero value for the drop-size parameter causes a pixel to influence a larger region of the effective image than its point-sampled nature would suggest. There are sophisticated versions of the algorithm, such as `Multi-Drizzle`, that allow the user to empirically vary the drop-size and output-image sampling in an effort to come as close as possible to the true "effective" image for data sets where there exists an optimal dither (see Koekemoer et al. 2002). However, at best the `Drizzle` procedure can only output an image that represents the effective scene convolved with an output pixel, since even with a negligible drop-size, where input pixels are treated as delta-function samplings of the effective scene, these drops will in general not always land at the centers of the output pixels, and therefore the output pixels will have samplings that correspond to various locations within that output pixel. As such, `Drizzle` can only attain the true resolution of the effective scene in the limit of infinitely small output pixels, which would require an infinite set of dithers to constrain.

Equally concerning to Lauer is the fact that input pixels are not always "dropped" precisely at the centers of the destination pixels. This means that the input-image pixels that contribute to a given destination pixel could easily have some average offset, in the flux-weighted sense, from the center of the destination pixel. As a result, the value determined for the destination pixel would not correspond to the value of the "effective" image at the destination pixel's center in the output array, but rather the arrived at pixel value would correspond to that of the weighted centroid of the contributing pixels. The destination image will thus in general have uneven sampling, and it could be dangerous to analyze it by traditional routines, which generally assume regular sampling (such as those that do PSF-fitting, deconvolution, parametric shape-modeling, etc). These caveats are well described in the `Drizzle` documentation.

It is worth underlining that the mandate of `Drizzle` was considerably broader than simple reconstruction of the effective scene in optimal circumstances, and it has accomplished its many goals admirably, as can be witnessed by its longevity and continued popularity. It is currently anticipated that `Drizzle` will be a major part of the JWST pipeline, so it is not going anywhere. Nevertheless, in acknowledgement of the possibility of improvement, a new beta version, called `iDrizzle`, has recently been developed (Fruchter 2010) in an effort to address many of Lauer's sampling concerns.

A final concern voiced in Lauer (1999) is there are correlations between the pixels in `Drizzle`'s output image when there is a non-zero drop-size . To some extent, some correlations between output pixels cannot be avoided. If an image pixel samples a location of the scene exactly between two output pixels, then by symmetry, it will

contribute equally to both of them. Lauer's main concern is not that correlations are present (apart from the case of perfect interlacing, they *must* be), but rather that they are not quantified and are often ignored by users. Drizzle does weight the input pixels by exposure time, and uses this information to estimate the ultimate error in the flux in the pixel, but the correlations between pixels are not considered.

### 4.3 Fourier reconstruction

Lauer (1999) recommends a Fourier-based approach to combine individual dithered images into a super-sampled composite. He takes a set of dithered images and maps each pixel into the reference frame. He then determines how many Fourier components are needed to represent all the structure in the scene, and solves for the band-limited transform that best represents the mapped samplings. The inverse of this transform is the estimate of the super-sampled composite. The sampling of the destination frame is rigorously regular, since it comes directly from the Fourier transform.

This Fourier approach works well over small regions, but it breaks down in the presence of distortion, which causes the spacing between dithers to change across the region being solved for. The Fourier approach is also not optimized to identify and remove artifacts, or to geometrically rectify the scene, though such modifications could be accomplished through iteration.

### 4.4 The need for improvement

Lauer's Fourier approach comes the closest to generating a super-sampled version of the effective scene, $S_E$. Unlike `Drizzle`, the Fourier approach is able to rigorously constrain the sampling in the output image to be regular. It is also able to make some assessment of covariances between pixels. However, it does have several limitations. It does not have a direct mechanism for dealing with artifacts such as bad pixels or cosmic rays in the input images, or pixels with different weights. Also, in its current formulation, it can operate only over a limited region of the image, since it assumes the dithers to be fixed relative to each other and to have the same orientation. Although one of Lauer's concerns with `Drizzle` has to do with correlations among pixels, some correlations should be present among Fourier-reconstructed pixels, but he does not provide an explicit mechanism for evaluating such correlations in his formalism. Finally, Lauer approach has no mechanism to deal with geometric distortion, so the image-rectification phase would have to be done at some later time.

`Drizzle`, on the other hand, does properly account for distortion and is able to merge images taken at different orientations. It is also able to evaluate data quality and reject previously known bad pixels and pixels that are found to be discordant (such as CR-impacted pixels). Its main limitations are threefold. First, while it aims to improve sampling, the finite drop-size of the pixels and the way it constructs the (strictly positive) weights for the output pixels means that the output pixels do not correspond to the effective image, but rather to a version of it that is blurred by the output-pixel and the drop-size profiles. Second, it does not rigorously preserve the output-image sampling. This means that the effective center-of-light for each pixel can be offset by a fraction of a pixel from the nominal output-pixel center. Finally, `Drizzle` has no mechanism to estimate covariances among the output pixels. Understanding the covariances and

properly including them in the analysis will be important for any algorithm that fits models to pixels. None of these concerns should take anything away from `Drizzle`'s universal appeal, but if our aim is to construct a properly sampled composite image, we must improve on this.

In the next section, I will come up with a least-squares-based image-restoration algorithm that takes the best aspects of both approaches. The image it constructs will correspond as closely as possible to the "effective" scene, which is sensed directly in each individual exposure, albeit with inadequate sampling. The approach will allow for distortion and different orientations among the input image pixels. It will also incorporate an iterative approach in an effort to reject image artifacts and otherwise discordant pixels, and can include proper error-weighting for pixels. Finally, it will rigorously preserve the regular sampling of the output image, and will also allow an explicit calculation of the covariances among the neighboring pixels.

## 5 A Least-Squares Approach

At the heart of the new approach that will be presented here is a least-squares formalism. The goal will be to take a large number of samplings of the effective scene (which come from multiple dithered exposures) and determine the single composite image that best represents all the observations.

We will assume in the following derivation that either using the telemetry information in the image headers or using previous star-based transformations, we can transform the location of the center of each pixel in each individual exposure into the reference frame with essentially no error. This is a standard assumption of both `Drizzle` and Lauer's algorithm. We also assume that the scaling and background of the pixel values in each exposure have been adjusted to correspond to the same exposure time and sky level. We will further assume that the PSF is the same for all exposures, so that each exposure is in fact sampling the same "effective" scene. It would be interesting to evaluate how imperfections in the transformations or variability in the PSF (with time or orientation) will impact image reconstruction, but since we do not yet know how stable JWST (PSF-wise and distortion-wise), it is hard to consider these issues at present. Here, we will

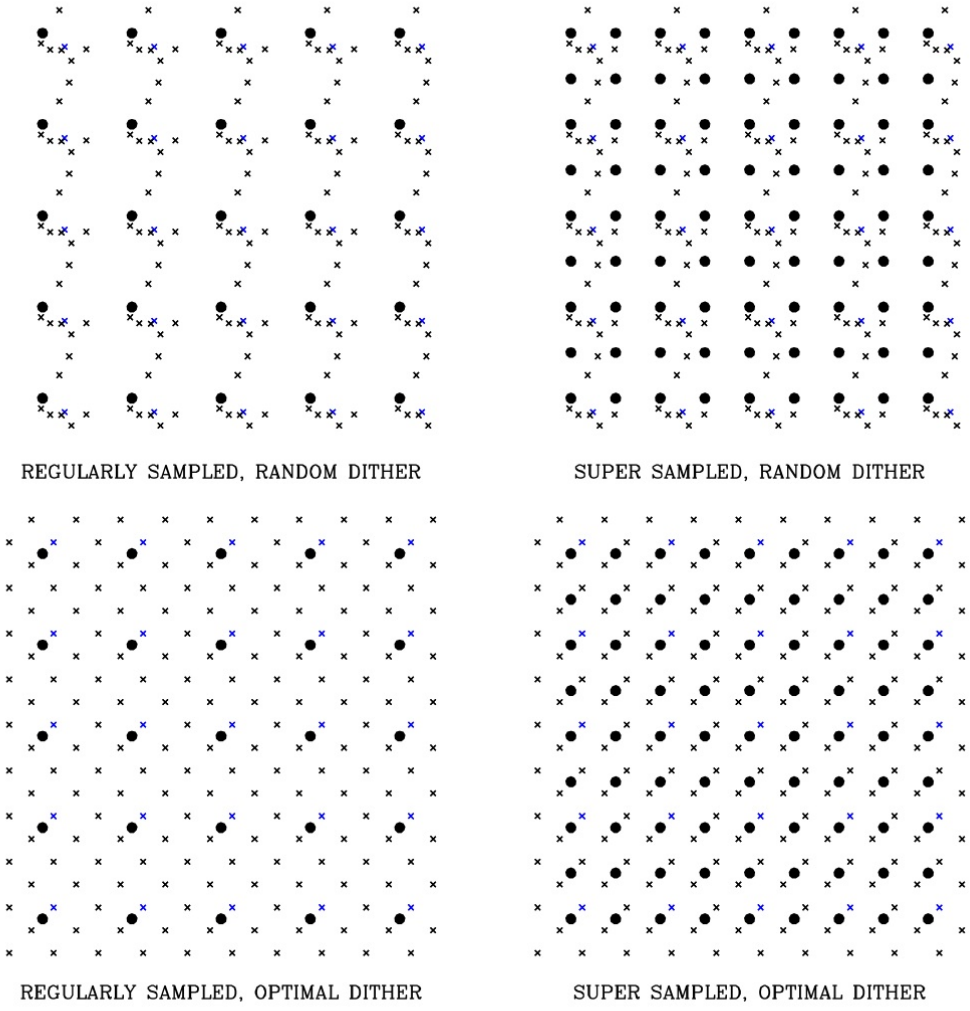



**Figure 4: (Top) The sampling of the reference frame achieved by a random 8-point dither pattern. The solid dots indicate ×1 reconstruction sampling on the left and ×2 reconstruction sampling on the right. (Bottom) An optimal 8-point dither pattern with the same sampling. In all panels, the first exposure is shown as the blue crosses.**

confine ourselves to the general issue of reconstructing a single effective scene from a general assortment of samplings.

Once all these conditions are met, we should have a well-posed problem. We have a set of samplings, as shown by the ×'s in Figure 4. Each of these samplings comes from one pixel in one of the input images and consists of a location in the reference frame $(x,y)$ and a pixel value (P). The samplings from the "first" image are highlighted as blue crosses. The samplings might also include an estimate in the error in the pixel value, but for simplicity here, we will assume they all have the same error. Each sampling represents a point-measurement of the "effective" scene at its location in the reference frame, in that it literally is a measurement of the flux recorded by a pixel placed at a particular point in the scene. Our task is to find a representative value for the scene at the locations of the ● 's, which represent the centers of the pixels in the destination image.

To distinguish them from the image pixels, we will refer to them as "grid-points" here. We have the flexibility to increase or decrease the sampling frequency of the destination image, as is shown schematically in the right-hand panels. The two sampling regimes shown are ×1 and ×2, but there is no reason we could not have ×1.5 or even ×1.31.

If we want to solve for the set of grid-point values that best represent the observed samplings at the × locations, we need to know how the values of the grid-points can be used to predict the effective image $S_E$ at the location of the samples. This will involve finding some way to interpolate among the grid-points at the locations of the observed constraints. If we can do that, then we can optimize the values of the grid-points to best represent the collection of samples. Our additional aim is to linearize the problem, so that we can use simple linear least-squares to arrive at the set of ● 's that best represent the set of ×'s.

Clearly if there are as many grid-points as sample points, then the output image will be just barely constrained. If there are more grid-points than samples, we will be under-constrained, and if there are fewer, we will be over-constrained. We can afford to space out the grid-points as we see fit in order to both ensure a sufficiently constrained problem, but also to properly correspond to the anticipated structure in the "effective" image. There is no sense in sampling $S_E$ more finely than is justified by the structure in $\psi_E$, so the goal should be to set the output pixel scale to match this, provided there are enough contributing images and provided they are adequately dithered.

An additional benefit of linearizing the problem is that as a by-product of inverting the least-squares normal equations, we will get estimates of the variances and co-variances in the pixel values. This will figure prominently in the second report in this series.

### 5.1 Setting up the least-squares problem

In solving for the values of the grid-points, our goal is to find the set of grid-point values such that when the grid is interpolated at the location of the samples, the residuals between the interpolated prediction and the observation are minimized (in a least-squares sense). So, before we can solve for the grid points in terms of the observed samples, we must first determine how to estimate the value of the grid at the location of each of the samples. In general, the samples will find themselves in-between grid-points. We must therefore do a two-dimensional interpolation within the grid. Under the formalism we adopt below, the interpolated value will simply be a *linear* combination of the neighboring grid-points:

$$S_E(x,y) \;=\; \sum_{\substack{abs(I-x)<2 \\ abs(J-y)<2}} a_{ij} \times G[I,J] \quad , \tag{1}$$

where $S_E(x,y)$ is the value of the "effective"-image grid interpolated at position $(x,y)$ in the output-image frame. The coefficients $a_{ij}$ are fixed for a given $(x,y)$ location and tell us how the value of each grid-point $G[I,J]$ contributes to the interpolated value of the 2-D function. With the bi-cubic spline interpolation that we will adopt here, only the grid-points within ±2 pixels of a given location have any impact on the interpolated value at that location. If we have a well-sampled image that we want to interpolate, the above

equation will allow us to sub-sample it in a way that is continuous with continuous first derivatives.

### 5.2 The big-picture plan

Once we have a way to calculate the coefficients in the above linear relation, we will know how to estimate the value of each pixel sample from the values of the grid-points, since we assume that we have perfect knowledge of the reference-frame location $(x,y)$ for each pixel. So, since we can express the value of the each of the observed pixels in terms of a linear combination of the grid-point values, it should be possible to invert the problem and treat each pixel as a constraint on the grid-points. We will of course need many observed-pixel constraints before we can solve uniquely for the many grid-point values, but the concept is clear.

To solve for the grid, at a minimum, we need at least as many constraints as we have grid-point values to solve for, but ideally we will over-constrain the grid with many more observed pixels than grid-points. Each pixel in each contributing exposure represents a single constraint on the grid:

$$P_{ijn} = \sum_{IJ} a_{IJ;ijn} \times G[I,J]. \qquad (2)$$

Here, pixel $P_{ijn}$ is located at location $[i,j]$ in exposure $n$. We assume that we have access to transformations that will tell us where this pixel maps to in the reference frame. Its position will be $(x_{ijn}, y_{ijn})$ in this output frame. The above relation then gives us a set of coefficients, $a_{IJ;ijn}$, that tell us how each grid-point influences the value of the interpolated function at the sampled location. The estimated pixel value is simply a linear combination of the grid-point values.

In general, we will have a great many samplings of the scene, and each one will be located at a different place within the grid. As such, the model value for each sampling will be interpolated by a different linear combination of grid-points. This can be thought of as a system of $N_n \times N_i \times N_j$ equations (where $N_n$ is the number of images, $N_i$ is the dimension of pixels in $i$, and $N_j$ is the dimension in $j$; this product is simply the total number of pixels) that relate the values of the grid-points to the values of the pixels. If we can invert this system of equations, we can solve for $G[I,J]$ as a function of the observed pixels $P_{ijn}$: $G_{IJ} \Rightarrow \Sigma_{ijn}\, b_{ijn;IJ}\, P_{ijn}$.

But we are getting ahead of ourselves. Before we think of solving for the grid-points, we must come up with an algorithm to interpolate the grid at an arbitrary location within the grid, and thus give us the value of the interpolated function, as in equation 1.

### 5.3 An example from one dimension

There is no unique way to interpolate functions, particularly in two dimensions. So, before looking at the difficult two-dimensional case, let us consider what we can learn from one-dimensional splines.

The goal of a 1-D spline is to provide the simplest function that goes through the specified set of points exactly, while simultaneously being continuous and smooth to

whatever degree is desired. Usually smoothness in the first derivative is sufficient, but higher-order constraints are also straightforward to implement.

One way to construct such a function is to record for a set of nodes along curve G the value of the function (trivial), and the value of the first derivative (obtained by fitting a polynomial to the neighboring nodes). With these data in hand, we can now construct the function in-between any two nodes (M and N) by simply solving for the function that satisfies these four boundary conditions:

$$\begin{aligned} f_{MN}(M) &= G[M] \\ f'_{MN}(M) &= G'[M] \\ f_{MN}(N) &= G[N] \\ f'_{MN}(N) &= G'[N] \end{aligned} \quad . \tag{3}$$

Here, $f_{MN}(x)$ is the function that we wish to constrain between $x = M$ and $x = N$, and $G$ and $G'$ represent the value and the derivative of the curve at the given nodes. Note that in none of this have we required the nodes to be evenly spaced. The solution becomes easy and regular when that is the case, but it is not necessary. For simplicity, though, we will now assume regular-spaced grid-points in what follows.

At the moment, we do not care what the function $f_{MN}(x)$ does outside of $\{M,N\}$, since this function will only be used for evaluating the curve between these two points. Since we constrain the value and the derivative of the function at each of the endpoints, then when we evaluate the function in a similar manner between the next two points ($N$ and $O$), the spline will be guaranteed to be smooth and continuous across the boundary.

We have four constraints on $f_{MN}(x)$, so we will have to represent it with a cubic function of the form:

$$f_{MN}(x) = A_{MN} + B_{MN}x + C_{MN}x^2 + D_{MN}x^3 \tag{4}.$$

For simplicity, let us define $\phi = x - M$ and assume that the grid-points are evenly spaced by "1" unit, so that $\phi = 0$ corresponds to the point at $x = M$, and $\phi = 1$ corresponds to the point at $x = N$. So, we are now seeking the function: $f(\phi) = A\phi + B\phi^2 + C\phi^3 + D\phi^4$. We can determine values for the coefficients $A$, $B$, $C$ and $D$ by evaluating the function $f$ and its first derivative at points $\phi = 0$ and 1:

$$\begin{aligned} f(0) &= G[M] &&= A \\ f'(0) &= G'[M] &&= B \\ f(1) &= G[N] &&= A + B + C + D \\ f'(1) &= G'[N] &&= B + 2C + 3D \end{aligned} \tag{5}$$

We can now solve for $A$, $B$, $C$, and $D$ to get:

$$\begin{aligned} A &= G(M) \\ B &= G'(M) \\ C &= -3G(M) - 2G'(M) + 3G(N) - G'(N) \\ D &= 2G(M) + G'(M) - 2G(N) + G'(N) \end{aligned} \tag{6}$$

Then if we plug the above equation into the equation for $f(\phi)$ we find that:

$$\begin{aligned} f(\phi) &= 1\times( \quad G[M] \qquad\qquad\qquad\quad ) \\ &+ \phi\times( \qquad\qquad G'[M] \qquad\qquad ) \\ &+ \phi^2\times( -3G[M] \quad -2G'[M] \quad +3G[N] \quad -G'[N]) \\ &+ \phi^3\times( \quad 2G[M] \quad +G'[M] \quad -2G[N] \quad +G'[N]) \end{aligned} . \tag{7}$$

It can easily be shown that this satisfies our four constraints on the function and its derivatives at $x$ = M and $x$ = N from Equation (5).

The right-hand side of Equation (7) contains the node points and their derivatives. To get the node-point derivatives, we can use simple finite-difference techniques. If we fit a parabola to the three points centered on each node $n$, we get G'[$n$] = (G[$n$+1]-G[$n$-1])/2. Specifically, G'[$M$] = (G[$N$]-G[$L$])/2 and G'[$N$] = (G[$O$]-G[$M$])/2. We can then plug-in the formulae for these derivatives into the above equation to get *everything* in terms of the values of the node-points at L, M, N, and O:

$$f(\phi) = \begin{pmatrix} 1 \\ \phi \\ \phi^2 \\ \phi^3 \end{pmatrix}^T \begin{bmatrix} 0 & +1 & 0 & 0 \\ -\frac{1}{2} & 0 & +\frac{1}{2} & 0 \\ +1 & -\frac{5}{2} & +2 & -\frac{1}{2} \\ -\frac{1}{2} & \frac{3}{2} & \frac{3}{2} & +\frac{1}{2} \end{bmatrix} \begin{pmatrix} G[L] \\ G[M] \\ G[N] \\ G[O] \end{pmatrix} \tag{8}$$

This is a powerful equation. It gives us the spline-fit value for any point $x$ ($\phi = x - $ M) between node-points M and N, simply in terms of the four surrounding node-points at L, M, N, and O. It is a completely closed-form solution for the smooth continuous function that takes us from grid-point M to grid-point N. Note that, if we use the same procedure to compute the function over the adjoining region between N and O, the spline is guaranteed to be continuous across the node-N boundary up to and including the first derivative, since both branches will constrain it to go through N at G[N] with the derivative G′[N]. We would like to come up with such a convenient formalism for the two-dimensional case.

### 5.4 Going to two dimensions

The 1-D family of splines is closed-form and can be nicely scaled in terms of polynomial order. We can define a quadratic fit to the three node-points centered on a particular node-point to get the first two derivatives, or a quartic fit to five points centered on a point for derivatives up to and including $G^{(4)}$. We can then design a polynomial to respect these derivatives and it is guaranteed to be smooth in-between the specified nodes.

Things become more complicated with two dimensions. In the 1-D case, we are constrained to go through the node points as we move along the function. Thus, we were able to constrain the function at each node and we were certain that nothing could go "around" the nodes. In the 2-D case, one can move around the (x,y) space of the grid

from the domain of one grid-point to that of another and never actually go through any grid points. Rather than simply constraining the function and its derivatives at the grid points, we will have to constrain it in many more places (for example, along the lines between the pixels).

In the 1-D case above, we chose to constrain the function and its derivative at each node point, which gave us four boundary-value constraints to satisfy for the function between two nodes. This required a third-order polynomial, and naturally the four nearest node-points total to constrain that. The 2-D Cartesian polynomials have 1 term to $0^{th}$ order, 3 terms to $1^{st}$ order, 6 terms to $2^{nd}$ order, 10 terms to $3^{rd}$ order, 15 terms to $4^{th}$ order, and 21 terms to $5^{th}$ order:

$$\begin{aligned} f(x,y) &= A \\ &+ Bx + Cy \\ &+ Dx^2 + Exy + Fy^2 \\ &+ Hx^3 + Ix^2y + Jxy^2 + Ky^3 \\ &+ Lx^4 + Mx^3y + Nx^2y^2 + Oxy^3 + Py^4 \\ &+ Qx^5 + Rx^4y + Sx^3y^2 + Tx^2y^3 + Uxy^4 + Vy^5 \end{aligned} \tag{9}$$

Once we decide what degree of polynomial to use, we must consider which pixels to include. Unfortunately, with 2 dimensions, it is not trivial to define a symmetric region about a given point of interest. Figure 5 shows several symmetric regions about various points, which are either centered upon a pixel or upon the gap between pixels. There are ways to get 4, 9, 12, 13, 16, 21, 24, and 25 in a symmetric fashion.

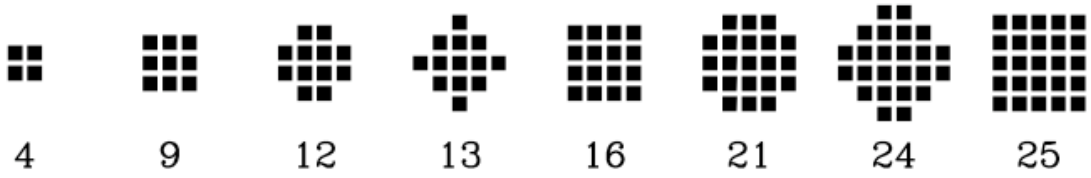



**Figure 5: Symmetric 2-D regions of pixels that could be fitted with 2-D polynomials.**

In our spline formalism, we would ideally like to map the number of parameters to the number of constraints exactly, so that we can ensure rigorously that the model will go through the grid points. As mentioned above, we managed to do this in the 1-D case with the 4 constraints (values and derivatives at surrounding node points) and 4 contributing pixels.

Note that in two dimensions, the only one of the above pixel distributions that has the same number of elements as one of the polynomial orders is the 21-pixel region, which matches the $5^{th}$-order polynomial. If we consider constraining this just along the x axis, then we have 5 pixels and 6 unknowns (A, B, D, H, L, and Q for 1, $x$, $x^2$, ... $x^5$). So, the two are not well-matched order-wise: we have no way to constrain all six coefficients. The non-commensurate nature of the 2-D case will force us to do things somewhat differently.

### 5.5 A bi-cubic spline

One approach is to do a bi-cubic spline. The bi-cubic spline will have the same benefits as the 1-D spline, namely that it will be the simplest function (in terms of polynomial order) that goes through all the grid-points while being continuous down to the first derivative. The bi-cubic spline will satisfy this in both dimensions.

The aim would be to come up with a function that represents the value of the grid within the square region between the 4 grid-points: **F**, **G**, **J**, and **K** in Figure 6. This involves solving for four horizontal cubic splines, one through each row, using the 1-D formalism above. These four splines are labeled SPLINE-$X_1$ through SPLINE-$X_4$ above. We then evaluate the splines at the $\Delta x$ location of the target point to get a series of four equally spaced values at $y = -1$, $y = 0$, $y = 1$, and $y = 2$ (blue squares). Next, a vertical spline is fit through these four values. Finally, we can evaluate this spline at the $y$ location of the target point of interest to get the value of the grid at location $(x,y) = (I_{\mathbf{F}}+\Delta x, J_{\mathbf{F}}+\Delta y)$. One could also do the same thing with $y$ first then $x$. It turns out that the results are identical.

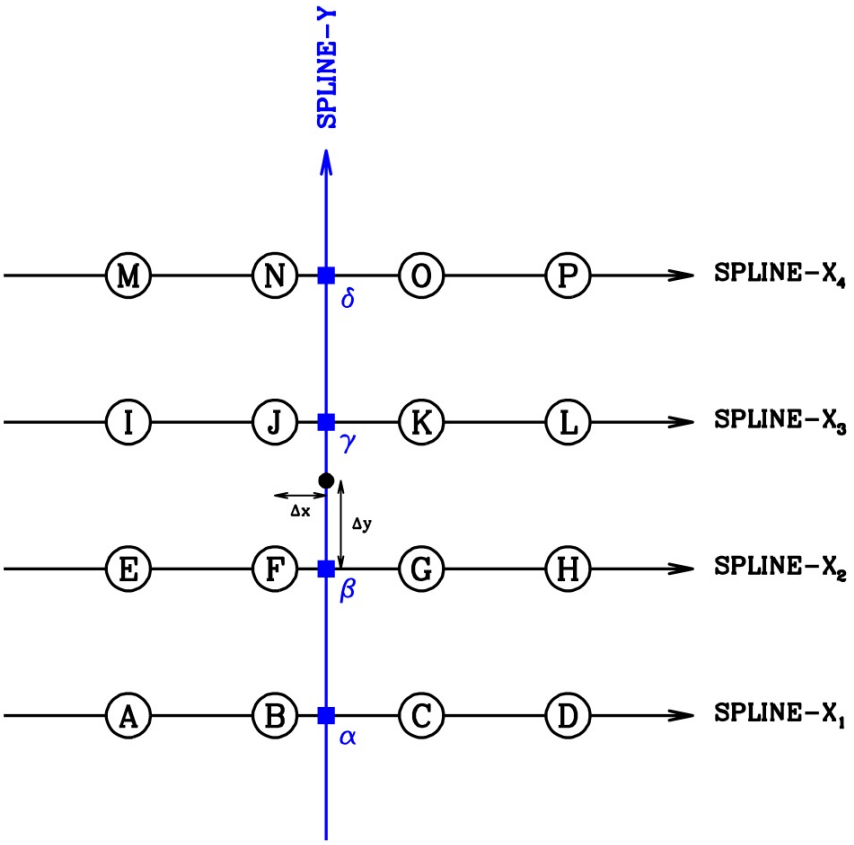



**Figure 6: The process of evaluating a bi-cubic spline for the point at the solid black circle. First, four 1-D splines in *x* are constructed in order to evaluate the function at the array of crosses, which are at the *x*-coordinate of the desired point. Then, a spline is fit to these four points (dashed line) in order to determine the value at the *y*-coordinate.**

The equations for the four splines along *x* can be written out, directly from Equation 8 above:

$$f_1(\Delta x) = \begin{pmatrix} 1 \\ \Delta x \\ \Delta x^2 \\ \Delta x^3 \end{pmatrix}^T \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 1 & -\frac{5}{2} & 2 & -\frac{1}{2} \\ -\frac{1}{2} & \frac{3}{2} & -\frac{3}{2} & \frac{1}{2} \end{bmatrix} \begin{pmatrix} A \\ B \\ C \\ D \end{pmatrix} \qquad f_3(\Delta x) = \begin{pmatrix} 1 \\ \Delta x \\ \Delta x^2 \\ \Delta x^3 \end{pmatrix}^T \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 1 & -\frac{5}{2} & 2 & -\frac{1}{2} \\ -\frac{1}{2} & \frac{3}{2} & -\frac{3}{2} & \frac{1}{2} \end{bmatrix} \begin{pmatrix} I \\ J \\ K \\ L \end{pmatrix} \quad . (10)$$

$$f_2(\Delta x) = \begin{pmatrix} 1 \\ \Delta x \\ \Delta x^2 \\ \Delta x^3 \end{pmatrix}^T \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 1 & -\frac{5}{2} & 2 & -\frac{1}{2} \\ -\frac{1}{2} & \frac{3}{2} & -\frac{3}{2} & \frac{1}{2} \end{bmatrix} \begin{pmatrix} E \\ F \\ G \\ H \end{pmatrix} \qquad f_4(\Delta x) = \begin{pmatrix} 1 \\ \Delta x \\ \Delta x^2 \\ \Delta x^3 \end{pmatrix}^T \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 1 & -\frac{5}{2} & 2 & -\frac{1}{2} \\ -\frac{1}{2} & \frac{3}{2} & -\frac{3}{2} & \frac{1}{2} \end{bmatrix} \begin{pmatrix} M \\ N \\ O \\ P \end{pmatrix}$$

The equation for the spline along *y* is then:

$$f(\Delta x,\Delta y)=\begin{pmatrix}1\\ \Delta y\\ \Delta y^2\\ \Delta y^3\end{pmatrix}^T\begin{bmatrix}0 & 1 & 0 & 0\\ -\frac{1}{2} & 0 & \frac{1}{2} & 0\\ 1 & -\frac{5}{2} & 2 & -\frac{1}{2}\\ -\frac{1}{2} & -\frac{3}{2} & -\frac{3}{2} & \frac{1}{2}\end{bmatrix}\begin{pmatrix}\alpha\\ \beta\\ \gamma\\ \delta\end{pmatrix}. \tag{11}$$

Plugging in for $\alpha=f_1(x)$, $\beta=f_2(x)$, $\gamma=f_3(x)$, and $\delta=f_4(x)$, we get:

$$f(x,y)=\begin{pmatrix}1\\ \Delta y\\ \Delta y^2\\ \Delta y^3\end{pmatrix}^T\begin{bmatrix}0 & 1 & 0 & 0\\ -\frac{1}{2} & 0 & \frac{1}{2} & 0\\ 1 & -\frac{5}{2} & 2 & -\frac{1}{2}\\ -\frac{1}{2} & \frac{3}{2} & -\frac{3}{2} & \frac{1}{2}\end{bmatrix}\begin{pmatrix}\left(\begin{pmatrix}1\\ \Delta x\\ \Delta x^2\\ \Delta x^3\end{pmatrix}^T\begin{bmatrix}0 & 1 & 0 & 0\\ -\frac{1}{2} & 0 & \frac{1}{2} & 0\\ 1 & -\frac{5}{2} & 2 & -\frac{1}{2}\\ -\frac{1}{2} & \frac{3}{2} & -\frac{3}{2} & \frac{1}{2}\end{bmatrix}\begin{pmatrix}A\\ B\\ C\\ D\end{pmatrix}\right) & \left(\begin{pmatrix}1\\ \Delta x\\ \Delta x^2\\ \Delta x^3\end{pmatrix}^T\begin{bmatrix}0 & 1 & 0 & 0\\ -\frac{1}{2} & 0 & \frac{1}{2} & 0\\ 1 & -\frac{5}{2} & 2 & -\frac{1}{2}\\ -\frac{1}{2} & \frac{3}{2} & -\frac{3}{2} & \frac{1}{2}\end{bmatrix}\begin{pmatrix}E\\ F\\ G\\ H\end{pmatrix}\right)\\ \left(\begin{pmatrix}1\\ \Delta x\\ \Delta x^2\\ \Delta x^3\end{pmatrix}^T\begin{bmatrix}0 & 1 & 0 & 0\\ -\frac{1}{2} & 0 & \frac{1}{2} & 0\\ 1 & -\frac{5}{2} & 2 & -\frac{1}{2}\\ -\frac{1}{2} & \frac{3}{2} & -\frac{3}{2} & \frac{1}{2}\end{bmatrix}\begin{pmatrix}I\\ J\\ K\\ L\end{pmatrix}\right) & \left(\begin{pmatrix}1\\ \Delta x\\ \Delta x^2\\ \Delta x^3\end{pmatrix}^T\begin{bmatrix}0 & 1 & 0 & 0\\ -\frac{1}{2} & 0 & \frac{1}{2} & 0\\ 1 & -\frac{5}{2} & 2 & -\frac{1}{2}\\ -\frac{1}{2} & \frac{3}{2} & -\frac{3}{2} & \frac{1}{2}\end{bmatrix}\begin{pmatrix}M\\ N\\ O\\ P\end{pmatrix}\right)\end{pmatrix}. \tag{12}$$

This can be solved for something of the form:

$$f(\Delta x,\Delta y)=\begin{pmatrix}1\\ \Delta y\\ \Delta y^2\\ \Delta y^3\\ \Delta x\\ \Delta x\Delta y\\ \Delta x\Delta y^2\\ \Delta x\Delta y^3\\ \Delta x^2\\ \Delta x^2\Delta y\\ \Delta x^2\Delta y^2\\ \Delta x^2\Delta y^3\\ \Delta x^3\\ \Delta x^3\Delta y\\ \Delta x^3\Delta y^2\\ \Delta x^3\Delta y^3\end{pmatrix}^T\begin{bmatrix}w_{1,1} & w_{2,1} & w_{3,1} & w_{4,1} & w_{5,1} & w_{6,1} & w_{7,1} & w_{8,1} & w_{9,1} & w_{10,1} & w_{11,1} & w_{12,1} & w_{13,1} & w_{14,1} & w_{15,1} & w_{16,1}\\ w_{1,2} & w_{2,2} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & w_{15,2} & w_{16,2}\\ w_{1,3} & \vdots & & & & & & & & & & & & & \vdots & w_{16,3}\\ w_{1,4} & \vdots & & & & & & & & & & & & & \vdots & w_{16,4}\\ w_{1,5} & \vdots & & & & & & & & & & & & & \vdots & w_{16,5}\\ w_{1,6} & \vdots & & & & & \ddots & & & & & & & & \vdots & w_{16,6}\\ w_{1,7} & \vdots & & & & & & & & & & & & & \vdots & w_{16,7}\\ w_{1,8} & \vdots & & & & & & & & & & & & & \vdots & w_{16,8}\\ w_{1,9} & \vdots & & & & & & & & & & & & & \vdots & w_{16,9}\\ w_{1,10} & \vdots & & & & & & & & & & & & & \vdots & w_{16,10}\\ w_{1,11} & \vdots & & & & & & & & & & & & & \vdots & w_{16,11}\\ w_{1,12} & \vdots & & & & & & & & & & & \ddots & & \vdots & w_{16,12}\\ w_{1,13} & \vdots & & & & & & & & & & & & & \vdots & w_{16,13}\\ w_{1,14} & \vdots & & & & & & & & & & & & & \vdots & w_{16,14}\\ w_{1,15} & w_{2,15} & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & w_{15,15} & w_{16,15}\\ w_{1,16} & w_{2,16} & w_{3,16} & w_{4,16} & w_{5,16} & w_{6,16} & w_{7,16} & w_{8,16} & w_{9,16} & w_{10,16} & w_{11,16} & w_{12,16} & w_{13,16} & w_{14,16} & w_{15,16} & w_{16,16}\end{bmatrix}\begin{pmatrix}A\\ B\\ C\\ D\\ E\\ F\\ G\\ H\\ I\\ J\\ K\\ L\\ M\\ N\\ O\\ P\end{pmatrix}. \tag{13}$$

This is the 2-D analog to equation (8). It tells us what the value of the bi-cubic-spline-interpolated grid is for any (*x*,*y*) within the central four grid-points (**F**, **G**, **J**, and **K**) in terms of all 16 relevant grid-points. The grid-points outside of this do not affect the value of the function within this inner square. If we write the 1-D spline evaluation matrix from equation (8) as **m**:

$$m = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -\frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 1 & -\frac{5}{2} & 2 & -\frac{1}{2} \\ -\frac{1}{2} & \frac{3}{2} & -\frac{3}{2} & \frac{1}{2} \end{pmatrix}, \tag{13}$$

then the elements of $\boldsymbol{w}$ can be seen to be:

$$w_{pq} = m_{PQ} \times m_{p'q'} \tag{14}$$

where $p'$ and $q'$ are the integer parts of $[1+(p\text{-}1)/4]$ and $[1+(q\text{-}1)/4]$, respectively, and $P$ and $Q$ are and $(p - 4{\cdot}p')$ and $(q - 4{\cdot}q')$, respectively. Written out fully, then:

$$\mathbf{w} = \frac{1}{4}\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 & -10 & 0 & 0 & 0 & 8 & 0 & 0 & 0 & -2 & 0 & 0 \\ 0 & -2 & 0 & 0 & 0 & 6 & 0 & 0 & 0 & -6 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & -2 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ -2 & 0 & 2 & 0 & 5 & 0 & -5 & 0 & -4 & 0 & 4 & 0 & 1 & 0 & -1 & 0 \\ 1 & 0 & -1 & 0 & -3 & 0 & 3 & 0 & 3 & 0 & -3 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 4 & -10 & 8 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -2 & 5 & -4 & 1 & 0 & 0 & 0 & 0 & 2 & -5 & 4 & -1 & 0 & 0 & 0 & 0 \\ 4 & -10 & 8 & -2 & -10 & 25 & -20 & 5 & 8 & -20 & 16 & -4 & -2 & 5 & -4 & 1 \\ -2 & 5 & -4 & 1 & 6 & -15 & 12 & -3 & -6 & 15 & -12 & 3 & 2 & -5 & 4 & -1 \\ 0 & 0 & 0 & 0 & -2 & 6 & -6 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -3 & 3 & -1 & 0 & 0 & 0 & 0 & -1 & 3 & -3 & 1 & 0 & 0 & 0 & 0 \\ -2 & 6 & -6 & 2 & 5 & -15 & 15 & -5 & -4 & 12 & -12 & 4 & 1 & -3 & 3 & -1 \\ 1 & -3 & 3 & -1 & -3 & 9 & -9 & 3 & 3 & -9 & 9 & -3 & -1 & 3 & -3 & 1 \end{pmatrix} \tag{15}$$

This is an extremely useful matrix. It means that we can accomplish our original objective by writing the function $f(x,y)$ as:

$$f(x,y) = \sum_{m=1}^{16} w_{mp} \left( \sum_{p=1}^{16} T_p(x - I_F, y - J_F) \right) G_m, \tag{16}$$

where the sum over $p$ represents the polynomial terms $T_p$, on the left of equation (13), and the sum over $m$ represents the grid-point values, $G_m$, with $G_1 = A$, $G_2 = B$, $G_3 = C$, ... and $G_{16} = P$. The 16 × 16 matrix represented by $w_{mp}$ is simply a set of numbers that will be fixed for all time. It relates the value of the grid-points to the value of the interpolated function in terms of the polynomial at the sample location. This compact expression provides the bi-cubic spline interpolation for the square region bounded by grid-points **F**, **G**, **J**, and **K**. In order to get the curvature of the interpolated function right within this region, the expression needs access to all 16 of the neighboring grid-points. It bears noting that $f(x,y)$ is *not* linear in $x$ and $y$, but it *is* linear in the $G_m$ grid-point values. This is what will make it possible to do least-squares optimization of them in the current situation where $x$ and $y$ are known and fixed for a given grid-construction.

Now that we have this matrix in hand, we can simply use it to evaluate how a grid would be interpolated for a particular sampling. This simple formalism has enormous benefits. We do not have to explicitly set-up and fit polynomials each time we want to interpolate the grid: the setting-up has already been done once and for all. We simply have to multiply a 16×16 matrix by two 16-element vectors. One of these vectors corresponds to the polynomial basis function, $T_p$, for the particular ($\Delta x$, $\Delta y$) offset, and the other corresponds to the 16 grid-point values.

### 5.6 Constraining the grid

The form of the above expression means that, as we had hoped, the interpolated grid value can be written down as a direct linear combination of the grid-point values. Thus if we have a fixed set of samples $P_1$ to $P_N$ of a 2-D function that correspond to a fixed number of points $(x_n,y_n)$ relative to an array of grid-points, then this formulation allows us to estimate the value of the samples in terms of the grid values:

$$\begin{aligned} P_1 &= f(x_1,y_1) = \sum_{m=1}^{12} \lambda_{1m} G_m \\ P_2 &= f(x_2,y_2) = \sum_{m=1}^{12} \lambda_{2m} G_m \\ &\cdots \\ P_n &= f(x_n,y_n) = \sum_{m=1}^{12} \lambda_{nm} G_m \end{aligned} \quad \rightarrow \quad \begin{pmatrix} P1 \\ P2 \\ \vdots \\ P_N \end{pmatrix} = \begin{bmatrix} \lambda_{11} & \lambda_{12} & \cdots & \lambda_{1M} \\ \lambda_{21} & \lambda_{22} & \cdots & \lambda_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{N1} & \lambda_{N2} & \cdots & \lambda_{MN} \end{bmatrix} \begin{pmatrix} G_1 \\ G_2 \\ \vdots \\ G_M \end{pmatrix} \qquad (17)$$

$$\text{where} \quad \lambda_{nm} = \sum_{p=1}^{16} T_n(x,y) w_{mp}$$

The $\lambda$ matrix must be constructed on-the-fly based on the specific location in (x,y) of each of the samples. It involves only matrix-vector multiplication, but no matrix inversion.

The system of equations in (17) can then be solved by least-squares, provided we have at least as many constraints as unknowns (N ≥ M). To get the least-squares result, we construct:

$$\chi^2 = \sum_{n=1}^{N} \left( P_n - \sum_{m=1}^{16} \lambda_{mn} G_m \right)^2 \qquad (18)$$

and set $\partial\chi^2/\partial G_m = 0$ for each grid-point $m$. This gives us a system of M = 16 equations:

$$\begin{pmatrix} \sum_n P_n \lambda_{n1} \\ \sum_n P_n \lambda_{n2} \\ \vdots \\ \sum_n P_n \lambda_{nM} \end{pmatrix} = \begin{bmatrix} \sum_i \sum_j \lambda_{i1}\lambda_{j1} & \sum_i \sum_j \lambda_{i1}\lambda_{j2} & \cdots & \sum_i \sum_j \lambda_{i1}\lambda_{jM} \\ \sum_i \sum_j \lambda_{i2}\lambda_{j1} & \sum_i \sum_j \lambda_{iM}\lambda_{j2} & & \sum_i \sum_j \lambda_{i2}\lambda_{jM} \\ \vdots & & \ddots & \vdots \\ \sum_i \sum_j \lambda_{iM}\lambda_{j1} & \sum_i \sum_j \lambda_{iM}\lambda_{j2} & \cdots & \sum_i \sum_j \lambda_{iM}\lambda_{jM} \end{bmatrix} \begin{pmatrix} G_1 \\ G_2 \\ \vdots \\ G_M \end{pmatrix} \quad (19)$$

$$= \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1M} \\ a_{21} & a_{22} & & a_{2M} \\ \vdots & & \ddots & \vdots \\ a_{M1} & a_{M2} & \cdots & a_{MM} \end{bmatrix} \begin{pmatrix} G_1 \\ G_2 \\ \vdots \\ G_M \end{pmatrix}$$

f there are more samples than grid-points (N>M), and the samples are not pathologically distributed, we can invert the symmetric matrix $a$ to get the $G_m$ values as a function of the sampling pixels $P_n$'s:

$$\begin{pmatrix} G_1 \\ G_2 \\ \vdots \\ G_M \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1M} \\ a_{21} & a_{22} & & a_{2M} \\ \vdots & & \ddots & \vdots \\ a_{M1} & a_{M2} & \cdots & a_{MM} \end{bmatrix}^{-1} \begin{pmatrix} \sum_n P_n w_{n1} \\ \sum_n P_n w_{n2} \\ \vdots \\ \sum_n P_n w_{nM} \end{pmatrix} \quad (20)$$

This gives us the least-squares solution for the grid-points $G_m$ given the pixels that sample the reference frame with their values $P_n$ at locations $(x_n,y_n)$. We note that the matrix $\boldsymbol{a}$ is a function of the positions $(x_n,y_n)$, so this matrix inversion *will* in general have to be done for the particulars of each particular scene we wish to recreate. To be specific, to find $\boldsymbol{a}$ we must multiply $w_{mn}$ by the $T_p$ polynomial to get the $\lambda_{mn}$ array. This will allow us to construct the $\boldsymbol{a}$ matrix, which is what we must invert to get the gridpoints $G_m$ from pixel-constraints $P_n$.

### 5.7 Covering larger areas

The above formalism is based on finding the grid-points that best represent the function $f(x,y)$ within the restricted square region between points **F**, **G**, **J**, and **K**. Thus far, we have no mechanism either for incorporating samples that come from outside of this region or for constraining the grid-points beyond this.

To extend this coverage, we can generalize the equations. Any sample within the grid will fall between four samples as above, and we can then express the interpolated function at that location in terms of the 16 surrounding grid-points. Each sample, then, will provide constraints on at most 16 different grid-points, and we can combine all the constraints into a single matrix inversion.

If we consider the samples that lie within the region of the grid-point space that extends from –3 to +3 in both $x$ and $y$, then in the above formalism, we will have to go out to –4 and +4 — a 9×9 array — in order to deal with all the grid points that are constrained. It

is awkward to constrain grid-points beyond the region of the samples, so within the border region, we can use bi-linear interpolation instead of our high-order polynomial interpolation. This will allow us to bound the reconstruction and incorporate samples that cover the entire region –4 to +4. Such an approach would have 9×9 = 81 grid points that need to be solved for, and would involve inverting an 81×81 matrix.

In principle we could reconstruct an entire image this way. For each pixel in the reconstructed image, we could find how its value contributes to estimating each of the pixels in the contributing exposures. We could then do a least-squares reconstruction all at once. Even for a "small" image that is just 1000×1000 pixels, this would involve inverting a one-million-element matrix, generally involving ~$10^{12}$ operations.

It would certainly be more efficient to do things in small patches at a time. Patches of 100×100 pixels would involve $10^8$ operations, which we would have to do 10×10 times to cover the original 1000×1000 pixels, so it would take only $10^{10}$ operations and would thus go 100 times faster than doing the entire scene at once. Clearly it is better to deal with small regions of the image at a time.

We can construct an equation that tells us how many operations will be required as a function of the patch size. For a given size $S \times S$ patch size, the inversion will take $S^4$ operations. We will not get good values for the entire grid, since the 2 grid-points in the outer border will not be fully constrained by the pixel samples; we may also need to do some experimentation to see how broad the border region really needs to be to have no impact on the reconstruction. At best, we will have adequate constraints only for the inner $(S-4)\times(S-4)$ points. We will have to do the operation $1000^2/S^2$ times to get well-constrained values for the entire 1000×1000 grid.

Inverting a general matrix of size S takes $S^4$ operations, and we will get from it $(S-4)^2$ constrained grid-points, so the number of operations per grid-point can be expressed as: $S^4/(S-4)^2$. This function is plotted in Figure 7 below. It turns out that the most efficient

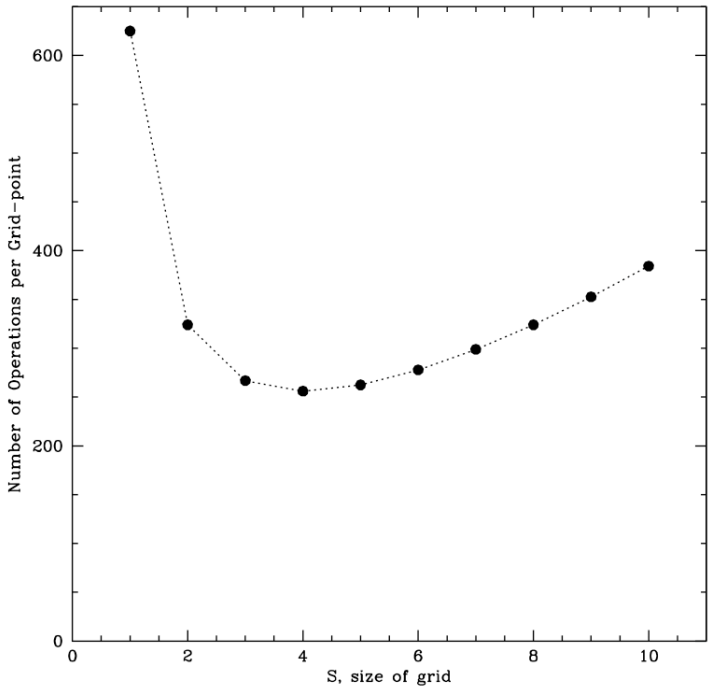



**Figure 7: The number of operations per grid-point solved for as a function of the number of grid-points solved for.**

way to construct a large grid is to use a 9×9 patch, which will give us 5×5 well-constrained grid-points at the center. The entire image can then be built up by constructing such a local patch for a tile of points across the image.

This report, and the companion report, will be more concerned with small-scale structure in reconstructed images, so we will not delve more into the build-up of larger reconstructions here. Our emphasis will be on how well a scene can be reconstructed over a very small region — a region small enough for point sources or marginally resolved sources. Thus, the 5×5 grid-point sweet-spot in the 9×9 grid will suffice.

It is worth noting that the above calculation did not take advantage of the fact that the matrix that must be solved for is sparse: each pixel constraint will affect at most 16 grid-points, and the correlations between gridpoints will be limited to nearby neighbors. It is likely that sparse-matrix algorithms could provide efficiencies that are much better than shown above, but that is beyond the scope of this document.

### 5.8 Additional benefits of the least-squares approach

The matrix formalism that we have developed here also allows us to solve for the covariance matrix, which will give us estimates of the errors in the solved-for grid-points (assuming we can estimate the errors in the samples) and will also give us the correlations between neighboring grid-points, so that we can assess how independent the values are that we find for the grid-points. If we find that there are large covariances between neighboring grid points, we might want to increase the spacing of the grid points. If this is discovered while evaluating the dither pattern, this information can help us to improve the dither pattern. Either way, knowledge of how the grid-point values are correlated will be good to have.

In ordinary linear least squares, the (symmetric) covariance matrix $Q$ is simply:

$$Q = \sigma^2 \begin{bmatrix} a_{11} & a_{12}' & \cdots & a_{1M} \\ a_{21} & a_{22} & & a_{2M} \\ \vdots & & \ddots & \vdots \\ a_{M1} & a_{M2} & \cdots & a_{MM} \end{bmatrix}^{-1} \tag{21}$$

where σ is the variance in the individual observations; it can be measured from the residuals, or estimated from a model for the S/N in each pixel[3]. The error in our value for grid-point *m*, should then be:

$$\sigma^2_{G_m} = \sigma^2 Q_{mm} \tag{22}$$

and its covariance with grid-point *m′* will be:

$$\sigma^2_{G_m G_{m'}} = \sigma^2 Q_{mm'} \tag{23}$$

The covariances can be particularly important when we want to assess whether we should believe gridpoint-to-gridpoint variations. The covariances can also be used when fitting models to the output grid.

It is interesting to note that the covariances flow directly from the overall error scaling and the location of the samples. The actual grid-point values and structure in the scene are not relevant to how the values of the grid-points may or may not be correlated. However, adopting a different scheme for weighting the pixels differently *will* affect the covariances.

### 5.9 Implicit assumptions

As was mentioned at the outset, the above procedure makes several assumptions. It assumes that all images represent a realization of the same scene, such that from image to image there is (1) no variation of the sky background, (2) no uncompensated-for variation of the exposure time, (3) no variation of the instrumental PSF, (4) no variation in the pixel shape, either due to dither-related distortion or due to different observations being at different roll angles, (5) no error in the distortion solution, which might cause pixels to be mapped to the wrong place in the reference frame, and finally (6) no intrinsic variation of the scene, such as variable stars or AGNs might generate.

Violation of any of these assumptions could compromise our ability to come up with a single "effective" scene. Some mitigation might be possible on several of these fronts, but we will not delve into that here.

[3] In this development, we have been assuming that all pixels have equal weights. This is not a necessary assumption and we could easily incorporate different weights for different pixels in Equation (18) and propagate that information into the other equations.

## 6 An Example

To show the power of this approach, we will assemble the F775W images from the UDF and will construct a least-squares composite image from them. There are 276 images with typical exposure times of 1200 s. We have chosen two targets, a bright star and a spiral galaxy.

Figure 8 shows the star and galaxy images from a stack generated with a `Drizzle`-like program with pixels that are 25 mas on a side (half the native ACS pixel scale). The star is located at (03:32:39.09, –27:46:01.8) and the galaxy at (03:32:39.09, –27:47:24.0). For each, the image on the left shows a wide 80×80-pixel region around the source. The middle panel shows a close up of the inner 19×19 pixels, and the right panel shows the same close up, but with the constituent samplings that come from the 276 contributing images. Each blue dot represents the master-frame location of one pixel in one of the contributing images. There is clearly a very dense sampling of the scene, and (by careful design) the dithers are evenly spread out.

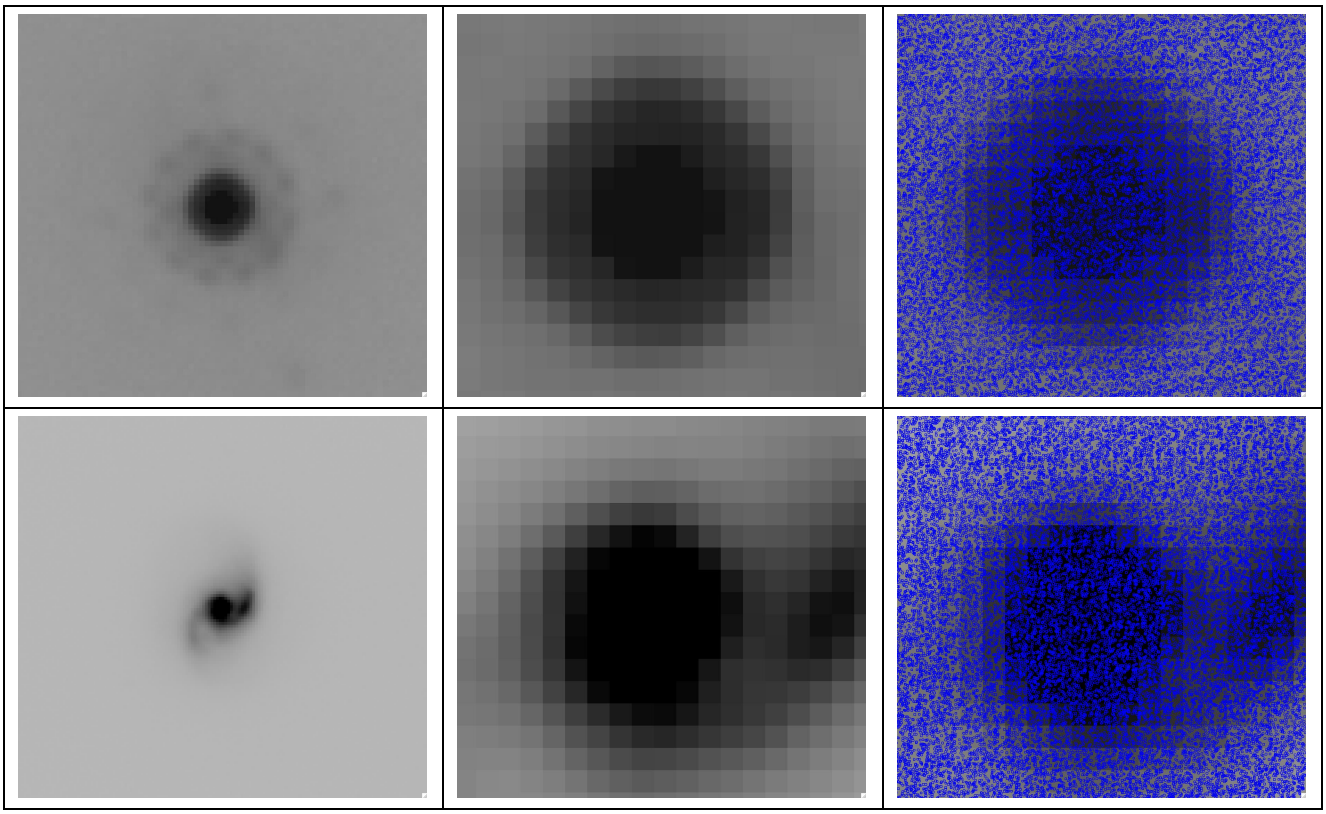


**Figure 8: We show stacks of the F775W UDF observations for a bright star (top) and a galaxy (bottom). The left panel shows a region that is about 2 arcseconds on a side. The middle panels show a region that is about 0.5 arcsecond on a side. The panels on the right add the samplings. The galaxy is shown with a linear scale, but because of its large dynamic range, the star was shown with a log intensity scale.**

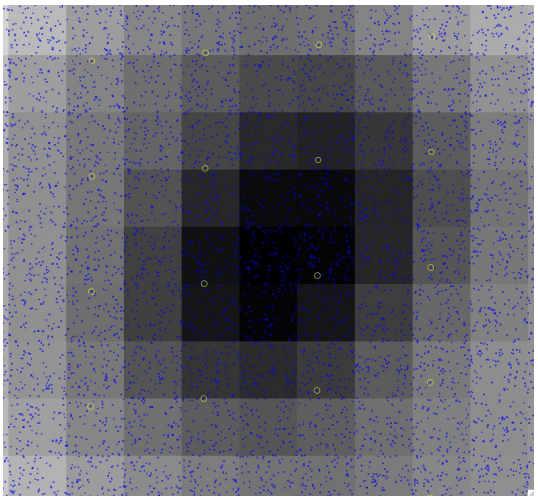


**Figure 9: This is a super close-up of the center of the galaxy, showing the locations of the many samplings of the scene available. The yellow circles are samplings from the first exposure.**

Figure 9 shows a close up of the central 9×9 super-sampled pixels of the galaxy. Again, the blue dots correspond to individual pixels from the 276 individual exposures in the data set. We clearly have many tens of samples in each super-pixel-sized region. We have highlighted in yellow the samples that come from the first exposure in the bunch. It is clear that they trace out a very regular pattern, with a spacing of 2 output-pixels (the super-sampling is a factor of 2). It is worth pointing out that the grid of pixels traced out by the first exposure is clearly not square: the x and y axes of WFC are skewed by almost 9 degrees and have different pixel scales, as well.

The next task will be to make use of these samplings and reconstruct the images using our new-found bi-cubic-spline-based algorithm. A least-squares grid solution was constructed for both objects. We solved for the grid in 9×9-pixel patches and spaced them every 5 pixels in $x$ and $y$ so that we could discard the outer 2 pixels of each patch, as the grid-points at the outside of the patch do not have samplings on all sides to constrain them. It remains to be seen whether this adequately solves for all 5×5 of the inner pixels, but we will assume it is adequate here.

The reconstructed image is shown in the top row of Figure 10. The samplings along two horizontal strips are also shown, one through the center of the galaxy/star and another that has been offset vertically. The samplings from the first image are shown in yellow to give a sense of how much information a single image provides. Since our horizontal strips are only ±0.5 pixels tall in the ×2 super-sampled frame, sometimes the samplings from image#1 straddle the strip, but have no samplings within the strip.

The top panel of plots shows the horizontal profile of the sources through their centermost pixels. The black dots correspond to the raw samplings of the "effective" image from the individual contributing pixels. The red dots are the samples that were discarded because they disagreed significantly with the model. The filled yellow dots

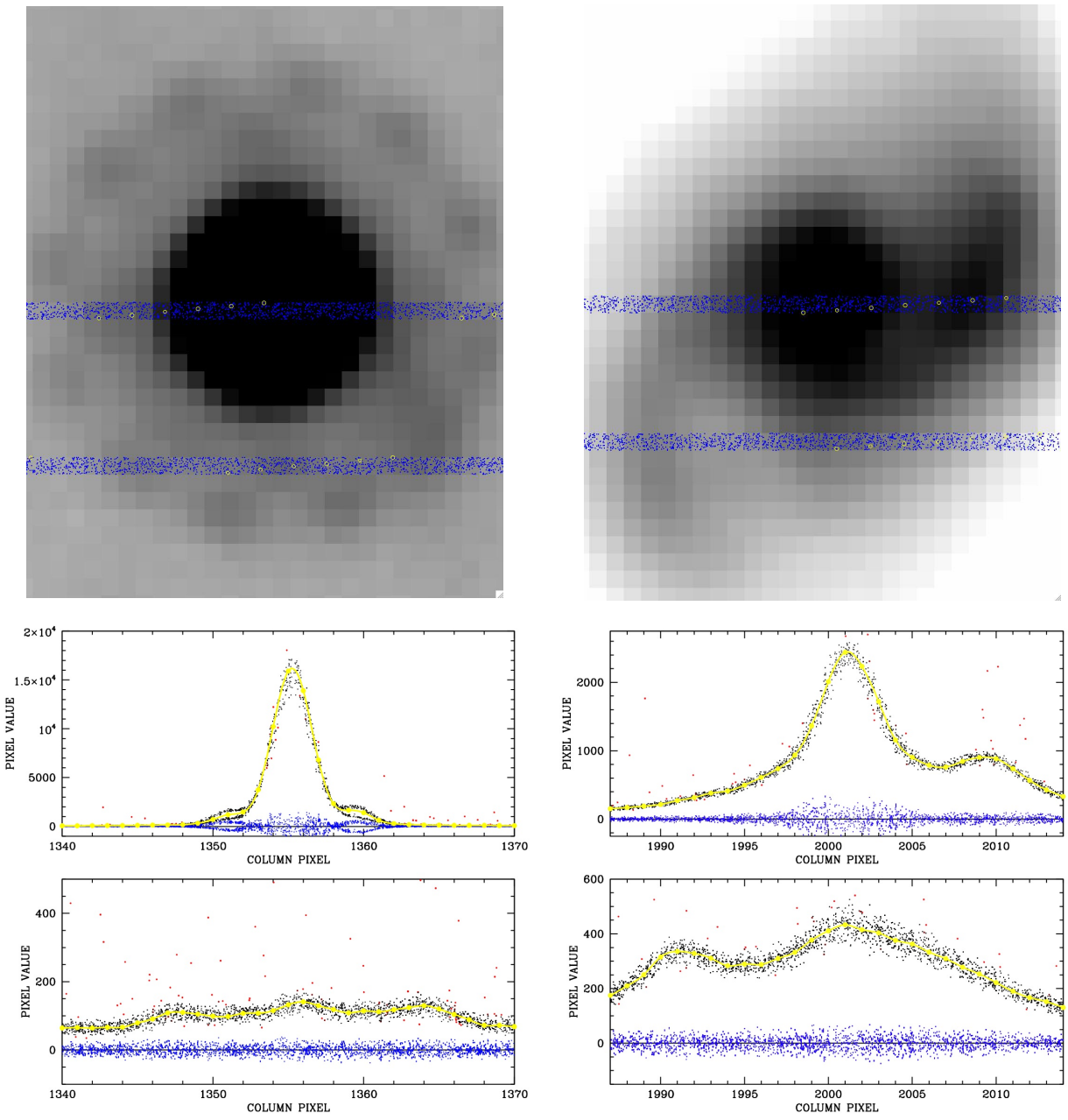



**Figure 10: Horizontal strips of samplings 0.5 super-sampled pixels tall across the galaxy and the star, one through the center and one offset. Image #1 is shown as yellow dots on the images. It is worth noting that the background can vary frame can vary from exposure to exposure; this variation has been removed by a constant offset for each exposure.**

show the grid-points of the model, and the yellow curve connects them with a 1-D spline. It is clear that the function goes nicely through the data. The difference between data and model is shown at the bottom as blue dots. There are no discernible trends remaining, either in the star image nor in the galaxy image.

The PSF appears to be well represented in the ×2-sampled "effective" scene. It is interesting to note that the raw samplings of the PSF exhibit a bifurcation around offsets of ±5 super pixels (2.5 raw WFC pixels). The reason for this is that the UDF was taken at two orientations differing by 90 degrees. The WFC PSF is not azimuthally symmetric; in particular, the inner halo can be extremely lop-sided[4]. The horizontal strip in the reference image goes from –x to +x in the first half of the UDF exposures, and from +y to –y in the other half. The asymmetry of the PSF then shows up as a bifurcation of the scene in the reference-frame profile. Note that while this is significant in the inner halo of the PSF, the core of the PSF is remarkably similar at both orientations. This difference would only be relevant for sources that are extremely sharp and bright. These features are 5 pixels away and down by a factor of 20 relative to the flux at the center of the star.

Note that the galaxy is very nicely fit by the smooth curve. The algorithm and the sampling easily resolve the central source, the bright arm to the right, and is able to track the smooth profile. It is clear that "effective scene" formalism is a powerful way to visualize data taken at different dithers.

The lower panels show the two scenes for a horizontal strip that is vertically offset downwards from the center. The scale of the plot has been expanded to show the structure of these lower pixels. Again, the model tracks the pixels extremely nicely. Even the PSF appears to be well modeled, even though half the samples come from orientations that differ by 90 degrees. Clearly the outer halo of the PSF at 10 pixels' radius has some 4-fold symmetry that the inner part at 5 pixels' radius does not.

The following array shows the covariance matrix for the central pixel of the star reconstruction:

```
 0.00000  -0.00001   0.00004  -0.00011   0.00046  -0.00011   0.00004  -0.00002   0.00001
-0.00001   0.00002  -0.00006   0.00018  -0.00076   0.00019  -0.00007   0.00003  -0.00002
 0.00005  -0.00008   0.00018  -0.00040   0.00193  -0.00039   0.00019  -0.00007   0.00006
-0.00012   0.00021  -0.00036   0.00089  -0.00430   0.00082  -0.00038   0.00018  -0.00014
 0.00043  -0.00077   0.00178  -0.00418   0.02355  -0.00381   0.00187  -0.00071   0.00044
-0.00011   0.00022  -0.00047   0.00061  -0.00453   0.00112  -0.00046   0.00022  -0.00015
 0.00004  -0.00008   0.00018  -0.00033   0.00185  -0.00042   0.00018  -0.00009   0.00006
-0.00002   0.00003  -0.00008   0.00013  -0.00089   0.00022  -0.00008   0.00003  -0.00002
 0.00001  -0.00002   0.00005  -0.00008   0.00057  -0.00012   0.00004  -0.00002   0.00001
```

The central value (0.02355) represents the variance. It is quite low and is consistent with there being about 70 samples within the domain of each grid-point[5]. The covariances of the central grid-point to the adjacent ones are also quite low (about 0.4%, roughly one fifth of the variance), indicating that there is very little correlation among the extracted grid-points.

[4] See Anderson & King (2006) for a discussion of how the ACS PSF varies with position.

[5] If this had been constructed by perfect interlacing, then all the samples for each pixel would be at the very center. If we were to combine them, we would expect the variance for the central pixel to be (1/70) $\sigma^2$, or 0.014 $\sigma^2$, and no co-variance.

It is worth noting that the covariance matrix is purely a function of the locations of the samplings and the weights that we assign to the pixels. The actual value of the grid-points in the scene do not enter into this at all. We assumed here that all the contributing pixels had the same error (scaled to $\sigma = 1$). If we had assumed that the brighter pixels had more Poisson error, then we would have put weights into the matrix constructions, and these values would change slightly, but the actual scene itself would still not directly affect the variances and covariances. In other words, in the absence of poisson noise, there should be as much covariance between pixels on a flat background as there is at the centers of stars.

## 7 Summary And Next Steps

This report has provided an overview of image-reconstruction techniques that stop short of deconvolution. It has shown that the multiple exposures in a dither pattern are all realizations of the same "effective" scene, and that it should be possible to reconstruct this fully-sampled parent image from its many children.

In this context, we have discussed `Drizzle` and Fourier-based approaches, and found that both were unable to construct regularly sampled scenes in real-world situations. In addition, neither could provide a direct estimate on how much neighboring pixels are correlated — an issue that will be important if we wish to evaluate dither patterns.

We then developed a new image-reconstruction algorithm, based on least-squares, that ensures rigorously regular sampling and also provides a mechanism for estimating the errors in each pixel of the output image, as well as covariances among the neighboring pixels. The machinery we have developed here will be used in a follow-on report on how to dither with NIRCam in a way to ensure adequate sampling of the effective scene. The subsequent report will explore two problems.

The first task will be to determine how much sampling is necessary in order to adequately represent all the structure in NIRCam images through various filters. We should expect that images through F200W, where the detector is essentially Nyquist sampled, will require less super-sampling than images through F070W, where the detector is woefully undersampled.

The second task involves evaluating the various dither patterns in order to determine how well each of them can be used to construct super-sampled images with various levels of over-sampling. The secondary dither patterns have been designed to provide optimal spacing of dithers from 1 to 64 points, but the net dither achieved for a given data set will be the product of the coherently-phased secondary dithers, and the un-phased primary dithers. It will be important to estimate whether a pattern that has 3 secondary dithers at 3 primary locations will achieve similar results to a campaign that has 9 coherently phased secondary dithers at only 1 primary location. In general, users will want to take as many primary dithers as possible, since large dithers mitigate L-flat errors and large artifacts better than small ones. So, it will be important to know how best to divide up the dithers among primary and secondary to achieve adequate sampling, while at the same time maximizing the number of better-mitigating primary dithers.

From a broader perspective, the techniques developed here could already be applied to existing data sets. Before this can be done, however, it will be necessary to evaluate the

assumptions that have been made, namely: (1) that the samplings have been accurately placed in the master frame, (2) that the PSF and the background do not vary from exposure to exposure, and (3) that all the data points are valid and have equal weights. Surely all of these will be violated at some level, and it will be worthwhile to explore how they impact the solution, or alternatively, how they might be mitigated. One can imagine iterating to improve the placement of the samplings or find a way to normalize the contributing images to all have the same PSF and background. As far as pixel weighting or rejection goes, this could also be done by iteration, starting with the anticipated S/N of each pixel. But these improvements are well beyond the scope of this document, where the aim is simply to set up a mechanism to explore how the placement of dithers may fundamentally impact our ability to recreate the scene.

## 8 References

Anderson, Jay & King, Ivan R. 2006 ACS/ISR 06-01. *PSFs, Photometry, and Astrometry for the ACS/WFC*

Anderson, Jay & King, Ivan R. 2000 PASP 112 1360. *Towards High-Precision Astrometry with WFPC2 I: Deriving an Effective Point-Spread Function*

Fruchter, Andrew S. & Hook, Richard N. 2002 PASP, 114, 144, *Drizzle: A Method for the Linear Reconstruction of Undersampled Images*

Fruchter, Andrew S. 2010, submitted to PASP.

Koekemoer, A. M., Fruchter, A. S. Hook, R. N., Hack, W. 2002, HST Calibration Workshop (eds. S. Arribas, A. M. Koekemoer, B. Whitmore), p 337

Lauer, Tod. 1999 PASP, 111, 227. *Combining Undersampled Dithered Images*