FOC f/48 Geometric Correction

M. Voit and W. Hack
October 29, 1996

ABSTRACT

This report summarizes our program to correct the geometric distortion of the f/48 imaging camera. We used several offset observations of 47 Tuc to correct the 256z x 1024 format, and this correction model was then used to determine the distortion of the 512z x 1024 format. The description of the geometric correction procedure presented here can serve as a primer for future determinations of FOC geometric distortion. Because the offsets used to correct the f/48 distortion were highly symmetric, spurious systematic 2-D modes crept into the correction algorithm when we used too fine a grid for the spline-model correction. With the newly corrected imaging format, we should be able to place targets directly into the f/48 slit in a single slew following the initial acquisition.

1. Introduction

Raw FOC images suffer from geometric distortions arising from both the off-axis optical system and the image-intensifying detector. Detector distortion dominates. Until recently, the FOC distortion corrections had been based on squaring the reseau grids observed in internal flat-field images. This type of correction fails to correct for distortion modes with higher spatial frequencies than the reseau grid. A superior distortion correction can be derived from overlapping observations of a crowded star field. Since 1995, the *f*/96 geometric correction has been based on a scheme of this kind (see ISRs FOC-086,087,088). This report describes a new geometric correction for the *f*/48 imaging mode derived in a similar way.

The report proceeds as follows. Section 2 lists the observations used. Section 3 describes the correction of the 256z x 1024 format. Section 4 tells how this correction was used to correct the 512z x 1024 format. Section 5 warns of the pitfalls of symmetric offset patterns. Section 6 describes how the geometric correction procedure aligns *f*/48 images. Section 7 outlines how the correction models can be used to improve the acquisition of spectroscopic targets. The primary purpose of these models is to improve our ability to place acquired targets in the long slit of the *f*/48 spectrograph. Because the full-format dis-

tortion model is unconstrained near the edges, it will not be used to generate a geometric correction file.

2. Calibration Observations

On 4 April 1996, the *f*/48 camera of the FOC gathered seven exposures of the crowded star field at the center of the globular cluster 47 Tuc (RA = 6.0251 deg, DEC = -72.0813 deg). Six of the exposures, covering five different overlapping pointings, used the 256z x 1024 format and the other exposure used the 512z x 1024 format. These seven exposures, all with the camera *y*-axis oriented 29 degrees east of north (ORIENTAT = 28.886), were taken through the F220W filter. Table 1 provides more details on the exposure times and offset pointings. Note that the exposure time at one of the pointings is split over two exposures (x34i0104t and x34i0105t). These were coadded and treated as a single image.

Table 1. Exposures & Pointings for f/48 Geometric Correction

Exposure	Format	Exp. Time (sec)	ΔRA (arcsec)	ΔDEC (arcsec)
x34i0101t	256z x 1024	597.25	0	0
x34i0102t	256z x 1024	597.25	- 0.3170	- 7.8198
x34i0103t	256z x 1024	597.25	- 6.4460	- 4.4380
x34i0104t	256z x 1024	318.25	6.4460	4.4386
x34i0105t	256z x 1024	276.25	6.4460	4.4386
x34i0106t	256z x 1024	597.12	0.3170	7.8198
x34i0101t	512z x 1024	597.12	0	0

3. Geometric Correction Procedures

The general scheme for correcting the geometric distortion of the *f*/48 imaging mode follows the method developed by P. Greenfield for the f/96 correction (ISRs FOC-086,087,088):

1. Determine offsets in camera *x*-*y* coordinate system
2. Measure star positions in the FOC images
3. Import star positions into IDL
4. Manually pair positions of stars that appear in multiple exposures
5. Prepare manually generated lists for automated pairing
6. Pair additional stars automatically

7. Select panels for spline fit to geometric distortion
8. Fit geometric correction model
9. Check residuals
10. Check geometric distortion model

The rest of this section describes each of these operations in detail.

Determining offsets in camera x-y coordinate system

To determine the image offsets in the camera *x-y* coordinate system, we assume that ORIENTAT angle in the image header correctly gives the angle of the *y*-axis east of north. In this set of exposures ORIENTAT ranges between 28.881 and 28.892 degrees. Table 2 gives the offsets determined from this orientation angle, assuming a plate scale of 0.02870 arcsec/pixel.

Table 2. Offsets in Camera Coordinates

Exposure	Δx (arcsec)	Δy (arcsec)	Δx (pixels)	Δy (pixels)	Suffix
x34i0101t	0	0	0	0	`cc`
x34i0102t	3.4993	-7.0000	122	-244	`se`
x34i0103t	-3.4995	-7.0000	-122	-244	`sw`
x34i0104t	3.5005	6.9999	122	244	`ne`
x34i0105t	3.5005	6.9999	122	244	`ne`
x34i0106t	-3.5000	7.0000	-122	244	`nw`
x34i0101t	0	0	0	0	`ff`

Measuring star positions in the FOC images

To measure the star centroids in each image, we use the `daofind` routine in IRAF. In zoomed images, it helps greatly to adjust the default parameters so that `sharplo = 0, sharphi = 9, roundlo = -10, roundhi = 10`. Otherwise, `daofind` misses many of the stars. This task also asks for the FWHM of the stellar images (1.5 pixels), the standard deviation of the background (2.5 counts), and the desired threshold (7 σ). When given the parameters in parentheses, `daofind` found between 1609 and 1809 star positions in each of the 256z x 1024 inages. This routine stored the star positions in data files of a type we will refer to generically as `posfile_dd`, where `dd` serves as a placeholder for the image suffixes `ne`, `nw`, `se`, `sw`, or `cc`. Following the convention established in ISR FOC-088, the first four suffixes refer to the approximate offset direction, and the last refers to the central image (i.e. zero offset).

Importing star positions into IDL

For the f/96 geometric correction, Perry Greenfield developed a number of useful IDL routines, described more fully in ISR FOC-088. To take advantage of these routines, we need to load the star positions from `daofind` into IDL using the following commands:

```
readcol,'posfile_dd',format='(f,f)',xdd,ydd
pos_dd = [[xdd],[ydd]]
```

The five `pos_dd` arrays contain the (*x*,y) positions for each identified star in each image.

Manual pairing of star positions

The next step in the geometric correction process is currently very labor-intensive. We need to look at each pair of images and identify instances where the same star appears in both. The end result for each image pair is an array called `pospair_dd_dd`, which contains [(x_1,y_1),(x_2,y_2)] position pairs for each star manually identified in both images.

Not all stars appearing in both images need to be identified by hand. This step merely prepares for the crude geometric correction perfomed before the automated pairing of star positions that comes next. In general, one wants to sample the edges of the frames as thoroughly as possible, because the outermost selected stars define the region over which stars will be paired automatically. In the 256z x 1024 correction, 100-200 stellar images were paired manually in each image pair.

The IDL routine `pairstars` used to do the manual pairing requires byte-scaled images as input. To prepare the input images, we follow these steps for each image:

```
readst,'exposure_dd.hhh',imdd
bimdd = bytscl(imdd)
```

Once the byte-scaled images and positions are ready, we run the routine `pairstars` for each image pair. The arguments supplied to `pairstars` are the two byte-scaled images, the star positions in the first image, and the relative offset of the two images. Later steps in the correction algorithm work most smoothly if the image pairs are ordered with the greatest y-offset values first. In this case, the image ordering was `ne`, `nw`, `cc`, `se`, `sw`. The sequence of pairings for the 256z x 1024 format ran as follows:

```
pairstars,bimne,bimnw,pos_ne,[122,0],pospair_ne_nw
pairstars,bimne,bimcc,pos_ne,[61,-244],pospair_ne_cc
pairstars,bimne,bimse,pos_ne,[0,-488],pospair_ne_se
pairstars,bimne,bimsw,pos_ne,[122,-488],pospair_ne_sw
pairstars,bimnw,bimcc,pos_ne,[-61,-244],pospair_nw_cc
pairstars,bimnw,bimse,pos_ne,[-122,-488],pospair_nw_se
```

```
pairstars,bimnw,bimsw,pos_ne,[0,-488],pospair_nw_sw
pairstars,bimcc,bimse,pos_ne,[-61,-244],pospair_cc_se
pairstars,bimcc,bimsw,pos_ne,[61,-244],pospair_cc_sw
pairstars,bimse,bimsw,pos_ne,[122,0],pospair_se_sw
```

Note that the relative x-offsets, in zoomed pixels, are half the size of the normal-pixel offsets in Table 2.

Each execution of `pairstars` displays a portion of the first image at the top of the screen and an appropriately registered piece of the second image below it. A click with the left mouse button identifies a star in the top panel, and a subsequent click on the same star in the lower panel pairs the *x-y* coordinates of this star in each image. The middle mouse button shifts the images. One click of the middle mouse button in the top panel activates the shifting routine. The second click selects a point in the first image. The third click chooses the display location to which the selected point will be moved. The lower image then shifts accordingly. Beware the right mouse button, which terminates the program.[1] Press this button only when you are sure you are finished, and the program will return a `pospair_dd_dd` array for the image pair. More details on this procedure are available in ISR FOC-088.

Prepare manually generated pairings for automated pairing

Prior to running `multistar`, the routine that pairs star positions automatically, several preparatory steps are necessary. First, a `pairmask` array must be generated to tell `multistar` which image pairs have `pospair_dd_dd` arrays. The indices of `pairmask` follow the image priority sequence established in the `pairstars` step. In the 256z x 1024 correction, we set

```
pairmask = [ [0,1,1,1,1], $
             [0,0,1,1,1], $
             [0,0,0,1,1], $
             [0,0,0,0,1], $
             [0,0,0,0,0] ]
```

The star positions and position pairs must be grouped in the same order. In this case, we set

```
pos_group = group(pos_ne, pos_nw, pos_cc, pos_se, pos_sw)
```

1. The `pairstars` routine is not particularly user-friendly. If you click the right mouse button prematurely, you must begin the pairing process over again from scratch. The `pospair_dd_dd` arrays must be created whole. If you mispair a star between the two images, your mistake cannot be excised automatically. Competence in IDL is required to remove offending entries after the pairing process is complete.

```
pospair_group = group( pospair_ne_nw, pospair_ne_cc, $
                       pospair_ne_se, pospair_ne_sw, $
                       pospair_nw_cc, pospair_nw_se, $
                       pospair_nw_sw, pospair_cc_se, $
                       pospair_cc_sw, pospair_se_sw )
```

Because the `multistar` routine needs to know the offset of each of the images, we set

```
offsets = [[-61,61,0,-61,61],[244,244,0,-244,-244]]
```

The final step is to check a particular line in the file `pmatch.pro` that looks like this:

```
t2(*,0)=2       ;COMMENT OUT IF USING NORMAL PIXELS!
```

This line was left uncommented because the 256z x 1024 correction works with zoomed pixels.

Pair additional stars automatically

After these preparatory steps, we run `multistar` by executing the following command:

```
multistar,pos_group,pospair_group,offsets,pos,q,o
```

The automatic pairing procedure found between 316 and 779 star-position pairs in each image pair, depending on the amount of image overlap. The output array `pos` holds the (x,y) coordinate pairs for each multiply observed star, the `q` array contains the star id number of each star in `pos`, and the `o` array contains image offsets corresponding to each coordinate pair in `pos`.

Select panels for spline fit to geometric distortion

After `multistar` has run, we are ready to fit a geometric correction model. This spline model is defined on a user-specified grid created by the function `pickpanel`. To define the grid, type

```
splinemodel = pickpanel(bimcc)
```

Any of the byte-scaled images works equally well as an argument. Follow the on-screen directions to select *x*-knots and *y*-knots. Sparse grids are usually best, for reasons we will

discuss in Section 5. For the 256z x 1024 correction, we ended up using a grid with 3 panels in the *x* direction and 4 panels in the *y* direction.

Fit a geometric correction model

The routine `twodfit` computes geometric correction models. It takes the input arrays `pos`, `o`, and `q`, and returns its best fit of the true star positions in the array `uvsol`, resulting from the geometric correction contained in the returned `splinemodel`. The array `resid` contains the (Δx,Δy) residuals corresponding to each star position in `uvsol`. To run `twodfit`, type

```
twodfit,pos,o,q,splinemodel,uvsol,resid
```

This routine prints the radial rms residuals to the screen, but when the *x*-pixels are zoomed, this number can be deceiving. To find the rms *x* and *y* residuals, respectively, in dezoomed pixels, type

```
print,stdev(resid(*,0)*2.0)
print,stdev(resid(*,1))
```

The rms residuals for the 256z x 1024 correction were 0.436 dezoomed pixels in *x* and 0.329 dezoomed pixels in *y*, giving a radial rms residual of 0.546 dezoomed pixels if the *x* and *y* residuals are uncorrelated.

Check residuals

To check visually for pathological systematic behavior in the residuals, one can generate postscript plots of the residuals using the routine `psplotresid` as follows:

```
psplotresid,pos,resid,splinemodel,10.0,filename='res.ps'
```

The numerical value, 10.0 in this case, specifies the scale of the tadpoles in the residual plot. Figure 1 shows the residuals in the 256z x 1024 correction.

Check geometric distortion model

Even when the residuals are small, the geometric distortion model can still be systematically wrong (see Section 5). To visualize the spline model itself, we use the routine `splinetest`. This routine creates a grid in the uncorrected (*x*,*y*) coordinate system with lines every 10 zoomed *x* pixels and 10 normal *y* pixels. The grid is then dezoomed, geometrically corrected, and displayed. To run `splinetest`, type

```
splinetest,splinemodel,0.0,250.0,0.0,1020.0
```

The last four arguments to `splinetest` specify the *x* and *y* limits of the uncorrected grid. If the spline model introduces an arbitrary offset, simple editing of the file `splinetest.pro` can properly register the corrected grid. Figure 2 shows the spline model adopted for the 256z x 1024 correction. Note that the edges are not as well constrained as the center.

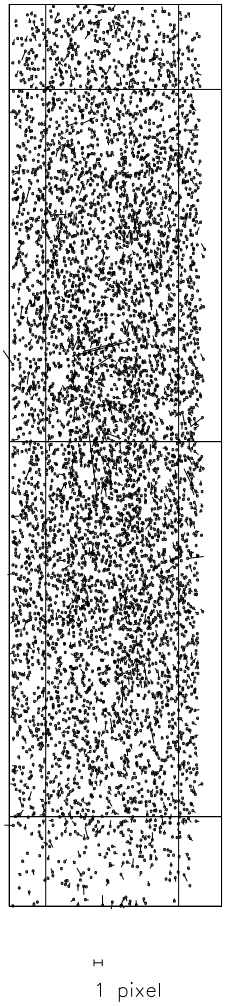

Figure 1: Residuals in the 256z x 1024 geometric correction. The grid shows the positions of the knots in the spline model.

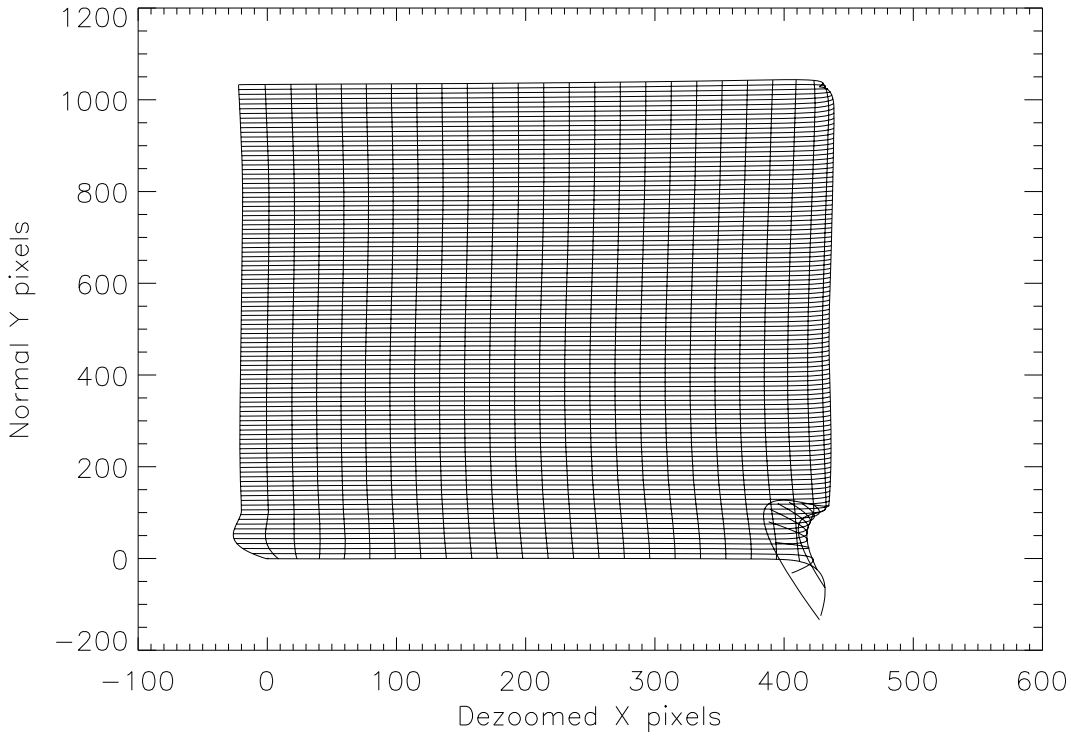

Figure 2: Spline model used in the 256z x 1024 geometric correction. The lines show how a grid of 10 pixel by 10 pixel squares in zoomed, uncorrected coordinates maps into dezoomed, corrected coordinates.

4. Correcting Other Formats

Geometrically corrected FOC formats can be used to correct other FOC formats. The array `uvsol` produced by `twodfit` holds the true positions of the stellar images used in the correction. These can serve as fiducial positions to correct images of the same starfield, taken with a different format.

The procedure for correcting additional formats is similar to, but simpler than, the one we followed to obtain the original correction. As before, we use `daofind` to measure star positions, `readcol` to bring the positions into IDL, `readst` to read the corresponding image into IDL, and `bytscl` to produce a byte-scaled version of that image:

```
readcol,'x34i0107t_d0f.coo.1',format='(f,f)',xff,yff
pos_ff = [[xff],[yff]]
readst,'x34i0107t.hhh',imff
bimff = bytscl(imff)
```

In general, the star positions in uvsol will not be aligned properly with the new image. Aligning the uvsol positions can be done quickly and dirtily by guessing the correct offset or correctly and tediously by identifying the position of a particular star in both coordinate systems.

To follow the tedious method, you must first identify a star that appears in both the image to be corrected and the pos list from the previous correction. Then, find the precise coordinates of this star in the image to be corrected and in one of the uncorrected images from the previously corrected format. For example, the star at (248.796,638.315) in image x34i0107t is identical to the star at (121.014,637.915) in image x34i0101t. In the present correction program, this star was number 1040 in the pos_cc array. Typing

```
print,where(pos(*,0) eq 121.014)
```

provides the index of this star in the pos array. In this case its index was 2347. To find this star's ID number in the q array, type

```
print,q(2347)
```

Here, the star's ID number turned out to be 214. Typing

```
print,uvsol(214,*)
```

then gave the corrected x-y coordinates (46.908,390.265) of this star in uvsol. The offset between uvsol and pos_ff was therefore (248.796,638.315) - (46.908,390.265) = (201.888,233.774). If we had preferred convoluted coding to careful bookkeeping, we could have typed

```
print,uvsol(q(where(pos(*,0) eq 121.014)),*)
```

to get the same result. No matter how the offset has been determined, a statement of the form

```
uvreg = [[uvsol(*,0)+201.888],[uvsol(*,1)+233.774]]
```

creates an array uvreg of corrected star positions that is properly registered with the uncorrected pos_ff positions.

Once the two sets of positions are aligned, we must again pair stars manually to prepare for the subsequent automated pairing. The command

```
pairuv,bimff,uvreg,uvreg,pospair_ff
```

initiates a routine that operates much like the `pairstars` routine described earlier. Once the manual pairing is complete, we type

```
pos_group = group(pos_ff)
pospair_group = group(pospair_ff)
uvoffsets = [[0],[0]]
offsets = uvoffsets
multiuvstar,pos_group,pospair_group,uvreg,uvreg, $
     uvoffsets,posset,q,o
```

This sequence prepares and executes `multiuvstar`, a companion program to `multistar`. The output arrays `posset`, `q`, and `o` correspond to the output arrays `pos`, `q`, and `o` from `multistar`. In the particular case described here, only one 512z x 1024 image has been used in the correction procedure. If there had been more images to include, the preparations would have been more complex. See ISR FOC-088 for details.

To complete the geometric correction procedure, we define another spline-model grid and run the routine `uvfit` as follows:

```
splinemodel = pickpanel(bimff)
uvfit,posset,splinemodel,uvresid
```

In the present case, the star positions in `uvreg` did not entirely cover the 512z x 1024 format. We therefore had to be careful to select a grid that had stars in every panel, so that the spline fits would be constrained. To keep the corners well behaved, we also added dummy position pairs to `posset` at the points (10,10), (10,1000), (500,10), and (500,1000). The rms residuals in the best fit were 0.384 dezoomed pixels in *x* and 0.361 normal pixels in *y* for a radial rms residual of 0.527 normal pixels, if the *x* and *y* residuals are uncorrelated.

The `psplotresid` and `splinetest` routines can be used to check the model. To produce illustrations of the residuals and the splinemodel, type

```
psplotresid,posset_reg(*,*,0),uvresid,splinemodel,10.0, $
       filename='uvresid.ps'
splinetest,splinemodel,0.0,510.0,0.0,1020.0
```

Figure 3 shows the residuals left by the 512z x 1024 geometric correction model, and Figure 4 shows the model itself.

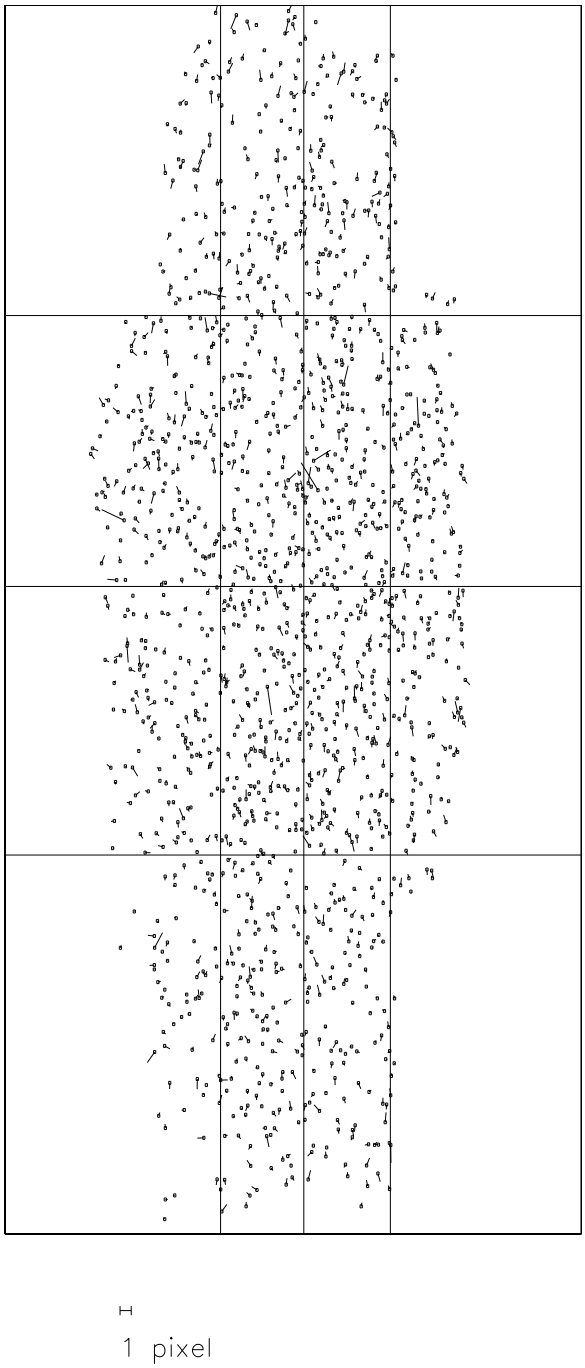

Figure 3: Residuals in the fit of the 512z x 1024 geometric distortion model. The fit is reliable only in the central region where the star positions are well sampled. For a well behaved spline fit, each panel needs to contain star positions.

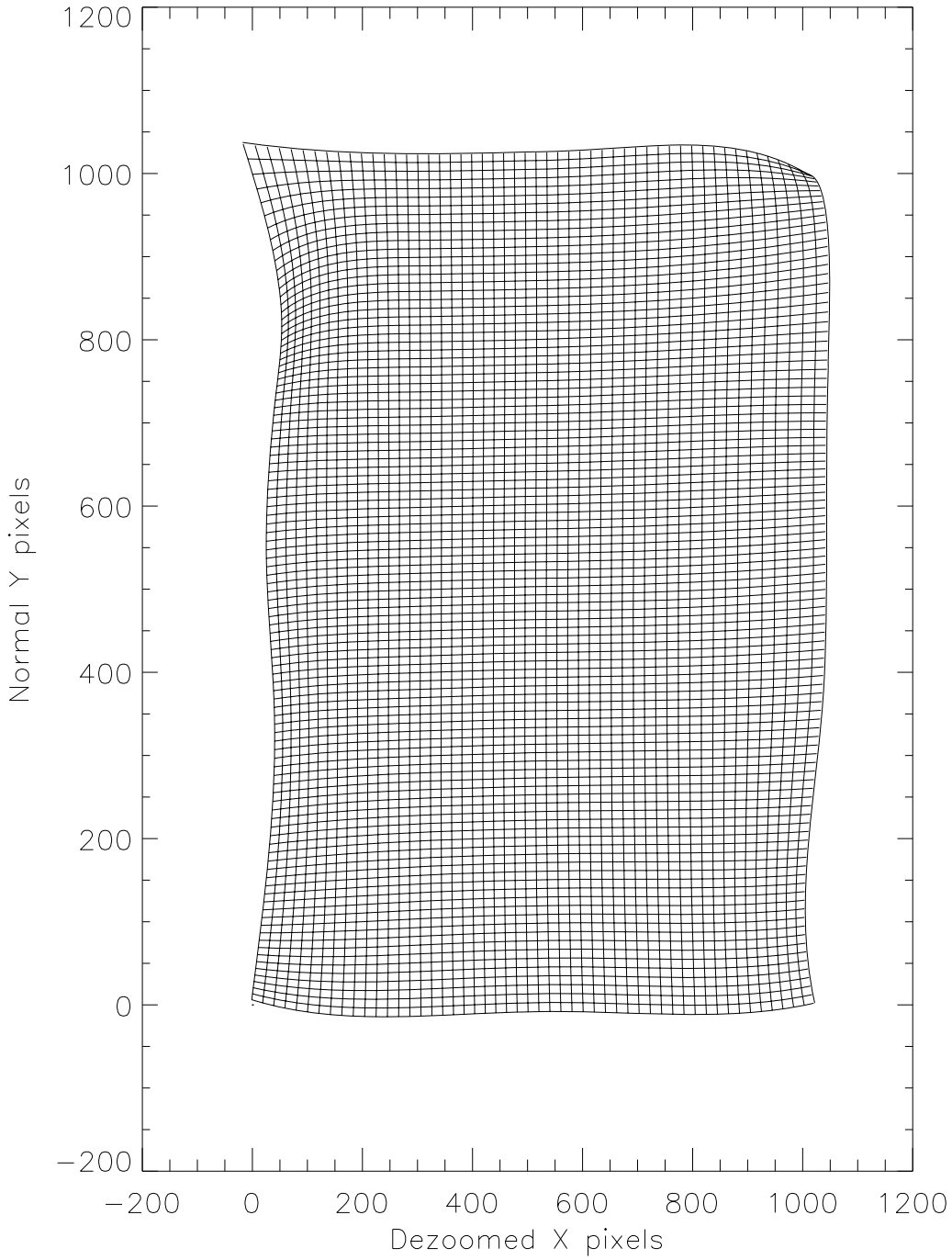

Figure 4: Spline model used in the 512z x 1024 geometric correction. The lines show how a grid of 10 pixel by 10 pixel squares in zoomed, uncorrected coordinates maps into dezoomed, corrected coordinates. Near the edges the fit is poorly constrained and cannot be trusted.

5. Problems with Symmetric Offsets

The regularity of the offset pattern in the 256z x 1024 correction led to some systematic problems in determining a valid distortion model. When the offset pattern is symmetric, the geometric correction algorithm can generate spurious distortion patterns that have the same symmetry as the offset pattern. An arbitrary periodic pattern that looks exactly the same in each region of image overlap passes through the algorithm undetected. Figure 5 shows an extreme example of the kind of faulty distortion correction that can result (see Figure 2 for the correct solution). A spurious 2-D wave with a period of 244 dezoomed pixels in the *x*-direction and 488 normal pixels in the *y*-direction dominates the distortion model.

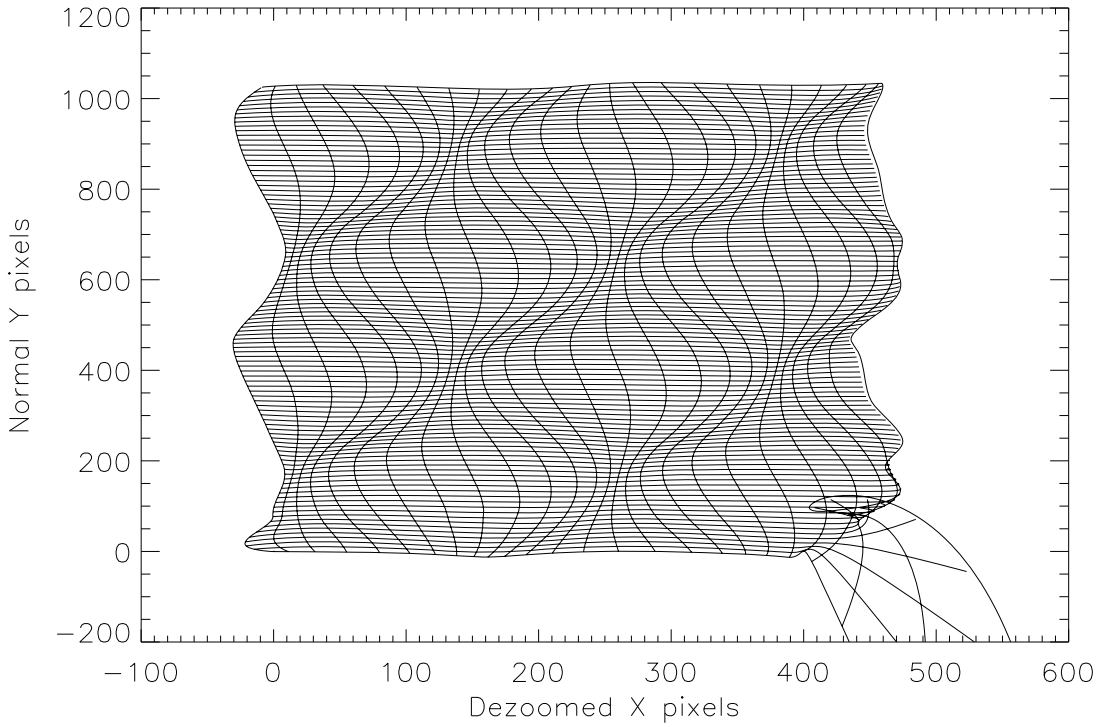

Figure 5: Pathological spline model for the 256z x 1024 distortion, computed over a dense grid with 6 panels in the *x*-direction and 11 panels in the *y*-direction.

To minimize the phony 2-D waves in the geometric correction, we needed to use a spline grid so sparse that the major panels were larger than the periodicity of the offset (see Figure 1). Then polynomials of low order have too low a spatial frequency to reproduce the bogus waves, damping them out of the fit. In the future, geometric correction programs relying on offset pointings should use an asymmetric offset pattern. An irregular pattern will not admit these spurious waves, allowing for more flexibility in the choice of the spline grid.

6. Image Alignment & Scale

The geometric correction procedure undertaken here defines the corrected *y*-axis of the FOC *f*/48 camera. The corrected *y*-axis is not an intrinsic property of the camera. Rather, it is a freely definable direction, lying at some fixed angle relative to the spacecraft V3 axis. Information about the *y*-axis enters the correction procedure when we use the ORIENTAT angle to convert RA and DEC offsets into *x* and *y* offsets (see Section 3). The ORIENTAT angle is the sum of the angle from north to V3, determined by the telescope pointing, and the angle from V3 to the *y*-axis, taken from the project database. Although the orientation angle $\theta(f/48)$ from the *y*-axis to V3 is arbitrarily definable, the most useful definition, currently $\theta(f/48) = 20.6$ degrees (ISR FOC-052), places the *y*-axis in the direction along which the *y*-values of uncorrected pixels increase most rapidly.

Images corrected via the scheme outlined here are automatically remapped into the *x*-*y* coordinate system defined by $\theta(f/48)$. At the beginning of the correction procedure, the algorithm ingests the image offsets in user-specified *x*-*y* coordinates. These offsets provide the only information the algorithm has about coordinate systems of any kind. In the end, the *y*-axis of the corrected image is identical to the *y*-axis of the coordinate system used to specify the offsets. Because the *x*-*y* offsets were derived from ORIENTAT, the *y*-axes of all images corrected in this way will lie at an angle $\theta(f/48)$ away from V3.

The ORIENTAT angle given in each image header will properly specify the angle from north to the corrected *y*-axis, as long as $\theta(f/48)$ remains unchanged in the project database and the geometric correction remains stable. The present correction will not properly account for time-dependent rotations added by the imaging system. We have compared the corrected *f*/48 images with corrected *f*/96 images of the same field and find no systematic rotation to a precision of better than 0.2 degrees.

The geometric correction procedure also sets the plate scale and absolute offset of the corrected images. When we convert the original offsets from arcseconds to pixels (Table 2), we need to assume a plate scale, in this case 0.02870 arsec/pixel. This assumption fixes the scale of the corrected images, so we need to check that it is correct.. The *f*/96 scale has been calibrated with astrometric fields and provides a suitable standard.Comparisions of corrected *f*/48 images with corrected *f*/96 images of the same field show that the assumed plate scale is indeed correct to about 1%. The registration procedure described in Section 4 aligns the corrected image to the uncorrected image at the position of the star used to register the `uvreg` array. In this case, the two images have nearly zero offset at the position (248,638).

7. Slewing to the Slit

Future target acquisitions for the *f*/48 spectrograph will take advantage of this new geometric correction. We should now be able to slew accurately from one position to

another within the *f*/48 field of view. However, if we want to move a target into the slit, we need to know where the slit is located. Our best information about the slit location currently comes from previous *f*/48 long-slit observations. In particular, there exists an observation of NGC 4151 in which the active galactic nucleus fell directly into the slit. Here we describe how we established a fiducial slit location by reconstructing the acquisition of that target.

NGC 4151 was initially acquired at an (S,L) position of (283.9,497.0), corresponding to a location of (229.1,497.0) in uncorrected *x*-*y* pixels. A slew of (ΔV2,ΔV3) = (1.071,-1.284), in arcseconds, followed by a POS TARG slew of (0.2,0) took the target into the slit. The POS TARG adjustment added a (V2,V3) slew of (0.1089,0.1677), so the total (V2,V3) slew was (1.180,-1.116) in arseconds.

Ultimately, we want to know which pixels in the *f*/48 imaging format correspond to the *f*/48 slit location after the spectroscopic mirror has moved into place. Thus, we need to convert the (V2,V3) slew to an (*x*,*y*) slew in pixel units. Adopting θ(*f*/48) = 20.6, we find that the (*x*,*y*) slew in arcseconds is (-1.497,-0.6295). The plate scale of corrected *f*/48 images is 0.02870 arcsec pixel^{-1} (see Section 6). The NGC 4151 slew therefore moved the target (Δ*x*,Δ*y*) = (-26.084,-21.93) in corrected zoomed pixels, from a corrected position of (227.7,499.2) to a corrected position of (253.8,521.1). Inverting the geometric correction, we find that the nucleus of NGC 4151 entered the slit at a fiducial (*x*,*y*) position of (256.2,518.3) in uncorrected pixels, equivalent to an (S,L) position of (256.8,518.3).

A new piece of IDL software called `f48_slew` computes the (V2,V3) slew needed to shift a target acquired in the *f*/48 512z x 1024 zoomed format to this fiducial position. This routine takes the uncorrected (S,L) acquisition position, in pixels, as input. For example,

```
f48_slew,283.9,497.0
```

would recalculate the NGC 4151 slew described above. The routine also draws a figure showing a map of the slew in corrected (*x*,*y*) pixels, the approximate slit location, and a piece of the uncorrected pixel grid in 10 x 10 blocks of zoomed pixels.