MIT Kavli Institute for Astrophysics and Space Research

Chandra X-Ray Center

MEMORANDUM

December 28, 2006

To:	Jonathan C. McDowell, SDS lead
From:	Bish K. Ishibashi, MIT/SDS
Subject:	Specification for finding zeroth order positions in grating data analysis
Revision:	1.2
URL:	`http://space.mit.edu/CXC/docs/memo_fzero_1.3.ps`
File:	`/nfs/cxc/h2/bish/CXC SDS/FINDZERO/memo_fzero_1.3.tex`

1 Locating Zero Wavelength Position

This memorandum is to provide specification for locating a 0^{th} order (ZO) position (i.e., deriving a zero-wavelength term) in Chandra HETGS/LETGS grating data analyses.

Currently the available tool for this purpose in CIAO is `tgdetect`, which locates the centroid position of point-like sources in a specified chip. The tool `tgdetect` is a wrapper script which sets up and executes several CIAO filter/copy tools, `celldetect`, and two grating specific DM tools (for details, see `''ahelp tgdetect''`). This tool is quite robust to use when the sources are truly point-like and not quite bright enough to pile up.

However, when the intended ZO target is fairly bright (e.g., Capella) in X-rays, the ZO image would be piled up. The resulting shape of the PSF would be distorted and no longer exactly point-like, hence leading to the difficulty in locating the precise centroid position of the source. Moreover, in occasion an on-board masking on the ZO image is applied so that the telemetry does not include any information on the ZO events, hence rendering the tool `tgdetect` useless.

To cope with these situations, a new tool needs to be invented in order to supplement the existing tool `tgdetect`. We propose such a new tool to derive an accurate ZO position based on grating arms and transfer streaks for ACIS (or diffraction structures in HRC).

2 Choosing ZO Detection Tools

While `tgdetect` performs well on the detection of a faint point-like source, the tool becomes less reliable once the source begins to pile up. The new ZO finding scheme, on the other hand, should perform better in the bright source regime, although the scheme would fail when the source is too faint. There is no omnipotent method to detect the ZO position accurately and one must *evaluate and choose* what method would work best for one's dataset.

Generally speaking, different observing conditions call for different schema of the ZO detection. Table 1 below enumerate possible different conditions that may arise in the `TIMED` exposure setting. The new ZO

Table 1. Different Conditions for 0^{Th}-Order Detection

Source Type	Data Characteristics	Recommended Tool	Example (ObsID)
0^{Th}-order imaged			
Extremely Bright	Cratered PSF, bright grating arms & streak	findzo	Cyg X-1 (3814)
Very Bright	Piled PSF, strong arms and streak	findzo	Capella (1103)
Bright	Weakly piled PSF, strong MEG arms	tgdetect or findzo	many
moderate	Possibly piled PSF, weak MEG arms and streak	tgdetect or findzo	Zeta Pup (640)
faint	Unpiled, faint arms and streak	tgdetect	HD206267 (1888)
Confused/multiple	Crowded point source regions	tgdetect	Orion field (3)
0^{Th}-order blocked			
Extremely Bright	Bright grating arms & streak	findzo	GRS1915+105 (660)
Very Bright	Strong arms and streak	findzo	Cyg X-2 (1016)
Bright	Strong MEG arms	findzo	??
moderate	Weak MEG arms and streak	findzo	Not applicable?
faint	Faint arms and streak	findzo?	Not applicable?
Off the chip	Grating arms only	dead reckoning	Sco X-1 (3505)

detection scheme (hereafter `findzo`) works for those "extremely-bright" to possibly "moderate" cases in both *ZO*-imaged and -blocked cases, whereas the tool `tgdetect` would cover "faint" and possibly "moderate" cases in the *ZO*-imaged case (possibly it works ok with the "bright" cases, though your mileage may vary).

It warrants to mention that no ZO detection tool would be 100% fail-proof. Especially the new scheme advocated here would not be any safer to apply to every grating dataset than `tgdetect`*; hence it may not be quite suitable for a pipeline processing (perhaps only suitable for those custom processing ObsIDs or user-reprocessing with CIAO).*

Note that this memorandum does not address the cases in the `CC` mode, in which `tgdetect` is often appropriate.

For all configurations involving HRC-S, it should be noted that the tool `tgdetect` is always suitable for detecting the *ZO* position as the detector technically does not suffer from pile-up. For that reason, this document will focus more on the configurations involving the ACIS. The new tool will be designed to work with the HRC as well.

3 Specification

Currently we intend to support three configurations: HETG + ACIS-S, LETG + ACIS-S, and LETG + HRC-S. Any other configuration may be supported[1] if there are high demands.

[1] A placeholder exists to support additional configurations.

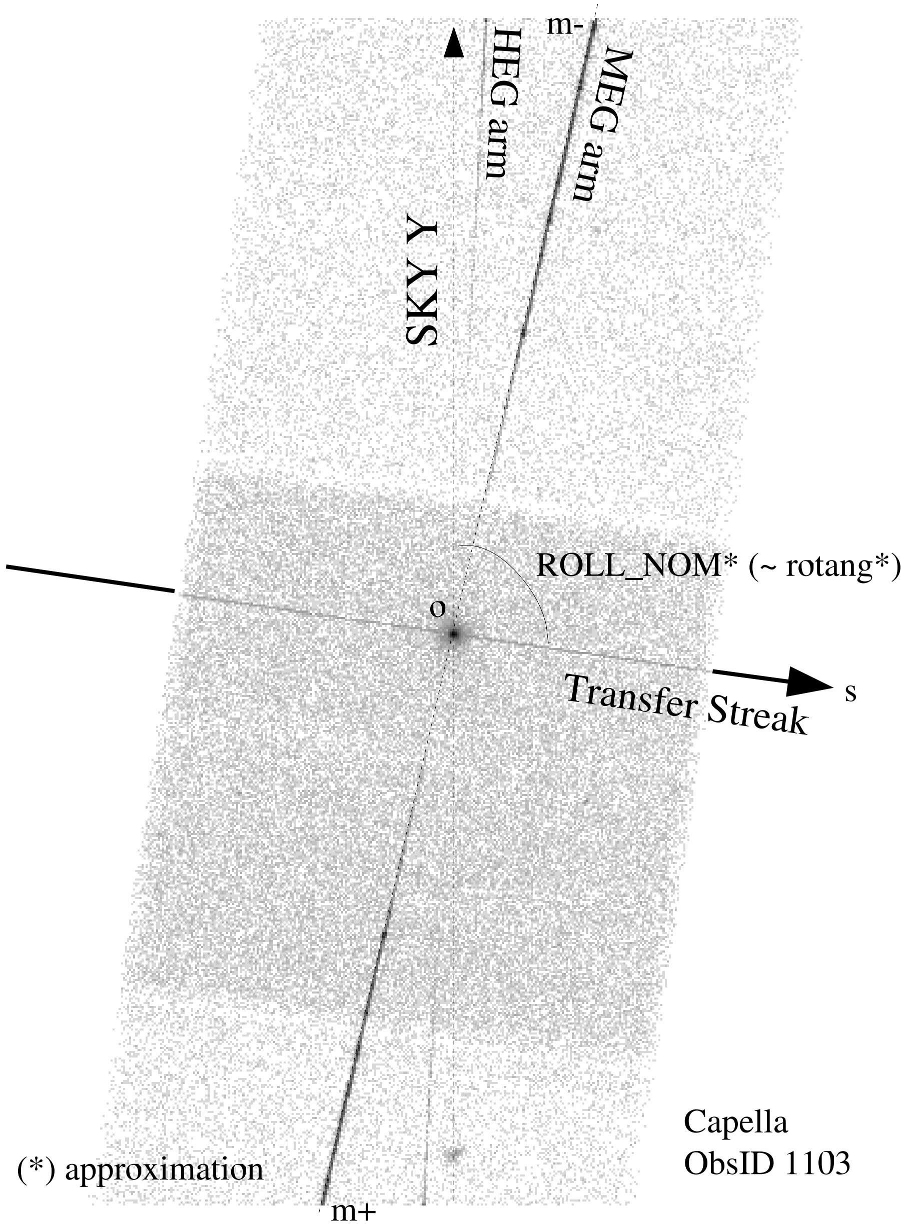

Figure 1: Finding the ZO position using a grating arm (e.g., MEG) and a data streak. The angle s–o–$m+$ (or $m-$) is fixed in the SKY frame, which is useful for a pattern match. Once the angle between the data streak and SKY-Y axis is known, the location of the MEG grating arm can be determined automatically.

We assume that the properties of the hardware do not change significantly over its mission lifetime. The current coding scheme may allow for time-dependent effect of the hardware, if required.

3.1 Big Picture

For a new *ZO* detection scheme, the name of the game is to locate an intersection of a grating arm and a *ZO* data transfer streak for ACIS, or, a diffraction pattern resulting from fine structure support for HRC[2]. The intersection position define the reference position for zero wavelength for the grating arm.

To derive an intersection, we take an advantage of the fact that the angles between a data streak and grating arms should be *fixed* in both SKY and TG coordinate spaces. For a given nominal roll angle of the spacecraft (FITS header keyword `ROLL_NOM`), the position angles of each grating arm and a data streak projected onto the SKY coordinate can be determined without tracing actual events. This enables us to find the intersection via a quasi "pattern match" scheme, where the linear fit onto the arm and streak are performed with user-specified slope values and the best fit parameters are then derived by minimizing the χ^2 statistics. Figure 1 provides an illustration of this procedure. In the case of finding a zero-wavelength position for MEG, the angle between streak (s), *ZO* position (o) and grating arm (m+ or m-) is determined from the CALDB geometry values (specified as hidden parameters in the code; see the subsection below) and then is used to lock on the lines over the arms and the streak events that minimizes the statistics. The intersection of the two fitted lines defines the zero wavelength reference position (o) for MEG. The goal of the method is set to achieve the ball-park accuracy of ± 0.1 ACIS-S pixels (or its HRC equivalent) to derive the *ZO* position.

Step 0: Initializing Parameters

In a truly ideal world, one would simply look up FITS header keywords to derive everything: the *ZO* position, the location of grating arms, etc.

But life isn't like that.

For instance, in the Chandra/HRMA system, each mirror shell *may* have a different focal point; and indeed at least one of the inner mirror shells is slightly more tilted than truly ideal. Now consider this: the HEG and MEG grating systems use two inner and two outer HRMA mirror shells, respectively. The two subsets of the HRMA shells may not necessarily have the common centroid peak, i.e., the zero-wavelength reference position for HEG and MEG may not be identical. Figure 2 illustrates this point.

Minor systematic errors may also be present in the values of the CALDB files; or, at least, we should allow for systematic errors to be present. (For example, an inaccurate rotation term defined for the SIM platform introduces the spurious rotation of 0.089° in the LETGS+ACIS-S configuration.) Also it is possible that the alpha clocking angle of a grating arm (e.g., HEG) may be somewhat inaccurate by a small degree (∼ -0.008deg). While these inaccuracies in the input values may not be critical for science, they may have somewhat a greater impact upon finding the *ZO* position to a high precision.

To account for many of these details, we have a few sets of hidden parameters hard-coded into the algorithm (which are bundled into one S-lang subroutine called `gparamio`). All the hidden parameters are tabulated in Table 2. The descriptions of each parameter set are as follows:

- **strc[1,2,3]**: as described above, the true *ZO* streak position of each grating element (i.e., HEG and MEG) is slightly offset from the observed *ZO* streak which is a superposition of the HEG and MEG streaks. Using MARX, the streak offset values are derived at a optimized energy range (around 1.5 keV). For LEG (**strc3**), the parameter should be set to zero.
- **alpha[1,2,3]**: alpha grating clock angle with respect to the spacecraft coordinate. The values are quoted from the latest CALDB geometry file (version 3.2.0).

[2] The six-pointed star pattern around the *ZO* is due to coarse structure support and should be excluded in the processing.

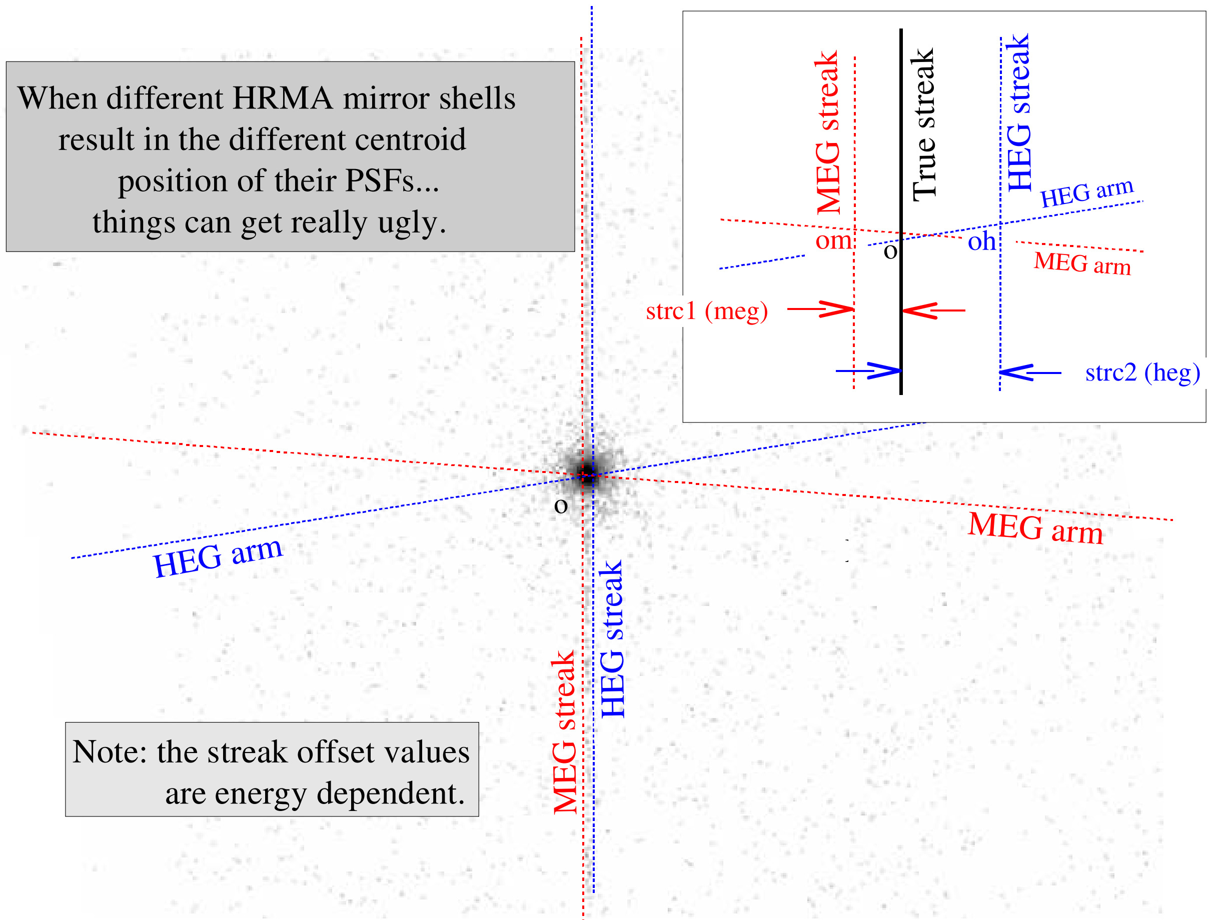

Figure 2: What consists of the ZO PSF? Because the HRMA consists of four independent mirror shells, each shell has its unique focal point and therefore the centroid of each PSF are not necessarily coincident. This means that the zero wavelength position of HEG and MEG can be very slightly different from the observed ZO centroid. The new scheme of ZO detection take this factor into account.

- **corr_a[1,2,3]**: *ad-hoc* rotation correction terms associated with alpha grating clock angle **alpha**. This set allows a temporary correction to the alpha grating clock angle (note: this should be deprecated in future).

- **tmp_a[1–4]**: *ad-hoc* rotation correction term associated with the geometry of detector or SIM platforms. The presence of spurious detector rotation (probably due to the inaccurate rotation term applied to the SIM corner positions) necessitates the introduction of a temporary rotation correction term for LETG+ACIS-S configuration (the rotation correction by -0.089°). Currently the LETG+ACIS-S configuration utilizes this parameter set (note: this should also be deprecated in future).

- **[aciss,hrcs]cor**: the rotation correction term of ACIS-S3 and HRC-S1 chips with respect to the spacecraft coordinate. The values are derived from the latest CALDB geometry file (version 3.2.0). We also provide dummy place-holders for ACIS-I and HRC-I (**[acisi,hrci]cor**) for future use (currently set to zero).

- **[acis,hrc]_plate**: ACIS and HRC plate scale in arcsec (defined in the imaging mode).

- **[boxw, boxl,rad]_strc**: the default geometry of the region box for data streak events.

- **[boxw, boxl,rad]_garm**: the default geometry of the region box for grating arm events.

- **n_del**: A threshold to use for data clipping upon selecting streak and grating events.

- **n_eng_[lo,hi]**: Lower and upper energy threshold used in the event filtering (in unit of eV).

The rotational angle is defined to be positive in CLOCKWISE direction around the X-axis in spacecraft coordinate system (hence in CCW direction in the SKY frame). Users need not to adjust any of the hidden parameters, unless directed otherwise by experts.

Step 1: Make A First Guess

The first action of every user is to make a guess on *roughly* where the *ZO* position is. The guess should be made either by looking the actual image with `ds9`, etc., and read off the SKY [X,Y] coordinate of the target, or, simply by choosing the nominal pointing coordinates given in the FITS header values (RA_NOM, DEC_NOM). So the new *ZO* detection scheme should allow users (1) to enter user-specific SKY coordinate values or (2) to use the nominal pointing coordinate as the first guess. A dumb guess (near a default center of each detector, i.e., [4096.5, 4096.5] for ACIS-S) may be chosen as well.

If the user chooses a user-specified SKY coordinate [X0,Y0], the coordinate would be used as the guess [gX, gY]. If one instead chooses to use the nominal coordinate, the coordinate info (RA_NOM, DEC_NOM) in the input event FITS file will be translated to the SKY coordinates. However, this may fail if the user has specified to put its nominal pointing wildly off-centered from an intended target (which seems to happen more often than we anticipate). But if the first guess is close enough (within a few dozen pixels away), the routine should be able to lock onto the intended point source in order to determine true *ZO* position. The new *ZO* detection scheme chooses this option by default.

Step 2. Derive the Position of the Streak

In ACIS detector, data transfer streak runs along the columns of a CCD chip. For the ACIS-S3 chip, the chip is slightly rotated (by +0.03°) with respect to the space craft coordinate frame (and hence the SKY coordinate frame as well). In order to rectify each data frame such that data streaks run vertically in the new rotated frame, one needs to rotate the whole event positions in the SKY coordinate by the angle (*rotang*):

$$Rotation\ Angle = (ROLL\_NOM) + (ACIS - S\ Rotation\ Correction)$$

where the ROLL_NOM value is the nominal roll angle value of the space craft found in the Chandra FITS header and the ACIS-S Rotation Correction term is the parameter value *acisscor* given in Table 2. Each

Table 2. User-Specific and Hidden Parameters

User-Specific Parameters

Parameter	Value	Description	Type
infile	"Event File"	Input Event file (CFITSIO filtering allowed)	file
grating	h,m, or l	Grating Type (h, m, or l for HEG, MEG, or LEG)	string
mode	1,2,3,or 4	Operation Mode (to make a first guess on the *ZO* position)	numeral
$x0$,$y0$	$x0$,$y0$	(Optional) guestimated *ZO* value (used only if $mode = 4$)	numeral

Hidden Parameters

Parameter	Value	Description	Reference
strc1	-0.011	Transfer streak offset position (pixels) for MEG	(1)
strc2	0.05	Transfer streak offset position (pixels) for HEG	(1)
strc3	0.00	Transfer streak offset position (pixels) for LEG	(0)
alpha1	4.755°	Alpha grating clocking angle for MEG	(2)
alpha2	-5.205°	Alpha grating clocking angle for HEG	(2)
alpha3	0.07°	Alpha grating clocking angle for LEG	(2)
corr_a1	-0.006°	Arbitrary rotation correction factor for alpha1	(0)
corr_a2	0.001°	Arbitrary rotation correction factor for alpha2	(1)
corr_a3	0.00°	Arbitrary rotation correction factor for alpha3	(0)
tmp_a1	0.00°	Arbitrary rotation factor for detector geometry (MEG)	(0)
tmp_a2	0.00°	Arbitrary rotation factor for detector geometry (MEG)	(0)
tmp_a3	-0.089°	Arbitrary rotation factor for detector geometry (LEG+ACIS-S)	(3)
tmp_a4	0.00°	Arbitrary rotation factor for detector geometry (LEG+HRC-S)	(0)
acisscor	0.03°	Rotational correction for ACIS-S3 chip	(2)
hrcscor	0.2744°	Rotational correction for HRC-S1 plate	(2)
acisicor	0.00°	Rotational correction for ACIS-I chip	(0)
hrcicor	0.00°	Rotational correction for HRC-I plate	(0)
acis_plate	0.492″	ACIS plate scale (in the imaging mode)	(2)
hrc_plate	0.01318″	HRC plate scale (in the imaging mode)	(2)
boxw_strc	100	Default box width (pixels) for Streak region	(1)
boxl_strc	2000	Default box length (pixels) for Streak region	(1)
rad_strc	50	Default exclusion radius (pixels) around *ZO* for Streak region	(1)
boxw_garm	50	Default box width (pixels) for grating arm region	(1)
boxl_garm	5000	Default ACIS-S box length (pixels) for grating arm region	(1)
boxl_garm	12000	Default HRC-S box length (pixels) for grating arm region	(1)
rad_garm	50	Default exclusion radius (pixels) around *ZO* for grating arm region	(1)
n_del	1.3	Significance threshold for array clipping upon selecting grating events	(1)
n_eng_hi	4000	Upper threshold for energy filtering (eV)	(1)
n_eng_lo	500	Lower threshold for energy filtering (eV)	(1)

[0.] Dummy placeholder for future use.
[1.] Derived by the author based on real observations.
[2.] Quoted/derived from the latest CALDB geometry file.
[3.] Adjustment for spurious SIM rotation term introduced in the LEG + ACIS-S configuration.

event position [X,Y] is rotated around the initial guess of *ZO* position [gX, gY] (see Step 1) to form the rotated vector array [Xp,Yp]. For the HRC-S, we choose a coarse diffraction pattern to be vertical in the rotated frame. The same equation for Rotation Angle (using *hrcscor* instead) will apply.

Then we apply spatial (rectangular) filtering ($boxw\_strc \times boxl_s trc$) to choose data streak events exclusively. The *ZO* events (radius of $rad_s trc$) are also excluded. To make the analysis simple, all the events with the energy *En* above *n_eng_hi* and below *n_eng_lo* are excluded as well. The actual filtering condition (in S-Lang) is given below:

```
i = where(
        Xp > (gX-boxw_strc/2) and Xp < (gX+boxw_strc/2) and
        Yp > (gY-boxl_strc/2) and Yp < (gY+boxl_strc/2) and
                  Xp > 0.0 and Yp > 0.0 and
            hypot(Xp - gX,Yp - gY) > rad_strc and
              (En > n_eng_lo and En < n_eng_hi)
);
```

The selected events [Xp[*i*],Yp[*i*]] are then used to derive median positions along the streak (see the subroutine `median_peak()`). We break up the streak into 20 pieces along the column and then find the median centroid position [mXp, mYp] in each bin. Because there may be a gap in the streak (block out, etc), some anomalous median data points in [mXp,mYp] are rejected via data clipping (with a significance of *n_del*, or 1.3 by default – see the subroutine `clip_array()`). The remainder of the data points [mXp,mYp] are then fitted with a vertical line to determine the centroid peak position *slp*[0] of the data streak (note that the subroutine `garm_fit()` allows to perform a linear fit with a fixed slope value, which is zero in this case). The process illustrated here is often iterated until the derived parameter *slp*[0] converges to the unique solution.

As illustrated in Figure 2, the peaks of data streak for HEG and MEG may be different. So we allow for ourselves to make a very small correction for each case:

$$strx = slp[0] + strc[\mathbf{n}]$$

where *strc*[**n**] [n = 1,2,3] is the correction factor given in Table 2. Again, this correction is not necessary for the LEG. The derived value *strx* will be used as the position of the streak in the rotated [Xp,Yp] frame.

Step 3. Determine the Location of Grating Arms

The method used in this step is basically identical to Step 2. To make it simpler for spatial region filtering, the data frame [Xp,Yp] is now rotated by the angle *grang* (see Figure 3):

$$grang = alpha[\mathbf{n}] - acisscor + tmp\_a[\mathbf{n}] \qquad [\mathbf{n} = 1, 2, 3]$$

so that all the grating events (now in the new frame [Xpp, Ypp]) are found in a rectangular region predefined by a spatial filter:

```
i = where(
        Xpp > (sXp-boxw_garm/2) and Xpp < (sXp+boxw_garm/2) and
        Ypp > (sYp-boxl_garm/2) and Ypp < (sYp+boxl_garm/2) and
                  Xpp > 0.0 and Ypp > 0.0 and
            hypot(Xpp - sXp,Ypp - sYp) > rad_garm and
                         En < 8000.
);
```

where $sXp = strx$, $sYp = gY$ initially, and others parameters are defined in Table 2. Note that the parameter sYp is what we want to derive in this step and the value sYp may change through a iterative loop

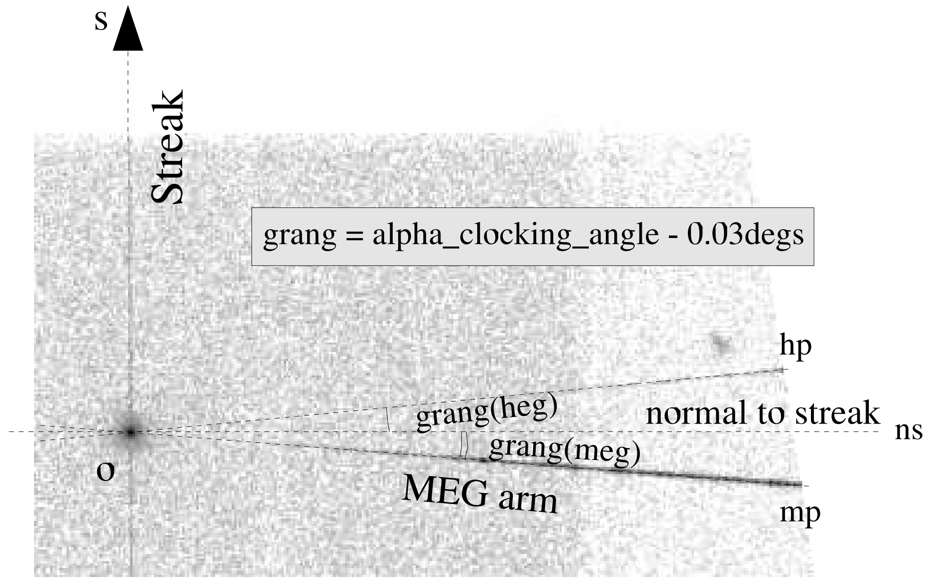

Figure 3: Grating angle *grang* defined w. r. t. S3 CHIP+Y (the image shown in the SKY frame). Since the grating alpha clock angle is defined w.r.t. spacecraft coordinate, the rotation of ACIS-S3 chip (0.03deg) is being subtracted in order to drive the angle *grang*.

(see below). The fixed energy filtering is applied here since there is no energy dependency in determining the sYp value.

Once again the selected events [Xp[i],Yp[i]] are then used to derive median positions along the streak with `median_peak()` (with a number of bins *bin_garm*), then find a new set of the median centroid position [mXp, mYp] in each bin. Just as in Step 2, the same data clipping is applied (`clip_array()`) and then the remainder of the data points [mXp, mYp] are fitted with a linear line with a fixed slope value *grang* (using `garm_fit()`). Note that we performs the fitting in the data coordinate frame [Xp,Yp], not in the *grang* rotated frame [Xpp,Ypp].

Now the position [sXp, sYp] of the intersection of the data streak and grating arms is computed and is fed back as the new ZO solution to repeat the entire processes in Step 3. This is repeated until the solution converges to the identical value.

Step 4. Computing the actual ZO position

The derived ZO position [sXp, sYp] in the coordinate frame [Xp, Yp] needs to be rotated back to the coordinate frame [X,Y]. This is done simply by rotating back the position vector by the angle *rotang* around the position [gX, gY]. The resulted position vector [zX, zY] will be returned as the ZO position used in conjunction with CIAO.

4 Evaluation

To verify the performance of this new ZO finding code `findzo`, three independent tests are performed on: (1) MARX, (2) Capella, and (3) randomly chosen sampling of HETGS and LETGS datasets.

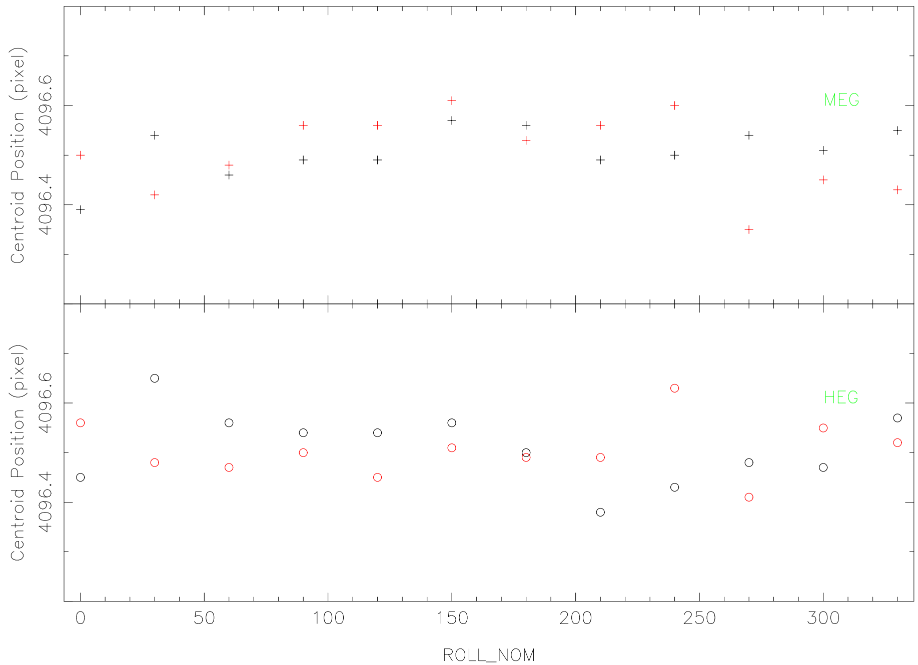

Figure 4: Derived *ZO* position with the MARX simulated datasets. The derived coodrinate vectors X and Y are marked in black and red, respectively. (Top): MEG and (bottom): HEG cases.

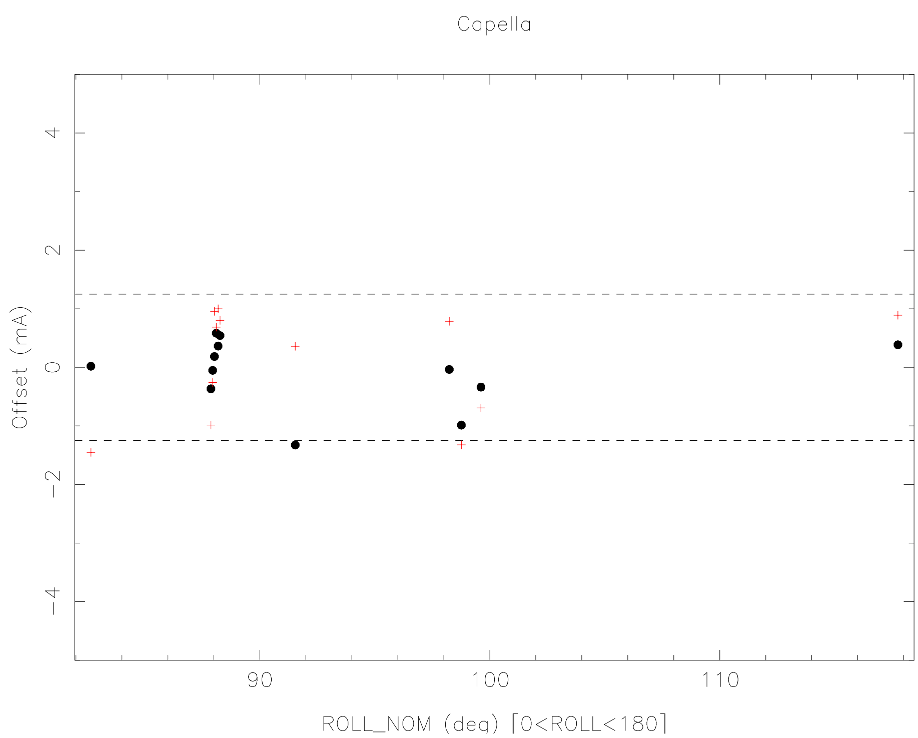

Figure 5: The mean differences (in mÅ) in the line centroid position of selected bright emission lines between plus and minus order spectra as a function of Roll_NOM angle. The dashed lines indicate the CIAO default width of HEG spectral bin size (2.5mÅ).

Table 3. Measured *ZO* Positions with `findzo` and `tgdetect` (Capella)

	findzo				tgdetect	
	MEG		HEG		*ZO* image	
OBSID	x0	y0	x0	y0	x0	y0
0057	4101.2	4133.33	4101.4	4133.39	4101.26	4133.19
1010	4100.81	4089.54	4101.11	4089.64	4100.97	4089.36
1099	4134.14	4061.61	4133.88	4061.54	4134.50	4061.61
1100	4134.37	4061.25	4133.92	4061.17	4133.92	4061.07
1101	4134.18	4061.24	4134.03	4061.18	4133.97	4061.11
1103	4127.37	4059.04	4127.07	4059.02	4127.69	4058.59
1235	4134.36	4061.2	4134.1	4061.13	4133.99	4061.04
1236	4134.31	4061.22	4133.92	4061.14	4133.66	4061.08
1237	4134.19	4061.12	4133.9	4061.05	4134.44	4061.02
1318	4126.83	4058.87	4126.6	4058.84	4127.09	4059.23
2583	4099.81	4091.17	4100.03	4091.13	4099.88	4091.20
3674	4129.75	4123.36	4129.42	4123.36	4129.49	4123.62

MARX Testing

MARX 4.2.1 is used to generate a flat spectrum (from 0.05 – 12 keV with the source flux of 0.04 photons/sec/cm^2) at its default *ZO* source position of SKY [X, Y] = [4096.5, 4096.5] at various ROLL_NOM angles.

Each simulation is processed to generate a Chandra EVENT L1 FITS file and then processed with the `findzo` routine to examine if it performs as expected. The results are illustrated in Figure 4. Basically the derived *ZO* positions (X & Y positions in black and red points, respectively) scatter about the expected sky coordinate value [zX,zY] = [4096.5, 4096.5]. The standard deviation of the derived *ZO* position from the expected position is about 0.1 pixels in radius. The scatter of this scale may be treated as a statistical uncertainty in determining a *ZO* position with `findzo`.

Capella Testing

Like in the MARX test case, all available Capella datasets are run with the `findzo` routine to determine the *ZO* position (for both HEG and MEG separately) and are reprocessed – with the new *ZO* position – using CIAO 3.3 to extract a PHA II spectrum in each case. Selected bright emission lines of Capella (e.g., Fe XVII and Ne XI lines) are the fitted with a Gaussian component in both plus and minus orders of HEG and MEG grating spectra and the mean difference in the centroid position of the emission lines between plus and minus 1^{st} orders is recorded for each ObsID. The results are plotted in Figure 5. The mean differences in the line centroid position for HEG and MEG are $-8.5\times10^{-2}\pm5.9\times10^{-1}$ and $-6.4\times10^{-2}\pm9.5\times10^{-1}$mÅ, respectively. The scale of the error in wavelength corresponds to 0.10 and 0.09 pixels, which are approximately the same as the uncertainty values derived from the simulated case. For the reference, Table 3 provides the derived *ZO* positions in the SKY coordinate [x0, y0] for both MEG and HEG, plus the derived *ZO* coordinate positions using `tgdetect` for comparison.

Random Picks of HETGS and LETGS datasets

Based on the random selection of Chandra grating datasets chosen by J. Nichols et al. (pipeline), we have examine the rate of failure of the `findzo` tool. Note that the most of the datasets chosen here (except for ObsID 2708, 3354, and 4148) are perhaps too faint to be used for the `findzo` tool. Despite of that, of 21 datasets, the tool has identified the *ZO* positions reasonably well for 11 times (marked as “ok”, “piled” or “blocked”). The cause of all the failures is solely due to the faintness of the source (marked as “too faint”). The outcome of the `findzo` usage is tabulated in Table 4.

Table 4. Measured *ZO* Positions with `findzo` and `tgdetect` (A la Carte)

	`findzo`				`tgdetect`		Remark
	MEG		HEG		*ZO* image		
OBSID	x0	y0	x0	y0	x0	y0	
2533	4050.88	4043.17	4050.71	4043.28	4050.65	4043.03	ok
2534	4044.78	4049.42	4044.56	4049.66	4044.67	4049.50	ok
2571	4071.18	4162.51	4074.31	4162.98	4071.10	4165.01	too faint
2689	4084.24	4152.52	4074.8	4151.48	4111.05	4137.92	too faint
2690	4078.58	4161.6	4080.32	4161.76	4114.86	4138.51	too faint
2691	4104.52	4163.64	4081.03	4161.31	4111.98	4138.21	too faint
2708	4085.42	4165.16	4085.66	4165.21	4086.85	4167.37	piled *ZO*
2710	4105.86	4165.94	4106.11	4166.01	4106.21	4166.26	ok
2730	4079.9	4082.58	4079.44	4082.64	4079.49	4079.29	too faint
2757	4086.34	4158.05	4086.54	4158.13	4086.46	4158.15	ok
2811	4035.93	4079.25	4035.89	4079.45	4036.32	4079.32	ok
2969	4108.88	4102.12	4108.49	4102.19	4108.88	4102.42	ok
2977	4082.1	4101.33	4082.24	4101.15	4082.05	4101.22	ok
3354	4124.02	4029.95	4123.82	4029.84	4152.89	3837.16	blocked *ZO*
4284	4108.21	4100.67	4108.21	4100.6	4108.91	4102.33	too faint
4364	4084.83	4094.22	4083.81	4094.41	4083.52	4087.98	too faint
4367	4071.13	4161.8	4072.46	4162.03	4071.10	4165.01	too faint
4381	4096.72	4078.45	4097.95	4082.76	4095.93	4079.01	too faint
3749	4059.55	4037.25	4059.37	4037.33	4059.39	4037.30	ok
2812	4103.2	4151.66	4103.48	4151.63	4170.19	4117.28	too faint
			LEG				
4148			4001.89	4220.18	4002.36	4218.60	ok

5 Summary and Recommended Actions

As demonstrated in the Evaluation section, this new *ZO* detection routine `findzo` is a robust tool to use for determining the *ZO* position accurate to one tenth of the ACIS-S CCD pixel. While the tool may not be so suitable (or stable) for the use as part of the pipeline processing, it would be useful to general users or to those who perform custom processing for achieving better fidelity in the zeroth wavelength calibration.

Appendix: Usage of FINDZO

The new *ZO* detection algorithm is written in S-Lang and can be used in both isis and sherpa. Its syntax, despite the complexity of the algorithm, is quite simple:

```
(x,y)=findzo(input_event_file, grating_type [, option_mode [, x0, y0]]);
```

where input_event_file is the input event L1 (or L2) file, grating_type is [“h”, “m”, “l”] (either HEG, MEG or LEG, respectively). Note that the value of grating_type needs to be string. The output coordinate value [X,Y] is returned by the script findzo(). For option_mode, users have four choices:

```
Option_mode: 1. (default) Use RA_NOM, DEC_NOM for the initial guess of ZO.
             2. Set the initial guess of ZO as [median(X), median(Y)].
             3. Use the dummy coodinate [4096.5, 4096.5].
             4. Use a user-specified coordinate [x0, y0].
```

By default, the option_mode is set to 1. For example, the script findzo() can be executed as simple as:

```
(x,y) = findzo(‘‘acisf0001N0001_evt1.fits’’,’’m’’)
```

to find out the *ZO* position of the point source near RA_NOM and DEC_NOM for MEG. For HEG, users should replace “m” with “h”.[3] If users wish to specify the initial guess more explicitly, then execute the script as:

```
(x,y) = findzo(‘‘acisf0001N0001_evt1.fits’’,’’m’’,4, 4090.2, 4061.7)
```

In this case, the algorithm will look for the source near the SKY coordinate [X0, Y0] = [4090.2, 4061.72]. This option may be useful when the specified nominal coordinate position differs significantly from the actual location of the point source. At the end of each run, two pgplot windows will pop to show where the estimated zeroth order position may be found. Users are encouraged to examine these plots to see if the algorithm has missed the zeroth order centroid position.

The use of simple CFITSIO extended file name syntax (filtering) is also allowed by fitsio S-lang module. For example, if the uesr wishes to filter events with the energy range of 1000 to 6000 eV:

```
(x,y) = findzo("acisf0001N0001_evt1.fits[energy=1000:6000]","m");
```

Note that the extension block name ([EVENTS]) is set by default and hence there is no need to type it in the syntax.

The derived *ZO* coordinate position [X, Y] is then used to create a region mask using the CIAO tool tg_create_mask. For example, to create a region mask for a user-specified coordinate position [X,Y] = [4090.23,4061.72], users need to set its parameters as follows:

```
pset tg_create_mask use_user_pars=yes last_source_toread=A
pset tg_create_mask sA_id=1 sA_zero_x=4090.23 sA_zero_y=4061.72
pset tg_create_mask sA_zero_rad=20 sA_width_heg=15 sA_width_meg=20
```

The width and radius parameters are specified here arbitrarily. Users should consult the CIAO/tg_create_mask documentation for details.

[3]However, it is generally adequate to determine the *ZO* position for MEG and use the same coordinate value for extracting HEG spectra.

FINDZO.SL:

```
%
% This file is part of findzo.sl (finding zeroth order).
% Copyright (C) 2006 Massachusetts Institute of Technology
%
% This software was developed by the MIT Center for Space Research under
% contract SV3-73016 from the Smithsonian Institution.
%
% This program is free software; you can redistribute it and/or modify
% it under the terms of the GNU General Public License as published by
% the Free Software Foundation; either < version number > of the
% License, or (at your option) any later version.
%
% This program is distributed in the hope that it will be useful, but
% WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
% General Public License for more details.
%
% You should have received a copy of the GNU General Public License
% along with this program; if not, write to the Free Software
% Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA
%
%-----------------------------------------------------------------------
%
% File: findzo.sl
%
% Version:
%
% Authors:  Bish Ishibashi
% Date Created: 21 APR 2005
%
%
% HISTORY:
%  This script has been modified numerous times.
%
% September 21, 2005:  Repair for LETG+HRC-S.
% Hidden / unnecessary parameters removed.
% Then some unnecessary parameters added as a placeholder
% for future support of HRC-I and ACIS-I.
%
% September 23, 2005:  Add a support for cfitsio subspace filtering.
% September 30, 2005: Remove the namespace setting since sherpa
% imports isis by default and is no
% longer necessary.
%
% March     21, 2006:     Changes in parameter names and correction
% on cfitsio subspace filtering.
%
%  August    24, 2006: Add a check on subarray condition. Scale rad_strc if necessary.
% August    31, 2006:  Replace "static" with "private". Add "provide()" at the end.
% October    3, 2006:     Add a better bowl_strc condition in the subarray situation.
% Allow NROWS to be set to 1024 when no subarray is used.
% October   17, 2006: Correct an error on deriving the initial guess [gx, gy] for
```

```
% ftype==1. I sloppily omitted cos(decnom) in evaluating an
% angular separation in RA.
% October   18, 2006: Per JED's advice, implement the conditions to control verbosity
% (i.e., to enable users to make it less chatty). The executable
% script testzo is also adjusted for this. A library called
% set_findzo_verbose is added to utilize this feature.
% October   23, 2006: Change in the grating angle parameters (consistent with the
% geometry update reported by me in July 2006.
%       December  27, 2006:     Adjust verbose mode settings and messages.
%
%
%========================
%
% Purpose: This program finds the centroid position (SKY_X,SKY_Y) of zeroth order
%     image by fitting lines to the MEG and readout streaks, and then determining
%    the intersection of the lines.
%
% Syntax:  (x,y)=findzo(evtfile_name, grating_type [,guessing_mode, gx, gy])
%
% Input:
%    evtfile_name == Event L1.5 or 2 file.
%    grating_type == "h" (HEG), "m", or "l" (LEG).
%    guessing_mode == 1 = astrometry based
%     2 = median filter based
%     3 = dumb guess ([X,Y]=[4096.5,4096.5])
%     4 = user specified guesses (sky x and y)
%    gx, gy == initial guess on the source position (in sky)
%
% Example
%  by default, flag == 1:
%    (x,y)=findzo("acisf0001N0001_evt1.fits","m")
%
% Or try something fancier. When using CFITSIO filtering,
%       do not specify the extention block name since it is set to
%       [EVENTS] by default (always true for Chandra Event files).
%
%    (x,y)=findzo("acisf0001N0001_evt1.fits[energy=2000:8000]","h",4,4090.2,4061.72)
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%
% Version: commented out for now.
%
private variable _version = [0, 9, 1 ]   ; % major, minor, patch
%



%
% Verbosity default value
%
private variable FINDZO_Verbose = 5;

private define vmsg ()
```

```
{
   variable args = __pop_args (_NARGS-1);
   variable vb = ();
   variable fp = stdout;
   if (FINDZO_Verbose >= vb)
      () = fprintf(fp, __push_args (args));
}

define set_findzo_verbose (vb)
{
   FINDZO_Verbose = vb;
}



%
% Control Display Setting
%
private variable FINDZO_Plot_Dev = "/xwin";

private define pdev ()
{
   variable pb = ();
   variable pw;

   if (FINDZO_Plot_Dev == "/xwin")
   {
      if (pb == 3) return;
      pw = open_plot("/xwin");resize(20,0.9);erase;
      window(pw);
   }
   else if (FINDZO_Plot_Dev == "/cps")
   {
      if (pb == 1) pw = open_plot("findzo.ps/cps");
      if (pb == 3) close_plot;
      else return;
   }
   else pw = open_plot("/xwin");
}

define set_findzo_plot_dev (pb)
{
   FINDZO_Plot_Dev = pb;
}


%
% Subroutines:
%
%-------------------------------------------------------------------------
% bfun:
%
%  S-Lang function: y = f(x,pars) = grang*(x + pars[0]) + b.
%
```

```
% The function will be fit to derive the best-fit parameter value "b".
% The slope of the function f(x,pars) is fixed in this function.
%

private define b_fun (x, pars)
{
   return pars[0] + x ;
}



%-----------------------------------------------------------------------
% garm_fit:
%
% Fit a line with a specificied slope value.
%
% This function depends on a S-lang function b_fun.
%
private define garm_fit (x,y,angle)
{
   variable pars, pars_min,pars_max;
   variable out,stat;
   pars=4000; % This is rather ACIS-S specific.
   pars_min=-1e5;
   pars_max=1e5;

   (out,stat) = array_fit (x*angle,y, NULL, pars, pars_min, pars_max, &b_fun);
   return [out,angle], NULL;
}



%-----------------------------------------------------------------------
% clip_array:
%
%  A customized S-lang function to perform clipping of array
% elements found NSIG x SDEV times above or below the AVERAGE
%  value of the array Y. This function will return the indices
% of the array Y that meets the given criterion NSIG.
%

private define clip_array( y, nsig )
{
   variable l, m, n, lprev;

   l = where( y == y ); lprev = 0 ;

   while ( length(l) - lprev )
   {
      lprev = length(l);
      m = moment( y[l] );
      l = where ( abs( y - m.ave ) < (nsig * m.sdev)  ) ;
   }
   return l ;
```

```
}



%-----------------------------------------------------------------------
% rotxy:
%
% A simple S-Lang function for 2-dimensional array rotation.
%
% The function rotates the input array (x,y) by the value ANGLE around
% the pivot point (x0, y0). For a given positive ANGLE value, the array
% is rotated in a CLOCK-WISE direction.
%

private define rotxy( x, y, x0, y0, angle )
{
   if (_NARGS !=5 )
   {
      message("USAGE: (newx, newy) = rotxy( x, y, x0, y0, angle );");
      message("    Rotate axes clockwise by angle (in radians)");
      message("    with center, x0, y0");
      return -1;
   }

   variable ca, sa, xca, xsa, yca, ysa ;

   ca = cos(angle);  sa = sin(angle);
   xca = (x-x0)*ca; xsa = (x-x0)*sa; yca = (y-y0)*ca; ysa = (y-y0)*sa;

   return xca+ysa + x0, -xsa+yca + y0 ;
}



%-----------------------------------------------------------------------
% median_peak:
%
%  A special S-Lang function to trace CHANDRA grating arms detected
%  by ACIS and HRC detectors.
%
% This function performs the following steps:
%
% (1) read in two data arrays (X,Y); the elements in Y is binned up
%     along X with a specified bin width BIN.
%
% (2) Use the clip_array function to clip data points that are considered
%     as insignificant (its significance controlled via DEL).
%
% (3) then derive the median value of data points in each data bin along X.
%
% (4) return the median value in each bin.
%
% This function is actually NOT a mission-specific one. It could work for
% any 2-D data array.
```

```
%

private define median_peak(x,y,bin,del)
{
   variable xp, yp, nx, j, k, l,m;

   nx=int((max(x)-min(x))/bin);
   xp=[0:nx-1:1]*bin +min(x)+bin*0.5;
   yp=[0:nx-1:1]*0.0-32768.; % Silly little hack.
% Sometimes the median values
% are not found in the routine;
% when that happens, the value YP
% stays as signed integer -32768.
% The array elements with this value
% are rejected and not returned.
   for (k=0; k < length(xp); k++)
   {
      j=where((x >= xp[k]-bin*0.5) and (x < xp[k]+bin*0.5));
      (m,,)=array_info(j);
      if (m[0] >= 2)
      {
         l=clip_array(y[j],del);
         (m,,)=array_info(l);
         if (m[0] >= 2)
         {
            yp[k]=median(y[j[l]]);
         }
      }
   }
   m=where(yp != -32768.);
   return xp[m],yp[m];
}



%-----------------------------------------------------------------------
% gparam_io:
%
%  A S-Lang function to set up both required and hidden parameters
%  to make the function "findzo".
%
% This function is essentially Bish's version of "param_io" that controls
% the parameter setting to run the function FINDZO. This one is CHANDRA
% mission specific. Almost all the hard-coded parameter values are found
% in this function (hence, this is where the users would want to mess with
% if he/she wants to customize parameters...I don't see why, though).
%

private define gparam_io (fitsfile, fitssub, gtype, ftype, x0 ,y0)
{
%
% Initialize Variables

   variable fp, nx, ny;
```

```
   variable rotkey="ROLL_NOM";
   variable grelem="GRATING";
   variable detnam="INSTRUME";
   variable ranom, denom, ratg, detg, rarf, derf;
   variable deg_rad=PI/180.;
   variable gx, gy, gflag;
   variable x, y, en, cy;
   variable rotang, grang;
   variable subarr;

%
% note on angles: positive angle == clockwise rotation angle in space craft coordinate.
%

   variable strc  = 0.0;  % Initialization
   variable strc1 =-0.011;  % Streak offset position for MEG
   variable strc2 = 0.05; % Streak offset position for HEG
   variable strc3 = 0.00; % Streak offset position for LEG (zero)

   variable acisscor = 0.03; % empirical acis-s rotation correction
   variable hrcscor  = 0.2744;     % empirical hrc-s rotation correction
   variable acisicor = 0.0;  % empirical acis-i rotation correction (dummy placeholder)
   variable hrcicor  = 0.0;  % empirical hrc-i rotation correction (dummy placeholder)

   variable alpha1 = 4.755;  % dispersion angle for MEG wrt CHIP_Y (from CALDB)
   variable alpha2 =-5.205; % dispersion angle for HEG wrt CHIP_Y (from CALDB)
   variable alpha3 = 0.07; % dispersion angle for LEG wrt CHIP_Y (from CALDB)

   variable corr_a1 =-0.006;  % supplemental correction factor for alpha1
   variable corr_a2 = 0.001; % supplemental correction factor for alpha2
   variable corr_a3 = 0.00; % supplemental correction factor for alpha3 (dummy placeholder)

   variable tmp_a1 = 0.00;  % ad-hoc correction factor allowed for alpha1 (dummy placeholder)
   variable tmp_a2 = 0.00; % ad-hoc correction factor allowed for alpha2 (dummy placeholder)
   variable tmp_a3 =-0.089; % ad-hoc correction factor allowed for alpha3
% ACIS-S or LETG arm is wicked and need
   % an additional bogus correction factor.
   variable tmp_a4 = 0.00; % ad-hoc correction factor allowed for alpha4
% (LETG+HRC-S: dummy placeholder)

%
%  Plate scale (in SKY)
%

   variable acis_plate = (0.492/3600.)*deg_rad; % in radian
   variable hrc_plate = (0.1318/3600.)*deg_rad; %



%
% First, check its grating configulation. And then check the nominal roll angle.
% If these information are not found (e.g., it is not a grating observation),
% then the NULL values will be returned.
%
```

```
   if (fits_key_exists(fitsfile,grelem))
   {
      grelem=fits_read_key(fitsfile+fitssub,grelem);
   }
   else return NULL,NULL; % no rotkey then return null

   if (fits_key_exists(fitsfile,rotkey))
   {
      fp=fits_open_file(fitsfile+fitssub,"r");
      ()=_fits_get_colnum(fp,"x",&nx);
      ()=_fits_get_colnum(fp,"y",&ny);
      ()=_fits_close_file(fp);

      (rotang,ranom,denom,ratg,detg,rarf,derf,detnam)=
           fits_read_key(fitsfile+fitssub,rotkey,"TCRVL"+string(nx),"TCRVL"+string(ny),
   "RA_TARG","DEC_TARG","TCRPX"+string(nx),"TCRPX"+string(ny),detnam);
   }
   else return NULL,NULL; % no grating then return null

%
% Case 1: HETG
%
%
% The lines below seem too complicated? Here is a synopsis.
%
% First for ACIS-S. The ACIS-S array happens to be tilted by -0.03deg, i.e.,
% transfer streak of zeroth order is tilted by +0.03 deg in Sky coordinate.
% And yet the alpha clock angles of HEG and MEG arms are -5.205 and 4.755 degs
% with respect to CHIP_Y axis across the detector array (reminder, positive
% angle goes in clockwise direction; i.e., -5.205 deg == 5.205 deg in ccw
% from CHIP_Y axis). So if the whole array is rotated by -0.03 such that
% the streak is now vertical in the mapped frame, then the alpha clocking
% angles must be rotated by -0.03 degrees. And that's what is happening.
%
% The same goes with LETG and ACIS-S, though there is an ad-hoc parameter
% to correct inconsistency (that rotation problem that plague LETG).
%
% Also a similar issue emerges in LETG+HRC-S. But things get much muddier
% in that configuration. Really really nasty and I have no idea how accurate
% the existing geometrical calibration on this particular configuration is.
%
%
% rotang, strc, and grang are determined here.
%

   if (grelem == "HETG")
   {
      alpha1 = alpha1 + corr_a1;  % Ad-hoc correction (empirical value)

      alpha2 = alpha2 + corr_a2;  % Ad-hoc correction (empirical value)
  % maybe set to zero (Sep 20, 2005)

      rotang = rotang + acisscor; % compensating a small ACIS-S detector
          % rotation so that the streak is vertical.
```

```
      if (gtype == "h")
      {
         strc = strc2;
         grang = alpha2 - acisscor + tmp_a2; % this is the corrected alpha clocking angle
          % in the final rotated frame (streak vertical).
      }
      if (gtype == "m")
      {
   strc = strc1;
         grang = alpha1 - acisscor + tmp_a1;    % Ditto
      }
         gflag=12;
    }    % the end of if (grelem == "HETG")


%
% Case 2: LETG + HRC-S or ACIS-S. This section is a mess.
%

   if (grelem == "LETG")
   {
      gflag=3;
      alpha3 = alpha3 + corr_a3;   % Ad-hoc alpha correction (empirical value)

      if (detnam == "HRC")
      {
 strc = strc3;
 rotang = rotang + hrcscor;
 grang = alpha3 + tmp_a4;
      }
      if (detnam == "ACIS")
      {
         strc = strc3;
 rotang = rotang + acisscor; % compensating a small ACIS-S detector rotation wrt Roll.
 grang = alpha3 + tmp_a3;  % ad-hoc correction for detector mismatch (acis vs. hrcs w/ letg)
      }


      if (detnam == "ACIS-I") % This is a dummy placeholder
      {
 message("This Mode is not Supported.\n");
 return -1;
      }
      if (detnam == "HRC-I")  % This is a dummy placeholder
      {
 message("This Mode is not Supported.\n");
 return -1;
      }
   }
   if (grelem == "NONE")
   {
      message("This is not a grating dataset.\n");   % This should not happen.
      return -1;
   }
```

```
%
% Do a quick sanity check to see if the right grating type is selected.
%
% gflag == 12 if the grating modes are 1 (HEG) or 2 (MEG),
%    or == 3  if the grating mode is 3 (LEG).
%
   if ((gflag == 12 and gtype == "l") or (gflag == 3 and (gtype == "h" or gtype == "m")))
   {
      message("Grating Type Mismatch. Please check your brain.\n");
      return -1;
   }

%
% Initial Guess (x0,y0) from _TARG coordinate
%
% Now read in the raw data points and re-rotate back such that
% the transfer streak is always vertical across the field in the resulted map.
%
   if (detnam == "ACIS")
   {
      (x,y,en,cy)=fits_read_col(fitsfile + fitssub,"x","y","energy","CHIPY");
   }
   if (detnam == "HRC")
   {
      (x,y,cy)=fits_read_col(fitsfile + fitssub,"x","y","CHIPY");
      en = x * 0.0 + 2000.00; % Fudge the energy column since HRC doesn't have one.
   }
   if (ftype == 1)
   {

      vmsg(2, "Assuming that the RA_TARG and DEC_TARG values are correct\n");
      vmsg(2, "  for this particular target.                             \n");

      if (detnam == "ACIS")
      {
         gx = -(atan((ratg - ranom)*deg_rad)/(acis_plate))*cos(denom*deg_rad)+rarf;
 gy =  (atan((detg - denom)*deg_rad)/(acis_plate))+derf;
      }
      if (detnam == "HRC")
      {
 gx = -(atan((ratg - ranom)*deg_rad)/(hrc_plate ))*cos(denom*deg_rad)+rarf;
 gy =  (atan((detg - denom)*deg_rad)/(hrc_plate ))+derf;
      }
   }
   if (ftype == 2)
   {
      vmsg(2, "Making the guess based on Median Sky X and Y values\n");

         gx = median(x); gy= median(y);
   }
```

```
   if (ftype == 3)
   {

      if (detnam == "ACIS")
      {
         vmsg(2, "Assume dumb coordinate values [4096.5, 4096.5]\n");
         gx = 4096.5; gy = 4096.5; % default dummy x,y values
      }
      if (detnam == "HRC")
      {
         vmsg(1, "Assume dumb coordinate values [32768.5,32768.5]\n");
 gx = 32768.5; gy = 32768.5; % default dummy x,y values
      }
   }
   if (ftype == 4)
   {
         vmsg(2, "User specified values will be used\n");
         gx = x0; % user specified values.
 gy = y0; %
   }



%
% Define the REGION BOX for filtering events.
%

   variable boxw_strc =  80.0;  % Box width for streak region
   variable boxl_strc = 2000.0;  % Box length for streak region

   variable boxw_garm =   50.0;  % Box width for grating arm region
   variable boxl_garm;  % Box length for grating arm region
   variable bin_garm;  % Bin size along X
   variable rad_garm  =    50.;  % Circle Radius for grating arm region
   variable rad_strc  =   50.0;  % Circle Radius for streak region


% Check the array usage

   subarr = max(cy) - min(cy);

   if (subarr <= 768)  % if 3/4 or less CCD area is used (subarray)
   { % then use a smaller radius of circle for streak.
      rad_strc *= 0.4;
      boxw_strc *= 0.5;
   }


%
% Set grating arm region
%

   if (detnam == "ACIS")
   {
```

```
      boxl_garm = 5000.0;
      bin_garm  =     20;
   }
   if (detnam == "HRC")
   {
      boxl_garm = 12000.0;
      bin_garm =      500;
   }



%
% Control the display box size, etc.
%
   variable disp_n_elem = 50000;  % Number of elements displayed.
   variable disp_bx_size = 1000;        % Define the range of (X,Y) box size
% in the plot display
%
% Misc. filtering (energy cut, significance setting for data clipping)
%
   variable n_del    =  1.3;  % significance setting for array clipping
   variable n_eng_hi = 4000;  % energy filtering (high threshold)
   variable n_eng_lo =  500; % energy filtering (low threshold)

%
% and that's it. Return the parameter values required for findzo.
%

   return, gx, gy, x, y, en, grang, rotang, detnam, strc, boxw_strc, boxl_strc, rad_strc,
boxw_garm, boxl_garm, rad_garm, bin_garm, disp_bx_size, disp_n_elem, n_del, n_eng_hi, n_eng_lo;
}





%
% Main Routine:
%
%------------------------------------------------------------------------
% findzo:
%
% A S-Lang function to determine the position of zeroth-order
%  centroid (CHANDRA/HETGS and LETGS specific) by determining
% the intersection of grating arms and data transfer streak
% (or fine support structure diffraction pattern in LETG/HRC).
%
%
% Usage: (x,y) = findzo(fits_event_file,gtype [,ftype, x0, y0]);
%
% where:
%
%  gtype == h, m or l.
%  ftype == 1, 2, 3, or 4.
```

```
%
%    1: use RA_ and DEC_TARG to derive the guesstimated source position [gx, gy]
%       2: use the median values as [gx, gy]
%          3: use the dumb guess [x0,y0] as the initial guesses for [gx, gy]
%          4: use the user specified coordinate [x0, y0] as [gx, gy]
%

define findzo( )
{

   vmsg(1, "Running TG_FINDZO version "+ string(_version[0])+
"."+string(_version[1])+"."+string(_version[2])+"\n");
%
% here is one hack for isis 1.2.6 or later.
%

   set_fit_method("marquardt;delta=1.0e-12");

%
% Initialize Variables
%
   variable fitsfile, gtype, ftype;
   variable gx, gy;
   variable x0, y0;
   variable boxl_strc, boxw_strc, rad_strc;
   variable boxl_garm, boxw_garm, rad_garm, bin_garm;
   variable detnam="INSTRUME";
   variable deg_rad=3.1416/180.;
   variable en, x, y, xp, yp, xpp, ypp;
   variable sxp, syp, sxp2, syp2, cnt, cnt2;
   variable cltmp, l, ll, mxp, myp, strx, slp, islp, zx, zy;
   variable nx, ny;
   variable disp_bx_size, disp_n_elem;
   variable n_del, n_eng_hi, n_eng_lo;
   variable p1, p2;
   variable strc;

%
% note on angles: positive angle == clockwise rotation angle.
%

   variable grang, rotang;

%
% Read input from command line
%

   if ((_NARGS < 2) or (_NARGS == 4) or (_NARGS > 5))
   {
      vmsg(1, "USAGE: (x, y) = findzo(fits_file,\"l\",[guessing_mode,gX,gY]);\n");
      vmsg(1, "    where \"l\" (LEG) can be replaced with \"h\" (HEG)\n"  );
      vmsg(1, "    or \"m\" (MEG). guessing_mode, gx and gy are optional.\n"       );
      return -1;
   }
```

```
   if (_NARGS ==2 )
   {
      (fitsfile,gtype)=();
      ftype = 1;
      x0=4096.5; y0=4096.5;  % These are dummy values. Not actually used.
   }

   if (_NARGS ==3 )
   {
      (fitsfile,gtype,ftype)=();        % here, ftype can be any integer [1,2,3, or 4].
      if (ftype == NULL)
      {
         vmsg(1,"ftype = NULL invalid\n");
         return -1;
      }
      x0=4096.5; y0=4096.5;             % Initial guesses. Will not be used.
   }

   if (_NARGS ==5 )
   {
      (fitsfile,gtype,ftype,x0,y0)=();  % here ftype should be 4. [x0, y0] needs
% to be specified.
   }



%
% Now allowing to accept the use of cfitsio extended name syntax.
%
%
% The extention block name [EVENTS] should always be used for Chandra Event files.
%

   variable tmp_name = strchop(fitsfile,'[',0);
   variable fitsname = tmp_name[0];
   variable fitssub = "[EVENTS]";

   if ( length(tmp_name) > 1 )
   {
      tmp_name = tmp_name[[1:length(tmp_name)-1]]; % remove the first element since
                                                   % they should be a fits filename

%
% Dummy proof
%
      if ((strup(tmp_name[0]) == "EVENTS]") or (strup(tmp_name[0]) == "1]"))
      {
 vmsg(2,"                                             \n");
         vmsg(2,"#############################################\n");
         vmsg(2,"# Please do not specify the extention name. #\n");
         vmsg(2,"#############################################\n");
 vmsg(2,"                                             \n");
         tmp_name = tmp_name[[1:length(tmp_name)-1]];
```

```
      }
%
%
      vmsg(2,"                                                                  \n");
      vmsg(2,"The extention block name is always set to be [EVENTS] by default. \n");
      vmsg(2,"                                                                  \n");

      if ( length(tmp_name) > 0 )
      {
         fitssub  = fitssub + "[" + strjoin(tmp_name,"[");
      }
   }


%
% Initialize the required parameters specific to this configuration.
%
% gx, gy == guesstimated coordinate position of the source (in SKY)
% x, y   == event data arrays
% en     == event energy
% grang  == grating arm angle
% rotang == ajusted ROLL_NOM angle
% detnam == DETECTOR name.
% strc   == Streak Correction Factor
% box[w,l]_strc == Region Box Size for transfer streak (for fitting)
% rad_strc == Circle of radius exclusion zone around the zeroth order image.
% box[w,l]_garm == Region Box Size for grating arms (for fitting)
% rad_garm == Circle of radius exclusion zone around the zeroth order image.
% bin_garm == binning width for histogramming.
% disp_bx_size == display box size to show the fidelity of fit
% disp_n_elem == a total number of events shown in the pgplot screen.
% n_del = a level of statistical significance used in filtering (e.g., clip_array).
% n_eng_[hi,low] == Upper and lower threshold levels for energy filtering.
%
   (gx, gy, x, y, en, grang, rotang, detnam, strc, boxw_strc, boxl_strc, rad_strc,
         boxw_garm, boxl_garm, rad_garm, bin_garm, disp_bx_size, disp_n_elem, n_del,
         n_eng_hi, n_eng_lo) = gparam_io(fitsname, fitssub, gtype, ftype, x0 ,y0);

%
% Rotate the SKY event array (X,Y) by (-rotang) degree around (gx,gy).
% This will make the transfer streak go vertical in the new frame.
%
   (xp,yp)=rotxy(x,y,gx,gy,-rotang*deg_rad);

%
% And draw a first region box to isolate transfer streak events
% without zero order and with energy filtering.
%
   l=where(xp > (gx-boxw_strc/2) and xp < (gx+boxw_strc/2) and
   yp > (gy-boxl_strc/2) and yp < (gy+boxl_strc/2) and
   x > 0.0            and y    > 0.0         and
           hypot(xp-gx,yp-gy)>rad_strc and (en > n_eng_lo and en < n_eng_hi));

%
```

```
% Fit the streak profile
%
   (myp,mxp) = median_peak(yp[l], xp[l],20,n_del);

   (slp,) = garm_fit(myp,mxp,0.0);
   islp = 0;
   while (islp != slp[0])
      {
         islp=slp[0];
         cltmp=clip_array(mxp-myp*slp[1]+slp[0],2); % loosely filtered here. hardcode OK.
         (slp,)=garm_fit(myp[cltmp],mxp[cltmp],0.0);
      }
   strx = slp[0]+strc;


%
% Save some parameter values for later use
%
   sxp=strx;syp=gy;sxp2=0.;syp2=0.;cnt=0; cnt2=0;

%
% Draw the events in the selected region box
%

   pdev(1);
   connect_points(0);
   ll=where(x > 0.0 and y > 0.0);
   xrange(sxp-disp_bx_size,sxp+disp_bx_size);
   yrange(syp-disp_bx_size,syp+disp_bx_size);
   plot(xp[ll[[0:disp_n_elem]]],yp[ll[[0:disp_n_elem]]]);

%
% here, the while loop, is to check whether the position
% solution points to the same place or not. Until it does,
% it'll keep runing (til 20th cycle).
%
   while ((sxp2 != sxp or syp2 != syp) and (cnt < 20))
   {
      sxp2=sxp; syp2=syp;

%
% Rotate the region box again to rectify the selected grating arm.
%
      (xpp,ypp)=rotxy(xp,yp,sxp,syp,-grang*deg_rad);

%
% Then index the events in the newly selected region box.
% Here, the energy filtering is hardcoded (and the user
% does not need to control this).
%
      l=where(xpp>(sxp-boxl_garm/2) and xpp<(sxp+boxl_garm/2) and
              ypp>(syp-boxw_garm/2) and ypp<(syp+boxw_garm/2) and
        x > 0.0           and y > 0.0           and
              hypot(xpp-sxp,ypp-syp)>rad_garm and (en < 8000.));
```

```
%
% Call median_peak function.
%
% The output are the point array (mxp,myp) (median point along the arm).
%
      (mxp,myp)=median_peak(xp[l],yp[l],bin_garm,n_del);

%
% Fit the vector (mxp, myp).
%
       (slp,)=garm_fit(mxp,myp,-grang*deg_rad);

%
% Do interative filtering; this may be redundant.
%

      cltmp=clip_array(myp-mxp*slp[1]+slp[0],2); % again, hardcode OK
      (slp,)=garm_fit(mxp[cltmp],myp[cltmp],-grang*deg_rad);
      islp=slp[1];

      variable flag=1;
      while (flag and cnt2 < 20)
      {
         cltmp=clip_array(myp-mxp*slp[1]+slp[0],2); % hardcode ok.
 (slp,)=garm_fit(mxp[cltmp],myp[cltmp],-grang*deg_rad);
 if (slp[1] == islp)
 {
    flag = 0;
 }
 islp=slp[1];
 cnt2++;
      }

      if (cnt2 == 20)
      {
         vmsg(5,"the slope solution did not converge after "+string(cnt2)+" trials.\n");
      }

      connect_points(1);
      sxp = strx; syp = slp[1]*sxp+slp[0];
      cnt++;
   }

%
% Plot out the resulting line fits.
%
   oplot([strx,strx],[min(yp),max(yp)]);
   oplot(mxp,mxp*slp[1]+slp[0]);

%
% Zoom in onto the zeroth order region.
%
   pdev(2);
```

```
   connect_points(0);
   xrange(sxp-disp_bx_size*0.15,sxp+disp_bx_size*0.15);
   yrange(syp-disp_bx_size*0.15,syp+disp_bx_size*0.15);
   ll=where(x > 0.0 and y > 0.0);
   plot(xp[ll[[0:disp_n_elem]]],yp[ll[[0:disp_n_elem]]]);
   connect_points(1);
   oplot([strx,strx],[min(yp),max(yp)]);
   oplot(mxp,mxp*slp[1]+slp[0]);
   pdev(3);

%
% Return error message if the fit did not converge.
%
   if (cnt == 20)
   {
      vmsg(5,"the position solution did not converge after "+string(cnt)+" trials.\n");
   }

%
% Rotate back to the original [X,Y] coordinate frame.
%
   (zx,zy)=rotxy([sxp],[syp],gx,gy,rotang*deg_rad);

%
% return the values. DONE.
%

   vmsg(5, "Derived zeroth order position: X0 = "+string(zx[0])+" Y0 = "+string(zy[0])+" \n");
   return zx[0], zy[0];
}


provide("findzo");
provide("set_findzo_verbose");
provide("set_findzo_plot_dev");

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

TG_FINDZO:

```
#! /usr/bin/env isis-script
%
% This file is part of tg_find (finding zeroth order).
% Copyright (C) 2006 Massachusetts Institute of Technology
%
% This software was developed by the MIT Center for Space Research under
% contract SV3-73016 from the Smithsonian Institution.
%
% This program is free software; you can redistribute it and/or modify
% it under the terms of the GNU General Public License as published by
% the Free Software Foundation; either < version number > of the
% License, or (at your option) any later version.
%
% This program is distributed in the hope that it will be useful, but
% WITHOUT ANY WARRANTY; without even the implied warranty of
% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
% General Public License for more details.
%
% You should have received a copy of the GNU General Public License
% along with this program; if not, write to the Free Software
% Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA
%
%------------------------------------------------------------------------
%
% File: tg_findzo
%
% Version:
%
% Authors:  Bish Ishibashi
% Date Created: 23 October 2006
%
%
% HISTORY:
%  This script is designed as a prototype.
%
% October     23, 2006:     Semi-official introduction of the tool
%
% December    27, 2006:   A small bug fix (non-CIAO part)
%
%========================
%
% Purpose: This program is a s-lang wrapper for executing findzo.sl.
%
% Syntax:  tg_find evt_file_name grating_type [guessing_mode, gx, gy]
%
% Input:
%    evtfile_name == Event L1.5 or 2 file.
%    grating_type == "h" (HEG), "m", or "l" (LEG).
%    guessing_mode == 1 = astrometry based
%     2 = median filter based
%     3 = dumb guess ([gx,gy]=[4096.5,4096.5])
%     4 = user specified guesses (sky gx and gy)
```

```
%    gx, gy == user-specified guess for source's sky position
%
% Example
%  by default, guessing_mode == 1:
%
%    $> tg_findzo acisf0001N0001_evt1.fits m
%
% Or try something fancier. When using CFITSIO filtering,
%       do not specify the extention block name since it is set to
%       [EVENTS] by default (always true for Chandra Event files).
%
%    $> tg_findzo acisf0001N0001_evt1.fits[energy=2000:8000] h 4 4090.2 4061.72
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Version: commented out for now.
%
% private variable _version = [0, 9, 1 ]   ; % major, minor, patch
%



%
% Load the s-lang script findzo.sl first. The tool has to be
% available somewhere in your ISIS path.
%

require("findzo");


%
% Check for existence of CIAO environment; if present, load paramio
%
#if$ASCDS_INSTALL


%+ paramio dependencies:

add_to_isis_load_path( getenv("ASCDS_INSTALL") + "/share/slsh/local-packages" );

if ( ( _slang_version / 10000 ) >= 2 )
   add_to_isis_module_path( getenv("ASCDS_INSTALL") + "/lib/slang/v2/modules" ) ;
else
   add_to_isis_module_path( getenv("ASCDS_INSTALL") + "/lib/slang/modules" ) ;

%-

%
% load paramio
%
require("paramio");


%
```

```
% Parameter setting
%

private variable fp = paramopen(NULL, "rw", __argv);

if (fp == NULL)
   verror("\n %% %S: No par file, or param specification error.\n", path_basename(__argv[0]));

private variable infile, grttype, fltype, ix, iy, pl_device, verbose;

infile  = pget(fp, "infile");
grttype = pget(fp, "grttype");
fltype  = pgeti(fp, "fltype");
ix      = pgetf(fp, "ix_value");
iy      = pgetf(fp, "iy_value");
pl_device = pget(fp, "pl_device");
verbose = pgeti(fp, "verbose");


%
% pass on verbose and pl_device variables onto findzo.sl
%

set_findzo_verbose(verbose);
set_findzo_plot_dev(pl_device);


%
% Run findzo
%
variable ox, oy;
(ox,oy) = findzo(infile,grttype,fltype,ix,iy);


%
% Obtain output (x,y) values and dump them into a parameter file
%
pset(fp, "ox_value", ox);
pset(fp, "oy_value", oy);
paramclose(fp);



%
% If CIAO is not available do the following
%
#else

#ifnexists atof
private define atof (x)
{
   () = sscanf (x, "%lf", &x);
   return x;
}
```

```
#endif

#ifexists __argc
#ifndef __INTERACTIVE__

   switch (__argc)
      { case 3 :
         findzo(__argv[1],__argv[2]);
 exit(0);
      }
      { case 4 :
         findzo(__argv[1],__argv[2],atof(__argv[3]));
         exit(0);
      }
      { case 6:
         findzo(__argv[1],__argv[2],atof(__argv[3]),atof(__argv[4]),atof(__argv[5]));
         exit(0);
      }
      {
         findzo();
         exit(0);
      }
#endif % ifndef __INTERACTIVE__
#endif  % ifexists __argc


#endif % else
```

TG_FINDZO.PAR:

```
#
# parameter file for tg_findzo
#
infile,f,a,"",,,"Input Event 1.5 or 2 file"
grttype,s,a,"h",h|m|l,,"Grating Type"
fltype,i,a,1,1,4,"Guessing Mode"
#
ix_value,r,l,4096.5,0,10000,"X Guess value"
iy_value,r,l,4096.5,0,10000,"Y Guess value"
ox_value,r,l,0,0,10000,"X value"
oy_value,r,l,0,0,10000,"Y value"
#
pl_device,s,a,"/cps",/cps|/xwin,,"Plotting Mode (default to /cps)"
verbose,i,a,5,0,5,"verbosity"
mode,s,h,"hl",,,"paramio mode flags"
```

TG_FINDZO.HELP:

```
SUBJECT(tg_findzo)

SYNOPSIS

    Derive zeroth order position using grating arms and data transfer
    streak (HETGS) or coarse diffraction pattern (LETGS).

SYNTAX

    tg_findzo infile grttype fltype [ix_value] [iy_value] [pl_device]
    [verbose]


DESCRIPTION

    Tg_findzo derives the intersection position for a specified grating
    arm and data transfer streak (or coarse diffraction pattern in LETGS
    + HRC-S configuration). This tool is designed to supplement another
    zero order detection routine tgdetect, which would likely fail to
    identify the source position properly when the source is either
    blocked out or extremely piled.

    Tg_findzo is a S-Lang wrapper script which executes findzo.sl, which
    is a main driver for identifying the pattern signals to lock onto
    and then derive their intersections in the SKY coordinate.

    As mentioned above, tg_findzo is extremely useful in cases where
    the zeroth order image is blocked or the source is so bright that
    the effect of pileup severely distorts its profile (e.g., a crater
    in the middle of its PSF). Since the tool requires grating arms and
    data streaks to lock onto, the tool is not ideal for identifying
    the zeroth order position for faint sources. It is also not ideal
    for the crowded field. This tool does not support CC mode.

    To locate the source, the tool uses the coordinate given by FITS
    header keywords RA_TARG and DEC_TARG by default (fltype == 1).
    The users may choose to apply median filters on input events
    to localize the brightest feature in the infile (fltype == 2),
    or just apply a dumb guess (SKY [X,Y] = [4096.5, 4096.5] for
    HETGS+ACIS-S or = [32768.5, 32768.5] for LETGS+HRC-S; fltype == 3).
    Alternatively, it also allows a user specified guess to be used
    (fltype == 4 with [X,Y] = [ix_value, iy_value]).

    By default, the tool will dump the graphic output file "findzo.ps"
    which allows users to check the fidelity of the zeroth order position
    derived with the tool.

    This tool supports the following grating + detector configurations:
    HETGS + ACIS-S, LETGS + HRC-S, and LETGS + ACIS-S.

EXAMPLES
```

EXAMPLE 1

```
tg_findzo acisf0001N0001_evt1.fits m
```

This is an example of running tg_findzo to identify the zeroth order position for MEG using RA_TARG and DEC_TARG coordinate values.

EXAMPLE 2

```
tg_findzo acisf0001N0001_evt1.fits h 3
```

This is an example of running tg_findzo to identify the zeroth order position for HEG by assuming that the source is near the SKY coordinate [X,Y] = [4096.5, 4096.5].

EXAMPLE 3

```
tg_findzo "acisf0002N0001_evt2.fits[energy=2000:8000]" l 4 4092.21 4061.72
```

This is an exaple of running tg_findzo for LETGS + ACIS-S configration using a user specified coordinate [X,Y] = [4092.21, 4061.72] and simple CFITSIO event filtering.

PARAMETERS

DETAILED PARAMETER DESCRIPTIONS

1. PARAM infile

   type=string
   filetype=input
   reqd=yes
   stacks=no

   Input grating event file (L1, L1.5, or L2).

   The tool will accept only a grating event file. No stack is allowed. A simple use of CFITSIO event filtering is allowed.

2. PARAM grttype

   type=string
   reqd=yes

   Enter Grating Arm of your choice.

   Name of the grating that the zeroth order position will be computed for. Valid values are h, m, and l (HEG, MEG and LEG, respectively).

3. PARAM fltype

   type=integer
   def=1
   reqd=yes

   Enter a choice of flag (mode) for an initial guess on the zeroth order position.

   Choose which type of an initial guess to use for computation. Valid choices are 1 through 4 (integer). By default, fltype is set to 1 which utilizes the astrometric information in the FITS header (e.g., RA_TARG and DEC_TARG).

   The available flags are:

   o) "1" Utilize the astrometric coordinate values for the target source (default).

   o) "2" Utilize median filter to identify the brightest source in the field.

   o) "3" Apply a hard-coded dumb guess ([X,Y] = [4096.5,4096.5] for ACIS-S and = [32768.5, 32768.5] for HRC-S).

   o) "4" Utilize a user specified position as an initial guess.

4. PARAM ix_value

   type=float
   units=pixels

   Enter initial guess value for SKY X.

   This value is used only when fltype == 4.

5. PARAM iy_value

   type=float
   units=pixels

   Enter initial guess value for SKY Y.

   This value is used only when fltype == 4.

6. PARAM ox_value

   type=float
   units=pixels

   Computed zeroth order position value for SKY X.

7. PARAM oy_value

type=float
units=pixels

Computed zeroth order position value for SKY Y.

8. PARAM pl_device

   type=string
   def=/cps

   Enter the plotting mode.

   When running from a command line, users are encouraged to use the default value "/cps" so that the tool will dump the graphic output "findzo.ps" for examining the fidelity of the zeroth order location derived with the tool. The other option, "/xwin", may be used in the interactive mode in ISIS.

9. PARAM verbose

   type=integer
   def=5
   min=0
   max=5

   Verbosity.

   The verbose parameter is used to set the level of output from tg_findzo.

10. PARAM mode

    type=string
    def=hl

    Enter mode for parameter file.

BUGS

While no bugs are known, the tool is not certainly a dummy-proof.

VERSION

```
FINDZO.SL        0.9.1
TG_FINDZO        0.9.1
```

LAST MODIFIED

NOVEMBER 2006