Specifications: tgfindzo Algorithm and Interface

David Huenemoerder

December 13, 2011

Revision History

1. 2011.12.13 version 1.0; corrections from JC; and more from DPH prototype (§ 4 steps 25, 26, & 30).
2. 2011.09.26 Initial version 0.1

1 Introduction

The zeroth order of a Chandra grating observation defines the origin of the dispersed spectrum’s wavelength scale. Hence, accurate determination of the zeroth order centroid is a fundamental step of the pipeline processing. The ACIS detector, however, can become saturated by bright sources. The brighter the source, the greater the distortion of the zeroth order due to rejection of core events, eventually leading to a “cratered” central region with few or no events. To mitigate telemetry saturation for observations of extremely bright sources, ACIS itself is sometimes configured to exclude the zeroth order from telemetry.

The current CIAO program for determining the zeroth order centroid, tgdetect (based on the celldetect program), will fail to give an accurate result for such distorted or blocked zeroth orders. In these cases, we rely on an alternate method which determines the intersection of a grating order with the ACIS CCD frame shift streak. This program is called findzo and is external to CIAO.

Background material on the ISIS script, findzo, can be found at the following URLs (including a full source listing):

- findzo web page: <http://space.mit.edu/cxc/analysis/findzo/>
- findzo memo: <http://space.mit.edu/cxc/analysis/findzo/Data/memo_fzero_1.4.ps.gz>

Here we specify the interface and algorithm in detail for development of an equivalent CIAO program. The specifications which follow supercede the above referenced code; we have removed unnecessary parts, referred to relevant calibration data files, defined the output as a standard data product, and tried to use longer, more descriptive variable names.

1.1 Scope

The new program specified here, tgfindzo, applies to Chandra Level 1 or 2 event files obtained with a grating instrument (HETG or LETG) in conjunction with the ACIS-S detector. Use with HRC-I, HRC-S, or ACIS-I is not specified. The output product will be a standard CIAO FITS product compatible with CIAO tools, in particular, tg_create_mask, so that it can be used in pipeline processing as well as by general
users. Output will also contain a CIAO FITS region for use with ancillary programs which can read that format (such as ds9).

2 Interface

The interface will be the standard CIAO parameter i/o with the following parameters:

infile (string) Input event file name;

outfile (string) Output source table file name;

(zo_pos_x) (real) Initial guess for the sky-\(x\) position (default: \(x\) pixel corresponding to RA_TARG, DEC_TARG);

(zo_pos_y) (real) Initial guess for the sky-\(y\) position (default: \(y\) pixel corresponding to RA_TARG, DEC_TARG).

(Note: as per paramio conventions, parameters are required as inputs unless they are enclosed in parentheses, in which case they are optional or have defaults. Other standard program-control parameters (such as mode, verbose) are not listed.)

Some features of the findzo.sl script interface have been removed, specifically the option to use either HEG or MEG type grating for HETG, and the "guessing mode".

3 Data Product

3.1 HDU Components

The primary HDU will be a null PRIMARY with standard Chandra minimal header components. There will be one extension with a standard Chandra header and the source list as a binary table. The table and header will have fields required by tg_create_mask as well as a FITS region. Some header keywords and fields present in current src1a files will be omitted, since they are specific to the celldetect algorithm and are not required by tg_create_mask.

The following table describes the file structure by Header-Data Unit number, type, extension name, content, and HDU classes. An asterisk (*) denotes the principal HDU.

<table>
<thead>
<tr>
<th>HDU</th>
<th>HDU Type</th>
<th>EXTNAME</th>
<th>EXTVER</th>
<th>CONTENT</th>
<th>HDCLASS</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>PRIMARY</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>NULL</td>
</tr>
<tr>
<td>1 (*)</td>
<td>BINTABLE</td>
<td>SRCLIST</td>
<td>1</td>
<td>TGSRC</td>
<td>OPGIP</td>
<td>Binary table extension source coordinates and region.</td>
</tr>
</tbody>
</table>
3.2 Columns

Columns for the source list are given in the following table.

<table>
<thead>
<tr>
<th>TTYPE</th>
<th>TUNIT</th>
<th>TFORM</th>
<th>TLMIN</th>
<th>TMAX</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RA</td>
<td>deg</td>
<td>1D</td>
<td>0</td>
<td>360</td>
<td>right ascension</td>
</tr>
<tr>
<td>DEC</td>
<td>deg</td>
<td>1D</td>
<td>-90</td>
<td>90</td>
<td>declination</td>
</tr>
<tr>
<td>X</td>
<td>pixel</td>
<td>1D</td>
<td>0.5</td>
<td>N/A</td>
<td>zeroth order sky $x$ centroid</td>
</tr>
<tr>
<td>Y</td>
<td>pixel</td>
<td>1D</td>
<td>0.5</td>
<td>N/A</td>
<td>zeroth order sky $y$ centroid</td>
</tr>
<tr>
<td>NET_COUNTS</td>
<td>count</td>
<td>1E</td>
<td>0</td>
<td>N/A</td>
<td>Counts in the zeroth order region</td>
</tr>
<tr>
<td>NET_RATE</td>
<td>count/s</td>
<td>1E</td>
<td>0.0</td>
<td>N/A</td>
<td>count rate in the zeroth order</td>
</tr>
<tr>
<td>SHAPE</td>
<td>N/A</td>
<td>10A</td>
<td>N/A</td>
<td>N/A</td>
<td>Region descriptor: shape of the region; usually CIRCLE</td>
</tr>
<tr>
<td>R</td>
<td>PIXEL</td>
<td>2E</td>
<td>0</td>
<td>N/A</td>
<td>Region descriptor: generalized radii, for each dimension of $R$. (Here, $[z_0, \text{radius}, 0]$.)</td>
</tr>
<tr>
<td>ROTANG</td>
<td>deg</td>
<td>1E</td>
<td>0</td>
<td>180</td>
<td>Region descriptor: rotation angle of source region (0.0 for a circle).</td>
</tr>
<tr>
<td>COMPONENT</td>
<td>N/A</td>
<td>1J</td>
<td>1</td>
<td>N/A</td>
<td>Region descriptor: serial index of region components (typically 1).</td>
</tr>
<tr>
<td>TG_SRCID</td>
<td>N/A</td>
<td>II</td>
<td>1</td>
<td>N/A</td>
<td>Zeroth order serial source count (typically 1).</td>
</tr>
</tbody>
</table>

3.3 Relevant Header Keywords

The header shall contain all the standard *Chandra* header components as defined by the CXC FITS Guide (e.g., mandatory FITS, configuration control, timing, observation information, coordinate systems, table column descriptors). New keywords output by `tgfindzo` are:

- **COUNT_ST**: the counts in the ACIS CCD frame-shift streak.
- **COUNT_TG**: the counts in the grating arm.

3.4 Relationships with Other Interfaces

Output of `tgfindzo` is required to be read by `tg_create_mask`, which requires the $x$, $y$ sky pixel values, TG_SRCID, and header keywords GRATING, ROLL_NOM, and other standard *Chandra* keywords.

Region information is useful to easily display the location of the zeroth order on a sky image (such as with ds9).
4 Algorithm

Here we step through the algorithm from start to finish in pseudo-code, adapted from the current findzo.sl script. We omit intermediate output, algorithmic components which are unnecessary, we simplify some parts, and we use longer, more descriptive names for variables. Standard infrastructure components (such as FITS file input/output, pixlib coordinate transformations, or CALDB interfaces are not described). In the pseudo-code below, we use array arithmetic and assume that it is clear from context which are array values. This is, of course, a language dependent feature, and may require loop constructs to implement.

4.1 Streak Processing

1. Read configuration from header:

   GRATING = HETG | LETG | NONE
   if NONE, error

   INSTRUME = ACIS | HRC
   if HRC, error.
   if ACIS, read DETNAM to see if aimpoint is on ACIS-I or ACIS-S
   if ACIS-I, error.

   READMODE
   if not TIMED, error.

   ROLL_NOM [deg]

2. Read grating angle alpha from CALDB geom file:

   default/geom/telD1999-07-23geomN0006.fits[GRATINGS]

   alpha (where INSTRUMENT==instrument, GRATING==grating, GRATING_ARM==(MEG|LEG))

   alpha_meg
   alpha_leg

3. Define some constants:

   streak_offset_meg = -0.011
   streak_offset_leg = 0.000

   acis_s_rotation_corr_meg = 0.03
   acis_s_rotation_corr_leg = 0.00

   alpha_corr_meg = -0.006
   alpha_corr_leg = 0.00

   pixel_tolerance = 1.e-4 /* threshold for convergence to a pixel */

4. Read coords RA_TARG, DEC_TARG (if absent, read RA_NOM, DEC_NOM).

   ra [ deg ]
   dec [ deg ]

5. Compute the rotation angle which will make the readout streak vertical. This is contingent upon grating type. (We’re only considering ACIS and MEG or LEG rotations).
Set variables as per grating type:

\[
\begin{align*}
\alpha & : \alpha_{\text{meg}} \quad | \quad \alpha_{\text{leg}} \\
\alpha_{\text{corr}} & : \alpha_{\text{corr\_meg}} \quad | \quad \alpha_{\text{corr\_leg}} \\
streak_{\text{offset}} & : \text{streak\_offset\_meg} \quad | \quad \text{streak\_offset\_leg} \\
\text{acis\_s\_rotation\_corr} & : \text{acis\_s\_rotation\_corr\_meg} \quad | \quad \text{acis\_s\_rotation\_corr\_leg}
\end{align*}
\]

\[\text{rotang} = \text{ROLL\_NOM} + \text{acis\_s\_rotation\_corr} ;\]

\[\text{grating\_angle} = \alpha + \alpha_{\text{corr}} - \text{acis\_s\_rotation\_corr} \]

6. Read events:

\[X, \; Y, \; \text{ENERGY}, \; \text{CHIPY}\]

7. If coordinates were specified by the parameters zo\_pos\_x, zo\_pos\_y, use those as initial guess. Otherwise (if blank, 0, or negative), use the header coordinates and convert to sky \((x, y)\) pixels.

If using the header coordinates, guess the zeroth order location by converting the RA\_TARG, DEC\_TARG to \(x, y\) coords:

\[(zo\_pos\_x, zo\_pos\_y) \Rightarrow f(\text{RA\_TARG, DEC\_TARG})\]

8. Define a priori region definition parameters (specific to ACIS). If a sub-array is in use, then decrease some of the values:

\[
\begin{align*}
\text{box\_width\_streak} & = 80.0 \\
\text{box\_length\_streak} & = 2000.0 \\
\text{box\_width\_grating} & = 80.0 \\
\text{box\_length\_grating} & = 5000.0 \\
\text{bin\_size\_grating} & = 20 \\
\text{bin\_size\_streak} & = 20 \\
\text{radius\_grating} & = 50.0 \\
\text{radius\_streak} & = 50.0 \\
\text{radius\_zo} & = 30.0
\end{align*}
\]

\[
\text{if ( max(CHIPY) - min(CHIPY) < 768 ) /* sub-array in use */}
\text{radius\_streak} *= 0.4
\text{box\_width\_streak} *= 0.5
\]

9. Define some additional constants::

\[
\begin{align*}
\text{clipping\_factor} & = 1.3 \quad % \text{empirically determined} \\
\text{energy\_filter\_low} & = 500 \\
\text{energy\_filter\_high} & = 4000
\end{align*}
\]

10. Rotate the X,Y event sky coordinates around the target coords: (positive angle gives clockwise rotation of points):

\[(xrot\_streak, yrot\_streak) = \]

\[
\text{rotate} (X, Y, zo\_pos\_x, zo\_pos\_y, -\text{rotang} \times \text{PI}/180)
\]

11. Select streak events via geometry and energy, excluding zeroth order:

\[
\text{streak\_indices} = \text{where} (\]
\[
\begin{align*}
( \text{xrot\_streak} > ( \text{zo\_pos\_x} - \text{box\_width\_streak}/2 ) ) \\
\text{and} ( \text{xrot\_streak} < ( \text{zo\_pos\_x} + \text{box\_width\_streak}/2 ) ) \\
\text{and} ( \text{yrot\_streak} > ( \text{zo\_pos\_y} - \text{box\_length\_streak}/2 ) ) \\
\text{and} ( \text{yrot\_streak} < ( \text{zo\_pos\_y} + \text{box\_length\_streak}/2 ) )
\end{align*}
\]

5
and ( X > 0.0 )
and ( Y > 0.0 )
and ( hypot( xrot_streak-zo_pos_x, yrot_streak-zo_pos_y ) > radius_streak)
and ( ENERGY > energy_filter_low )
and ( ENERGY < energy_filter_high ))

12. Determine streak position (assumed vertical): Find the median in coarse bins (see median_peak() and clip_array() functions in original S-Lang script, findzo-dev.sl):

\[
y_{\text{streak}} = yrot_{\text{streak}}[\text{streak_indices}]
x_{\text{streak}} = xrot_{\text{streak}}[\text{streak_indices}]
\]

\[(y_{\text{median}}, x_{\text{median}}) = \text{median_peak}(y_{\text{streak}}, x_{\text{streak}}, \text{bin_size}_{\text{streak}}, \text{clipping_factor})\]

ERROR if no points to fit.

13. Find the mean of the x values:

\[x_{\text{fit}_{\text{streak}}} = \text{mean}(x_{\text{median}})\]

14. Iterate, by discarding 2σ outliers and repeating:

\[\text{old}_x = 0.0\]
\[\text{while } (\text{abs}(\text{old}_x - x_{\text{fit}_{\text{streak}}}) > \text{pixel}_\text{tolerance})\]
\[\text{old}_x = x_{\text{fit}_{\text{streak}}}\]
\[\text{indices} = \text{clip_array}(x_{\text{median}}, 2);\]
\[x_{\text{fit}_{\text{streak}}} = \text{mean}(x_{\text{median}}[\text{indices}]);\]

15. Repeat the filter to get streak counts (plus background):

\[\text{streak}_\text{indices} = \text{where}\]
\[ (xrot_{\text{streak}} > (x_{\text{fit}_{\text{streak}}} - \text{box}_\text{width}_{\text{streak}}/2))\]
\[\text{and}(xrot_{\text{streak}} < (x_{\text{fit}_{\text{streak}}} + \text{box}_\text{width}_{\text{streak}}/2))\]
\[\text{and}(yrot_{\text{streak}} > (\text{zo}_\text{pos}_y - \text{box}_\text{length}_{\text{streak}}/2))\]
\[\text{and}(yrot_{\text{streak}} < (\text{zo}_\text{pos}_y + \text{box}_\text{length}_{\text{streak}}/2))\]
\[\text{and}(X > 0.0)\]
\[\text{and}(Y > 0.0)\]
\[\text{and}(\text{hypot}(xrot_{\text{streak}}-x_{\text{fit}_{\text{streak}}}, yrot_{\text{streak}}-\text{zo}_\text{pos}_y) > \text{radius}_{\text{streak}})\]
\[\text{and}(\text{ENERGY} > \text{energy}_\text{filter}_{\text{low}})\]
\[\text{and}(\text{ENERGY} < \text{energy}_\text{filter}_{\text{high}}))\]

16. Save the number of streak counts (for output to tool param):

\[\text{streak}_\text{counts} = \text{length}(\text{streak}_\text{indices})\]
\[\text{*** output to header: } \text{COUNT}_{\text{ST}}\]

17. Apply empirical offset (compensates for MEG, HEG alignments):

\[x_{\text{fit}_{\text{streak}}} += \text{streak}_{\text{offset}}\]

### 4.2 Grating Arm Processing

Determine where the grating arm is. Similar to streak processing, only we rotate so grating trace is horizontal to filter, then fit in the streak-vertical coordinates.

18. Set initial guess for the \( y \) coordinate of zeroth order, and a saved previous value; set an iteration counter and its limit:
y_fit_zo = zo_pos_y
prev_y_fit_zo = 0.0 ;
num_iter = 0
max_iter = 20

outer iteration begin:

19. While the y_fit_zo value changes, iterate, up to max_iter tries:
   
   while( (num_iter < max_iter)
           and ( abs(y_fit_zo - prev_y_fit_zo) > pixel_tolerance ) )

20. Copy the y-value:

   prev_y_fit_zo = y_fit_zo

21. Rotate event coordinates (xrot_streak, yrot_streak) about the current center (y-coord is being iterated) so the grating arm is horizontal (starting with events rotated for vertical streak):

   ( xrot_grat, yrot_grat ) = rotate( xrot_streak, yrot_streak,
                                       x_fit_streak, y_fit_zo,
                                       -grating_angle*PI/180)

22. Index the grating arm events in the box region, excluding zeroth order and energies above a limit:

   grating_indices = where(
       xrot_grat > ( x_fit_streak - box_length_grating/2 )
       and xrot_grat < ( x_fit_streak + box_length_grating/2 )
       and yrot_grat > ( y_fit_zo - box_width_grating/2 )
       and yrot_grat < ( y_fit_zo + box_width_grating/2 )
       and X > 0.0
       and Y > 0.0
       and hypot(xrot_grat - x_fit_streak, yrot_grat - y_fit_zo) > radius_grating
       and (ENERGY < 8000.) )

23. Select the filtered events (in [xy]rot_streak coords):

   x_grat = xrot_streak[ grating_indices ]
   y_grat = yrot_streak[ grating_indices ]

24. Find the median of the grating arm in coarse bins:

   ( x_median, y_median ) =
   median_peak( x_grat, y_grat, bin_size_grating, clipping_factor )

25. Do a linear fit, with slope assumed to be 1.0. Really fitting $y = slope \times x + const$ with slope frozen to 1.0:

   angle = -grating_angle * PI / 180
   x_values = angle * x_median
   y_values = y_median
   y_intercept_guess = 4000.0 ;
   slope = 1.0
   y_intercept = fit_linear( x_values, y_values, y_intercept_guess, slope )
26. Iterate (within the outer iteration) on this by clipping > 2\(\sigma\) outliers from the resulting line (may want some max iteration limit, say 20 tries):

**inner iteration begin:**

\[
\text{prev}_y\text{_intercept} = y\text{_intercept} \\
\text{indices} = \text{clip\_array}(y\text{_values} - x\text{_values} + y\text{_intercept}, 2) \\
x\text{_values} = x\text{_values}[\text{indices}] \\
y\text{_values} = y\text{_values}[\text{indices}] \\
y\text{_intercept} = \text{fit\_linear}(x\text{_values}, y\text{_values}, y\text{_intercept}, \text{slope}) \\
\text{if} \ ((y\text{_intercept} - \text{prev}_y\text{_intercept}) < \text{pixel\_tolerance}) \ \text{break}
\]

**inner iteration end**

27. When done, compute the zeroth order \(y\) value:

\[
y\text{_fit\_zo} = y\text{_intercept} + \text{angle} \times x\text{_fit\_streak}
\]

**outer iteration end**

28. Count the grating events using the final coordinates:

\[
grating\text{\_indices} = \text{where}( \\
xrot\text{\_grat} > (x\text{\_fit\_streak} - \text{box\_length\_grating}/2) \\
\text{and} \ xrot\text{\_grat} < (x\text{\_fit\_streak} + \text{box\_length\_grating}/2) \\
\text{and} \ yrot\text{\_grat} > (y\text{\_fit\_zo} - \text{box\_width\_grating}/2) \\
\text{and} \ yrot\text{\_grat} < (y\text{\_fit\_zo} + \text{box\_width\_grating}/2) \\
\text{and} \ X > 0.0 \\
\text{and} \ Y > 0.0 \\
\text{and} \ \text{hypot(xrot\text{\_grat} - x\text{\_fit\_streak}, yrot\text{\_grat} - y\text{\_fit\_zo})} > \text{radius\_grating} \\
\text{and} \ (\text{ENERGY} < 8000.) )
\]

29. Save the number of counts in the grating-selected events (this is approximate, but will be useful for downstream heuristics which compare accuracy of findzo and celldetect):

\[
grating\text{\_counts} = \text{length}(grating\text{\_indices})
\]

*** output to header: COUNT\_TG

30. Rotate the values back to the original sky frame:

\[
(x\text{\_zo}, y\text{\_zo}) = \text{rotate}(x\text{\_fit\_streak}, y\text{\_fit\_zo}, \\
zo\text{\_pos\_x}, zo\text{\_pos\_y}, \\
\text{rotang} \times \text{PI} / 180)
\]

*** output to table

31. Convert coords:

\[
(\text{RA}, \text{DEC}) = f(x\text{\_zo}, y\text{\_zo}, ...)
\]

*** output to table

32. Determine the number of zeroth order counts by filtering in a circle centered on \(x\text{\_zo}, y\text{\_zo}\) (and other data needed to be compatible w/ the src1a file):

\[
zo\text{\_indices} = \text{where}(\text{hypot}(X - x\text{\_zo}, Y - y\text{\_zo}) < \text{radius\_zo})
\]

\[
zo\text{\_counts} = \text{length}(zo\text{\_indices})
\]

*** output to table