

NAME

deticpt - intercept rays at detector plane

DESCRIPTION

deticpt models the intersection of rays with a detector. The detector is made up of one or more elements, modelled as flat rectangles oriented at arbitrary angles in three-space. **deticpt** determines which element a ray will intercept, and projects the ray to the elements's plane.

deticpt takes a BPIPE formatted input. Transformations to various coordinate systems are determined by values in the chipposdb and chipsizedb. Output is written in BPIPE format.

deticpt places the detector aimpoint at the user specified X,Y,Z offset. Chip position and chip size databases for the Chandra ACIS-I, ACIS-S, HRC-I, and HRC-S are available at </proj/axaf/simul/databases/detectors/deticpt>.

USAGE OPTIONS

deticpt uses an IRAF-compatible parameter interface, i.e. the parameters can be set in the parameter file (deticpt.par) or they can be set on the command line at run-time.

The options are :

input *string*

Name of the BPIPE format ray file from which to read. If **input** equals 'stdin', input is read from STDIN.

output *string*

Name of the BPIPE format ray file to which to write. If **output** equals 'stdout', output is written to STDOUT.

xaimpoint *string*

The X(OSAC) position of the aimpoint. The aimpoint may be specified in three ways. First, the user may simply instruct the program to use the default value in the **aimpointdb** by specifying the string *default*. Alternatively, the user may specify an absolute position by providing the position as the argument to **xaimpoint**. Finally, the user may specify an offset to the **aimpointdb** value by prefacing the offset with the 'f' character. For more information of coordinate systems see the COORDINATE SYSTEMS section.

yaimpoint *string*

The Y(OSAC) position of the aimpoint. The aimpoint may be specified in three ways. First, the user may simply instruct the program to use the default value in the **aimpointdb** by specifying the string *default*. Alternatively, the user may specify an absolute position by providing the position as the argument to **yaimpoint**. Finally, the user may specify an offset to the **aimpointdb** value by prefacing the offset with the 'f' character. For more information of coordinate systems see the COORDINATE SYSTEMS section.

zaimpoint *string*

The Z(OSAC) position of the aimpoint. The aimpoint may be specified in three ways. First, the user may simply instruct the program to use the default value in the **aimpointdb** by specifying the string *default*. Alternatively, the user may specify an absolute position by providing the position as the argument to **zaimpoint**. Finally, the user may specify an offset to the **aimpointdb** value by prefacing the offset with the 'f' character. For more information of coordinate systems see the COORDINATE SYSTEMS section.

xaimoffset *string*

The X(SIM) offset of the aimpoint. The aimpoint offset may be specified in three ways. First, the user may simply instruct the program to use the default value in the **chipposdb** by specifying the string *default*. Alternatively, the user may specify an absolute position by providing the position as the argument to **xaimoffset**. Finally, the user may specify an offset to the **chipposdb** value by prefacing the offset with the 'f' character. For more information of coordinate systems see the COORDINATE SYSTEMS section.

yaimoffset *string*

The Y(SIM) offset of the aimpoint. The aimpoint offset may be specified in three ways. First, the user may simply instruct the program to use the default value in the **chipposdb** by specifying the string *default*. Alternatively, the user may specify an absolute position by providing the position as the argument to **yaimoffset**. Finally, the user may specify an offset to the **chipposdb** value by prefacing the offset with the 'f' character. For more information of coordinate systems see the COORDINATE SYSTEMS section.

zaimoffset *string*

The Z(SIM) offset of the aimpoint. The aimpoint offset may be specified in three ways. First, the user may simply instruct the program to use the default value in the **chipposdb** by specifying the string *default*. Alternatively, the user may specify an absolute position by providing the position as the argument to **zaimoffset**. Finally, the user may specify an offset to the **chipposdb** value by prefacing the offset with the 'f' character. For more information of coordinate systems see the COORDINATE SYSTEMS section.

flength *string*

The focal length of the telescope. The focal length may be specified in three ways. First, the user may simply instruct the program to use the default value in the **aimpointdb** by specifying the string *default*. Alternatively, the user may specify an absolute position by providing the position as the argument to **focal_length**. Finally, the user may specify an offset to the **aimpointdb** value by prefacing the offset with the 'f' character. Note that **flength** will affect the value calculated for the DET coordinate system.

xnode *string*

The X(OSAC) position of the aimpoint. The aimpoint may be specified in three ways. First, the user may simply instruct the program to use the default value in the **aimpointdb** by specifying the string *default*. Alternatively, the user may specify an absolute position by providing the position as the argument to **xnode**. Finally, the user may specify an offset to the **aimpointdb** value by prefacing the offset with the 'f' character. For more information of coordinate systems see the COORDINATE SYSTEMS section.

ynode *string*

The Y(OSAC) position of the aimpoint. The aimpoint may be specified in three ways. First, the user may simply instruct the program to use the default value in the **aimpointdb** by specifying the string *default*. Alternatively, the user may specify an absolute position by providing the position as the argument to **ynode**. Finally, the user may specify an offset to the **aimpointdb** value by prefacing the offset with the 'f' character. For more information of coordinate systems see the COORDINATE SYSTEMS section.

znode *string*

The Z(OSAC) position of the aimpoint. The aimpoint may be specified in three ways. First, the user may simply instruct the program to use the default value in the **aimpointdb** by specifying the string *default*. Alternatively, the user may specify an absolute position by providing the position as the argument to **znode**. Finally, the user may specify an offset to the **aimpointdb** value by prefacing the offset with the 'f' character. For more information of coordinate systems see the

COORDINATE SYSTEMS section.
aimpointdb *string*

Name of the RDB file containing the OSAC position of the aimpoint. It consists of three header variables, xaimpoint, yaimpoint and zaimpoint. It should be noted that the OSAC position of the aimpoint is generated by determining the best focus for an on-axis raytrace.

chipposdb *string*

Name of the RDB file containing the chip physical coordinate(CPC) and local science instrument(LSI) positions of the detector corners. It contains:

- chip physical coordinate(CPC) positions of the detector corners.
- local science instrument(LSI) positions of the detector corners.
- offset between the LSI origin and the detector aimpoint.
- X and Y CHIP coords, position of the origin of the SKY coords system.
- size of a SKY coordinate pixel in arcseconds.

chipsizedb *string*

Name of the RDB file containing the various dimensions of the detector. It contains:

- size of the chip in CHIP coords.
- size of CHIP pixel in microns.
- size of the chip in millimeters.
- rotation of the CHIP axes wrt DET coords.
- CHIP pixel resolution.
- origin of DET coords in CHIP coords.
- handedness of the planar rotation between CHIP and TDET coords.

quickcheck *boolean*

Currently it is recommended that the user set this parameter to *no*. For more info see the **DETECTION** section.

infiniteplane *boolean*

If set to yes, check only that a photon intersects the plane of a chip. If set to no, check to ensure it lands within the clipping corners.

flength *real*

The focal length of the telescope. The focal length may be specified in three ways. First, the user may simply instruct the program to use the default value in the **aimpointdb** by specifying the string *default*. Alternatively, the user may specify an absolute position by providing the position as the argument to **focal_length**. Finally, the user may specify an offset to the **aimpointdb** value by prefacing the offset with the 'f' character. Note that **flength** will affect the value calculated for the DET coordinate system.

filter *boolean*

If set to yes, rays are filtered through the code. The BPIPE output has an addition field, *icpt*, which is set to 1 if the ray intercepted the detector and 0 otherwise. If **filter** is set to no, only rays which

hit the detecting surface are passed out the backend.

pixprim *boolean*

If equal to yes, the position_x, position_y, and position_z fields in the output BPipe stream contain the pixelized positions of the ray where it intercepted the detector. The non-pixelized positions are placed in fields named position_np_x, position_np_y, and position_np_z. If set to no, the position_x, position_y, and position_z fields in the output BPipe stream contain the non-pixelized positions of the ray. The pixelized positions are placed in fields names position_pix_x, position_pix_y, and position_pix_z.

diagnostic *boolean*

Perform a diagnostic of the chips fiducial coordinate positions and orientations.

help *boolean*

Print this usage information and exit.

version *boolean*

Print the version information and exit.

COORDINATE SYSTEMS

deticpt makes use of a number of coordinate systems. For an in-depth look at all the coordinate systems and how they relate, see SDS-2.0: Coordinate Systems (Rev 5.0) by Jonathan McDowell. I will assume that the user is familiar with the various coordinate systems, and outline how **deticpt** processes rays with respect to the coordinate systems.

In the physical detectors we are attempting to simulate, incoming rays are detected in the CHIP coordinate system. CHIP coords is a pixel based system and as such, a detector is unable to resolve the position of a ray to less than a pixel(or the **delta** parameter in the **chipsizedb**). In order to maintain as much position information as possible, **deticpt** calculates the CHIP position and a non-pixelized CHIP position(CHIP_np). These two systems are the basis for a series of transformations to all other systems. So, **deticpt** will calculate the position as the physical detector would, unable to resolve positions to less than a pixel. But it will also calculate positions with full resolution, storing those positions in the *_np* fields.

AIMPOINT DATABASE

The **aimpointdb** consists of seven header variables: xaimpoint, yaimpoint, zaimpoint, xnode, ynode, znode, and focal_length. These variables are the OSAC position of the aimpoint and the focal_length in millimeters.

CHIP POSITION DATABASE

The **chipposdb** contains the following header variables which apply detector-wide.

delta

size of a SKY coordinate pixel in arcseconds

X0

CHIP coords position of the SKY coords system

Y0

CHIP coords position of the SKY coords system

xaimoffset

offset between the LSI origin and the detector aimpoint

yaimoffset

offset between the LSI origin and the detector aimpoint

zaimoffset

offset between the LSI origin and the detector aimpoint

The following information is contained in the table, one line per detecting chip corner.

chip

chip name

corner

corner name, LL, LR, UR, UL

xcpc

X position of corner in CPC coords(mm)

ycpc

Y position of corner in CPC coords(mm)

zcpc

Z position of corner in CPC coords(mm)

xcpcclip

X position of corner of detecting surface within the chip in CPC coords(mm)

ycpcclip

Y position of corner of detecting surface within the chip in CPC coords(mm)

zcpcclip

Z position of corner of detecting surface within the chip in CPC coords(mm)

xlsi

X position of corner in LSI coords(mm)

ylsi

Y position of corner in LSI coords(mm)

zlsi

Z position of corner in LSI coords(mm)

The **chipposdb** maps the incoming rays LSI coords to CPC coords. It contains one line for each of the four corner points of the detecting chip.

CHIP SIZE DATABASE

The **chipsizedb** contains dimensions of the detector. The information is contained in a file, one line per detecting chip.

chip

chip name

id

chip id number

xsize_pix	size of the chip in CHIP coords
ysize_pix	size of the chip in CHIP coords
xpixelsize_micron	size of CHIP pixel in microns
ypixelsize_micron	size of CHIP pixel in microns
xsize_mm	size of the chip in millimeters
ysize_mm	size of the chip in millimeters
theta	rotation of the CHIP axes wrt DET coords
delta	CHIP pixel resolution
X0	origin of DET coords in CHIP coords
Y0	origin of DET coords in CHIP coords
H	handedness of the planar rotation between CHIP and TDET coords

POSITIONING THE DETECTOR

The detector is position relative to its aimpoint. The user may specify where the aimpoint is in OSAC coordinates via the **xyzaimpoint** parameters. From there **deticpt** uses the chip position and chip size databases to determine the position of the aimpoint on the detecting surface. The Chandra detector databases provide the position of the aimpoint on the detector surface for each of the following detectors: ACIS-I, ACIS-S, HRC-I, HRC-S. The aimpoint database provides a starting point for where on-axis best focus is located.

To use the default OSAC aimpoint position, simply set each of the **xyzaimpoint** parameters to the string *default*. To apply an offset to the default OSAC aimpoint position, prepend the character *f* to argument of **xyzaimpoint**. To provide an absolute position for the aimpoint, simply specify the absolute OSAC position with no additional characters prepended.

OUTPUT

The program outputs the intersection positions in CHIP, TDET, DET, CPC, and OSAC coordinate systems. For each coordinate system, the program generates a pixelized position and a non-pixelized position. The pixelized position reflects the fact that we can not distinguish the position of a photon in the actual detector at a subpixel level. In the raytraces though, we can determine the exact location of the intersection. So the non-pixelized position retains the exact position of the intersection.

The names of the output fields are dependent on the value of the **pixprim** parameter. **pixprim** allows the user to specify if the pixelized position is considered the primary position. If **pixprim** is set to yes, then the pixelized position is placed in the BPipe position fields. The non-pixelized positions are placed in fields with the tag *_np* included to indicate they are non-pixelized values.

If the user specifies **pixprim** equals no, then they are requesting that the primary position fields, *position_x* etc., are non-pixelized positions. In this case, the pixelized positions are placed in fields with the tag *_px* included in the name to indicate they are pixelized.

The names of the various output positions are listed below. The names reflect those found when the **pixprim** parameter is set to yes.

chip

CHIP coords, note that this position reflects only as much resolution as the **delta** parameter in the **chipsizedb**. This is a IVector2.

chip_np

CHIP coords, full resolution. This is a DVector2.

tdet

TDET coords, assuming intersection at the pixelized CHIP position. This is a IVector2.

tdet_np

TDET coords, assuming intersection at the non-pixelized CHIP position. This is a DVector2.

det

DET coords, assuming intersection at the pixelized CHIP position. This is a DVector2.

det_np

DET coords, assuming intersection at the non-pixelized CHIP position. This is a DVector2.

cpcpos

CPC coords, assuming intersection at the pixelized CHIP position. This is a DVector3.

cpcpos_np

CPC coords, assuming intersection at the non-pixelized CHIP position. This is a DVector3.

position

OSAC coords, assuming intersection at the pixelized CHIP position. This is a DVector3.

position_np

OSAC coords, assuming intersection at the non-pixelized CHIP position. This is a DVector3.

DETECTION

If a ray intercepts a detecting surface, the above coordinate systems are calculated and placed in the output data.

If a ray does not intercept a detecting surface, **deticpt** behavior is determined by the **filter** flag. If **filter** is on, an additional field is in the output stream, *icpt*. *icpt* is set to 0 for rays which do not intercept a detecting surface, and 1 for rays which do intercept a detecting surface. The values for the output coordinate systems are undefined if the ray does not intercept a detecting surface, i.e. *icpt* is 0.

If **infiniteplane** is off, the detecting surface is the area described by the four corners in the **chipposdb** and the **chipsizedb**. See elsewhere in this document for a description of how the four corners are

manipulated to assure they are coplanar. If **infiniteplane** is on, the detecting surface is defined as the entire plane defined by the corners in the **chipposdb**.

quickcheck default value is off. It is a parameter offered for the advanced user. It attempts to speed the execution of the code by removing a number of sanity checks. Therefore, it is strongly recommended that the user leaves **quickcheck** off unless they are willing to perform the sanity checks on the input and output data themselves. That said, if **quickcheck** is off, **deticpt** loops through all the detecting surfaces determining which ones the ray intercepts. It then determines which surface is furthest upstream and outputs the intercept information for that surface. If **quickcheck** is on, **deticpt** exits when it determines the ray intercepts any detecting surface. It is important to note that there is no assurance the surface the ray intercepted is the first surface it would encounter if it continued moving in the directions it was going. Currently it depends on the order the detecting surfaces are listed in the **chipposdb**. So, **quickcheck** may be used in cases where the position of the detection is not important. It will provide a quicker means of determining that the ray intercepted any surface. But in cases where the position of the intersection is important, the user must assure that the geometry of the detecting surfaces and direction of the input rays preclude any overlapping detecting surfaces. If not, then the user must assure that the order the detecting surfaces appear in the **chipposdb** and the direction of the input rays will result in detection by the appropriate surface. To reiterate, if **quickcheck** is on, the user is takes responsibility for the accuracy of the results.

CHANDRA DETECTOR DATABASES

As mentioned above, aimpoint, chip position, and size databases are available at `/proj/axaf/simul/databases/detectors/deticpt/` for the ACIS-I, ACIS-S, HRC-I, and HRC-S. These databases are generated from values in the CALDB. For more insight into how these databases are constructed the user should see the **deticpt_chipdb** CVS module.

AUTHOR

M. Tibbetts (mtibbetts@head-cfa.harvard.edu)