

calibrate

SBC01-????-00

August 1, 1996

Prepared by: J. W. Pflugrath
Molecular Structure Corporation
3200 Research Forest Drive
The Woodlands, TX 77381
(713) 363-1033
(713 364-3628 (fax)
jwp@msc.com

Prepared for: Contract No. 943072401
Mary Westbrook
Argonne National Laboratory
9700 South Cass Avenue
Argonne, IL 60439
(708) 252-8914
(708) 252-4021 (fax)

westbrook@anlel.el.anl.gov

Reviewed by:

Approved by:

Copyright © 1996, 1995, 1994 Molecular Structure Corporation
RESTRICTED RIGHTS NOTICE SHORT FORM (JUNE 1987)

Use, reproduction, or disclosure is subject to restrictions set forth in Contract No. W-31-109-ENG-38 and Contract No. 943072401 with the University of Chicago, Operator of Argonne National Laboratory.

TABLE OF CONTENTS

1 SCOPE AND PURPOSE	4
1.1. References.....	4
1.2 Definitions and Abbreviations.....	4
2 BACKGROUND	5
3 INPUT IMAGES	6
3.1 Dark or background images	6
3.2 Mask image.....	6
3.3. Flood image.....	7
3.4 An image to determine the position of the primary beam	7
4 RUNNING CALIBRATE	8
4.1 Calibrate input.....	8
4.2 Mouse input and the display	9
4.2.1 The menubar	10
4.2.1.1 File menu	10
4.2.1.2 View menu	11
4.2.1.3 Calibrate menu	12
4.2.1.4 Help button	13
4.2.2 Input fields	13
4.2.2.1 Beam position and detector distance	14
4.2.2.2 Image search area and limits	14
4.2.2.3 Mask scanning parameters	14
4.2.2.4 Bad pixel criteria.....	16
4.2.2.5 Non-uniformity reference algorithm	16
4.2.2.6 Filenames.....	17
4.2.3 The image display.....	18
4.2.3.1 Display Min, Max	18
4.2.3.2 Zoom factor	18
4.2.3.3 Display output	18
4.3 Save file commands.....	19
4.4 Command line options.....	20

5 OUTPUT	21
6 ERRORS.....	21
7 ACCESSORY PROGRAMS	21
7.1 dtdecompose.....	22
7.2 dtcompose.....	23
7.3 make_marks.....	24
7.4 multi	24
7.5 make_interpolation	25
7.6 dtaverage	27
8 EXAMPLE: MULTIPLE MODULE DETECTOR CALIBRATION.....	28
8.1 Input save files	28
8.1.2 Single module spatial distortion savefile	29
8.1.2 Multiple module non-uniformity of response savefile	30
8.1.3 Input C-shell script calibrate.com	31
8.1.4 Image files	33
8.2 Output.....	34
9 FILE FORMATS.....	42
9.1 Image files	42
9.1.1 Image header format	42
9.1.2 Image data format	44
9.2 Bad pixel list file.....	44
9.3 Calibration files	45
9.3.1 The non-uniformity file	45
9.3.2 The spatial distortion fileset	45
APPENDIX A INSTALLATION	47
APPENDIX B RESOURCE FILE	48

1 Scope and Purpose

This document describes how to use the calibrate software in order to calibrate an area detector for spatial distortion and non-uniformity of response. The calibrate software was originally developed by Dr. Marty Stanton at Brandeis University. The calibrate software produces files that are used by the d*TREK program to convert from millimeters to pixels, from pixels to millimeters, to flag bad pixels, and to correct pixel intensities for variations in response.

Appendix A describes how to install the calibrate software on a computer system.

1.1. References

- Stanton, M. (1992) Correcting Spatial Distortions and Nonuniform Response in Area Detectors. *J. Appl. Cryst.* **25**, 549-558.
- Thomas, D.J. (1989) Calibrating an area detector diffractometer: imaging geometry. *Proc. R. Soc. Lond.* **A425**, 129-167.
- Thomas, D.J. (1989) Calibrating an area detector diffractometer: integral response. *Proc. R. Soc. Lond.* **A428**, 181-214.
- Kabsch, W. (1988) Evaluation of Single-Crystal X-ray Diffraction Data from a Position-Sensitive Detector. *J. Appl. Cryst.* **21**, 916-924.
- NCSA Mosaic User Guide.

1.2 Definitions and Abbreviations

- SIT silicon-intensified target
- CCD charge-coupled device
- CID charge-injection device
- ADC analog-to-digital converter
- DC direct current
- px pixel or picture element
- mm millimeters
- stdin standard input, SYS\$INPUT or Fortran unit 5. Most programs get their input from this device, file or stream.
- stdout standard output, SYS\$OUTPUT or Fortran unit 6. Most programs write text output to this device, file or stream.
- stderr standard error, SYS\$ERROR. Most programs write error messages to this device, file or stream.

2 Background

Electronic position-sensitive detectors generally geometrically distort the detected signal such as a diffraction pattern. The components of these detectors -- phosphor, fiber-optic tapers, lens, image intensifiers, SIT tubes and CCDs -- each introduces potential problems that are difficult to avoid when designing and building detectors. Software is used to correct for the deficiencies in images introduced by the hardware. The deficiencies are spatial distortions, pixel inhomogeneity of response and defective pixels.

Spatial distortions need to be corrected so that pixels can be accurately mapped to millimeters and the reverse. These mapping functions predict where Bragg reflections will fall on the detector and the millimeter coordinates of peaks found in images. Also the correction for inhomogeneity of response needs to accurately determine the position and area of each pixel. In the calibrate program, it is assumed that the spatial distortion of the detector does not vary between the time of calibration and the end of an experiment.

The inhomogeneity or non-uniformity of response needs correction so that counts in individual pixels can be correlated with the incoming signal (photons). Response might vary over the detector because of variations in phosphor thickness, fiber-optic taper properties, pixel area, fixed-pattern noise, and other electronic noise.

Defective pixels arise for a number of reasons. For our purposes, a defective pixel is one that does not give a count proportional to the photons registered. Some pixels are "dark", that is they always give zero or very low counts. Other pixels are always "bright"; they give high counts independent of the photons detected. Still other pixels can give random counts. Defective pixels must be flagged and not used by a data processing program.

A saturated pixel is one where the counts are beyond the range where they are proportional to the detected photons. Each detector can exhibit both local and global saturations. The data processing software needs to know how to detect and treat Bragg reflections with saturated pixels.

Finally, for most diffraction data processing software, the origin of the detector millimeter coordinate system is where the primary beam hits the detector when it is normal to the beam.

For further description of calibration, consult the references listed above, especially Stanton et al. 1992.

3 Input images

These calibrate programs require several images as inputs that must be created before running calibrate. These images are:

1. A dark or background image
2. A mask image
3. A flood image
4. An image to determine the position of the primary beam.

Image file formats are discussed in section 9. Images are not created by the calibrate program. They are created by a data acquisition program such as **dtcollect**. Furthermore, for best results, the input images should be averages of several images. You can use the **dtdisplay** and **dtaverage** (see section 7.6) programs to average images.

3.1 Dark or background images

A dark or background image is used to subtract out the DC offset of the ADC, readout noise, other fixed-pattern noise and dark current. This image is taken with the source shutter closed. The exposure time should be long enough to give good statistics. Since dark current varies with temperature, the temperature should be held constant during calibration and during a data collection experiment. Readout noise can often be measured by making an image with the shutter closed for a zero exposure time. These images will be subtracted from data images and from the other calibration images to yield images with only X-ray signal in them.

3.2 Mask image

Calibrate requires a mask image created by placing a metal plate with a square lattice of holes in it over the front of the detector and exposing the detector to a flood field of X-rays. The holes must be evenly spaced in both directions over the entire active area of the detector. The mask should be situated so that the square lattice is aligned with the pixel directions. Calibrate has been used successfully with masks that have had spacing between the holes from 1 mm to 5 mm. For a multimodule detector, the mask should have a blocked hole every 50 mm. The position of the blocked hole is used to reference the spatial distortion of an individual module to the entire multimodule. Each individual module needs such a reference or fiducial mark. Calibrate will search

the mask image for peaks, so expose the image long enough to allow for effective and precise peak searching. A few missing or blocked holes are allowed in the mask.

3.3 Flood image

The flood image is taken without the mask in place. All active pixels in the detector need to be exposed to X-rays. The detector should be in the same position as during diffraction experiment. The exposure time of the flood image should be long enough to obtain accurate Poissonian counting statistics without saturating the detector. Remember that the dark current accumulates during the exposure, too.

A flood field can be generated in several ways. One method uses a radioactive ^{55}Fe source positioned at the sample position to create an isotropic flood field of X-ray photons. Some problems with this method are the photons are neither numerous enough to give good statistics for a reasonable exposure time nor are they at the same energy of the photons in the diffraction experiment. Another method exposes an amorphous material (metal foil, powder or concentrated solution) at the sample position. The sample fluoresces creating a flood field of X-rays. This field is not isotropic and needs to be separately characterized for use by the calibrate software. An appropriate choice of sample can yield a flood field of the proper energy photons. Since the source is used to create the flood field, the detector must be swung out of the shadow of the beam stop, so that the flood field hits all active pixels. Or the beamstop must remain in the same place for the actual diffraction experiment, as the pixels behind the beamstop will not be calibrated. A final method for generating a flood field is to remove all the collimation of the source and to position the detector far enough away from the source so that it sees a flood field. This method has the problems of absorption and safety.

3.4 An image to determine the position of the primary beam

This image will determine the initial origin of reciprocal space on the detector. This origin is refined by the data processing software, so it need not be super accurate. Often the beam position can be selected manually from a diffraction image by looking at the beam stop shadow. However, many problems in data processing can be traced back to an inaccurate primary beam position. To create this image, the X-rays should be highly attenuated, the beam stop

removed and a very short exposure made. Be sure to replace the beam stop and check its position after making such images.

4 Running calibrate

After calibrate has been properly installed, it may be run interactively from a command line or from a script file (command procedure). In either case, calibrate **requires** an X Window display to run. Also, for the help command to work, NCSA Mosaic must be installed and the command to run it made known to calibrate (see Appendix A). Enter 'calibrate' on the command line or select 'calibrate...' from the command menu. The following list shows the steps to take:

1. Create Dark, Mask and Flood images
2. Run calibrate
3. Enter Beam Position
4. Enter Crystal-Detector Distance
5. Review image search limits
6. Review masking scanning parameters
7. Review bad pixel parameters, read in bad pixel list if necessary
8. Select reference image criteria
9. Enter image filenames
10. Calibrate - Go Calibrate

All steps will need to be taken when calibrating a detector for the first time. Subsequent calibrations can make use of a save file, so that steps 5,6,7,8,9 can be skipped.

4.1 Calibrate input

Calibrate can get input from the mouse, save files on disk, and/or the command line. If appropriate input comes from the command line, then the mouse is not used, but the display is still required and used. When calibrate first starts up, it tries to read input from the following files in order:

- 1a. A file defined by the environment variable or logical name CALSAV. This is **not** a file named CALSAV; it **must be** an environment variable or logical name.

-or-

- 1b. A file named CALIBRATE_SITE if it exists (this may be the file itself, or one defined as this environment variable or logical name), then a file named CALIBRATE_USER if it exists (this may be the file itself, or one defined as this environment variable or logical name).

-then-

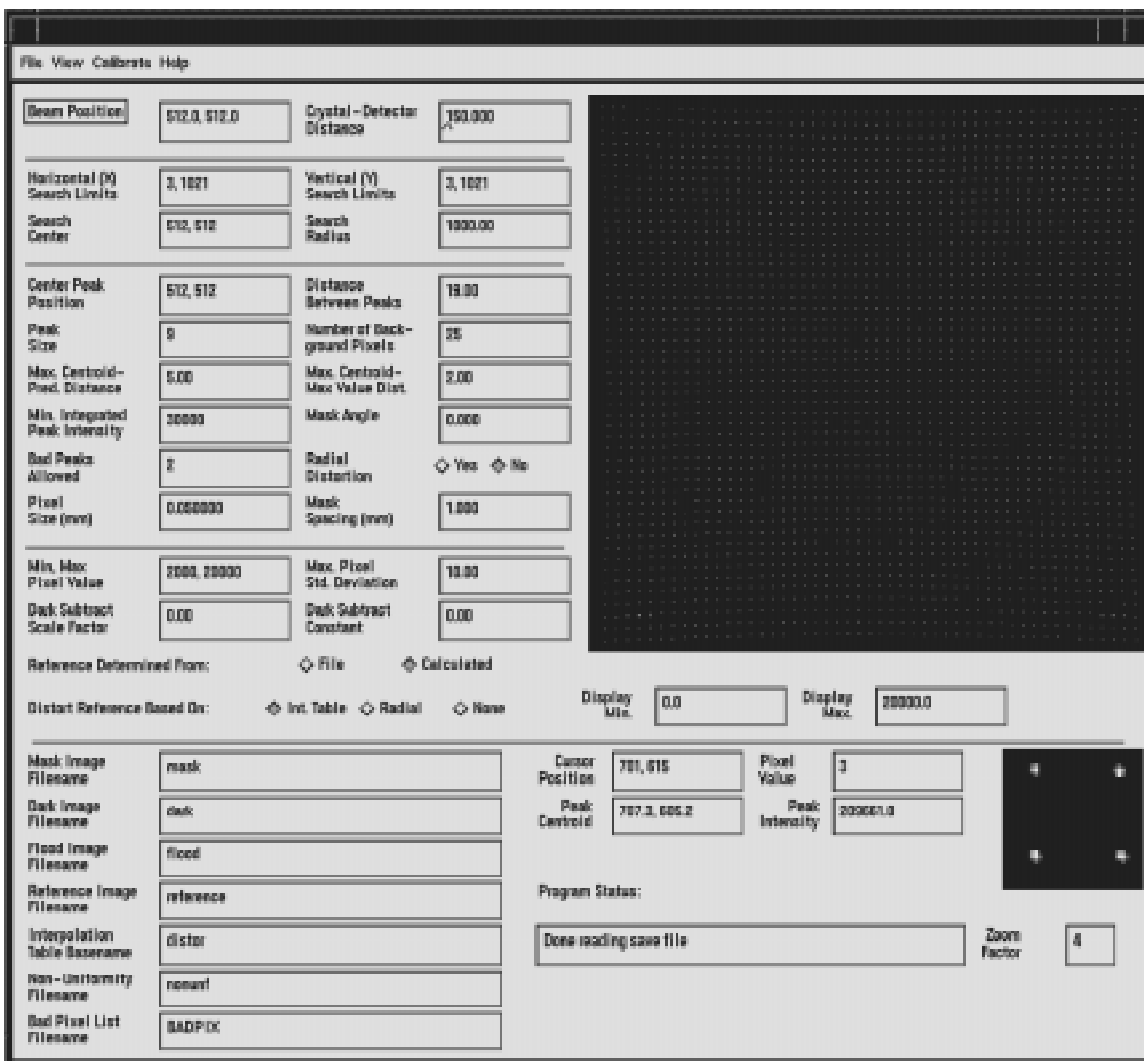
2. Files specified by any -init filename option on the command line. These are read in the order they appear on the command line after the files in 1 are (possibly) read in.

The format of the save file is discussed in section 4.2 below.

After reading in any save or initialization files, a calibrate window appears on the display and mouse input is accepted (unless -autoexit is specified on the command line).

4.2 Mouse input and the display

The calibrate window displays a menubar across the top, parameter input fields along the left and bottom sides, a drawing area on the left for images, and a small zoomed area as show below:



If the display does not appear as shown, check that the X Window server has accessed the calibrate resource file (see Appendix B).

4.2.1 The menubar

The menubar has four buttons labeled **File**, **View**, **Calibrate**, and **Help**. All except the **Help** button have associated cascading menus.

4.2.1.1 File menu

The File menu is used to read files, write files and to exit from calibrate. Select **Read** to read in files used by calibrate. A pullright menu appears with choices for the file to read in. The choices are:

- Mask
- Flood
- Dark
- Reference
- DISTOR
- NONUNF

Select the type of file to read. Select the name of the file to read in the file selection dialog box that appears.

To write a file, select **Write** from the File menu. Proceed as for reading files.

To read in a save or script file, select **Read Save...** from the File menu.

To read in a bad pixel list file, select **Read Bad Pixel List...** from the File menu. The format of this file is described in section 4.

To write the current values of all calibrate parameters to a file, select **Write Save...** from the File menu. The save file is an ASCII file that can be edited (see section 4.2).

To exit from calibrate, select **Exit** from the File menu.

4.2.1.2 View menu

The View menu is used to select the image displayed in the drawing area along with any bad pixels. The possible choices are

- Mask image
- Dark image
- Flood image
- Reference image
- DISTOR files (they are in the form of images)
- Non-uniformity images
- Bad pixels (not really an image, but can be viewed)
- Search limits

Only images that have been read in or calculated by calibrate can be viewed. If a choice is inactivated, this means that calibrate does not have the image available. When an image is selected, it appears in the drawing area.

4.2.1.3 Calibrate menu

The Calibrate menu is used to perform the actual spatial distortion and non-uniformity of response calibration. The current values of all input parameters are used during these tasks, so they should be set properly beforehand. The selections in the Calibrate menu are:

- | | |
|------------------|---|
| Go Calibrate | Performs all steps to write spatial distortion and non-uniformity of response files. This is a Go DISTOR and Go Nonunf combined into a single command. |
| Go DISTOR | Performs the entire sequence of steps to calibrate the detector for spatial distortion. A mask image is read in and scanned for peaks. The pixel coordinates of the peaks are correlated with the expected millimeter coordinates of the holes in the mask to yield tables that can be used to convert between pixels and millimeters and vice versa. |
| Scan Mask | Scans the mask image for peaks, but does not calculate the spatial distortion. This is used to check the peak searching parameters. |
| Calculate DISTOR | Calculates the spatial distortion as described under Go DISTOR above. The interpolation files are written if this is successful. |
| Go NONUNF | Performs the entire sequence of steps to calibrate the detector for non-uniformity of response. The spatial distortion must have been previously calibrated. Typically, a flood image and a dark image are read in. Bad pixels are flagged in both images. The dark image is subtracted from the flood image. A reference image is calculated, then distorted to match the actual spatial distortion of the detector. The ratio of the dark-subtracted flood image to this distorted reference image is the non-uniformity of response information. |

Find bad pixels Bad pixels in the flood or dark images are found. Any pixel with a value below Min Pixel and above Max Pixel is set as bad. Any pixel with a value more than Std. Deviation away from the average of its neighbors is set as bad. See section 4.2.2.4.

Flood - Dark The dark image is subtracted from the flood image.

Calculate reference The reference image is calculated. Once calculated it can be viewed by selecting it in the View menu.

Distort reference The reference image is distorted to match the available spatial distortion information.

Calculate NONUNF The flood image is scaled to the reference image.

MASK - DARK The Dark image is subtraced from the mask image and the result is placed in the mask image.

Correct MASK NONUNF
The mask image is corrected for non-uniformity of response and the result is placed in the mask image.

Correct MASK DISTOR
The Mask image is corrected for spatial distortion and the result is placed in the Mask image. This is a good check of the algorithm.

4.2.1.4 Help button

The Help button activates the help command specified in the calibrate resource file. Usually this command is set to be NCSA Mosaic that is then used to view a hypertext help document.

4.2.2 Input fields

The left-side of the display contains many text entry fields to display and changes the current values of parameters that are used by the calibration program. These values may be initialized or changed by reading in a save file, either at program startup or subsequently with the **Read Save...** command in the **File** menu. To change a value, simply select the field and enter a new value

followed by the **Enter** key. The input fields are grouped together by separator bars.

4.2.2.1 Beam position and detector distance

These must be set for every calibration, so they are at the top.

Beam position The pixel coordinates of the primary beam when the detector is positioned at a swing angle of 0. The input is given as the fast pixel coordinates first, then the slow pixel coordinates second, in the input images. The X and Y refer to these fast and slow directions, not to the physical laboratory coordinate system.

Crystal to detector distance
The crystal to detector distance in millimeters.

4.2.2.2 Image search area and limits

Horizontal (X) Search Limits
The minimum and maximum pixel coordinate in the horizontal direction as displayed by calibrate.

Vertical (Y) Search Limits
The minimum and maximum pixel coordinate in the vertical direction as displayed by calibrate.

Search Center The center pixel of a circle

Search Radius The radius in pixels of a circle, only pixels inside the circle are considered in subsequent calibration steps.

4.2.2.3 Mask scanning parameters

Center Peak Position
The center peak position in the mask in pixels. The mask is scanned in a special way starting from this peak position. The mask is scanned first in the horizontal direction for peaks, then in the vertical direction. In both cases, peaks are found by searching outward from known peaks and

applying a crude fit of the distortion from the known peaks. Thus the algorithm needs to start with a good peak and that is the purpose of this entry.

Distance Between Peaks

The distance between peaks in pixels. When searching for peaks, the next peak is this many pixels from the previous peak, modified by any known distortion

Peak Size

The size of peaks in pixels.

Number of Background Pixels

The number of pixels used to calculate the background around a peak during the peak centroid determination.

Max. Centroid-Pred. Distance

Mask peaks whose centroids are more than these many pixels away from the expected centers are discarded.

Max. Centroid-Max Value Dist.

Mask peaks whose centroids are more than these many pixels from the maximum value in the peak are ill-shaped and discarded.

Min. Integrated Peak Intensity

A peak with an integrated intensity below this value is discarded.

Mask Angle

The angle the mask makes relative to the pixel coordinate system of the detector.

Bad Peaks Allowed

Scanning along a line of peaks stops when this number of bad or discarded peaks is reached. It is assumed that the edge of the image, the mask or the active area of the detector has been reached.

Radial Distortion Yes|No

Select whether the detector exhibits radial distortion. Normally choose No.

Pixel Size (mm) Set the nominal pixel size near the center of the detector in millimeters. Normally set this to 0.0, so that calibrate determines this value for itself.

Mask Spacing (mm) The spacing between mask holes in millimeters. The holes must be spaced evenly in two orthogonal directions.

4.2.2.4 Bad pixel criteria

Min, Max Pixel Values

This text field takes 6 values separated by commas. The values are (1) Min Pixel in Dark, (2) Max Pixel in Dark, (3) Min Pixel in Flood, (4) Max Pixel in Flood, (5) Min Pixel (or truncate value) in Flood-Dark and (6) unused at present. Pixels in the dark image or in the flood image below the Min value and above the Max value will be flagged as bad. Any pixel in a dark-subtracted image below the Min Pixel in Flood-Dark. is set to the Min Pixel in Flood-Dark value.

Max Pixel Std. Deviation

Pixels which deviation by more than this number of standard deviations from the neighborhood average (a neighborhood is 32 x 32 pixels)

Dark Subtract Scale Factor

The dark image is multiplied by this factor before subtracting from the flood and/or mask image.

Dark Subtract Constant

This constant value is added to the dark-subtracted flood or mask image.

4.2.2.5 Non-uniformity reference algorithm

Reference Determined from File | Calculated

If File is selected, then a reference image calculated elsewhere is used to scale the dark-subtracted flood field and calculate the non-uniformity of response.

If Calculated is selected, then a simple reference field based on an isotropic point source is calculated and used to scale the dark-subtracted flood field.

Distort Reference Based On: Int. Table | Radial | None

The reference image is distorted based on the
Interpolation Tables,
A radial distortion,
or None
before scaling to the dark-subtracted flood field. In general, chose Int. Table.

4.2.2.6 Filenames

The current names of all files used by calibrate are displayed. This is no guarantee that these files exist, since calibrate tries to read or write them only as needed. The filenames are listed for the following files:

Mask Image	The filename of the mask image
Dark Image	The filename of the dark image
Flood Image	The filename of the flood image
Reference Image	The filename of the reference image

Interpolation Table Basename
The basename of the interpolation table fileset.

There are five files in the interpolation table fileset:

basename.calpar	Calibration parameters
basename.x_int	Coded Xmm for a pixel
basename.y_int	Coded Ymm for a pixel
basename.inv_x_int	X pixel for a coded Xmm,Ymm
basename.inv_y_int	Y pixel for a coded Xmm,Ymm

Non-uniformity of response
The filename of the non-uniformity of response image

Bad Pixel
The filename of the bad pixel image

4.2.3 The image display

The drawing area on the right displays images selected by the View menu or images being processed by the Calibrate menu. The area is only 512 by 512 pixels, so for larger images, entire rows and columns of pixels are skipped in order to fit the entire image into the area. The images are displayed as a gray-scale if the X Window display is capable. The image search area and limits can also be displayed. Bad pixels are displayed with the following color code:

Blue	Outside search limits (section 4.1.2.2)
Red	Outside allowed Min, Max pixel value limit (section 4.1.2.3)
Yellow	Exceeded maximum standard deviation

4.2.3.1 Display Min, Max Min, Max

These set how the pixel values are mapped to the gray-scale of the image display. Pixels with values below Display Min are displayed as black, while those with values above Display Max are displayed as white. Pixels with intermediate values are given a corresponding gray color.

4.2.3.2 Zoom factor

A small zoom area of 128x128 pixels displays a zoomed area of the parent image. Select the zoomed area with the mouse. The selected area is zoomed by the factor entered in the Zoom factor field. Pixels may be selected in this area, just as in the large drawing area.

4.2.3.3 Display output

When a pixel in the image is selected with the mouse, the following information is calculated and displayed below the image:

Cursor Position	Coordinate of the selected pixel as a FAST, SLOW pair. The first pixel in the image has coordinate 1,1 and the second 2, 1.
Pixel Value	Value of the selected pixel. Use this to help evaluate the proper choice of Min and Max Pixel Value 4.2.2.4.
Peak Centroid	The center of gravity of a selected peak.
Peak Intensity	The integrated peak intensity. Use this to evaluate the proper choice of Min. Peak Intens. 4.2.2.3.

Program status A single line giving the current status including any error message from calibrate.

4.3 Save file commands

A calibrate save file is an editable ASCII file that contains one command followed by parameters per line. A save file with the current values of calibrate parameters can be created by clicking on 'File Write Save...' with the mouse. The save file **must end with the word** exit. Commands in the save file are case insensitive as are filenames on VMS platforms (but filenames are case-sensitive on Unix platforms). A single apostrophe just prevents the parser from interpret the following character as special. The commands in the save file map to entry fields described earlier. The *_pattern and *_dirname commands are used to save the results of the file selection dialog boxes used by the **File** menu. A radio button is selected as on with a value of 1 and is off with a value of 0:

For example:

```
flood_radial      0  
flood_geometry    1  
flood_film        0  
  flood_interpolate 1
```

mean that the Radial Distortion is set to No (flood_radial 0); Reference Determined From is set to Calculated (flood_geometry 1; flood_film 0); and the Distort Reference Based On is set to Int. Table (flood_interpolate 1).

```
nonunf_filename 'nonunf  
nonunf_pattern '*nonunf*  
nonunf_dirname './  
  
distor_filename 'jim_mask  
distor_pattern '*distor*  
distor_dirname '/rxdat2/people/raxis/jwp/gold/  
reference_filename 'reference  
reference_pattern '*reference*  
reference_dirname './  
flood_filename '/jwp/gold/nflood.ds  
flood_pattern '*flood*  
flood_dirname '/jwp/gold/  
dark_filename '/jwp/gold/dark.img  
dark_pattern '*dark*  
dark_dirname '/jwp/gold/  
mask_filename 'mask  
mask_pattern '*.mad
```

```
mask_dirname './
save_filename 'jim4.sav` .sav
save_pattern '*.sav*
save_dirname '/jwp/gold/
badpix_filename 'BADPIX
badpix_pattern '*BADPIX*
badpix_dirname './
flood_radial 0
flood_interpolate 1
flood_geometry 1
flood_film 0
darksub_const 0.000000E+00
darksub_scale 1.000000E+00
pixel_sd 40.00000
max_pixel 65534
min_pixel 4000
radial_distortion 0
bad_peaks 2
mask_angle 0.000000E+00
min_peak 30000
cent_to_maxval 2.000000
cent_to_pred 5.000000
background_pixels 25
peak_size 9
peak_distance 19.00000
center_peak 1536 1536
search_radius 2200.000
search_center 1536 1536
vertical_limits 32 3050
horizontal_limits 32 3050
xtod_distance 150.0000
masktod_distance 0.000000E+00
beam_position 1546.60 1490.500
mask_spacing 1.000000
pixel_size 5.0000001E-02
display_min 0 0 0 0 0 0 0 0 0
display_max 20000 30000 10000 10000 15000 100000 100000 100000 100000
edges 0
dump_mode 0
dump_filename 'DMPFIL
exit
```

4.4 Command line options

The options below may be placed on the calibrate command line. Judicious choice of options allows calibrate to operate in the absence of mouse input. Command line options are used when in calibrate script files.

-beamx fpx	The fast pixel coordinate of the beam position.
-beamy spx	The slow pixel coordinate of the beam position.
-dist d	The sample to detector distance in millimeters.
-init file	An initialization file (save file) is read in when this command line argument is processed. Be careful, this will override any previous settings including those made by a previous command line argument.
-dump file	Turn on dump mode and set output dump filename.
-flood file	Sets input flood filename.
-mask file	Sets input mask filename.
-dark file	Sets input dark filename.
-distor file	Sets output spatial distortion fileset basename.
-nonunf file	Sets output nonuniformity filename.
-badpix file	Sets input bad pixel list filename.
-refer file	Sets input reference filename.
-godistor	Run the Calibrate Go DISTOR command automatically.
-gononunf	Run the Calibrate Go NONUNF command automatically.
-autoexit	Automatically exit before getting mouse input. This is only useful when combined with -godistor or -gononunf.
-help	Print command line help text.

5 Output

Calibrate creates a spatial distortion calibrate fileset that consists of five files and a non-uniformity of response file. These files can be used in the MADNES program, the d*TREK program and other programs to correct images for spatial distortion and non-uniformity of response.

6 Errors

Most errors can be traced to unable to read the input files. The Program Status field will display an error message when a file cannot be read.

7 Accessory programs

Although calibrate is the only program needed to calibrate a single module detector, additional programs and scripts are required to calibrate a multiple module detector. These are:

dtdecompose	Decompose (extract) subimages from an image
dtcompose	Compose (build) an image from subimages
make_marks	Mark single modules in a large multimodule image
multi	Read a calibrate dump file and determine the pixel position of the fiducial blocked hole by interpolating from the nearest neighbor unblocked holes.
make_interpolation	Combine interpolation tables from single module calibrations into a multimodule calibration table.

7.1 dtdecompose

dtdecompose reads from stdin the filename of the input image to decompose. It reads this image into memory. Next it reads image filenames and the origin and extent (in pixels) of the rectangle of pixels in the input image to be copied into the output image. If the requested rectangle is completely contained within the input image, dtdecompose creates the new image and writes it out. Otherwise it prints an error message and exits. New images have the same Data_type (short int, long int, etc) as the input image. dtdecompose continues to process input until there is an end-of-file on stdin, whereupon it exits.

Note: The size of the input image is only limited by the system resources, not the software.

Note: The first pixel in the image has coordinates 0,0. All coordinate pairs are expressed as the fastest varying pixel coordinate first, then the slowest varying pixel coordinate. In other words, pixel 1,0 is the second pixel in the file; 2,0 the third and so on.

Example: Extract nine non-overlapping images each 1024 x 1024 pixels from a 3072 x 3072 image.

```
% dtdecompose <<EOF
big.img
 0  0 1024 1024 sub1.img
1024  0 1024 1024 sub2.img
2048  0 1024 1024 sub3.img
 0 1024 1024 1024 sub4.img
1024 1024 1024 1024 sub5.img
2048 1024 1024 1024 sub6.img
 0 2048 1024 1024 sub7.img
```

1024 2048 1024 1024 sub8.img
2048 2048 1024 1024 sub9.img
EOF

7.2 dtcompose

dtcompose is the functional opposite of dtdecompose. It reads from stdin the filename and dimensions of an image to create. Next it reads additional image filenames and their position (in pixels) in the new image. The pixel values of these additional images are placed in the composed image if the image has large enough dimensions to contain it. Otherwise an error is reported and dtcompose exits. The new image has the same Data_type as the first image placed in it. The Data_types of the subsequent images are converted to this Data_type if necessary. When there is an end-of-file on stdin, the composed image is written to the filename given at the beginning. Since each subsequent image is placed in the composed image in the order given, it is possible to replace values in the composed image that came from a previous image file. Information about the progress of dtcompose is written to stderr.

Note: The size of the output image is only limited by the system resources, not the software. Any pixels in the output image that are not replaced by input images will be given the value 0.

Note: The first pixel in the image has coordinates 0,0. All coordinate pairs are expressed as the fastest varying pixel coordinate first, then the slowest varying pixel coordinate. In other words, pixel 1,0 is the second pixel in the file; 2,0 the third and so on.

Example: Compose a 3072 x 3072 image from nine 1024 x 1024 images:

```
% dtcompose << EOF  
big.img 3072 3072  
sub1.img 0 0  
sub2.img 1024 0  
sub3.img 2048 0  
sub4.img 0 1024  
  
sub5.img 1024 1024  
sub6.img 2048 1024  
sub7.img 0 2048  
sub8.img 1024 2048
```

sub9.img 2048 2048
EOF

7.3 make_marks

Make_marks reads a 3072x3072 flood field image, minimum pixel value and the name of a so-called output "mark" image from stdin. It creates a new 3072x3072 mark image and writes it to disk. The mark image is used by the make_interpolation program. The mark image constructed by make_marks has pixel values 1000, 2000, 3000, ..., 9000 in nine 1024x1024 evenly spaced areas of the image. Any pixel value that is below the input minimum pixel value is given a value of 0. Thus, the mark image contains only pixel values of 0, 1000, ..., 9000.

Example:

```
% make_marks <<EOF  
flood.img  
10000  
mark.img  
EOF
```

7.4 multi

Multi reads from a file DMPFIL (actual name or environment variable) or if it does not exist then from stdin the dump output created by the single module calibrate Go DISTOR command. The dump file lists the positions of all the mask holes (peaks in the mask image) found and the expected positions of mask holes that were not found. Multi then interpolates the position of the missing holes from the positions of its immediate neighbors and writes this position to stdout preceded by the text "Correct position." This becomes part of the input to the make_interpolation program (section 7.5).

Example: Run multi on the dump file output of calibrate:

```
% multi <nmask1.dmp >multi.log  
% cat multi.log  
Trying : DMPFIL  
Done reading dump file  
Min, max xpeak      76      124  
Min, max ypeak      76      124  
Peak      99      97 is missing
```

Observed position 491.2100 448.8500
Correct position 492.5512 449.4068

The dump file looks like in part:

! Searching for center point at : 512.0000 512.0000

Peak 100 100 511.53 506.82 210238.0 1

! Searching for right point at :

Peak 101 100 530.74 506.99 215126.0 1

...

Peak 124 100 993.96 510.20 242448.0 1

Peak 125 100 1021.00 511.12 187.0 14

! Peak to far from maximum value in box

! Peak moved to far from predicted position

! Peak intensity below minimum

! Predicted position at: 1014.383 511.3594

Peak 99 100 492.26 506.89 231131.0 1

Peak 98 100 472.52 507.19 212596.0 1

...

Peak 99 97 491.21 448.85 349.0 8

! Peak intensity below minimum

! Predicted position at: 491.4299 449.3658

Peak 99 96 491.81 429.78 207573.0 1

Peak 99 95 491.70 410.42 204774.0 1

! Peak intensity below minimum

! Predicted position at: 398.0293 1012.762

Peak 94 99 395.54 487.80 232009.0 1

Peak 94 98 395.47 468.45 217801.0 1

Peak 94 97 395.36 449.05 206068.0 1

Peak 94 96 395.27 429.72 217458.0 1

Peak 94 95 395.17 410.30 202119.0 1

....

7.5 make_interpolation

Make_interpolation reads 9 single module spatial distortion calibration filesets and combines them into a multiple module spatial distortion fileset. Presently, the single modules must be dimensioned 1024 x 1024 pixels and the multiple module must be composed of a 3 x 3 array of these single modules.

Input on stdin consists of:

1. A template basename for the single module filesets. The character # in the template will be replaced with the characters 1, 2, ..., 9 in order to derive the spatial distortion fileset basename for each input single module.

For example, if the template basename is “distor#”, then the files read in are:

```
distor1.calpar
distor1.x_int
distor1.y_int

distor1.inv_x_int
distor1.inv_y_int
distor2.calpar
...
distor9.inv_y_int
```

2. The name of the so-called “mark” image file created by make_marks (section 7.3)
3. The word “yes” to write out a filled image file. This file is not used further.
4. The name of the filled image file.
5. The basename of the multiple module spatial distortion fileset. Make_interpolation will create five output files beginning with this basename: *.calpar, *.x_int, *.y_int, *.inv_x_int, *.inv_y_int.
6. The fast pixel coordinate of the primary beam position.
7. The slow pixel coordinate of the primary beam position.

In addition to the input on stdin, a file named MISSING (actual name or environment variable) is read with 9 lines from the multi output that contain the positions of the single module fiducials. The fiducials are assumed to be 50 mm apart. The MISSING file can be created with the following Unix C-shell scripts assuming the calibrate dump files are named nmask1.dmp,..., nmask9.dmp:

```
% touch multi.log
% multi < nmask1.dmp >> multi.log
...
% multi < nmask9.dmp >> multi.log
% grep Correct multi.log > MISSING
```

WARNING: If the mask contains more blocked holes than just the expected fiducials, then MISSING must be edited, so that it contains only the fiducial positions.

7.6 dtaverage

Dtaverage averages a scan or series of images on a pixel by pixel basis. It rejects outliers in the averaging process and saves rejected pixel positions in a separate image file. Its main purpose is to create the best possible averaged images for input to calibrate. To use dtaverage, first collect the images you wish to average with dtcollect. Next use dtdisplay to compute initial average and standard deviation images. The command syntax for dtaverage is:

```
dtaverage [-scan scanfile] [ options ...]
```

Option	Description
-scan sName	File sName (no default) has the scan definition.
-template sInTemplate	Input image file template. If sInTemplate contains ? be sure to enclose it in quotes. Default: image.???
-start nSeqStart	Sequence start number. Default: 1.
-inc nSeqIncr	Sequence increment. Default: 1.
-num nNumImages	Number of images to process. Default: 1000.
-avg sAvgFile	Input averaged image file name. Default is formed from the input file template by stripping off any leading directory and appending av to the name.
-sd sSDFile	Input standard deviation image file name. Default is formed from the input file template by stripping off any leading directory and appending sd to the name.
-sigma fSigma	Pixels which differ by more than fSigma from the average value will be excluded from the calculation of the new average and standard deviation images. Default 3.
-maxsd fMaxSD	Any pixel input standard deviation larger than fMaxSD will be set to fMaxSD. Default 1000000.
-minsd fMinSD	Any pixel input standard deviation smaller than fMinSD will be set to fMinSD. Default 1.

-
- min fMin Pixels with values below fMin will be excluded from the calculation of the new average and standard deviation images.
Default -1000000.
- max fMax Pixels with values above fMax will be excluded from the calculation of the new average and standard deviation images. Default 1000000.
- out sOutFile Output averaged image file name. Default is formed from the input file template by stripping off any leading directory and appending dtav to the name.
- outsd sOutSDFFile Output standard deviation image file name. Default is formed from the input file template by stripping off any leading directory and appending dtsd to the name.
- help Prints a description of the command line options.

8 Example: Multiple module detector calibration

Below is an example of a complete multiple module calibration. First the input files and the command script are listed, then the output is shown. To run this example, use

```
calibrate.com mask.img >&! calibrate.log
```

at the Unix shell prompt after changing the directory to `.../src/calibrate/test`.

8.1 Input save files

The C-shell script in section 8.1.3 below requires two input save files for the two separate runs of the calibrate program. The first run calibrates a single module spatial distortion. The second run calibrates the multiple module for non-uniformity of response. Since the input images are different sizes for these two runs, the save files need to reflect this.

8.1.1 Single module spatial distortion save file

The items in **bold** below should be checked to make sure that the mask will be scanned properly. A method to do this is to run calibrate interactively, read in the save file and then choose Scan Mask from the Calibrate menu. If the

mask is scanned successfully, then the parameters are correct. If it is not scanned correctly, adjust the parameters, re-scan the mask and then save the parameters to a new save file. These are the mask scanning parameters that will be used in the C-shell script below.

```
nonunf_filename 'nonunf
nonunf_pattern '*nonunf*
nonunf_dirname './
distor_filename 'distor
distor_pattern '*distor*
distor_dirname './
reference_filename 'reference
reference_pattern '*reference*
reference_dirname './
flood_filename 'flood
flood_pattern '*flood*
flood_dirname './
dark_filename 'dark
dark_pattern '*dark*
dark_dirname './
mask_filename 'mask
mask_pattern '*.mad
mask_dirname './
save_filename 'single_module.sav
save_pattern '*.sav*
save_dirname './
badpix_filename 'BADPIX
badpix_pattern '*BADPIX*
badpix_dirname './
flood_radial      0
flood_interpolate 1
flood_geometry    1
flood_film        0
darksub_const    1.0
darksub_scale    0.0

pixel_sd        10.0
max_pixel       20000
min_pixel       2000
radial_distortion 0
bad_peaks         2
mask_angle        0.0
min_peak        30000
cent_to_maxval    2.0
cent_to_pred      5.0
background_pixels 25
peak_size         9
peak_distance    19.0
```

```
center_peak      512    512
search_radius  1000.0
search_center   512    512
vertical_limits 3     1021
horizontal_limits 3    1021
xtod_distance   150.0
masktod_distance 0.0
beam_position   512.0   512.0

mask_spacing    1.00
pixel_size        0.05
dump_mode         0
dump_filename     DMPFIL
display_min       0 0 0 0 0 0 0 0 0
display_max       20000 10000 10000 1000 15000 100000 100000 100000 100000
exit
```

8.1.2 Multiple module non-uniformity of response save file

The items in **bold** below should be checked to make sure that bad pixels in the dark image and the flood image are properly detected. A method to do this is to run calibrate interactively, read in the save file and then choose Find Bad Pixels from the Calibrate menu. If this makes physical sense then the parameters are correct. Otherwise, adjust the parameters, re-determine the bad pixels and then save the parameters to a new save file.

```
nonunf_filename 'nonunf
nonunf_pattern '*nonunf*
nonunf_dirname './
distor_filename 'distor
distor_pattern '*distor*
distor_dirname './
reference_filename 'reference
reference_pattern '*reference*
reference_dirname './
flood_filename 'flood
flood_pattern '*flood*
flood_dirname './
dark_filename 'dark

dark_pattern '*dark*
dark_dirname './
mask_filename 'mask
mask_pattern '*mask
mask_dirname './
save_filename 'nonunf.sav
save_pattern '*sav*
```

```
save_dirname './
badpix_filename 'BADPIX
badpix_pattern '*BADPIX*
badpix_dirname './
flood_radial      0
flood_interpolate 1
flood_geometry   1
flood_film       0
darksub_const    0.0
darksub_scale    1.0
pixel_sd        10.0
max_pixel       65534
min_pixel       2000

radial_distortion  0
bad_peaks          2
mask_angle         0.0
min_peak           30000
cent_to_maxval     2.0
cent_to_pred       5.0
background_pixels  25
peak_size          9
peak_distance      19.0
center_peak      1536    1536
search_radius   2200.0
search_center   1536    1536
vertical_limits 32     3050
horizontal_limits 32    3050
xtod_distance   150.0
masktod_distance 0.0
beam_position   1536.0    1536.0
mask_spacing    1.00
pixel_size         0.05
dump_mode          0
dump_filename      DMPFIL
display_min        0 0 0 0 0 0 0 0 0
display_max        20000 10000 10000 1000 15000 100000 100000 100000 100000
exit
```

8.1.3 Input C-shell script calibrate.com

Below is an input script to do a complete calibration. Items in **bold** need to be edited for each detector calibration. Execute the script with the following command:

```
% calibrate.com mask.img>&! calibrate.log
```

```
#!/bin/csh -f
#
set echo
set verify

# Routines used (they must be in the PATH):
#
# dtdecompose
# calibrate
# multi
# make_marks
# make_interpolate

if ($1 == ) then
  echo -n "Enter multimodule mask filename : "
  set multimask = $<
else
  set multimask = $1
endif
if (!(-e $multimask) ) then
  echo "File $multimask does not exist."
  exit 1
endif

set nmod = 9          # Number of modules
dtdecompose << EOF    # Decompose big image into smaller images
$multimask
  0  0 1024 1024 $multimask:r1.tmp
1024 0 1024 1024 $multimask:r2.tmp
2048 0 1024 1024 $multimask:r3.tmp
  0 1024 1024 1024 $multimask:r4.tmp
1024 1024 1024 1024 $multimask:r5.tmp
2048 1024 1024 1024 $multimask:r6.tmp
  0 2048 1024 1024 $multimask:r7.tmp
1024 2048 1024 1024 $multimask:r8.tmp
2048 2048 1024 1024 $multimask:r9.tmp
EOF

rm -f multi.log      # Remove any existing multi.log
touch multi.log      # Create an empty multi.log to append to later

#
# Run calibrate on each individual mask image.
#
set n = 1
while ( $n <= $nmod )

  set mask = $multimask:r$n.tmp
  echo "Mask filename for module number " $n ": " $mask
```

```
if ( -e $mask ) then
  calibrate -geom +0+0 -init single_module.sav -mask $mask \
    -distor $mask:r \

    -dump $mask:r.dmp -dark dark.img \
    -godistor -autoexit

    multi < $mask:r.dmp >> multi.log
else
  echo "Bad filename ", $mask
endif
@ n++
end

grep "Correct" multi.log >! MISSING
emacs MISSING # Check the MISSING file to make sure there are lines

make_marks <<EOF
flood.img
10000
mark.img
EOF

make_interpolation << EOF
$multimask:r#
mark.img
yes
filled.img
distor
1546.60
1490.50
EOF
calibrate -geom +0+0 -init nonunf.sav -distor distor \
  -dark dark.img \
  -nonunf nonunf.img \
  -gononunf -flood flood.img -autoexit
exit
```

8.1.4 Image files

The above scripts use the following image files:

```
mask.img
flood.img
dark.img
```

The format of these files is described in section 9.

8.2 Output

The following output is generated by the script in section 8.1.3:

```
set verify
if ( mask.img == ) then
```

```
set multimask = mask.img
endif
if ( ! ( -e mask.img ) ) then
set nmod = 9
dtdecompose
```

dtdecompose: Copyright (c) 1995 Molecular Structure Corporation

```
dtdecompose: Enter input image name:
Cimage::nRead filename is mask.img
File mask.img successfully opened.
Header of file mask.img successfully read.
Image is 3072 by 3072 pixels.
Data_type in header is short int.
Compression_type is None.
Byte_order is big_endian.
```

```
dtdecompose: Enter origin and
extents to extract and output filename:
File mask1.tmp successfully opened.
Success writing file mask1.tmp!
dtdecompose: Wrote image mask1.tmp with origin at 0, 0 and extents 1024, 1024
```

```
dtdecompose: Enter origin and
extents to extract and output filename:
File mask2.tmp successfully opened.
Success writing file mask2.tmp!
dtdecompose: Wrote image mask2.tmp with origin at 1024, 0 and extents 1024, 1024
```

```
dtdecompose: Enter origin and
extents to extract and output filename:
File mask3.tmp successfully opened.
Success writing file mask3.tmp!
dtdecompose: Wrote image mask3.tmp with origin at 2048, 0 and extents 1024, 1024
```

```
dtdecompose: Enter origin and
extents to extract and output filename:
File mask4.tmp successfully opened.
Success writing file mask4.tmp!
dtdecompose: Wrote image mask4.tmp with origin at 0, 1024 and extents 1024, 1024
```

```
dtdecompose: Enter origin and
               extents to extract and output filename:
File mask5.tmp successfully opened.
Success writing file mask5.tmp!
dtdecompose: Wrote image mask5.tmp with origin at 1024, 1024 and extents 1024, 1024

dtdecompose: Enter origin and
               extents to extract and output filename:
File mask6.tmp successfully opened.

Success writing file mask6.tmp!
dtdecompose: Wrote image mask6.tmp with origin at 2048, 1024 and extents 1024, 1024

dtdecompose: Enter origin and
               extents to extract and output filename:
File mask7.tmp successfully opened.
Success writing file mask7.tmp!
dtdecompose: Wrote image mask7.tmp with origin at 0, 2048 and extents 1024, 1024

dtdecompose: Enter origin and
               extents to extract and output filename:
File mask8.tmp successfully opened.
Success writing file mask8.tmp!
dtdecompose: Wrote image mask8.tmp with origin at 1024, 2048 and extents 1024, 1024

dtdecompose: Enter origin and
               extents to extract and output filename:
File mask9.tmp successfully opened.
Success writing file mask9.tmp!
dtdecompose: Wrote image mask9.tmp with origin at 2048, 2048 and extents 1024, 1024

dtdecompose: Enter origin and
               extents to extract and output filename:
dtdecompose: Done
rm -f multi.log
touch multi.log
set n = 1
while ( 1 <= 9 )
set mask = mask1.tmp
echo Mask filename for module number 1 : mask1.tmp
Mask filename for module number 1 : mask1.tmp
if ( -e mask1.tmp ) then
calibrate -geom +0+0 -init single_module.sav -mask mask1.tmp -distor mask1 -dump mask1.dmp
-dark dark.img -nonunf mask1.nonunf -godistor -autoexit
multi
else
@ n++
end
```

```
while ( 2 <= 9 )
set mask = mask2.tmp
echo Mask filename for module number 2 : mask2.tmp
Mask filename for module number 2 : mask2.tmp
if ( -e mask2.tmp ) then
calibrate -geom +0+0 -init single_module.sav -mask mask2.tmp -distor mask2 -dump mask2.dmp
-dark dark.img -nonunf mask2.nonunf -godistor -autoexit
multi
else
@ n++
end
while ( 3 <= 9 )
set mask = mask3.tmp
echo Mask filename for module number 3 : mask3.tmp
Mask filename for module number 3 : mask3.tmp
if ( -e mask3.tmp ) then
calibrate -geom +0+0 -init single_module.sav -mask mask3.tmp -distor mask3 -dump mask3.dmp
-dark dark.img -nonunf mask3.nonunf -godistor -autoexit
multi
else
@ n++
end
while ( 4 <= 9 )
set mask = mask4.tmp
echo Mask filename for module number 4 : mask4.tmp
Mask filename for module number 4 : mask4.tmp
if ( -e mask4.tmp ) then
calibrate -geom +0+0 -init single_module.sav -mask mask4.tmp -distor mask4 -dump mask4.dmp
-dark dark.img -nonunf mask4.nonunf -godistor -autoexit
multi
else
@ n++
end
while ( 5 <= 9 )
set mask = mask5.tmp
echo Mask filename for module number 5 : mask5.tmp
Mask filename for module number 5 : mask5.tmp
if ( -e mask5.tmp ) then
calibrate -geom +0+0 -init single_module.sav -mask mask5.tmp -distor mask5 -dump mask5.dmp
-dark dark.img -nonunf mask5.nonunf -godistor -autoexit
multi
else
@ n++
end
while ( 6 <= 9 )
set mask = mask6.tmp
echo Mask filename for module number 6 : mask6.tmp
Mask filename for module number 6 : mask6.tmp
if ( -e mask6.tmp ) then
```

```
calibrate -geom +0+0 -init single_module.sav -mask mask6.tmp -distor mask6 -dump mask6.dmp
-dark dark.img -nonunf mask6.nonunf -godistor -autoexit
multi
else
@ n++
end
while ( 7 <= 9 )
set mask = mask7.tmp
echo Mask filename for module number 7 : mask7.tmp
Mask filename for module number 7 : mask7.tmp
if ( -e mask7.tmp ) then
calibrate -geom +0+0 -init single_module.sav -mask mask7.tmp -distor mask7 -dump mask7.dmp
-dark dark.img -nonunf mask7.nonunf -godistor -autoexit
multi
else
@ n++
end
while ( 8 <= 9 )
set mask = mask8.tmp
echo Mask filename for module number 8 : mask8.tmp
Mask filename for module number 8 : mask8.tmp
if ( -e mask8.tmp ) then
calibrate -geom +0+0 -init single_module.sav -mask mask8.tmp -distor mask8 -dump mask8.dmp
-dark dark.img -nonunf mask8.nonunf -godistor -autoexit
multi
else
@ n++
end
while ( 9 <= 9 )
set mask = mask9.tmp
echo Mask filename for module number 9 : mask9.tmp
Mask filename for module number 9 : mask9.tmp
if ( -e mask9.tmp ) then
calibrate -geom +0+0 -init single_module.sav -mask mask9.tmp -distor mask9 -dump mask9.dmp
-dark dark.img -nonunf mask9.nonunf -godistor -autoexit
multi
else
@ n++
end
while ( 10 <= 9 )
grep Correct multi.log
emacs MISSING
make_marks
Flood field filename [flood.mad] : Minimum flood field value [10000] : Image Size      3072
3072
Number bad pixels      708868
Number good pixels    8728316
Mark filename [mark.mad] : make_interpolation
Interpolation template [mask_#] : Reading : mask1
```

4.999997E-02 1024 1024
Reading : mask2
5.000016E-02 1024 1024
Reading : mask3
4.999990E-02 1024 1024
Reading : mask4
4.999993E-02 1024 1024
Reading : mask5
4.999997E-02 1024 1024
Reading : mask6
5.000001E-02 1024 1024
Reading : mask7
4.999982E-02 1024 1024
Reading : mask8
5.000016E-02 1024 1024

Reading : mask9
4.999982E-02 1024 1024

Missing reference mask points:

Module	X	Y
1	492.55	449.41
2	516.78	460.34
3	536.49	452.26
4	501.62	473.38
5	522.62	466.53
6	538.65	465.15
7	497.06	499.58
8	525.37	498.19
9	539.61	498.76

1000.000
Module 1
Actual 1
Index 1 1
Start 1 1
Step 4 4
Position 492.5512 449.4068
Shifted 492.5512 449.4068
Expected 546.6213 490.5266
Difference 54.07013 41.11981
Move 54.07013 41.11981
(Move) 5407.013 4111.981
Offset 0.000000E+00 0.000000E+00

Module 2
Actual 2
Index 2 1
Start 1 1

Step	4	4
Position	516.7823	460.3420
Shifted	1540.782	460.3420
Expected	1546.622	490.5266
Difference	5.839478	30.18460
Move	1029.839	30.18460
(Move)	102984.0	3018.460
Offset	256.0000	0.000000E+00

Module	3	
Actual	3	
Index	3	1
Start	1	1
Step	4	4
Position	536.4874	452.2554
Shifted	2584.487	452.2554
Expected	2546.622	490.5266
Difference	-37.86523	38.27121
Move	2010.135	38.27121
(Move)	201013.5	3827.121
Offset	512.0000	0.000000E+00

Module	4	
Actual	4	
Index	1	2
Start	1	1
Step	4	4
Position	501.6197	473.3830
Shifted	501.6197	1497.383
Expected	546.6213	1490.527
Difference	45.00165	-6.856079
Move	45.00165	1017.144
(Move)	4500.165	101714.4
Offset	0.000000E+00	256.0000

Module	5	
Actual	5	
Index	2	2
Start	1	1
Step	4	4
Position	522.6217	466.5270
Shifted	1546.622	1490.527
Expected	1546.622	1490.527
Difference	0.000000E+00	0.000000E+00
Move	1024.000	1024.000
(Move)	102400.0	102400.0
Offset	256.0000	256.0000

Module	6	
--------	---	--

Actual 6
Index 3 2
Start 1 1
Step 4 4
Position 538.6456 465.1524
Shifted 2586.646 1489.152
Expected 2546.622 1490.527
Difference -40.02344 1.374634
Move 2007.977 1025.375
(Move) 200797.7 102537.5
Offset 512.0000 256.0000

Module 7
Actual 7
Index 1 3
Start 1 1
Step 4 4
Position 497.0619 499.5789
Shifted 497.0619 2547.579
Expected 546.6213 2490.527
Difference 49.55945 -57.05151
Move 49.55945 1990.948
(Move) 4955.945 199094.9
Offset 0.000000E+00 512.0000

Module 8
Actual 8
Index 2 3
Start 1 1
Step 4 4
Position 525.3714 498.1926
Shifted 1549.371 2546.193
Expected 1546.622 2490.527
Difference -2.749634 -55.66528
Move 1021.250 1992.335
(Move) 102125.0 199233.5
Offset 256.0000 512.0000

Module 9
Actual 9
Index 3 3
Start 1 1
Step 4 4
Position 539.6090 498.7557
Shifted 2587.609 2546.756
Expected 2546.622 2490.527
Difference -40.98682 -56.22827
Move 2007.013 1991.772
(Move) 200701.3 199177.2

Offset 512.0000 512.0000

Mark filename [mark.mad] : Correcting mark image distortions

Image Size	:	3072	3072
Image Array Size	:	3072	3072
Int Table Size	:	768	768
Int Table Array Size	:	768	768
Int X start, step	:	1	4
Int Y start, step	:	1	4

Clearing output image

Starting main loop

Line : 100
Line : 200
Line : 300
Line : 400
Line : 500
Line : 600
Line : 700
Line : 800
Line : 900
Line : 1000
Line : 1100
Line : 1200
Line : 1300
Line : 1400
Line : 1500
Line : 1600
Line : 1700
Line : 1800
Line : 1900
Line : 2000
Line : 2100
Line : 2200
Line : 2300
Line : 2400
Line : 2500
Line : 2600
Line : 2700
Line : 2800
Line : 2900
Line : 3000

Filling holes in mark image

=====

Iteration	1
New good	2332
Number of bad	868833

=====

Iteration	2
New good	23

```
Number of bad      866501
=====
Iteration          3
New good           5
Number of bad      866478
Write MARK file [no] ?
  Filled MARK filename [mark.filled] :
  Starting REVERSE interpolation tables
Output template filename [mask] :
  Enter X beam position [1536] :
  Enter Y beam position [1536] :
calibrate -geom +0+0 -init nonunf.sav -distor distort -dark dark.img -nonunf nonunf.img -gononunf
-flood flood.img -autoexit
Maximum reference/floodfield ratio :  4.725791
Minimum reference/floodfield ratio :  3.5424564E-05
Scale factor          :  6771.354
exit
```

9 File formats

Calibrate works with image files, spatial distortion filesets, bad pixel list files and save files. The save file format has been described in section 4.3 and two examples have been shown in sections 8.1.1 and 8.1.2. In the following sections, the image file format, the bad pixel list file format and the spatial distortion fileset formats are described. The non-uniformity file has the same format as an image file. The calibrate resource file is described in Appendix B.

9.1 Image files

Image files use a format first developed by Jon Cristy of Dupont and then modified by Jim Pflugrath and Marty Stanton for area detector images. The files consist of an ASCII header described in section 9.1.1 followed by binary data.

9.1.1 Image header format

The image header format consists of an ASCII string that has a length that is a multiple of 512 characters with a minimum length of 512 and maximum length of an integer whose ASCII representation fits in 5 digits. (when using Stanton's routines, the limit is 2048 characters). Padding (space ' ') characters will be used to pad the image header so its length is a multiple of 512 characters.

The string consists of human readable text in the form:
KEYWORD=VALUE;^J

KEYWORD is a case-sensitive string bounded by whitespace on the left, with no whitespace within. KEYWORD must begin with a letter or underscore and can contain only letters, digits and underscores. The length of KEYWORD may not exceed 32 characters.

= is the equal sign; there is no whitespace on the left side, but may or may not be on the right.

VALUE is the value of the keyword, which may be a number (integer or float), a string or arrays of numbers. A string value cannot contain the {, } and ; characters. Elements in arrays are separated by whitespace.

; is the semicolon character; it may or may not be preceded by whitespace.

^J is the newline character (ASCII 10).

Whitespace is any length sequence of space, tab, and/or newline characters.

Other image header requirements:

1. The header begins with the 2-character sequence '{ ^J', where ^J is ASCII 10.

2. The third character is the position of the start of the required keyword=pair:
HEADER_BYTES=value;
which gives the length in bytes of the entire header. The unpadding part of the header always ends with the 3 character sequence:
} ^J ^L

where ^J is ASCII 10 and ^L is ASCII 12. With this one can use the Unix 'more' command to view image headers and avoid viewing binary data. Padding will occur after the ^L if necessary.

3. The following keywords are also required for images and must be placed in the image header when an image is constructed:

DIM = value; The number of dimensions in the image
(usually 2)
SIZE1 = value; The number of pixels along the 1st direction.
SIZE2 = value; The number of pixels along the 2nd direction.

TYPE = mad; Required to read these images with MADNES
and the binary image data is of type short integer.
BYTE_ORDER = big_endian | little_endian
The byte order of the data
Data_type = signed char | unsigned char | short int | long int | unsigned
short int
| unsigned long int | float IEEE | Compressed | Other

where

signed char	is signed one byte integer value
unsigned char	is unsigned one byte integer value
short int	is signed 2-byte integer value
long int	is signed 4-byte integer value
unsigned short int	is signed 2-byte integer value
unsigned long int	is signed 4-byte integer value
float IEEE	is IEEE floating point values
Compressed	means the data is compressed. Then there must be a COMPRESSION keyword for the algorithm.
Other	is some other representation which will make the data unusable by most programs. There should be an DATA_OTHER keyword with more information.

9.1.2 Image data format

After the image header, the binary values of all the pixels in the image occur. The binary may be of any type allow by the header and may have any byte order.

9.2 Bad pixel list file

The bad pixel list file consists of the following ASCII keywords followed by the indicated parameters. The file can have any length.

PIXEL *pxfast pxslow badflag*
LINE *pxline pxstart pxend badflag*
COLUMN *pxcolumn pxstart pxend badflag*

where *pxfast*, *pxslow*, *pxstart* and *pxend* are pixel coordinates. *Pxline* is the pixel value of a line. *Pxcolumn* is the pixel value of a column. The first pixel has

coordinate (1,1). A LINE indicates a contiguous set of pixels along the fastest varying image dimension. Pixel coordinates occur as (LINE, COLUMN) pairs.

Badflg is an integer which indicates the type of pixel defect:

- 0 Dark (below min value, 4.2.2.4)
- 1 Bright (above max value, 4.2.2.4)
- 2 Out of bounds (outside search limits, 4.2.2.2)
- 3 Bad standard deviation (4.2.2.4)

The file can contain comments that begin with an exclamation point '!'.

9.3 Calibration files

9.3.1 The non-uniformity file

The non-uniformity file has the same format as image files, but with the additional keywords NONUNF_FLAG1, NONUNF_FLAG2, NONUNF_FLAG3, NONUNF_FLAG4 which give the pixel values of flagged bad pixels in the file. To correct an image for non-uniformity of response, first compute the average of the non-uniformity file pixels near the center of the image, next subtract away any dark current and non-X-ray signal from all the pixels, then multiple each pixel by the ratio of the NONUNF value to the average NONUNF value near the center:

$$\text{Corrected value} = (\text{observed value} - \text{dark value}) * \text{NONUNF}(\text{px1}, \text{px2}) / (\text{average NONUNF near center})$$

Any pixel with a value equal to one of the NONUNF_FLAG* values is a bad pixel.

9.3.2 The spatial distortion fileset

The spatial distortion fileset consists of five files which have a format based on the image file format. The files are named *basename.**, where *basename* is chosen by the user and * is:

calpar	Calibration parameters
x_int	Coded Xmm for a pixel
y_int	Coded Ymm for a pixel
inv_x_int	X pixel for a coded Xmm,Ymm
inv_y_int	Y pixel for a coded Xmm,Ymm

The .calpar file has the same format as an image file header and contains information about how the fileset was generated. It contains the primary beam position and other information needed by d*TREK and MADNES. The four other

files have the image file format as described in section 9.1. They contain pixel to millimeter and millimeter to pixel information for everything fourth pixel. Thus they have different dimensions than the images used to create them. For example, spatial distortion files for a 3072 x 3072 pixel image are dimensioned 768 x 768.

Appendix A Installation

The calibrate files are a mixture of C source code, C++ source code and older Fortran code. The files are distributed as a compressed tar file. To install the calibration files, first have the Free Software Foundation GNU gmake tool and gcc compiler along with its class libraries installed on the system. Also install NSCA Mosaic. Then choose a directory for the installation of the calibrate software, uncompress and untar the tar file and change directory to the `.../src/calibrate` and type use the make utility to build executables images::

```
mkdir .../DTREK
cd .../DTREK
zcat calibrate.tar.Z | tar xvfo
cd src/calibrate
make
make install
cd ../c++
gmake
gmake install
```

Edit the calibrate resource file as described in Appendix B and copy it to an application defaults directory. Finally, place `.../DTREK/bin` in your path.

The example shown in section 8 can be found in the `.../src/calibrate/test` directory.

Appendix B Resource file

The calibrate resource file is necessary to specify the text of the labels and the name of the help command and the help document. The resource file must reside in a directory normally searched by the X Window server for resource files. On Unix platforms the file is called Calibrate. On VMS platforms the file is called CALIBRATE.DAT. Here is a copy of a typical resource file.

```
*fontList:          -*-helvetica-bold-r-*-17-*-  
*XmPushButtonfontList:  -*-helvetica-bold-r-*-17-*-  
*XmTextField.fontList:  -*-helvetica-bold-r-*-14-*-  
*XmLabel.fontList:     -*-helvetica-bold-r-*-14-*-  
*XmToggleButton.fontList:  -*-helvetica-bold-r-*-14-*-  
*background: gray95  
  
*helpDocument:  
file://localhost/rxd1/people/jwp/DTREK/doc/calibrate.html  
!*helpDocument:  file://localhost/jwp_disk:[jwp.DTREK.doc]calibrate.html  
! Do not change the next *helpTmpdir value from /tmp/.  
*helpTmpdir:      /tmp/  
!*helpTmpdir:     sys$login:  
*helpMosaicCommand: mosaic  
!*helpMosaicCommand:  echo "NCSA Mosaic not installed." Cannot view\n "  
!*helpMosaicCommand: write sys$error "NCSA Mosaic not installed."\n!  
*w_help_cb.labelString: Help  
  
*w_beam_position_label.labelString: Beam Position  
*w_xtod_distance_label.labelString: Crystal-Detector\nDistance  
*w_horizontal_limits_label.labelString: Horizontal (X)\nSearch Limits  
*w_vertical_limits_label.labelString: Vertical (Y)\nSearch Limits  
*w_search_center_label.labelString: Search\nCenter  
*w_search_radius_label.labelString: Search\nRadius  
  
*w_center_peak_label.labelString: Center Peak\nPosition  
*w_peak_distance_label.labelString: Distance\nBetween Peaks  
*w_peak_size_label.labelString: Peak\nSize  
*w_background_pixels_label.labelString: Number of Back-\nground Pixels  
*w_cent_to_pred_label.labelString: Max. Centroid-\nPred. Distance  
*w_cent_to_maxval_label.labelString: Max. Centroid-\nMax Value Dist.  
*w_min_peak_label.labelString: Min. Integrated\nPeak Intensity  
*w_mask_angle_label.labelString: Mask Angle  
*w_bad_peaks_label.labelString: Bad Peaks\nAllowed  
*w_radial_label.labelString: Radial\nDistortion  
*w_radial_distortion.labelString: Yes  
*w_noradial_distortion.labelString: No  
*w_pixel_size_label.labelString: Pixel\nSize (mm)  
*w_mask_spacing_label.labelString: Mask\nSpacing (mm)  
*w_minmax_pixel_label.labelString: Min, Max\nPixel Value
```

*w_pixel_sd_label.labelString: Max. Pixel\nStd. Deviation
*w_darksub_scale_label.labelString: Dark Subtract\nScale Factor
*w_darksub_const_label.labelString: Dark Subtract\nConstant
*w_reffrom_label.labelString: Reference Determined From:
*w_flood_film.labelString: File
*w_flood_geometry.labelString: Calculated
*w_flood_label.labelString: Distort Reference Based On:
*w_flood_interpolate.labelString: Int. Table
*w_flood_radial.labelString: Radial
*w_flood_none.labelString: None

*w_mask_filename_label.labelString: Mask Image\nFilename
*w_dark_filename_label.labelString: Dark Image\nFilename
*w_flood_filename_label.labelString: Flood Image\nFilename
*w_reference_filename_label.labelString: Reference Image\nFilename
*w_distor_filename_label.labelString: Interpolation\nTable Basename
*w_badpix_filename_label.labelString: Bad Pixel List\nFilename
*w_nonunf_filename_label.labelString: Non-Uniformity\nFilename

*w_cursor_position_label.labelString: Cursor\nPosition
*w_pixel_value_label.labelString: Pixel\nValue
*w_peak_centroid_label.labelString: Peak\nCentroid
*w_peak_intensity_label.labelString: Peak\nIntensity
*w_display_minimum_label.labelString: Display\nMin.
*w_display_maximum_label.labelString: Display\nMax.
*w_zoom_factor_label.labelString: Zoom\nFactor
*w_program_status_label.labelString: Program Status:

*w_file_cb.labelString: File

*w_read_cb.labelString: Read
*w_file_readmask.labelString: Mask...
*w_file_readflood.labelString: Flood...
*w_file_readdark.labelString: Dark...
*w_file_readreference.labelString: Reference...
*w_file_readdistor.labelString: DISTOR...
*w_file_readnonunf.labelString: NONUNF...

*w_write_cb.labelString: Write
*w_file_writemask.labelString: Mask...
*w_file_writeflood.labelString: Flood...
*w_file_writedark.labelString: Dark...
*w_file_writereference.labelString: Reference...
*w_file_writedistor.labelString: DISTOR...
*w_file_writenonunf.labelString: NONUNF...

*w_file_readsava.labelString: Read Save...
*w_file_writesava.labelString: Write Save...
*w_file_readbadpix.labelString: Read Bad Pixel List...

*w_file_exit.labelString: Exit

*w_action_cb.labelString: Calibrate
*w_action_go.labelString: Go Calibrate
*w_action_godistor.labelString: Go DISTOR
*w_action_scanmask.labelString: Scan Mask
*w_action_interpolate.labelString: Calculate DISTOR
*w_action_gononunf.labelString: Go NONUNF
*w_badpix_cb.labelString: Find Bad Pixels
*w_action_darkimage.labelString: Dark Image
*w_action_floodimage.labelString: Flood Image
*w_action_clearbadpix.labelString: Clear Bad Pixels
*w_action_darksub.labelString: Flood - Dark
*w_action_calcref.labelString: Calculate Reference
*w_action_distortref.labelString: Distort Reference
*w_action_scale.labelString: Calculate NONUNF
*w_action_maskdarksub.labelString: MASK - DARK
*w_action_cornuf.labelString: Correct MASK NONUNF
*w_action_cordis.labelString: Correct MASK DISTOR

*w_display_cb.labelString: View
*w_display_mask.labelString: Mask
*w_display_dark.labelString: Dark
*w_display_flood.labelString: Flood
*w_display_reference.labelString: Reference
*w_displaydistor_cb.labelString: DISTOR
*w_display_xint.labelString: x_int
*w_display_yint.labelString: y_int
*w_display_invxint.labelString: inv_x_int
*w_display_invyint.labelString: inv_y_int
*w_display_nonunf.labelString: NONUNF
*w_display_limits.labelString: Search Limits
*w_display_badpix.labelString: Bad Pixels