THE UNIVERSITY OF ARIZONA®

Center for Gamma-Ray Imaging

Image Processing Code Documentation

August 8, 2006

Jacob Hesterman

Image Processing Code by Jacob Hesterman, Jean Chen, & William Hunter

Table of Contents

Introduction

The documentation included in this packet is the companion to a library of image processing code. The code is designed to accompany the data acquisition and electronics used in several SPECT imaging systems at the University of Arizona's Center for Gamma Ray Imaging. The code library contains almost seventy standard C files, many of which can be run as stand-alone processes. For ease of use, six overarching Perl scripts have been included that should account for most of the desired image processing. These scripts are described briefly in the bulleted list below and more extensively in the body of the document.

- ProcessMDRF - Processes the Mean Detector Response Function (MDRF) calibration data. The resulting files are used in position estimation.
- ProcessPSF - Processes the Point Spread Function (PSF) calibration data. The resulting files are used in reconstruction.
- ProcessImage - Performs position estimation on tomographic image data to produce a file of projection images.
- ProcessRecon - Performs position estimation on tomographic image data to produce a file of projection images and goes on to reconstruct the original object using Ordered Subset Maximum-Likelihood Expectation Maximization (OSEM).
- ProcessReconWithCOR - Similar to ProcessRecon with the benefit that center of rotation corrections are possible in the reconstruction.
- ProcessInterp - Performs Gaussian interpolation on a single-pinhole PSF, producing a more finely sampled PSF.
- ProcessInterpMultPinhole - Performs Gaussian interpolation on a multiple-pinhole PSF, producing a more finely sampled PSF.

The documentation focuses on these Perl scripts. First, a brief description of the script is provided. The usage of the script is then detailed. Next, a brief description of each individual function called by the script is given. Finally, there appears a list of all output files produced by the script, including mention of their format and the individual function that generated them. Good luck!

ProcessMDRF.pl

Description

ProcessMDRF is designed to take a raw MDRF file and generate several files from the data. The most important of these files are the mean file and the likelihood threshold file. These two files are used in position estimation with likelihood windowing.

Usage

```
ProcessMDRF <MDRFFile> <Camera> <Dimension> <Cutoff> <Output Directo-
ry>
```

e.g. `/utils/ProcessMDRF.pl RawData/mdrf_1123_01.dat 1 79 4000 MDRF_1123`

`<MDRFFile>` --> The location and name of the raw MDRF data file.

`<Camera>` --> The camera number. It is required for correctly naming files.

`<Dimension>` --> The dimension of the MDRF (e.g. 79 for a 79x79 MDRF).

`<Cutoff>` --> Up to the user. In the example, a value of 4000 is given for a 5000 event MDRF, resulting in an 80% likelihood window.

`<Output Directory>` --> The location for files created by this routine.

Code

GenMean --> Generates an unsmoothed mean file. Means are computed for each PMT from the raw, list mode data. A shrinking window (based on the standard deviation of the list mode data) is applied to improve the accuracy of the mean.

pfit2ad_interp --> Take the unsmoothed mean file and smooths it. This code (written by Bill Hunter) takes the mean file generated by GenMean and interpolates/smooths it. ProcessMDRF currently employs only smoothing, but interpolation is easily implemented. See the comments in pfit2ad_interp.c for more details.

GenLogFact --> Produces a log factorial file that will be used to generate likelihood thresholds. An 11-bit A/D conversion is used for the PMT outputs so the LogFact file contains the log factorial value for integers ranging from 0 to 2047 (2^{11}-1).

Gen_LW_int --> Performs position estimation with likelihood windowing on the raw MDRF data and produces files containing the pixel position and likelihood value of each event in the MDRF.

sort_like_thresh --> Takes the two files returned by Gen_LW_int and a cutoff value to determine a likelihood threshold for each pixel. The likelihood values for each event binned into a given pixel are sorted and the likelihood value occurring at the specified cutoff point becomes the likelihood threshold for that pixel.

medfilt_2d_float --> Median filters the likelihood thresholds.

Files Produced

MeanFilePreSmooth (GenMean) --> The unsmoothed mean values for each PMT at each pixel. (float)

MeanFile (pfit2ad_interp) --> The smoothed (and possibly interpolated) mean file. These mean values are used in future position estimation routines. (float)

LogFact (GenLogFact) --> The log factorial values for each integer from 0-2047. (float)

Pos (Gen_LW_int) --> The pixel location for each event in the MDRF following position estimation. (short)

Like (Gen_LW_int) --> The likelihood value for each event in the MDRF following position estimation. (long)

Thresholds (sort_like_thresh) --> The initial threshold value for each pixel. (long)

FiltThresholds (medfilt_2d_float) --> The median filtered threshold values. These likelihood thresholds are used in future position estimation routines. (float)

Note: All files except for LogFact are camera specific.

ProcessPSF.pl

Description

PSF preparation is a two-step process. First, position estimation with likelihood windowing is performed on the raw PSF data, producing a relatively large, 'crunched' PSF file. Second, the crunched PSF file is split into several smaller files (described below) that are used in the reconstruction code.

Usage

```
ProcessPSF <PSF File> <Camera> <Dimension> <Number of Coordinates>
<Fudge> <MDRF Directory> <Output Directory>
```

e.g. `/utils/ProcessPSF.pl Data/PSF_1211_01.dat 1 79 41888 90 MDRF_1123 PSF_1211`

`<PSF File>` --> The raw PSF data file.

`<Camera>` --> The camera number. It is required for correctly naming files.

`<Dimension>` --> The dimension of the MDRF (e.g. 79 for a 79x79 MDRF).

`<Number of Coordinates>` --> The number of voxels in the PSF (e.g. 41888)

`<Fudge>` --> The PSF file stores coordinates as 100*(step size) where the step size is in millimeters. For example, a 0.9 mm step size results in a fudge factor of 90.

`<MDRF Directory>` --> The directory where the MDRF files are stored.

`<Output Directory>` --> The location for files created by this routine.

Code

psf_LW_cruncher --> Performs position estimation with likelihood windowing on the raw PSF data. A 'crunched' PSF file is returned. Its format is described below.

PSF_sort --> The large 'crunched' PSF file is split into six files used by the reconstruction code. These files are described below.

Files Produced

CrunchFile (psf_LW_cruncher) --> The PSF file following position estimation. For each voxel in the PSF, (x,y,z) coordinates are given (3 short ints), followed by the number of non-zero values (call it nnz) in the projection image (1 short int), followed by the pixel locations for those non-zero values (nnz short ints), followed by the values at those pixel locations (nnz short ints).

coords (PSF_sort) --> Contains the (x,y,z) coordinates for each voxel in the PSF with ranges of 1-->X, 1-->Y, and 1-->Z. (short).

num_vals (PSF_sort) --> The number of non-zero values in the projection image for that voxel. (short)

num_vals_sum (PSF_sort) --> A running sum of the non-zero values for each projection image. (long)

pos (PSF_sort) --> The pixel locations for each non-zero value in the PSF. (short)

counts (PSF_sort) --> The value at each non-zero pixel location in the PSF. (short)

sn_cam (PSF_sort) --> The sum of the events for each voxel (sensitivity map). (float)

Note: All files are camera specific.

ProcessImage.pl

Description

Position estimation with likelihood windowing is performed on raw image data and the resulting projection images are prepared for reconstruction.

Usage

```
ProcessImage <Image File Base> <Cameras> <Dimension> <Number of An-
gles>  <MDRF Directory> <Output Directory>
```

e.g. `/utils/ProcessImage.pl Data/Tomo_image 012 79 40 MDRF_1123 IMG_1215`

`<Image File Base>` --> The base name for the image file. The acquisition software automatically appends the camera number and file extension (i.e. _01.dat) to the image data. The ProcessImage routine accounts for that appendage and so only the base name for the image should be included.

`<Cameras>` --> Projection images for multiple cameras may be generated with this routine. Simply enter the desired camera numbers (no spaces, please).

`<Dimension>` --> The dimension of the MDRF (e.g. 79 for a 79x79 MDRF).

`<Number of Angles>` --> Number of angles over which projection data was collected.

`<MDRF Directory>` --> The directory where the MDRF files are stored.

`<Output Directory>` --> The location for files created by this routine.

Code

psf_LW_cruncher --> Identical to the code used in ProcessPSF.pl. Performs position estimation with likelihood windowing on the raw image data. A 'crunched' image file is returned. Its format is described above in the ProcessPSF.pl documentation.

proj_prep_psf_style_data --> Combines projection image data from several cameras and prepares it for use in the reconstruction algorithm.

Files Produced

<Image Base Name>_0*.img (psf_LW_cruncher) --> The camera number and file extension (i.e. _01.img) are appended to <Image Base Name>. The format is identical to that of the 'crunched' PSF described in the ProcessPSF.pl documentation. (short)

all_<Image Base Name>_cam_* (proj_prep_psf_style_data) --> Contains all of the projection image data for the cameras of interest. The projection image data is no longer compressed. (float)

ProcessRecon.pl

Description

Position estimation with likelihood windowing is performed on raw image data and the resulting projection images are used with the PSF files to generate a reconstruction of the object.

Usage

```
ProcessRecon <Image File Base> <Cameras> <Dimension> <Number Coordi-
nates> <Number of Angles>  <Number Subsets>  <Number Iterations> <MDRF
Directory> <PSF Directory> <Output Directory>
```

e.g. `/utils/ProcessRecon.pl Data/Tomo_image 012 79 41888 40 5 10 MDRF_1123 PSF_1211 IMG_1215`

`<Image File Base>` --> The base name for the image file. The acquisition software automatically appends the camera number and file extension (i.e. _01.dat) to the image data. The ProcessImage routine accounts for that appendage and so only the base name for the image should be included.

`<Cameras>` --> Projection images for multiple cameras may be generated with this routine. Simply enter the desired camera numbers (no spaces, please).

`<Dimension>` --> The dimension of the MDRF (e.g. 79 for a 79x79 MDRF).

`<Number of Coordinates>` --> The number of voxels in the PSF (e.g. 41888)

`<Number of Angles>` --> Number of angles over which projection data was collected.

`<Number of Subsets>` --> Number of subsets to use in the OS-EM reconstruction.

`<Number of Iterations>` --> Number of iterations to use in the OS_EM reconstruction.

`<MDRF Directory>` --> The directory where the MDRF files are stored.

`<PSF Directory>` --> The directory where the PSF files are stored.

`<Output Directory>` --> The location for files created by this routine.

Code

Note: Several of the subroutines will run only if the files that it produces do not yet exist.

psf_LW_cruncher --> Identical to the code used in ProcessPSF.pl. Performs position estimation with likelihood windowing on the raw image data. A ‘crunched’ image file is returned. Its format is described above in the ProcessPSF.pl documentation.

proj_prep_psf_style_data --> Combines projection image data from several cameras and prepares it for use in the reconstruction algorithm.

gen_coords_bank --> Contains the pre-computed coordinates for rotation of the PSF during reconstruction.

gen_sn_files --> Generates the sensitivity file for the selected camera combination by summing the sensitivity files of each individual camera.

OSEM_*cam --> The reconstruction code. Uses the projection image files and PSF files to produce a reconstruction of the original object.

prep_recon_3D --> Transfers the reconstruction to a rectangular grid so that it will be usable on a viewing platform such as Amide or MRICro.

Files Produced

<Image Base Name>_0*.img (psf_LW_cruncher) --> The camera number and file extension (i.e. _01.img) are appended to <Image Base Name>. The format is identical to that of the 'crunched' PSF described in the ProcessPSF.pl documentation. (short)

all_<Image Base Name>_cam_* (proj_prep_psf_style_data) --> Contains all of the projection image data for the cameras of interest. The projection image data is no longer compressed. (float)

coordsbank_* (gen_coords_bank) --> The * represents the number of angles. Contains the coordinate rotation for the PSF for each angle in the reconstruction. (long)

sn_cam_* (gen_sn_files) --> Sensitivity file for the selected camera combination. (float)

recon_<Image Base Name>_cam_* (OSEM_*cam) --> Contains the reconstruction of the original object. All iterations are present. (float)

recon_<Image Base Name>_cam_*_3D (prep_recon_3D) --> Contains the final iteration of the reconstruction transferred onto a rectangular grid for use in a program like Amide or MRICro (necessary because most object spaces will be cylindrical). (float)

ProcessReconWithCOR.pl

Description

Position estimation with likelihood windowing is performed on raw image data and the resulting projection images are used with the PSF files to generate a reconstruction of the object. Center of rotation correction is implemented.

Usage

```
ProcessReconWithCOR <Image File Base> <Cameras> <Dimension> <Number
Coordinates> <X offset> <Y offset> <Number of Angles>  <Number Sub-
sets>  <Number Iterations> <MDRF Directory> <PSF Directory> <Output
Directory>
```

e.g. `/utils/ProcessRecon.pl Data/Tomo_image 012 79 41888 -0.5 1.0 40 5 10 MDRF_1123 PSF_1211 IMG_1215`

`<Image File Base>` --> The base name for the image file. The acquisition software automatically appends the camera number and file extension (i.e. _01.dat) to the image data. The ProcessImage routine accounts for that appendage and so only the base name for the image should be included.

`<Cameras>` --> Projection images for multiple cameras may be generated with this routine. Simply enter the desired camera numbers (no spaces, please).

`<Dimension>` --> The dimension of the MDRF (e.g. 79 for a 79x79 MDRF).

`<Number of Coordinates>` --> The number of voxels in the PSF (e.g. 41888)

`<X offset>,<Y offset>` --> Center of rotation corrections.

`<Number of Angles>` --> Number of angles over which projection data was collected.

`<Number of Subsets>` --> Number of subsets to use in the OS-EM reconstruction.

`<Number of Iterations>` --> Number of iterations to use in the OS_EM reconstruction.

`<MDRF Directory>` --> The directory where the MDRF files are stored.

`<PSF Directory>` --> The directory where the PSF files are stored.

`<Output Directory>` --> The location for files created by this routine.

Code

Note: Several of the subroutines will run only if the files that it produces do not yet exist.

psf_LW_cruncher --> Identical to the code used in ProcessPSF.pl. Performs position estimation with likelihood windowing on the raw image data. A 'crunched' image file is returned. Its format is described above in the ProcessPSF.pl documentation.

proj_prep_psf_style_data --> Combines projection image data from several cameras and prepares it for use in the reconstruction algorithm.

gen_coords_bank --> Contains the pre-computed coordinates for rotation of the PSF during reconstruction.

gen_sn_files --> Generates the sensitivity file for the selected camera combination by summing the sensitivity files of each individual camera.

OSEM_testCOR_*cam --> The reconstruction code. Uses the projection image files and PSF files to produce a reconstruction of the original object.

prep_recon_3D --> Transfers the reconstruction to a rectangular grid so that it will be usable on a viewing platform such as Amide or MRICro.

Files Produced

<Image Base Name>_0*.img (psf_LW_cruncher) --> The camera number and file extension (i.e. _01.img) are appended to <Image Base Name>. The format is identical to that of the 'crunched' PSF described in the ProcessPSF.pl documentation. (short)

all_<Image Base Name>_cam_* (proj_prep_psf_style_data) --> Contains all of the projection image data for the cameras of interest. The projection image data is no longer compressed. (float)

coordsbank_*_*_* (gen_coords_bank) --> e.g. coordsbank_40_m0p5_1p0. Contains the coordinate rotation for the PSF for each angle in the reconstruction. (long)

sn_cam_* (gen_sn_files) --> Sensitivity file for the selected camera combination. (float)

recon_<Image Base Name>_cam_* (OSEM_*cam_testCOR)--> Contains the reconstruction of the original object. All iterations are present. (float)

recon_<Image Base Name>_cam_*_3D (prep_recon_3D) --> Contains the final iteration of the reconstruction transferred onto a rectangular grid for use in a program like Amide or MRICro (necessary because most object spaces will be cylindrical). (float)

ProcessInterp.pl

Description

Jean Chen's code for performing interpolation on a single-pinhole PSF. Gaussian coefficients are computed for a PSF and used to interpolate the PSF onto a finer grid.

Usage

```
ProcessInterp <Input PSF File> <Cameras> <Dimension> <Number Coordinates> <Radial Steps> <Z Steps> <Output Directory>
```

e.g. `/utils/ProcessInterp.pl Data/PSF_1211 012 79 41888 34 44 PSF_INTERP_1211`

`<PSF File>` --> The raw PSF data file.

`<Cameras>` --> Projection images for multiple cameras may be generated with this routine. Simply enter the desired camera numbers (no spaces, please).

`<Dimension>` --> The dimension of the MDRF (e.g. 79 for a 79x79 MDRF).

`<Number of Coordinates>` --> The number of voxels in the PSF (e.g. 41888)

`<Radial Steps>` --> The number of steps in the radial direction.

`<Z steps>` --> The number of steps in the z direction.

`<Output Directory>` --> The location for files created by this routine.

Code

Note: Several of the subroutines will run only if the files that it produces do not yet exist.

Note: All code written by Jean Chen except for PSF_sort_float.

psf_gcoef_LM_2 --> Generates Gaussian coefficients for each column of the PSF. Recall that each column is the projection image of the point source at a given voxel. Least squares fitting is used.

gen_xyzidx --> Generates voxel indices for each column of the PSF.

psf_interp_gcoef(1,2,4,8) --> Gradual interpolation of the Gaussian PSF coefficients.

comb_gcoef --> Combines the interpolated coefficient files created by psf_interp_gcoef(1,2,4,8).

gcoef_vox_sort --> Sorts through all of the coefficients and orders them in preparation for generation of the new PSF.

gen_gauspsf --> Takes the new Gaussian coefficients and transforms them into a PSF file with the more standard format.

PSF_sort_float --> Splits the interpolated PSF into several files for use in reconstruction. Note: This code is slightly different from PSF_sort in order to accommodate slight differences in PSF file format between the original and interpolated PSFs.

Files Produced

psf_0(cam)_gcoef (psf_gcoef_LM_2) --> The Gaussian coefficients of the original PSF. For each voxel, three unsigned short coordinate values are written, followed by six float Gaussian coefficients.

xyz_0(cam) (psf_gcoef_LM_2) --> The coordinate values for the original PSF. (unsigned long)

xyzidx_0(cam) (gen_xyzidx) --> A listing of voxel indices generated from the coordinates in xyz_0(cam). (unsigned long)

fpos_0(cam) (psf_gcoef_LM_2) --> Keeps a listing of file pointers for psf_0(cam)_gcoef. (long)

psf_0(cam)_gcoef(1,2,4,8) (gcoef_interp(1,2,4,8)) --> The interpolated Gaussian coefficients. See psf_0(cam)_gcoef (just above) for details on file format.

psf_0(cam)_combo (comb_gcoef) --> The combination of the four interpolated Gaussian coefficient files. See psf_0(cam)_gcoef (just above) for details on file format.

psf_0(cam)_sorted (gcoef_vox_sort) --> psf_0(cam)_combo is sorted in order to properly align the voxels. See psf_0(cam)_gcoef (just above) for details on file format.

psf_0(cam)_interp.dat (gen_gauspsf) --> The interpolated PSF file. The file format is similar to the file format of the input PSF with a couple of differences. The number of non-zero values and the pixel locations of these values are given as longs while the count values at these locations are given as floats.

PSF files (PSF_sort_float) --> Exactly the same PSF files as described in ProcessPSF.pl are produced by PSF_sort_float as by PSF_sort. The only difference in the functions is that PSF_sort_float has been modified to accommodate the difference in file format between the measured and interpolated PSFs.

ProcessInterpMultPinhole.pl

Description

Jean Chen's code for performing interpolation adapted for a multiple-pinhole PSF. The multiple-pinhole PSF is split, using known system geometry, and Gaussian coefficients are computed for each split PSF and used to interpolate the PSF onto a finer grid.

Usage

```
ProcessInterpMultPinhole <PSFInFile> <CoordsFile> <PinholeFile> <Di-
mension> <Number Coordinates> <Radial Steps> <Z Steps> <Output Direc-
tory>
```

e.g. `/utils/ProcessInterpMultPinhole.pl Data/PSF_1211/CrunchFile_01 Data/PSF_1211/Coords_01 Data/PSF_1211/PH_Info_01 79 41888 34 44 PSF_INTERP_1211`

`<PSFInFile>` --> The crunched PSF data file returned by ProcessPSF.pl.

`<CoordsFile>` --> The coordinates file returned by ProcessPSF.pl.

`<PinholeFile>` --> A text file (see attached example) containing system geometry information.

`<Dimension>` --> The dimension of the MDRF (e.g. 79 for a 79x79 MDRF).

`<Number of Coordinates>` --> The number of voxels in the PSF (e.g. 41888)

`<Radial Steps>` --> The number of steps in the radial direction.

`<Z steps>` --> The number of steps in the z direction.

`<Output Directory>` --> The location for files created by this routine.

Code

Note: Several of the subroutines will run only if the files that it produces do not yet exist.

Note: This perl routine calls ProcessInterp.pl for each pinhole.

Predict_PH_Centers --> Makes use of the system geometry (described in the Pinhole Information File) to predict the projection centers for each pinhole.

Gen_Split_PSF --> Splits the PSF into new PSF files, one for each pinhole.

ProcessInterp.pl --> For each pinhole, this perl routine is called to perform the single-pinhole interpolation.

Recombine_PH --> Recombines the interpolated PSF files for each pinhole into one large interpolated PSF.

PSF_sort_float --> Splits the interpolated PSF into several files for use in reconstruction. Note: This code is slightly different from PSF_sort in order to accommodate slight differences in PSF file format between the original and interpolated PSFs.

Files and Directories Produced

The following directories are created:

psf_interp_cam_(cam)_ph_(pinhole #) (inside < Output Directory >)

< Output Directory >_Interp --> e.g. PSF_1211_Interp

For each pinhole, the following files are created inside the corresponding pinhole directory:

pred_cens (Predict_PH_Centers) --> The predicted projection centers (in pixels) for that particular pixels. The size is (`<Number of Coordinates> x 2`) (float).

CrunchFile (Gen_Split_PSF) --> The PSF file following position estimation. For each voxel in the PSF, (x,y,z) coordinates are given (3 short ints), followed by the number of non-zero values (call it nnz) in the projection image (1 short int), followed by the pixel locations for those non-zero values (nnz short ints), followed by the values at those pixel locations (nnz short ints).

psf_0(cam)_gcoef (psf_gcoef_LM_2) --> The Gaussian coefficients of the original PSF. For each voxel, three unsigned short coordinate values are written, followed by six float Gaussian coefficients.

xyz_0(cam) (psf_gcoef_LM_2) --> The coordinate values for the original PSF. (unsigned long)

xyzidx_0(cam) (gen_xyzidx) --> A listing of voxel indices generated from the coordinates in xyz_0(cam). (unsigned long)

fpos_0(cam) (psf_gcoef_LM_2) --> Keeps a listing of file pointers for psf_0(cam)_gcoef. (long)

psf_0(cam)_gcoef(1,2,4,8) (gcoef_interp(1,2,4,8)) --> The interpolated Gaussian coefficients. See psf_0(cam)_gcoef (just above) for details on file format.

psf_0(cam)_combo (comb_gcoef) --> The combination of the four interpolated Gaussian coefficient files. See psf_0(cam)_gcoef (just above) for details on file format.

psf_0(cam)_sorted (gcoef_vox_sort) --> psf_0(cam)_combo is sorted in order to properly align the voxels. See psf_0(cam)_gcoef (just above) for details on file format.

psf_0(cam)_interp.dat (gen_gauspsf) --> The interpolated PSF file. The file format is similar to the file format of the input PSF with a couple of differences. The number of non-zero values and the pixel locations of these values are given as longs while the count values at these locations are given as floats.

The following files are created inside < Output Directory >_Interp:

psf_0(cam)_interp.dat (gen_gauspsf) --> The interpolated PSF file. The file format is similar to the file format of the input PSF with a couple of differences. The number of non-zero values and the pixel locations of these values are given as longs while the count values at these locations are given as floats.

PSF files (PSF_sort_float) --> Exactly the same PSF files as described in ProcessPSF.pl are produced by PSF_sort_float as by PSF_sort. The only difference in the functions is that PSF_sort_float has been modified to accommodate the difference in file format between the measured and interpolated PSFs.

Pinhole Information File

A simple text file used to provide system geometry information. **Note:** The perl script is not robust and this file must be entered precisely or there could be trouble.

EXAMPLE FILE --> PH_Info_Cam_01.dat (with example values included)

```
Camera Number (0-3 for M3R) :1
PSF step size (mm) :1.0
MDRF step size (mm) :1.5
Number of pinhole :4
X pinhole coordinates (mm) ([x1,x2,...,xN]) :[3.61,6.11,-3.61,-6.11]
Y pinhole coordinates (mm) ([x1,x2,...,xN]) :[8.43,-2.62,-8.43,2.62]
Object space to pinhole distance (mm) :21
Pinhole to detector distance (mm) :61.55
X coordinate correction factor :0.0
Y coordinate correction factor :0.0
Z coordinate correction factor :0.0
Range :10
Sum Check :30
```

pfit2ad_interp.c

Description

This piece of code (written by Bill Hunter) is useful for interpolating the MDRF. The interpolated MDRF is aesthetically pleasing in 2D projection imaging (its task-based usefulness has not been evaluated).

Usage

```
pfit2ad_interp  <path/file_in> <header offset> <# rows in> <# columns in> <#chn> <order> <win min size> <win max size> <# rows out> <# columns out> <path/file_out>
```

e.g. `pfit2ad_interp MDRF_0906/MeanFile_00 0 79 79 9 3 5 15 158 158 MDRF_0906/MeanFile_Interp_00`

`<path/file_in>` --> The uninterpolated mean file.

`<header offset>` --> Number of offset bytes before data begins (usually zero).

`<#rows/#columns in>` --> The dimension of the uninterpolated mean file (e.g. 79 for a 79x79 MDRF).

`<#chn>` --> PMT number (9 for the modcams)

`<order>` --> The order of the interpolation (Bill recommends 3).

`<win min/max size>` --> Size of the interpolation window (Bill recommends 5/15).

`<#rows/#columns out>` --> The dimension of the interpolated mean file.

`<path/file_out>` --> The interpolated mean file.

Notes

- The likelihood thresholds will also need to be interpolated in order to use the interpolated mean file. This code does not work well for that interpolation. Matlab's interp2 function has been used for the likelihood threshold interpolation and appears to work well.
- A 2x interpolation works nicely. Larger interpolations tend to become unwieldy.
- The interpolated mean file will also be in the form of a big endian float.