bhc Documentation

Release

Ronald Fowler

Feb 08, 2018

CONTENTS:

This is a brief outline of the software package for fitting carousel or crown image data to a model of the energy dependent reponse function of the X-ray system (source, filters, detector). Using this fit a correction curve can be defined to map observed attenuation to expected monochromatic attenuation for a given material at a defined energy. This mapping function can then be used in either pre-processing the X-ray projections or as an input to the reconstruction software.

Requirements:

This software is written in Python 2. Non-standard python modules needed are:

- `scipy`
- `matplotlib`
- `tifffile`

CHAPTER

ONE

Beam Hardening Correction based on sample imaging

This python code is based on a IDL package written by Anthony Evershed at QMUL. The purpose is to take a set of X-ray images obtained under the same conditions and at the same time as the sample that is being imaged on the X-ray CT machine. By fitting a model of the energy dependent response of the system to the observed attenuation of a set of well characterised flat plates it is them possible to generate a correction function that maps from observed attenuation of the polychromatic beam to the expected attenuation of a monochromatic beam at a specified energy. Using this correction function as a pre-processing step before CT reconstruction then helps to minimise beam hardening artifacts in single material samples.

A user guide is included in the software to help set up the description of the crown or carousel device that is needed to measure the attenuation. The software offers a simple interactive command line interface, or can be run with predefined script files.

The software consists of two main source files and a post-processing file:

- `runCarouselFit.py` This is the main program to run which acts as a simple command line iterpreter. A set of commands read the necessary data files, sets parameters and then solves the least squares problem. Plotting of the input data and the output results is supported to allow evaluation of the fit.
- `carouselUtils.py` This file defines a few class objects which read and fit the data. Use is made of scipy module for least squares fitting. This file is imported by `runCarouselFit`.
- `applyTransform.py` This is a support program to apply the fitted correction transform to images for the reconstruction. It is run independently to the above program after that has produced correction polynomials.

CHAPTER

TWO

Src

2.1 runCarouselFit module

Control file to run simple carousel analysis and return attenuation correction function. Based on IDL code by Anthony Evershed.

This is the command line interface which is executed by: python ../src/runCarouselFit.py

within the "test" directory. The location is important since the code assumes that spectral data is available in the "spectra" subdirectory which is currently in test. Most of the fitting operations are implemented in the file carouselUtils.py. This code should run on both Python 2 and Python 3.

runCarouselFit.**checkVersions**()
: Check version of matplotlib and exit if too old

runCarouselFit.**debugToggle**(*cmd*)
: toggle debug

runCarouselFit.**fitAtt**(*string*)
: Check necessary data has been set then fit model to carousel data. Finally generate curves for attenuation over each line using the correction material (corMat/corEn) and then fit a polynomial to this for correction purposes.

runCarouselFit.**helpCar**(*cmd*, *string*)
: just print list of commands

runCarouselFit.**initGuess**(*words*)
: Set initial values to use for the variables of the target absortion width, detector width and filter width

runCarouselFit.**loadAll**(*string*)
: read both the carousel definition file and the data file with the calibration data

runCarouselFit.**mask**(*words*)
: show or set a mask array which is used to define if some of the sample data is not to be used in the fit. e.g. "mask 7" will mean sample 7 is omitted from the fit. "mask off" will restore all samples to the fit.

runCarouselFit.**quitCarousel**(*string*)
: exit the command level

runCarouselFit.**setCorMat**(*words*)
: Input the name of the material that will be the target of the attenuation correction e.g. Calcium hydroxyapatite which might be defined in a file cahypa.txt, precalculated using the program xcom. Without arguments, list current setting, if any.

runCarouselFit.**setFilter**(*string*)
: List filters defined or change settings of an existing filter. Can not at present add a new filter

runCarouselFit.**setOptions**(*words*)
: Set options controlling the fit process. Currently just select between the new and old versions of the least squares solver.

runCarouselFit.**setVary**(*strlst*)
define polynomial order of parameters to fit

runCarouselFit.**setWidth**(*words*)
set the half width of area along row to be averaged

runCarouselFit.**showAtt**(*string*)
1D plots of attenuation of sample n

runCarouselFit.**showCalConf**(*string*)
prints out some calibration data

runCarouselFit.**showCor**(*string*)
plot the fitted correction curve from the polynomial data

runCarouselFit.**showImg**(*string*)
plot the n calibration images on one plot; user must kill window to continue

runCarouselFit.**showSpec**(*string*)
plot spectra of source along with filtered spectra and response spectra. Note that these just use input data, not fitted data.

runCarouselFit.**transform**(*words*)
TEST code to try and fix up some data; should get right data in first place, Assume we have "I" data and want log(I0/I) values, where I0 is provided by the user.

2.2 carouselUtils module

Utilities for beam hardening correction method. Classes are provided to load the various data types that are needed by the algorithm. classes:

specData object to load and store X-ray spectrum for specified case carousel object to load and store description of test carousel carouselCalibrationData object to load and store the calibration data fit-Data object with methods and data related to the fitting process

class carouselUtils.**carousel**(*defFile*)
Bases: object

class for data describing the test carousel

getSamples()
elements in the carousel

isValid()
is object ok

class carouselUtils.**carouselCalibrationData**(*calFile*, *carouselInfo*)
Bases: object

This class reads the calibration data from a carousel calibration file.

The calibration file contains information about the X-ray voltage, take-off angle, target material and image resolution. It also gives the name & possibly the format of the image data file for the carousel.

getAvAtten(*line*, *sample*)
average over a named range of rows

getCentrePos(*line*, *sample*)
get centre point

getImage(*imgNum*)
access a given image by number

getLines()
return the number of lines of the data

getRows()
return the number of rows of the data

isValid()
is object ok

plotCalData(*showplt*, *markWidth*)
plot calibration images in one window using matplotlib. Only return when window closed. Mark image with width of "averaging", markWidth

plotCalProf()
will be used to plot profile along a row; not yet complete

printImageStats(*carInf*)
print out some data for each frame in the set of images

setWidthAve(*width*)
set the (half) width to be used when calculating the average attenuation along a row This is measured either size of the centre point. All points should lie on the sample image.

class carouselUtils.**fitData**(*carInfo*, *carCal*, *defMat*)
Bases: object

This object contains fit related data and functions

calcWidths(*x0*, *nlines*, *xe*)
Function to return the 3 widths for the target, the detector and the filter from the set of fitting variables. Each is assumed to be a polynomial of some order in the line number. Global varables are zero order polynomials. Also returns the energy array which can be a polynomial: E+aE**2+... ; this should be constrained >=0, not done at present.

dofit(*nlines*, *lstep*, *xin*)
perform fit

linesPolyFit(*soln*, *corMat*, *corEn*, *npoints*, *attrange*)
Function to calculate the attenuation over "npoints" for attenuation up to "attrange" for each line. Uses the fitted parameters for attenuation in "soln". Having calculated apparent attenuation for the correction material "corMat", map the observed attenuation to the true attenuation at the corEn energy. Then fit a polynomial to the data and save the coefficients.

objFunSq(*x*)
The function to minimize; returns the squared error for every point on each selected line

class carouselUtils.**materialAtt**(*formula*, *density*)
Bases: object

Attenuation for a material expressed as formula

For each material of interest will create this object to represent attenuation as a function of energy. Material is idenitified by chemical symbol and data is read from file produced by xcom.f.

getE()
return array of energy data

getEnergyRes()
energy resolution, assumed constant

getMaxEnergy()
highest energy value recorded

getMu()
attenuation array

getMuByE(*energyVal*)
return attenuation for the given energy

isValid()
if object ok

readFile (*formula*, *density*)
load the mu data for this material from a simple ascii file in ./xcom mu is mulitplied by density

class carouselUtils.**specData** (*target*, *voltage*, *angle*)
Bases: object

Spectral data for given condition

Data referes to the initial values of target, voltage and angle. This object contains the spectral data for the given target material, applied voltage and take off angle. Currently this is only available for Tungsten in the range 1-150KeV from data obtained from Spekcalc at fixed set of values. Interpolation may be used for intermeidate points. The resolution is as set in the data files. Note that only one spectrum is loaded as we do not vary the voltage in this fitting procedure.

Attributes

target [string] X-ray tube target material, chemical symbol e.g. W

voltage [float] applied voltage to the X-ray tube in KeV

angle [float] take off angle in degrees for the X-ray beam. This is kept constant

Methods

getE() Get array of energies over which the spectrum is defined (KeV)

getS() Get array of spectral intensities corresponding to energies of getE()

getEnergyRes() Return the step size between spectral points, assumed constant.

getMaxEnergy() Return the applied voltage (KeV)

isValid() Flag to indicate if object correctly set up.

getE()
return energy array; should remove

getEnergyRes()
return the (assumed constant) energy resolution

getMaxEnergy()
voltage is max energy

getS()
return intensity of spectra at energy points

isValid()
has object been set up OK

readData (*target*, *voltage*, *angle*)
Read the spectra for this voltage/angle pair from file. If file not found, return spec=0

2.3 applyTrans module

Program to read in the polynomial fit(s) from carousel analysis and apply them to a set of data. The Data can be float values of I0/I or int16 values of I0/I with I0 defined in some way. e.g. single I0 value or matrix of I0 values Corrected values will be written in same format as input.

applyTrans.**checkArgs** (*argstr*)
set default lines and rows for QMUL data

applyTrans.**correct** (*lineData*, *polyData*)
simply apply polynomial to data

applyTrans.**genbht** (*bhtFile*, *whiteLevel*, *polyC*)
Generate the bht file from the first polynomial data. Only one correction curve can be used in a bht file, so

line wise correction is not possible. Any values above the whiteLevel are set as 1.0, i.e. no attenuation. Use first correction polynomial at present.

applyTrans.**processRaw**(*infile*, *outfile*, *polyC*, *rows*, *lines*)
Read a binary raw file containing float32 data of the ln(I0/I) image data and apply the correction given by the polynomial(s) in polyC. Results output written to outfile in same raw format.

applyTrans.**processTif**(*infile*, *outpre*, *polyC*, *rows*, *lines*, *whiteLevel*, *debug*)
Read a tif file containing uint16 data of "I" image data and apply the correction given by the polynomial(s) in polyC. White level "whiteLevel" must be provided. Have to convert to float format for correction and retur to uint16 afterwards. A correction table in the reconstruction process would be more efficient. Results output written to outfile in same tif format.

applyTrans.**readPolyCoeff**(*fileParam*)
read the "param.log" type data of polynomial coeffs retunrs numpy array polyC

applyTrans.**trans**()
Main fuction of applyTrans to read command options and load data files, then write results

CHAPTER

THREE

Indices and tables

- genindex
- modindex
- search

PYTHON MODULE INDEX