Beam Hardening Correction CarouselFit: User Guide

Ronald Fowler

STFC Rutherford Appleton Laboratory

February 7, 2018

Abstract

This document is a brief user guide to the Python software package CarouselFit. This software takes image data from a number of known samples, e.g. from an X-ray CT machine, and fits them to a model of the beam hardening process which occurs when a broad spectrum source is used to image a sample. This model can then be used to generate corrections appropriate for a single material to convert the observed attenuation values into the actual attenuation that would be observed for that material with monochromatic X-rays at a given energy. This software is based on the IDL package and ideas described in [1].

1 Introduction

Beam hardening is well known problem that is described in many works, see [1]. This software takes as input a number of images of well characterised samples and uses these to fit a simple model of the expected beam hardening (BH) to the observed data. The result is an estimate of the "response function", $R(E)$, which gives the expected output signal from the detector as a function of X-ray energy for the selected combination of X-ray source, voltage, filters and detector. Note that due to aging effects of the X-ray source, detector, etc., this function may change over time, so ideally the calibration measurements should be made before and after each CT scan. In addition the model allows for variation in the form of $R(E)$ with the number of the scan line. This can occur due to the way the emitted X-ray spectra is known to depend on the "take-off" angle[1]. Use of pre-filtering the X-ray reduces both beam hardening effects and the variation of these with take-off angle. Low energy X-rays show the greatest variation with take-off angle. Davis recommends using both pre-filtering as well as the software correction described in this guide to best minimise beam hardening artifacts.

Using the fitted response function of the system it is then possible to determine a correction curve that will map from the observed attenuation to the true attenuation that would be seen at a given monochromatic X-ray energy. This correction curve is calculated assuming the sample is composed of a single material type for which the X-ray attenuation coefficient can be determined, using a program such as XCOM[2]. This allows for compound materials, as long as the composition is constant. In the case of samples made of more than one compound the correction curve will only be applicable if one material is the dominant absorber and corrections made to that material.

The next section describes how to download and run the software. In section 3 we describe how to set up the necessary files that give information on the number and type of test images that are used for calibration and the image formats. Section 4 details how to the run the fitting and post-processing image modules of the software. The first of these fits the model to the data while the second applies the correction directly to the CT image data. An alternative to processing all the images is to generates a look-up table or a 4th order polynomial to map observed attenuation values to mono-chromatic attenuation. Such data can be used as input to some reconstuction software, such the standard XTek filtered back projection package and the CCPi CGLS iterative code.

2 Downloading and running the software

The software is available from the CCPForge repository. It consists of a Python software package along with a number of data files that are used to help model the X-ray beams and the material attenuation. As well as a Python environment the software depends on a number of additional packages being available. An easy way to access most of the required packages is to download the Anaconda Python environment which is available for Linux, MacOS and Windows systems from `https://www.continuum.io/downloads`. The software has been developed using Python version 2.7. It is recommended that the user installs this before installing the CarouselFit software. Alternatively the user may install the required packages in their local Python installation, if they are not already available. The main Python modules that may need to be added to a local installation are:

- numpy - needed for array operations
- matplotlib - needed for plotting

- scipy - needed for optimization
- tifffile - only needed if corrections are to be applied to tiff images

The CarouselFit software can be checked out to a suitable directory using the command:

```
svn co https://ccpforge.cse.rl.ac.uk/svn/tomo_bhc/branches/release01 carouselFit
```

This will create a set of three directories under `carouselFit`:

- `src`: this contains the Python source code
- `doc`: this contains documentation of the software
- `test`: this contains several sub-directories with information on attenuation and X-ray spectra. The source code must be executed from this directory and any updates to the carousel or crown information should be made in the `carouselData` sub-directory.

After downloading the software the installation can be checked by running Python in the `test` directory and reading the example script file. On Linux and MacOS this could be done from a command prompt, assuming that a suitable version of Python is in the system PATH by typing:

```
python ../src/runCarouselFit.py
read script.short
quit
```

This set of commands should run without generating any error messages, such as failure to import modules. If missing modules are reported it will be necessary to add these to the Python system and run the test script again. Check the documentation for your Python system to see how to add modules.

On Windows systems Anaconda python can be accessed from the Start Menu after it has been installed. The software can be downloaded using the command line version of svn, e.g. via Cygwin, or using the GUI provided by TortoiseSVN `https://tortoisesvn.net/`. Once the software is installed, start an IP Python window (or similar) from the Start Menu and navigate to the `test` directory as mentioned above. Then use the `%run` command to execute the code, e.g.:

```
cd c:/svnpath/test
%run ../src/runCarousel.py
read script.short
quit
```

3 Configuration files

The original calibration device described in [1] was called a carousal as it was built from a set of 9 test samples arranged between two circular supports allowing for each of the samples to be imaged individually by the scanner. The samples would cover the full range of lines in the scanner, but not the full range of each row; typically only the centre half of each row would be covered by the sample.

A more recent calibration device has been developed at staff at the Research Centre at Harwell (RCaH) which is known as a crown. This device allows a larger number of samples to be mounted. In this case the sample usually covers all lines and rows of the image.

3.1 Carousel sample definition file

The materials mounted on the carousel, or crown, must be described in a simple ASCII file which is stored in the `test/carouselData` directory. An example of the format that was used for the carousel from QMUL is shown below.

```
# carousel definition file based on data from QMUL 17/11/14
10
Cu,Ti,Ti,Ti,Al,Al,Al,Al,Al,NOTHING
8.92,4.506,4.506,4.506,2.698,2.698,2.698,2.698,2.698,1
0.2093,0.4420,0.2210,0.1105,0.3976,0.1988,0.0994,0.0497,0.02,0.
```

This illustrates a case where there are 9 sample materials in the carousel. In this case all the samples are pure metals of known thickness and density. It is important to emphasize that the calibration depends on the sample materials been very well characterised. If a large error exists in either the thickness or purity of a sample this can undermine the accuracy of the fitting process. No exact guidelines have yet been defined on the best set of test materials to use, but obviously samples of the material the forms the dominant absorber in the imaged target would be ideal. However, this is often not practical in many cases, such as bone and teeth studies, where calcium metal is the prime absorber, but samples of the pure metal are subject chemical reactions in air. As long as the energy dependence of the sample attenuation coefficient, $\mu(E)$, is not too different to that of target dominant absorber then the calibration method should work. Some possible problems may occur if the sample has sharp steps in $\mu(E)$ due to band edges that lie in the response range of the system which are not seen in the target material. For example, compare the attenuation of Sn with that of Ca in the range 0 to 75KeV.

The above file uses the simple format:

- line1: a comment line, starting with #, to describe the file
- line2: a single integer giving the number of sample materials plus 1
- line3: a set of comma separated strings giving the names of each sample, with no spaces. the number of names must be the same as the previous number, with the final one named "NOTHING". In this case the samples are all pure metals and the chemical symbol has been used as the name. However any name be used as long as a corresponding file with the extension `.txt` exists in the directory `test/xcom`. This file gives the energy dependent $\mu(E)$ for this sample in steps of 0.5KeV from 0 to the maximum expected energy.
- line4: a set of comma separated values giving the density (in g/cm3) of each sample. A dummy value of 1 is used for the final material.
- line5: a set of comma separated values giving the thickness of each sample in cm. A dummy value of 0. is added on the end.

If a sample type other then the ones already described in `test/xcom` is used it is necessary to create a file of the attenuation values of that sample. See the Readme file in that directory for details.

The thickness range of the samples should aim to cover the range of attenuations that are expected in the test sample.

3.2 Sample image data file

In addition to a description of the samples in the carousel it is also necessary to define the format of the sample images and details of the X-ray source, filters and detector. This is done via another file in the directory `test/carouselData` which has the default extension `.data`. One such file must be generated for each calibration case, while the above carousel definition file will only change if the samples are changed.

Again a simple ASCII format is used to define the necessary values. An example is shown below:

```
# data for one QMUL calibration run
80              # voltage
22              # take of angle [not used by default]
W               # target material
19.25           # target density
600             # image res rows
800             # image res lines
carouselData/run001.img         # image file
```

```
float32         # data type in image file
2               # number of filters
Al              # filter material
0.12            # filter width
2.698           # filter density
Cu              # filter material
0.1             # filter width - 0.1
8.92            # filter density
CsI             # detector material
0.01            # starting value for detector thickness
4.51            # detector density
```

The format has one value per line with a comment to described the value. Most of these are self describing, such as the accelerating voltage, the take-off angle, the target material (tungsten, W) and its density, for the X-ray source.

The path to the file containing the sample images must be included in this file. All the images must currently be in a single file. The format used above, `float32`, assumes a binary format with 9 separate images of 600×800 32bit floating point values. Each value is $log(I_0/I)$ for that pixel with flat/dark field corrections.

Another supported format is `uint16`. In this case the sample images values are unsigned 16 bit values of the I value. Again these are all packed in order in a single file. The first image of the file is the (shading corrected) flat field image. The I_0 value is taken as the average of this initial image.

A variation on `uint16` format, which is slightly more compact, is labelled as `uint64_65535`. This format is again unsigned 16 bit images, but it assumes that the data has been corrected for flat and dark fields and that it has been normalised to a white level of 65535. This means that the raw binary file no longer needs an initial image giving the white level. This is the format that is generated by the Python script `average_mat.py` which converts tif image files into this format. See Appendix A for details of using this program.

Usually a set of filters are used to limit the energy range of the X-ray beam. In the case of the QMUL data they normally employ two filters with 0.12cm of Al and 0.1cm of Cu, as shown in the above file. As the fitting process includes varying the exact Cu filter width it is recommended that a zero width Cu filter element is included even if no Cu was used in the actual imaging.

The definition of the detector material is important and tests to date have been made with CsI. However other materials may be used if their attenuation profile is included in the `text/xcom` directory. Since the width of the detector maybe used as a fitting parameter it is not essential to specify an accurate value, though this will be used in the command `showspec`, if it is run before a fit has been performed.

4 The command line interface

4.1 Command list

When the Python software is started from Python or a similar environment, a simple command prompt is issued. Typing `help` will give a list of the available commands.

The commands are:

- **read** *filename* This command opens the given file and reads commands from it until end of file. Control is then returned to the command line. Do not include blank lines in the command script.
- **load** *file.def file.data* This reads the definition file for the carousel and the data relating to the actual calibration images. These two files must exist and are described in the previous section. they are normally located in the `test/carouselData` directory. This is usually the first command to issue since most others need this data to be present.
- **quit** Exit the program.
- **help** Give a list of available commands.
- **showcor** *[l1 l2...]* This command will plot the attenuation correction curve for any one or more lines. If no arguments are given it will plot the first, middle and last correction lines. The matplotlib zoom feature can be used to focus on a particular region of the plot. It can only be used after a fit has been performed. The correction is shown in the space of log(I0/I).
- **showimag** This command will plot the images of each sample in one window. It may be useful to check for problems with the samples. It can only be used after data has been loaded.
- **fitatt nlines** *[linestep]* This command attempts to fit the model to the selected samples (see mask command). The number of lines of data to fit must be given. This maybe followed by a "step", e.g. 10 to use every 10th line. This can be useful when using many lines as fitting all of them can be very slow and the fit may not be improved using more data. The time to fit also increases with the number of variables that have been selected with the "vary" command. Fitting to a few lines can be a good way to see if the model fits and give a better initial guess for a fit to a larger subset of the data.

- **vary** *[target—detector—filter—energy—spectra npoly]* On its own this command lists the order of polynomial used in fitting the line wise dependence of each of the three main parameters, *target width*, *detector width* and *filter width.* The setting "-1" indicates that the value should be held constant, as set by the initguess command. Using "0" indicates the value will be fitted, but is independent of the line number. Setting to "1" gives a fit allowing a linear variation of the value with line number. For example:

  ```
  vary filter 0
  vary detector -1
  vary target 1
  ```

  will allow a single fitted value for the filter width, the detector width held constant, and the target (filter) width to vary linearly with the line number. The fit time increases significantly with the order used and values greater than 1 are not recommended. An experimental option is to allow extra terms to be added to the normally linear dependence of the detector response to the photon energy, e.g. $E + \alpha E^2$. Note that energy dependence is NOT related to line number in this case. However this polynomial is not constrained to be positive and the fit may fail. Keeping energy variation off (-1) is recommended. The final option called "spectra", which defaults to 0, i.e. on, when no pre-defined spectra are present, which is the case for the open source release of the package. Setting spectra to 0 causes the calculated spectra to be modelled as a simple non-symmetric Gaussian form with 3 parameters, *peak*, *inverse left width* and *inverse right width.* If pre-computed spectra are available, e.g. from spekCalc, these can be used in preference to the Gaussian by setting vary spectra -1.

- **initguess** *[s_1 s_2 s_3[s_4 s_5 s_6 s_7]]* Set the initial guess to be used by fitatt. s_1 is the width of the target filter (usually tungsten), s_2 is the log width of the detector (usually CsI) and s_3 is the width of the fitted filter (usually copper). Commonly used values for the initial guess are 0.01 -6.0 0.01. If using the experimental feature "vary spectra 0" than 4 additional values can be given which are the initial value of the energy term (should be zero) plus the Gaussian centre and widths, e.g. 0.01 -6.0 0.01 0.0 40.0 0.05 0.05. When loading data the Gaussian peak is set to half the maximum X-ray energy. Using this command with no parameters gives the current settings on the values.

- **mask** *[n1 n2..]* Without arguments this shows the set of masks that control if a given sample will be used in the next fit operation. By default all values are true which means that sample will be used in the fit. Samples are labeled from 1 to n and to mask the m sample that number should be given as an argument to the mask command. A negative value can be used to unmask a previously masked sample.

- **setcormat** *material energy* This command must be used before a fit operation to define the material and energy to which the correction curve should be determined. For example `setcormat Al 40` sets the correction curve to be calculated for Aluminium at 40KeV. At present if the correction material or energy are altered it is necessary to rerun the fit command.

- **transform** This is an experimental command which will be removed in future.

- **showspec** *[line]* - plot three spectra, the input X-ray spectrum, the filtered spectrum and the response spectrum. Should only be used after a fit has been made. This command needs improving since the "filtered" plot is not meaningful. Also the printed attenuation values are not useful since these are not fitted to. If a line number is given, plots are for that line. The default is line 0.

- **showatt** *[nsamp nline]* - plot the sample attenuations along a specific line. By default this shows the attenuation for all samples at line 400. Samples are labeled 0 to $n-1$ in this case.

- **debug** - set debugging option, for diagnostic purposes only.

- **showconf** - list some of the settings, such as the filters, detector, source and voltage.

- **setwidth** *[width]* - without arguments, prints the width, in pixels, used to average over each line to get the mean attenuation. For the QMUL data, where the sample does not cover the whole image, it is important to ensure this does not exceed the true sample width. For the RCaH data, where the image does cover the whole width, a larger value can be used.

- **setfilter** *[material width]* - without arguments lists the filters defined. Can also be used to change the width of existing filters, though not add new ones. Used for debugging.

- **setoptions** *[solver=old|new]* - set option. Currently only allows switching between old and new least square solvers in scipy. The old version is more widely available and is the default.

4.2 Using the software

As described in section 3 it is necessary to write the definition files that describe the carousel and the particular test case that is being treated. The latter file must also point to the data file that contains the sample images in a suitable format. It is assumed that corrections for dark and flat field images have being applied to the images before they are passed to the software. A simple partial analysis might consist of the following steps:

```
load carouselData/carousel0.def carouselData/run001.data

showimg
showatt
```

The first command loads the definition and run data from files, while the next two commands plot the 2D images and 1D cuts along line=400.

```
setcormat CaHydro 40

vary target 1
vary detector 0
vary filter 1

initguess .01 -6 .01
fitatt 800 10

showspec
showcor
```

These commands then set the material and energy to which we wish to correct the data via the `setcormat` command, and then alter the default orders of the fit variables. The `fitatt` command fits the given initial guess using the lines of the image data, 800, but only every 10th line. This fit may take 60 seconds. Finally the fitted spectrum and correction curves are plotted.

The correction curves are stored in the same format as used in the earlier IDL code as separate 8th order polynomial fits to the correction data in a file called polyfit.npz. These curves are the ones shown by the `showcor` command above. To actually apply the correction to image data requires the use of another Python program, `applyTrans.py`. In addition to the above 8th order polynomials, 4th order fits are also written to the output file *param.log*. The 4th order polynomial values are written at the end of the file, one set per line if the solution includes variation with line number. These values can be used in the xtekct file for the parameters X0 to X4, X0 being the rightmost value in *param.log*. If the variation of the correction with the line number is significant it would be better to correct each project individually as described in the next sectionn.

4.3 applyTrans.py

The Python script applyTrans.py can be used to update image files using the correction curves calculated by the above fitting process. It can also calculate a file of type `.bht` which can be used by XtekCT machines to correct the image data used in CT analysis. In latter case only one correction curve is applied to all the data, in the same way that using the using the 4th order polynomial fit does.

The syntax of the command can be seen using the `-h` option, which gives:

```
applyTrans.py [-r rows -l lines -p poly.npz -w whiteLevel -x file.bht]
              [-d] [file1.ext] [filen.ext]
```

In the above data it is usually necessary to specify the image size in rows and lines. If all the image data is stored in a single file with data type float32, as used for some data from QMUL, then the following command can be used to process it:

```
python ../src/applyTrans.py -r 600 -l 800 images.raw
```

In this case the default file `polyfit.npz` is read to find the correction curves. If 800 curves are present then one will be applied to each line in the image. If only one correction curve is present then this one correction will be used on all image lines. The processed output will be written to `bhc_images.raw`. Note that the `whiteLevel` parameter is not needed in this case as the `.raw` extension is taken to imply `float32` data of $log(I_0/I)$.

To generate a `.bht` correction file the following command can be used:

```
python ../src/applyTrans.py -b -x xtekct.bht -w 59200
```

In this case only the file `xtekct.bht` is generated. It is necessary to provide an accurate estimate of the white level since any pixels above this are mapped to no attenuation.

A Generating the averaged images of crown materials

The current approach used to get accurate images of the crowm materials on the Xtek CT scanner is to take 11 exposures of each material sample. These are saved as tif files into separate directories, one for each material sample. In addition it is necessary to the flat field and the dark field images for correction purposes. Again, 11 images are taken and saved into separate directories labelled "dark" and "flat". It is then necessary to average over each set of images, correct for flat and dark fields and then save the attenuation data in a suitable format for use with the fitting software.

A simple python script called `average_mat.py` to perform this operation is included the software distribution. It takes as its input a single file which contains a list of directories, one per line. The first two of these must be the dark and flat field image directories. The remaining lines give the material image directories in the order that is defined in `.def` file. Each directory must contain at least one tiff file of the appropriate image. In our case we use 11 tiff files, but the program should except any number of tiff files and take an average. The output of the program is a raw image file with one image per material directory listed. These images are in UINT16 format, normalised to a white level of 65535. For input to the fitting program, this format must be labelled as `uint16_65535`.

By default the output images will all be the same resolution as the input images. Ih the case of the Xtek system used in testing this is 2000x2000. In the case where variation with take-off is not being modelled, a central subset of the image is sufficent and more efficient to use. To do this four additional arguments can be given to the python program to select a subset of the data, e.g.

```
python average_mat.py dirlist.txt output.raw 500 500 1500 1500
```

This command will read the set of directories for the dark field, flat field and material images for the file dirlist.txt. These will be averaged and corrected with the images written to output.raw. In this case only the central pixels from (500,500) to (1500,1500) will be written so images are 1000x1000 rather than 2000x2000.

References

[1] Graham R. Davis, Anthony N.Z. Evershed, and David Mills, *Quantitative high contrast X-ray microtomography for dental research*, Journal of Dentistry, Volume 41, Issue 5, May 2013, Pages 475482.

[2] M.J. Berger, J.H. Hubbell, S.M. Seltzer, J. Chang, J.S. Coursey, R. Sukumar, D.S. Zucker, and K. Olsen, *https://www.nist.gov/pml/xcom-photon-cross-sections-database*