

# THE FILTERED BACK-PROJECTION ALGORITHM

## THE 'PEARL' UPGRADE

This document describes the algorithms which EMI developed in 1972-73 and launched in July 1974 as a retrofit to the first batches of EMI-Scanners which had been delivered with the iterative reconstruction algorithm. The retrofit was called the Pearl upgrade.

Pearl upgraded the image matrix size from 80x80 to 160x160 while simultaneously reducing the time taken to reconstruct the image. The two slices were available within 30 seconds of the scan finishing, instead of 4 minutes after the scan. Thus it was a very welcome upgrade for the hospital staff. The upgrade involved changing some hardware as well as software: the number of samples taken across the image was increased from 160 to 240 (by changing the optical graticule which marked where the samples should be taken). The electronics for converting the X-ray readings into numbers was upgraded to allow faster sampling.

The same algorithms were subsequently used on the Emerald / CT5000 body scanner and the second-generation brain scanner (Super Opal / CT1010). Those scanners all collected the X-ray data as sets of parallel beams. The algorithms were subsequently extended to reconstruct fan-beam data.

This document currently describes both the algorithm and the software which has recently been written to recreate it. Those two aspects may become separate documents in future.

## RE-ORDERING THE SIMULATED DATA

The software starts by reading simulated data from *SimEdgeReadings.csv*, re-ordering it, and writing the result to *EdgeReadings.csv*. If you print the file *EdgeReadings.csv* it has the different angular views across the page and the different radial positions down the page.

The file *SimEdgeReadings.csv* includes information about which of four "machine types" have been simulated. This "machine type" is either Prototype, Pearl, Emerald E320 or Custom. The machine type is used to select the detailed parameters of the machine geometry which are stored in file *Parameters3.csv*.

Some padding zeroes are added at the beginning and end of each angular view, and the data is normalised so that water is zero.

The remainder of the algorithm divides into two parts: the first part of the algorithm is a filter which counteracts the errors which are introduced by the second part of the algorithm, which is back-projection.

## CONVOLUTION

The back-projection algorithm blurs the picture by distributing the attenuation of each beam uniformly along the length of that beam. The back-projection can't do anything else, because it works on each view independently. If back-projection tried to work out how to correctly distribute the attenuation along the length of the beam then it would take hundreds of times longer to run, and it is already the longest process in CT reconstruction. It is easy to see that this type of back-projection blurs the image, and that it will reduce the higher frequency components in the image. The convolution filter seeks to exactly counteract this known error which is caused the back-projection. The derivation of the filter coefficients can be done mathematically or numerically, but this document simply assumes that the convolution filter has been supplied as a list of coefficients.

The filter is implemented by the mathematical process of convolution. This can be modelled in a spreadsheet using a “sum of products” function. In the BASIC language it is performed by the following two lines of code:

```
X = J + K
C(X) = C(X) + Data(J) * Filter(K)
```

These lines of code sit inside FOR loops which make the variables J and K take up all of the values which correspond to the positions of all of the measured X-ray beams.

In these lines of code, the array “C” is the result of the convolution. The code shows that each value within the C array is built up gradually as the J and K FOR loops go through all of their combinations.

The convolution section of the program picks up data from file *EdgeReadings.csv* and writes results to file *convolved.csv*. These files can be examined by Excel or Notepad or any other text editor, to check how the convolution process is working.

“Data” is a column extracted from the file *EdgeReadings.csv* at the particular angular view which is currently being convolved. It is an array of X-ray attenuation values. This is after taking logs of the readings from the detectors, so that the values represent attenuation rather than the number of X-ray photos.

“Filter” is the convolution filter. This is a symmetrical function, in the sense that it has a mirror image about the point at which K=0. In other words, Filter(-K)=Filter(+K). The filter can be adjusted to give different trade-offs between high spatial resolution and low noise. The central coefficients of the Ramp filter (which is designed for high spatial resolution) are:

0, -.40528, +1, -.40528, 0

The central terms for other filters are similar in that a central value of +1 is surrounded by two equal negative values. These two negative values are usually in the range -0.30 to -0.44.

Outside the central coefficients, the coefficients are all negative and they reduce towards zero as K increases. The coefficients tend towards  $F(K) = -B/K^2$ , where B is a constant.

Five choices of convolution filters are stored in the file *ConvolutionFilters.csv* which can be read and edited by Notepad or Excel. The first few lines of this file are shown below. In this table, line 1 is the filter number, 0-4. Line 2 is the name of the filter. The Smooth2 filter is left over from earlier attempts to optimise the smooth filter. The “NoFilter” filter applies no filtration, so it allows images to be reconstructed from data which has not been changed by the convolution filter. These filters can easily be replaced by alternative filters by editing the file *ConvolutionFilters.csv*.

0	1	2	3	4
CAG	Ramp	Smooth1	Smooth2	NoFilter
1	1.601379	0.646412	0.63012	1
1	1	1	1	1
0	-0.40528	0	0	0
-0.436		-0.33333	-0.33333	0
0	-0.04504	0	0	0
0	0	-0.06667	-0.06462	0
0	-0.01621	0	0	0
-0.01957	0	-0.02857	-0.02872	0
0	-0.00827	0	0	0

Each filter has an associated scale factor, which is shown in line 3 of the table. These were derived empirically, to give the expected attenuation values in the CT scan.

Line 4 in the table is the central coefficient of the convolution filter, in other words it is the value of Filter(K) when K=0. Line 5 in the table is the next coefficient of the convolution, in other words it is the value of Filter(K) when K=1. Due to symmetry, line 5 in the table is also the value of Filter(K) when K= minus 1. The following lines are the subsequent coefficients of the filter.

When the program is run, it selects one column of the above table (depending on what filter the user has asked for), and uses that column alone. A file called *Parameters3.csv* controls many parameters, including whether the user gets asked to select the convolution filter, or whether it defaults to the “Smooth1” filter.

Everything else in the convolution section of the program is related to the tedious housekeeping which seems inevitable when using this out-dated language. Housekeeping includes reading and writing information to disk, and ensuring that the rules of the language are obeyed. For example, the language does not allow negative array subscripts, which is inconvenient because CT most naturally treats the centre of the image as the origin of the axes for the picture matrix.

## BACK-PROJECTION

The back-projection algorithm distributes the attenuation of each beam uniformly to the pixels which lie along the length of that beam. It consists of two operations:

1. Interpolation / expansion
2. Mapping

### Interpolation / expansion

Few of the pixels will be exactly along the centre line of any selected beam. Accordingly, the back-projection applies an interpolation so that the beam contributes more to a pixel which lies near the centre line of the beam than it does to a pixel which is near to the edge of the beam. Earlier work by Godfrey Hounsfield and Stephen Bates had shown that the choice of such an interpolation function was an important factor in optimising the signal-to-noise ratio of the resulting CT scans. They used an interpolation filter which was modelled on the profile of the beam. Their model was a raised-cosine:

$$I(x) = 0.5 \cdot (1 + \cos(kx))$$

where  $x$  is the distance between the pixel and the closest approach of the beam to that pixel, and  $k$  is chosen such that the function has the same width as the X-ray beam has when it passes through the patient. If the beam width at half-maximum is about 2mm then  $k$  is about  $\pi/2$ .

The interpolation filter can be implemented by convolution. The range of parameter  $K$  in this convolution is only a few beam widths, whereas in the previous application of convolution both  $J$  and  $K$  ranged across the entire image.

It is preferable to perform this interpolation outside the inner loop of the back-projection, to save time. It was also found more efficient to quantise the interpolation so that it could be applied to sampled data instead of as a continuous function. It was found empirically that the interpolation needs to be quantised at least to an accuracy of one tenth of the distance between adjacent edge readings, and that there was little or no gain from increasing the

accuracy beyond one twentieth of the distance between adjacent edge readings. These were described as Expansion Factors of 10 or 20 respectively.

The software uses a look-up table to define the shape of the interpolation function. This is stored in file *ExpansionTable.csv*. In fact several different expansions are available, and these are selected via three parameters in the file *Parameters3.csv*. The parameter “ExpansionFactor” selects whether the expansion is by 10 or by 20. The parameter “ExpansionType” selects whether the interpolation curve is a sinusoid, linear, or “3<sup>rd</sup> difference” (this will be described in a later document). The parameter BeamOverlap describes whether the X-ray beam is sampled once or twice each time it moves by one FWHM beam width. FWHM is the distance between the points either side of the central line of the beam at which the intensity of X-rays is half of that measured at the central line of the beam.

## Mapping

Mapping involves a large number of repetitive computations. The back projection needs to add a contribution from each angular view into each pixel in the image. By simple multiplication if we have 80,500 pixels in the image and 540 angular views then we have to make 43.5 million additions during back-projection. These numbers represent EMI’s 1974 body scanner, which had 540 views spaced at a third of a degree. The 320x320 image matrix contained about 80,500 pixels because a circle of diameter 320 pixels was reconstructed.

The back-projection is three nested FOR loops. The outer loop runs through the angular views. The inner two loops run through every X,Y position in the image.

A method invented by Brian Lill and Paul Beaven allows all of the trigonometry to be taken out of the inner X,Y loop. Thus almost all operations in the inner loop are simple additions.

Within the square image, only the pixels which are inside a circle of diameter equal to the width of the square are reconstructed. The four corners of the square are not reconstructed because they are assumed to contain no attenuation from the patient and because they do not get adequately measured by X-ray beams. Before starting on its main work, the back-projection software calculates the location of this circle.

For example, in a 20x20 image there is a line of pixels for which Y=3 . For this line, the program will calculate that the pixels which need to be reconstructed start at X=5 and end at X=16. (The software calls these Xstart and Xend.) It also calculates the exact radius and angle with respect to the X-axis of the first pixel which will be reconstructed in this row. This radius and angle are subsequently used to calculate exactly the radial position at which this pixel is “seen” by the set of beams which were measured at each of the angular views which were collected. That radial position will follow a sinusoidal path across the angular views.

The main steps of back-projection are:

1. A “LillAndBeavenIncrement” is calculated:  
$$\text{LillAndBeavenIncrement} = (\text{PixelSize} * \text{ExpFactor} / \text{BeamPitch}) * \text{SIN}(\text{Radians})$$
  
(where Radians is the angle of the current angular view.)
2. The radial position and angle of Pixel(Xstart,Y) is used to find the LillAndBeavenOffset:  
$$\text{LillAndBeavenOffset} = (\text{radius of Pixel}) * \text{COS}(\text{Radians} - (\text{angle of Pixel}))$$
3. This is converted into an address in the expanded array of edge readings for this angle:  
$$\text{OffsetAsExpEdgeReadings} = \text{LillAndBeavenOffset} / \text{ExpPitch}$$
  
$$\text{DataAddress} = \text{CentralExpanded} - \text{OffsetAsExpEdgeReadings}$$

4. At this point, the inner X- loop starts:  

$$\text{DataAddress} = \text{DataAddress} + \text{LillAndBeavenIncrement}$$

$$\text{AddressInt} = \text{INT} (0.5 + \text{DataAddress})$$
5. The value found in array C2() at that address is added to the current pixel:  

$$\text{Pixel}(\text{xPixel}, \text{yPixel}) = \text{Pixel}(\text{xPixel}, \text{yPixel}) + \text{C2}(\text{AddressInt})$$
6. Steps 4 and 5 are repeated for each X value which lies between Xstart and Xend.

The mapping software produces three files, the CT picture and two files which may be useful in understanding the above explanation:

- *image.csv* is the picture from this CT scan
- *ExampleExpandedData.csv* is the expanded version of one of the angular views. Which angular view is stored is controlled by parameter *ExampleView* in file *Parameters3.csv*.
- *TestPixelLogs.csv* shows exactly how the pixel values build up for the nine pixels around  $X = X_{\text{test}}$  and  $Y = Y_{\text{test}}$ . The values of parameters  $X_{\text{test}}$  and  $Y_{\text{test}}$  can be set editing the file *Parameters3.csv*.

## SUMMARY

Data is read from file *SimEdgeReadings.csv*, re-ordered and the result is written to file *EdgeReadings.csv*.

The file *SimEdgeReadings.csv* includes information about which of four “machine types” have been simulated. This “machine type” is either Prototype, Pearl, Emerald E320 or Custom. The machine type is used to select the detailed parameters of the machine geometry which are stored in file *Parameters3.csv*.

The convolution process reads file *EdgeReadings.csv* and writes file *convolved.csv*.

The mapping process reads file *convolved.csv* and writes file *image.csv*. It also writes explanatory files *TestPixelLogs.csv* and *ExampleExpandedData.csv*

The convolution and expansion/interpolation filters can easily be changed (without re-compiling the BASIC program) by editing files *ConvolutionFilters.csv* and *ExpansionTable.csv*.

A lot remains to be done, including implementing Chris LeMay’s fast methods of convolution and 3<sup>rd</sup> difference interpolation.

## RESULTS

The images are close to what is expected. There are some streaks from sharp edges which have been proved to be due to the fact that the simulation (correctly) applies the beam profile to data “before logs”, whereas the reconstruction attempts to match that filter after the non-linear log stage. The streaks reflect the effect of that non-linearity on sharp edges in the data.