PROCESS TOMOGRAPHY Ltd.

APPLICATION NOTE AN4

AN ITERATIVE METHOD FOR IMPROVING ECT IMAGES

Issue 2

April 1999

SUMMARY

The method normally used to obtain ECT images from capacitance measurements is the **Linear Back-Projection (LBP)** algorithm, which produces relatively low-accuracy images. This note describes a method for improving the accuracy of LBP images using a simple **iterative image computation method**. This information is provided by Process Tomography Ltd. to encourage users of ECT systems to experiment with this technique. There are many possible ways in which this method can be developed and improved and we welcome feedback from customers with suggestions for how this technique can be further optimised.

PROCESS TOMOGRAPHY LTD

64, Courthill House, Water Lane, Wilmslow, Cheshire. SK9 5AJ United Kingdom.

Phone/Fax 01625-418722

(From outside UK +44-1625-418722)

email: enquiries@tomography.com Web site: www.tomography.com

Registered in England No. 2908507. Registered Office 15, Croft Road, Wilmslow, Cheshire. SK9 6JJ United Kingdom.

ACKNOWLEDGMENT

The work reported here was assisted by knowledge of the work of several other researchers. We would therefore like to acknowledge contributions from the following people:

1. Dr J. Salkeld of UMIST. Description of iterative method for ECT image reconstruction in PhD thesis, 1991.

2. Dr W.Q. Yang of UMIST. Patent application and publication and demonstration of an iterative algorithm using a forward field solving technique, 1997.

3. John Pendleton of John Pendleton Associates. Development of Matlab software algorithm for ECT image reconstruction, 1998.

AN ITERATIVE METHOD FOR IMPROVING ECT IMAGES

1. THE LBP ALGORITHM

Before considering how the iterative algorithm works, it is necessary to understand the principle of the simple linear back-projection (LBP) algorithm. In an ECT system, images are constructed of the cross-sectional distribution of dielectric material inside an ECT sensor from values of capacitance measured between combinations of electrodes located around the periphery of the sensor. In PTL ECT systems, the ECT image is normally based on a grid of 32 x 32 square pixels.

The method used to derive the values of each pixel in the image from the capacitance measurements is the so-called **Linear Back-Projection** (**LBP**) algorithm, which was originally developed for use in X-ray tomography systems. In this algorithm, it is assumed that the relationship between the measured values of inter-electrode capacitance **c** and the permittivity **k** of each pixel can be written in matrix form as follows:

$$\mathbf{C} = \mathbf{S}.\mathbf{K} \qquad (1)$$

where **C** is a column vector containing the set of **m** normalised inter-electrode capacitance measurements **c** for one image frame, **K** is a column vector containing the set of **n** normalised pixel permittivities **k** and **S** is a normalised **m x n** matrix known as the sensitivity matrix (or map). All of the values in these matrices and column vectors are normalised to lie within the nominal range 0 to 1.

The value of **m** (the number of possible unique inter-electrode capacitance measurements) is **28** for an 8-electrode sensor or **66** for a 12-electrode sensor. The value of **n** will be **1024** for a 32 x 32 pixel grid.

An ECT system is normally used to image a combination of two materials having different permittivities. The ECT system is calibrated at a lower point when full of the lower permittivity material and then at a higher point when full of the higher permittivity material.

The sensitivity matrix **S** is the normalised set of inter-electrode capacitance measurements obtained when all of the pixels except one are occupied by the lower permittivity material and the remaining pixel (which we will refer to as the probe pixel) is occupied by the higher permittivity material. The elements in the matrix **S** will be referred to as capacitance coefficients. The sensitivity matrix will consist of **m** (eg 66) sets of capacitance coefficients, each set containing 1024 capacitance coefficients corresponding to each individual pixel location. Note that the values of the coefficients for pixel locations outside the sensor boundary will be zero.

Equation 1 shows that for any arbitrary permittivity distribution inside the sensor, the individual inter-electrode capacitances are obtained by adding up the effects of each individual probe pixel, weighted by its actual permittivity, k. That is, by summating the combined efects of the 812 pixels inside the circle (out of the 1024 total number of pixels) on the measured inter-electrode capacitances, when each of these pixels is a probe pixel in turn, but weighted by its actual permittivity, rather than that used to derive the sensitivity matrix pixel (the higher calibration permittivity).

If each probe pixel contains the higher permittivity material, then the effect of adding up the inter-electrode capacitances which occur for each probe pixel location should give values of capacitances which equal those measured when the sensor is full of the higher permittivity material (the values measured at the upper calibration point). However, in practice, because of field distortion, this will only be approximately true and this is the first of several possible sources of error in the LBP algorithm. Clearly, if the probe pixels contain dielectric material of lower permittivity than that of the higher permittivity material, then the resultant inter-electrode capacitances will have lower values than those measured at the higher calibration point.

Equation 1 relates the measured capacitances to the permittivity distribution inside the sensor. However, the object of an ECT system is to derive the permittivity distribution from the measured inter-electrode capacitances. In principle, this can be done by simply inverting equation 1. That is:

$$\mathbf{K} = \mathbf{S}^{-1}.\mathbf{C} \qquad (2)$$

where $\mathbf{S}^{-1}$ is the inverse matrix of **S.**

Unfortunately, it is only possible to obtain the inverse of a matrix if it is square, that is, one where **m = n**. As this is not the case for ECT systems, it is not possible to invert the sensitivity matrix and hence some other method must be found to generate the permittivity distribution from the capacitance measurements.

The method used in the LBP algorithm is based on the assumption, already stated, that the measured inter-electrode capacitances are equal to the sum of the capacitances for that particular electrode combination caused by each pixel location inside the sensor acting individually, using the actual normalised permittivity for each pixel location in turn. This process is known as **forward projection**

Conversely, the image pixel values are obtained by assuming that the inverse relationship is true, namely that the permittivity of each individual pixel can be obtained by some relationship of the form:

$$\mathbf{K} = \mathbf{Q}.\mathbf{C} \qquad (3)$$

where **Q** is a matrix to be defined. This technique is known as **back projection**.

Now as **K** is an **n x 1** matrix and **C** is an **m x 1** matrix, it is clear that **Q** must be an **n x m** matrix if equation 3 is to be valid. The matrix which is chosen for **Q** in the LBP algorithm is the transpose sensitivity matrix $\mathbf{S}^{T}$ (formed by interchanging the rows and columns of **S**), which has the required dimensional form (**n x m**)

Hence equation 3 becomes:

$$\mathbf{K} = \mathbf{S}^{\mathbf{T}}.\mathbf{C} \qquad (4)$$

The transpose sensitivity matrix $\mathbf{S}^{\mathbf{T}}$ can be considered to be the matrix of normalised permittivity coefficients which, when multiplied by the matrix of inter-electrode capacitance measurements **C**, yields the required matrix of permittivity pixels **K**. The transpose sensitivity matrix $\mathbf{S}^{\mathbf{T}}$ will consist of **n** (1024) sets of permittivity coefficients, each set containing **m** (eg 66) permittivity coefficients corresponding to each individual inter-electrode capacitance measurement. Note that the values of the coefficients for pixel locations outside the sensor boundary will again be zero.

Equation 4 is the basis of the LBP algorithm and is used to calculate the required pixel permittivity distribution.

The LBP algorithm is simple and fast. However, the images produced by this algorithm are blurred because, unlike the case of X-rays, where a single ray path between source and detector will pass through only one set of pixels, the electric field between two capacitance electrodes spreads out and intercepts many pixels. The effect of this is to give a spurious and unwanted level of background permittivity to each pixel. This can be removed by some form of filtering or thresholding if required. In the standard PTL PCECT operating software, no thresholding or filtering is used at present.

2. AN ITERATIVE METHOD FOR IMPROVING IMAGE QUALITY

The iterative image reconstruction method is based on the use of equations 1 and 4. The idea is to use these two equations alternately to correct the sets of capacitance and pixel values in turn and hence produce a more accurate image from the capacitance measurements. The details are as follows:

1. Measure a set of (66) normalised capacitances $\mathbf{C_1}$ for one image frame.

2. Correct the set of measured capacitances $\mathbf{C_1}$ using the series model correction formula (or some other correction formula, see Appendix 1).

3. Calculate the set of (1024) pixel permittivities $\mathbf{K_1}$ corresponding to $\mathbf{C_1}$ using equation (4) ie.

$$\mathbf{K_1 = S^T.C_1} \qquad \textbf{(5)}$$

Note that the matrix $\mathbf{S^T}$ must be normalised as follows: As $\mathbf{K_1}$ is a set of 1024 pixels and $\mathbf{C_1}$ is a set of (eg 66) capacitance readings, then $\mathbf{S^T}$ must be normalised by dividing each of the elements (permittivity coefficients) in $\mathbf{S^T}$ which contribute to each specific pixel value in $\mathbf{K_1}$ by the sum of all of the values of permittivity coefficients in $\mathbf{S^T}$ which contribute to the specific pixel location (a sum of eg 66 permittivity coefficients).

4. Truncate the individual pixel values **k** so that they lie within the range $\mathbf{0 < k < 1}$ and save and display the image.

5. Use these new values of permittivity to back-calculate a new set of inter-electrode capacitances $\mathbf{C_2}$ using equation (1) ie:

$$\mathbf{C_2 = S.\ K_1} \qquad \textbf{(6)}$$

Note that in this case, the matrix **S** must be normalised as follows: As $\mathbf{C_2}$ is a set of eg 66 inter-electrode capacitance measurements and $\mathbf{K_1}$ is a set of 1024 pixel values, then **S** must be normalised by dividing each of the elements (capacitance coefficients) in **S** which contribute to each specific inter-electrode capacitance measurement by the sum of all of the capacitance coefficients in **S** which contribute to this capacitance measurement (a sum of 1024 capacitance coefficients).

6. Calculate a set of error capacitances $\mathbf{\Delta C}$ where:

$$\mathbf{\Delta C = (C_2 - C_1)} \qquad \textbf{(7)}$$

7. Truncate $\mathbf{\Delta C}$ to limit the maximum values of $\mathbf{\Delta C}$ so that they lie within the range ($-0.05 < \mathbf{\Delta C} < 0.05$) or some other pre-defined limits. This is necessary to prevent the feedback loop from becoming unstable.

8. Multiply **ΔC** by a gain factor (typically 2) which **is** determined by empirical means.

9. Use equation (4) and the error capacitances **ΔC** to calculate a set of error pixel values **ΔK**. ie:

$$\mathbf{\Delta K = S^T . \Delta C} \qquad \mathbf{(8)}$$

10. Use the set of error pixels to generate a new set of pixel values $\mathbf{K_2}$ where:

$$\mathbf{K_2 = (K_1 - \Delta K)} \qquad \mathbf{(9)}$$

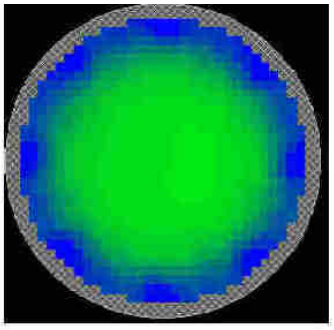
11. Truncate these pixel values to lie within the range $0 < \mathbf{k} < 1$ and save and display the image.

12. Repeat steps 5 to 11 using the new set of **K** values $\mathbf{K_2}$ (instead of $\mathbf{K_1}$) in equation (5). In equation 7, generate the error capacitances by subtracting the **original** measured capacitances from the current set.

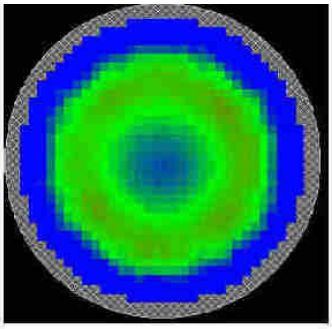
11. Repeat step 12 as many times as necessary to obtain an accurate image.

A simple Matlab program for implementing this procedure is described in Appendix 2.

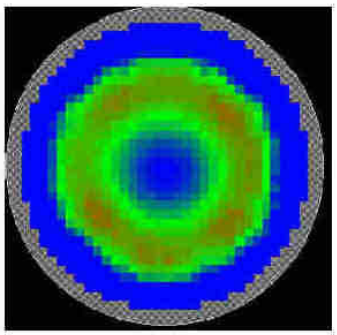
An illustration of the image improvement which can be obtained is shown in the following figure, which was obtained using the program described in Appendix 2. The image data is for a 60mm OD plexiglass tube with 5mm walls, placed approximately centrally inside an 8 electrode ECT sensor having an internal diameter of 100mm. The figure shows the improvements in the image as the number of iterations is steadily increased.



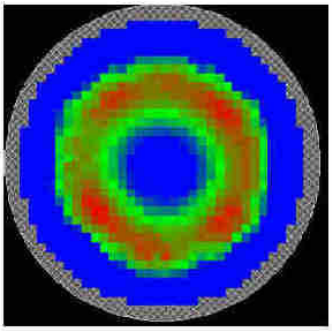
LBP IMAGE



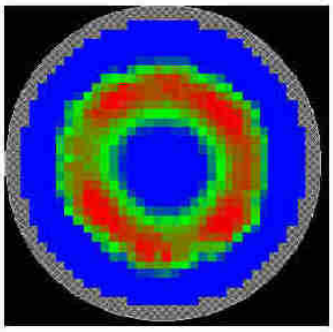
10 ITERATIONS



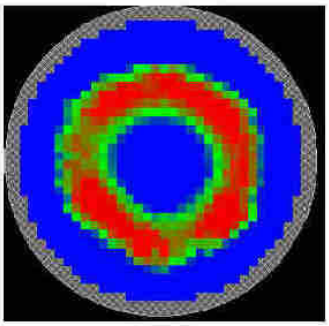
20 ITERATIONS



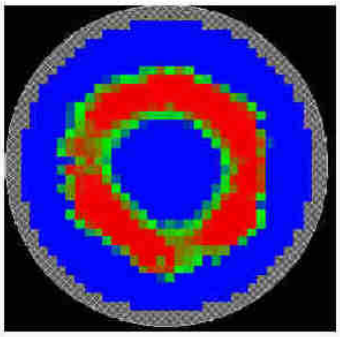
50 ITERATIONS



100 ITERATIONS



200 ITERATIONS



500 ITERATIONS

TUBE IMAGE SHOWING IMPROVEMENTS WITH INCREASING NUMBER OF ITERATIONS

APPENDIX 1

1. CORRECTION OF MEASURED CAPACITANCES FOR SERIES MODEL

If the measured values of normalised capacitance are **Cn,** then a new set of normalised capacitances **Cn(new)** should be calculated where **Cn(new)** are given by:

$$\mathbf{Cn(new) = R \,.\, Cn \,/\big(1 + Cn.(R - 1)\big)}$$

where **R** is the ratio of the permittivities (>1) of the two materials used for calibration.

2. CORRECTION OF MEASURED CAPACITANCES FOR MAXWELL MODEL

If the measured values of normalised capacitance are **Cn,** then a new set of normalised capacitances **Cn(new)** should be calculated where **Cn(new)** are given by:

$$\mathbf{Cn(new) = Cn.(2 + R)/\big(3 + Cn.(R - 1)\big)}$$

where **R** is the ratio of the permittivities (>1) of the two materials used for calibration.

APPENDIX 2

MATLAB PROGRAM FOR CALCULATING ITERATIVE ECT IMAGES

PROGRAM NOTES

The listing for the demonstration Matlab program (**ITERBP.M**) is given in the following pages. The program listing is a standard **Matlab M file** which can be run using **Matlab version 4** or later.

To run the program, copy all of the files on the demonstration disk to the **Matlab** directory. Then insert a blank floppy disk in the a: drive, start Matlab and type **ITERBP** in the Matlab screen. The program will run and two image files will be copied to the floppy disk.

The data file used in the demonstration program (8tube.ncp) was obtained using a perspex tube of OD 60 mmm and wall thickness 5mm, inside an 8-element sensor of internal diameter 100mm. The sensor had been calibrated with air and polypropylene beads.

FURTHER INFORMATION

The following data files must be available in the directory which contains the Matlab program files: (The actual data files and parameter settings used in the demonstration program are shown in *italics*.)

1. The program M file: (*iterbp.m*)

2. The colour scheme for the plot files: (*ptl.m*)

3. The Threshold function program: (thresh.m)

4. The appropriate sensitivity map for the sensor: (*e8_bin.p32*)

5. The normalised capacitance file for the image frame to be constructed: (*8tube.ncp*)

Note that the normalised capacitance file must include only the capacitance values. The first figure in the .ncp file obtained from the standard PTL PCECT software, which is the frame number, must be deleted from the file using a text editor.

All of these files (together with an extra file for 12 electrode sensitivity map) are included on the demonstration program disk. These files should be copied to the Matlab directory.

The program can be easily modified to process other data files by changing the capacitance file name in the program listing and by changing the input parameters (see below) and sensitvity map file name as appropriate.

INPUT PARAMETERS

In addition to the data files. the following input parameters must be defined on the first page of the program listing: (default values in demo program in italics).

Number of electrodes: (*Electrodes = 8*)
Permittivity ratio: (*PermRatio = 2*)
Feedback gain: (*gain = 2*)
Number of iterations: (*ITER = 50*)
Threshold parameter: (*Thresh = 1*)
Pixel Resolution: (*Resolution = 32*)

Note that the number of electrodes and pixel resolution must be compatible with the sensitivity map and measured capacitance data file.

The capacitance model to be used should be enabled by commenting out the program line referring to the unwanted model using a % character at the start of the line. In the listing here, the series model is used.

The program generates two image files image1.img and image2.img which can be saved to a floppy disk. These files can then be viewed using the PTL IMGCON utility. Image1.img is the simple LBP image and image2.img is the iterated image.

This program was developed by John Pendleton and Malcolm Byars.

DEMONSTRATION MATLAB PROGRAM ITERBP.M

```
%*********************************************
% Iterative backprojection with thresholding.
% MB/JDP 5/98.
%*********************************************

clear;
clc;
format;

%*********************************************
% Input Parameters **************************
%*********************************************

Electrodes = 8;
PermRatio = 2
gain = 2
ITER = 50
Thresh = 1
Resolution = 32;

n=Resolution^2;
PR = PermRatio;
NE = Electrodes;
Meas = NE * (NE -1)/2
m=Meas;

K = zeros(m,n);
M = zeros(1,m);
N = zeros(1,n);
R1 = zeros(1,n);
R2 = zeros(1,m);
C2 = zeros(1,m);


%*********************************************
% load sensitivity map.*********************
%*********************************************
fid = fopen('e8_bin.p32');
for i = 1:m
        S(i,1:1024) = fread(fid,n,'float')';
end
fclose (fid);
```

```
%*********************************************
% load normalised capacitance file.***********
%*********************************************
fid = fopen('8tube.ncp');
CMeas = fscanf (fid,'%f',[Meas]);
fclose (fid);



%*********************************************
% Correct measured capacitances according to **
% series or Maxwell model**********************
%*********************************************

for i = 1:m

%*********************************************
% Series Model.********************************
%*********************************************


        C(i,1) = (PermRatio*CMeas(i))/(1+CMeas(i)*(PermRatio-1));

%*********************************************
% Maxwell Model ********************************
%*********************************************

%       C(i,1) = CMeas(i)*(2 + PR)/(3 + CMeas(i)*(PR - 1));

end

C1 = C;


%*********************************************
% derive normalisation matrix for S by summing maps *
% avoiding devision by zero ********************
%*********************************************

'Normalising Matrix Denominators'

for i = 1:m
        N(1,:) = N(1,:) + S(i,:);
end

for j = 1:n
        if N(j) ~= 0
                R1(j) = 1 / N(j);
        end
end
```

```
%***********************************************
% derive normalisation matrix for ST by summing maps *
% avoiding devision by zero ******************
%***********************************************

        ST = S';

for i = 1:n
        M(1,:) = M(1,:) + ST(i,:);
end

for j = 1:m
        if M(j) ~= 0
                R2(j) = 1 / M(j);
        end
end


%***********************************************
% Normalise sensitivity map S ******************
%***********************************************

'Normalising Sensitivity Maps'

for i = 1:m
        SN(i,:) = S(i,:) .* R1(1,:);
end

%***********************************************
% Normalise sensitivity map ST *****************
%***********************************************

for i = 1:n
        SNT(i,:) = ST(i,:) .* R2(1,:);
end


%***********************************************
% Backproject **********************************
%***********************************************

'Backprojecting'

K = SN' * C1;
```

```
%*********************************************
% Threshold ***********************************
%*********************************************

K = thresh(K,0,Thresh);

K1 = K;

%*********************************************
% Calculate volume ratio for original image ***
%*********************************************

vr0 = 0;
for iv=1:n
vr0 = vr0 + K(iv);
end
vr0 = vr0/1024


%*********************************************
% Start iteration loop  ***********************
%*********************************************

for L = 1:ITER

C2 = SNT'* K1;

DC = (C2 - C1);

        capsum = 0;
        for i = 1:m;
        capsum = capsum + (DC(i))^2;
        end
        caperror = sqrt(capsum)/m

DC = gain * thresh(DC,-0.05,0.05);

DK = SN' *  DC;

K2 = (K1 - DK);

%*********************************************
% Threshold ***********************************
%*********************************************

K2 = thresh(K2,0,Thresh);

K1 = K2;
```

```
%*********************************************
% Calculate volume ratio for each iteration ***
%*********************************************

vrf = 0;
for iv=1:n
        vrf = vrf + K2(iv);
        end
vrf = vrf/1024

end

%*********************************************
% End of iteration loop **********************
%*********************************************


%*********************************************
% Construct image as an 32 by 32 array ********
%*********************************************

I1 = reshape(K,Resolution,Resolution);
I2 = reshape(K2,Resolution,Resolution);

%*********************************************
% Adjust image array so that it spans color ***
% map and set values outside void space to 1 **
% (1 maps to black, > 1 to colour in palette) *
%*********************************************

for x = 1:Resolution
        for y = 1:Resolution
                if N(x+(y-1)*Resolution) > 0
                        I1(x,y)=2+62*I1(x,y);
                else
                        I1(x,y)=1;
                end

        end
end
```

```
%*********************************************
% Adjust image array so that it spans color ***
% map and set values outside void space to 1 **
% (1 maps to black, > 1 to colour in palette) *
%*********************************************

for x = 1:Resolution
        for y = 1:Resolution
                if N(x+(y-1)*Resolution) > 0
                        I2(x,y)=2+62*I2(x,y);
                else
                        I2(x,y)=1;
                end

        end
end

%*********************************************
% Display original image in 2-D ***************
%*********************************************

figure
colormap(ptl);
image(I1);

% Ensure 1:1 aspect ratio,
% set height and width to be ave of current settings.
Win = get(gcf,'Position');
WinW = Win(3);WinH = Win(4);
Win(3)= (WinW + WinH)/2;
Win(4)= (WinW + WinH)/2;
set(gcf,'Position',Win)

'press a key'
pause


%*********************************************
% Display iterated image in 2-D ***************
%*********************************************

figure
colormap(ptl);
image(I2);

% Ensure 1:1 aspect ratio,
% set height and width to be ave of current settings.
Win = get(gcf,'Position');
WinW = Win(3);WinH = Win(4);
Win(3)= (WinW + WinH)/2;
Win(4)= (WinW + WinH)/2;
set(gcf,'Position',Win)
```

```
'press a key'
pause

%*********************************************
% Display original image in 3-D **************
%*********************************************

frame1 = reshape(K,Resolution,Resolution);

figure
colormap(ptl);
surf(frame1)

% Ensure 1:1 aspect ratio,
% set height and width to be ave of current settings.
Win = get(gcf,'Position');
WinW = Win(3);WinH = Win(4);
Win(3)= (WinW + WinH)/2;
Win(4)= (WinW + WinH)/2;
set(gcf,'Position',Win)

'press a key'
pause

%*********************************************
% Display iterated image in 3-D **************
%*********************************************

frame2 = reshape(K2,Resolution,Resolution);

figure
colormap(ptl);
surf(frame2);

% Ensure 1:1 aspect ratio,
% set height and width to be ave of current settings.
Win = get(gcf,'Position');
WinW = Win(3);WinH = Win(4);
Win(3)= (WinW + WinH)/2;
Win(4)= (WinW + WinH)/2;
set(gcf,'Position',Win)

'press a key'
pause
```

```
%*********************************************
% Write original and iterated images to files *
%*********************************************


'Writing original image to file'

fid = fopen('a:image1.img','w')
for x = 1 : Resolution;
for y = 1 : Resolution;
fprintf(fid,'%4.2f\t',frame1(x,y));
end
fprintf(fid,'\r\n');
end
fclose(fid);


'Writing iterated image to file'

fid = fopen('a:image2.img','w')
for x = 1 : Resolution;
for y = 1 : Resolution;
fprintf(fid,'%4.2f\t',frame2(x,y));
end
fprintf(fid,'\r\n');
end
fclose(fid);

%*********************************************
```