Pattern Recognition and Tomographic Reconstruction with Terahertz Signals for Applications in Biomedical Engineering

by

Xiaoxia (Sunny) Yin

Bachelor of Engineering (Industrial Electronics), Dalian University, 1996

Thesis submitted for the degree of

Doctor of Philosophy

in

Electrical and Electronic Engineering
University of Adelaide

2008
Appendix A

Oblique Projection Operation

This appendix introduces an oblique projection operation, which underpins the subspace system identification algorithms discussed in Chapter 7. In addition, a few figures are drawn to illustrate parts of the subspace identification procedures.
An oblique projection operation is illustrated in Fig. A.1. In this case, there are three rows spaces involved: \( Y, U \) and \( M \). The notation \( Y/_{U}M \) is used to denote the oblique projection of \( Y \) onto \( M \) along the direction of \( U \). The vector space \( Y \) in the oblique projection should lie in the span of \( \{U, M\} \), otherwise it should be first projected orthogonally onto that span before the oblique projection operation is carried out. The oblique projection is defined by two properties, which are obvious from Fig. A.1:

\[
\begin{align*}
U/_{U}M &= 0 \quad \text{(A.1)} \\
M/_{U}M &= M. \quad \text{(A.2)}
\end{align*}
\]

Any transformation that satisfies Eq. (A.1) and Eq. (A.2) can be adopted as an oblique projection operator. The following formula satisfies the conditions for the projection operator:

\[
M/_{U}M = (Y/_{U}^\perp)(M/_{U}^\perp)^*M \quad \text{(A.3)}
\]

where \( * \) denotes the Moore-Penrose pseudo-inverse, \( \perp \) indicates an orthogonal projection.

In fact, Eq. (A.1) follows from the fact that \( (U/_{U}^\perp) \) is zero by definition, whereas Eq. (A.3) is satisfied because \( (M/_{U}^\perp)(M/_{U}^\perp)^* \) equals the identity matrix.

Oblique projection vectors are also drawn in Fig. A.2(a) to represent the row spaces of the involved matrices in Eq. (7.20), Section 7.5.1. The row space \( Y_f \) is seen to be the sum
Appendix A

Oblique Projection Operation

Figure A.2. The projections regarding state-space sequences. Determination of (a) the future state sequence \( X_f \) and (b) the shifted state sequence \( X_{i+1} \). After Galvão et al. (2005).

of \( \Gamma_i X_f \) and \( \Omega_i U_f \). In this projection operation, the direction of \( \Omega_i U_f \) is known, since \( U_f \), consisting of input data, is known, and \( \Omega_i \), albeit unknown, \( \Omega_i U_f \) can be realised via the performance of rescalings and rotations of the rows of \( U_f \) in the hyperplane in which they lie. As a consequence, the oblique projection of \( Y_f \) along \( U_f \) or along \( \Omega_i U_f \) is the same. The direction of \( W_p \) is also known, composed of input and output data. Moreover, if the system is observable, it is easy to see that \( X_f \) lies in the \( W_p \) space, because \( x[i] \) can be estimated according to the input \( u[k] \) and output \( y[k] \) data up to the instant \( k = i - 1 \). As a result, we can obtain the direction with respect to the oblique projection of \( Y_f \). Owing to the availability of known deterministic input data, the identification allows to be carried out by performing an oblique projection.

The augmented matrix \( \tilde{A} \) has a column zeros, which results in an eigenvalue at \( z = 0 \). In fact, an eigenvalue at the origin corresponds to the \( z^{-1} \) delay factor, which introduces a pole at \( z = 0 \) in the transfer function \( \tilde{G}_d(z) \).

Fig. A.2(b) illustrates the projection procedure regarding the shifted state sequence, which is represented in Eq. (7.28), Section 7.5.1. It is similar to the reasoning related to Fig. A.2(a). The \( O_{i-1} \) is the oblique projection of \( Y_f^{-} \) (with omitting of the first row of the matrix) along \( H_{i-1} U_f^{-} \) direction. It must be performed on the row space of the expanded matrix of input-output data \( W_p^{+} \), which is obtained from \( W_p \) by adding one row at the bottom.
This Appendix provides further details about back projection algorithms. This is a specific supplement made for Chapter 10 in respect of computed tomography reconstruction.
B.1 Theory

The back projection is represented via parallel beam projections. Recalling the formula for the inverse Fourier transform, the object function, \( f(x, y) \), can be expressed as

\[
f(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(u, v) e^{i2\pi(ux + vy)} \, du \, dv. \tag{B.1}
\]

Exchanging the rectangular coordinate system in the frequency domain, \((u, v)\), for a polar coordinate system, \((w, \theta)\), by making the substitutions

\[
u = w \cos \theta \tag{B.2}
\]

\[
v = w \sin \theta \tag{B.3}
\]

and changing the differentials by using

\[
du \, dv = wd\,dw \, d\theta, \tag{B.4}
\]

we can write the inverse Fourier transform of a polar function as

\[
f(x, y) = \int_{0}^{2\pi} \int_{0}^{\infty} F(w, \theta) e^{i2\pi w(x \cos \theta + y \sin \theta)} \, wd\,dw \, d\theta. \tag{B.5}
\]

This integral can be split into two by considering \(\theta\) from 0° to 180° and then from 180° to 360°,

\[
f(x, y) = \int_{0}^{\pi} \int_{0}^{\infty} F(w, \theta) e^{i2\pi w(x \cos \theta + y \sin \theta)} \, wd\,dw \, d\theta
+ \int_{0}^{\pi} \int_{0}^{\infty} F(w, \theta) e^{i2\pi w[x \cos(\theta + 180^\circ) + y \sin(\theta + 180^\circ)]} \, wd\,dw \, d\theta \tag{B.6}
\]

and then using the property

\[F(w, \theta + 180^\circ) = F(-w, \theta)\] \(\tag{B.7}\)

the above expression for \(f(x, y)\) may be written as

\[
f(x, y) = \int_{0}^{\pi} \left[ \int_{-\infty}^{\infty} F(w, \theta) |w| e^{i2\pi wt} \, dw \right] \, d\theta. \tag{B.8}
\]

Here, we have simplified the expression by setting

\[t = x \cos \theta + y \sin \theta. \tag{B.9}\]

If we substitute the Fourier transform of the projection at angle \(\theta\), \(s_\theta(w)\), for the two-dimensional Fourier transform \(F(w, \theta)\), we get

\[
f(x, y) = \int_{0}^{\pi} \left[ \int_{-\infty}^{\infty} s_\theta(w) |w| e^{i2\pi wt} \, dw \right] \, d\theta. \tag{B.10}
\]
This integral in Eq. (B.10) may be expressed as

\[ f(x, y) = \int_0^\pi Q_\theta(x \cos \theta + y \sin \theta) d\theta \quad (B.11) \]

where

\[ Q_\theta = \int_{-\infty}^{\infty} S_\theta(w)|w|e^{j2\pi wt} dw. \quad (B.12) \]

Eq. (B.11) represents a filtering operation, where the frequency response of the filter is given by \(|w|\); therefore \(Q_\theta(w)\) is called a ‘filtered projection’. The resulting projections for different angles \(\theta\) are then back projected to form the estimate of \(f(x, y)\).

We relabel \(S_\theta(w)\) to \(S(\theta, \beta)\), \(t\) to \(\xi\), then we rewrite Eq. (B.11) to yield,

\[ I(x, y) = \int_0^\pi \left[ \int_{-\infty}^{\infty} S(\theta, \beta)|\beta|\exp[i2\pi \beta \xi] d\beta \right] d\theta \quad (B.13) \]

where, Eq. (B.13) is the same as Eq. (10.2) that we use in Section 10.2 for THz reconstruction.
This Appendix provides further details about error analysis with respect to wavelet based local reconstruction. This is a specific supplement made for Chapter 12 to validate local CT via wavelet transforms.
C.1 Methodology

Radon transform error is not negligible because of the nonlocal property of the derivative Hilbert transform (the impulse response of the filter $|\beta|$). In this case, even a small local ROI can be reconstructed by considering some data outside the region of interest for a negligible reconstruction error. In terms of the amount of nonlocal data applied in the reconstruction, an upper bound of the reconstruction error can be calculated. The comparison is made between the wavelet based reconstruction and the traditional reconstruction algorithm for the local tomography image recovery.

In the current algorithm, the region of interest (ROI) and the region of exposure (ROE), are assumed to be centered at the center of an image. The support of a completed image is a disk of radius $R$ pixels centered at the origin. Disks of radius $r_i$ pixels and $r_e$ pixels centered at the origin are used to denote the ROI and ROE, respectively. Consider the Eq. (C.5), the traditional filtered back projection algorithm, which is shifted to the time domain scheme.

\[
I_r(x, y) = \int_0^{\pi} s(\theta, \zeta) h_\theta(x \cos \theta + y \sin \theta) d\theta. \tag{C.1}
\]

The reconstructed function $I_r(x, y)$ is an approximation of the function $I(x, y)$ if $h_\theta$ is the angle dependent impulse response of the ramp filter, $\theta \in [0, 2\pi)$, and is an approximation of the wavelet and scaling coefficients if wavelet and scaling filters are substituted for $|\beta|$.

The discrete version is expressed as follows:

\[
I_r(x, y) = \frac{\pi}{k} \sum_{k=1}^{K} \frac{1}{R} \sum_{n=-R}^{R} s_{\theta_k}(n) h_{\theta_k}(m-n) \tag{C.2}
\]

where $m = (x \cos \theta + y \sin \theta) \in$ ROE, $K$ is the number of the measured projection angles, $s_{\theta_k}$ indicates the projection at $k$th angle and $\theta_k = k(\pi/k)$. The completed image based on global data consists of two parts: the ROE and its complement ROE,

\[
I_r(x, y) = \frac{\pi}{k} \sum_{k=1}^{K} \frac{1}{R} \sum_{|n| \leq r_e} s_{\theta_k}(n) h_{\theta_k}(m-n) + \frac{\pi}{k} \sum_{k=1}^{K} \frac{1}{R} \sum_{|n| > r_e} s_{\theta_k}(n) h_{\theta_k}(m-n). \tag{C.3}
\]

Therefore the magnitude of error regarding the ROE can be calculated as follows:

\[
|e(x, y)| = \left| \frac{\pi}{k} \sum_{k=1}^{K} \frac{1}{R} \sum_{|n| > r_e} s_{\theta_k}(n) h_{\theta_k}(m-n) \right|. \tag{C.4}
\]

The Cauchy-Schwartz inequality is used to achieve an upper bound of the error calculation,
Appendix C  Error Analysis Regarding Wavelet Based Local Reconstruction

\[
|\varepsilon(x, y)| = \left| \frac{2}{K} \sum_{k=1}^{K} \frac{1}{R} \sum_{|n| > r_e} s_{\theta_k}(n) h_{\theta_k}(m - n) \right|
\leq \frac{2}{K} \sum_{k=1}^{K} \frac{1}{R} \sum_{|n| > r_e} |s_{\theta_k}(n) h_{\theta_k}(m - n)|
\leq \frac{2}{K} \sum_{k=1}^{K} \frac{1}{R} \left( \sum_{|n| > r_e} |s_{\theta_k}(n)|^2 \right)^{1/2} \left( \sum_{|n| > r_e} |h_{\theta_k}(m - n)|^2 \right)^{1/2}. \tag{C.5}
\]

There exists such an approximation that \( |s_{\theta_k}| \leq 2 \max |I(x, y)| \).

\[
|\varepsilon(x, y)| = \frac{2\sqrt{2}\pi}{k} \max |I(x, y)| \frac{\sqrt{R - r_e}}{R} \cdot \sum_{k=1}^{K} \left( \sum_{|n| > r_e} |h_{\theta_k}(m - n)|^2 \right)^{1/2}. \tag{C.6}
\]

The relative error is defined as:

\[
|\varepsilon_{rel}(x, y)| = \frac{|\varepsilon(x, y)|}{\max |I(x, y)|}
= \frac{2\sqrt{2}\pi}{k} \frac{\sqrt{R - r_e}}{R} \cdot \sum_{k=1}^{K} \left( \sum_{|n| > r_e} |h_{\theta_k}(m - n)|^2 \right)^{1/2}. \tag{C.7}
\]

It is observed that after applying wavelet based ramp filter, the reconstructed intensity of an terahertz image is much higher than traditional ramp filtered based reconstruction. Thus in the calculation of relative error, normalisation of the reconstructed local images is important for comparison between the global and local reconstruction. A normalisation scaling factor is calculated via dividing maximum intensity of global reconstruction, denoted by \( I(x, y) \) by maximum intensity of local reconstruction, denoted by \( I_{local}(x, y) \). The function is as follows:

\[
N_{I_{g,l}} = \max |I(x, y)| / \max |I_{local}(x, y)|. \tag{C.8}
\]

Combine Eq. (C.7) and Eq. (C.8), and then we achieve a relative error calculation equation in terahertz image reconstruction,

\[
|\varepsilon_{rel}(x, y)| = N_{I_{g,l}} \cdot \frac{2\sqrt{2}\pi}{k} \frac{\sqrt{R - r_e}}{R} \cdot \sum_{k=1}^{K} \left( \sum_{|n| > r_e} |h_{\theta_k}(m - n)|^2 \right)^{1/2}. \tag{C.9}
\]

The ROI can be viewed as a point is its worst case. The function above can be described as:

\[
|\varepsilon_{rel}(x, y)| = N_{I_{g,l}} \cdot \frac{2\sqrt{2}\pi}{k} \frac{\sqrt{R - r_e}}{R} \cdot \sum_{k=1}^{K} \left( \sum_{|n| > r_e - r_i} |h_{\theta_k}(n)|^2 \right)^{1/2}. \tag{C.10}
\]
C.1 Methodology

The truncated filter is defined as:

\[
    h^T_{\theta k}(n) = \begin{cases} 
    h_{\theta k}(n) & \text{if } |n| < r_e - r_i, \\
    0 & \text{otherwise}. 
    \end{cases}
\] (C.11)

Hence,

\[
    |\epsilon_{rel}| = N_{l_s,l_c} \cdot \frac{2\sqrt{2\pi}}{k} \sqrt{R - r_e} \cdot \frac{K}{r_e} \sum_{k=1}^{K} \left( \sum_{|n|=-R}^{R} |h_{\theta k}(n) - h^T_{\theta k}(n)|^2 \right)^{1/2} 
\] (C.12)

The inner sum can be described in the frequency domain

\[
    |\epsilon_{rel}(x, y)| = N_{l_s,l_c} \cdot \frac{2\sqrt{2\pi}}{k} \sqrt{R - r_e} 
    \cdot \sum_{k=1}^{K} \left( \sum_{|n|=-R}^{R} \left| \{[H_{\theta k}(l) - H^T_{\theta k}(l)]\exp[i2\pi\beta\xi] \} \right|^2 \right)^{1/2} 
\] (C.13)

where \( H_{\theta k} \) and \( H^T_{\theta k} \) are the Fourier transform of \( h_{\theta k} \) and \( h^T_{\theta k} \), respectively. To calculation of upper bound for the error in standard filtered back projection algorithm, \( h_{\theta k} \) in Eq. (C.9) is replaced by the ramp filter \( |\beta| \). The upper bound of the relative error in the reconstructed image of wavelet and scaling coefficients can be obtained by replacing \( H_{\theta k} \) in Eq. (C.9) with \( H^C_{\theta k} \) in Eq. (C.14) and via multiplication by a normalization factor,

\[
    \begin{align*}
    H^C_{\theta} &= |\beta|\Phi_{2j}(\beta \cos \theta, \beta \sin \theta) = |\beta|\Phi_{2j}(\beta \cos \theta) \Phi_{2j}(\beta \sin \theta) \\
    H^{DH}_{\theta} &= |\beta|\Psi^h_{2j}(\beta \cos \theta, \beta \sin \theta) = |\beta|\Phi_{2j}(\beta \cos \theta) \Psi^h_{2j}(\beta \sin \theta) \\
    H^{DV}_{\theta} &= |\beta|\Phi^V_{2j}(\beta \cos \theta, \beta \sin \theta) = |\beta|\Phi_{2j}(\beta \cos \theta) \Phi^V_{2j}(\beta \sin \theta) \\
    H^{DD}_{\theta} &= |\beta|\Psi^d_{2j}(\beta \cos \theta, \beta \sin \theta) = |\beta|\Phi_{2j}(\beta \cos \theta) \Psi^d_{2j}(\beta \sin \theta)
    \end{align*}
\] (C.14)

where \( H^C_{\theta} \) and \( H^{Di}_{\theta}, (i = H, V, D) \) are called scaling and wavelet ramp filters.

The relative error in the reconstruction image from approximate reconstruction coefficients is as follows:

\[
    |\epsilon_{rel}(x, y)| = N_{l_s,l_c} \cdot \frac{2\sqrt{2\pi}}{k} \sqrt{R - r_e} 
    \cdot \sum_{k=1}^{K} \left( \sum_{|n|=-R}^{R} \left| \{[H^C_{\theta k}(l) - H^{CT}_{\theta k}(l)]\exp[i2\pi\beta\xi] \} \right|^2 \right)^{1/2} 
\] (C.15)
where $N_{l_g/l_C}$ is the normalised scale factor of an image in relation to approximate reconstruction coefficients, which is calculated via dividing maximum intensity of global reconstruction, denoted by $\{I(x,y)\}$, by maximum intensity of local reconstruction regarding approximate wavelet coefficients, denoted by $\{I_{local}^C(x,y)\}$. The normalised scale factor is as follows:

$$N_{l_g/l_C} = \max |I(x,y)| / \max |I_{local}^C(x,y)|.$$  

(C.16)
This Appendix provides further detail and specifications on the components of both the pulsed THz and continuous wave (CW) terahertz imaging system. The pulsed approach uses a conventional Ti:sapphire laser and relates to Chapter 2 (Sec. 2.3.1), Chapters 9, 11 and 12. The CW approach uses a THz quantum cascade laser (QCL) and relates to Chapter 2 (Sec. 2.5.2) and Chapter 13. It provides a list of the major hardware components along with their critical specifications and purpose. It also summarises part of the experimental procedure used for pulsed spectroscopy experiments in this Thesis.
D.1 Ultrafast T-ray pulsed imaging

This Section describes experimental equipment used in T-ray pulsed measurements in further detail along with model specifications. Moreover, the software tools that were designed to control the equipment during an experiment and to process the results are also described herein.

D.1.1 Ultrafast laser

The femtosecond laser illustrated in Fig. D.1 is the key part of the pulsed T-ray spectrometer. The Ti:sapphire laser is used for most of the spectroscopy experiments in this Thesis.

D.1.2 Crossed-polariser detection

A fundamental part of electrooptic sampling (EOS, see Section 2.5.1) is the photodiode detection circuit. A photograph of crossed polarisers in electrooptic Sampling used in a series of THz experiments in this Thesis is shown in Fig. D.2. The probe beam, entering the Figure from the left, passes though the EO crystal. The polarisation of the probe beam is modulated by the T-ray power, which, subsequently, causes an intensity modulation by a Wollaston polariser. Depending on the rotation of the probe polarisation, the optical power shifts between the two output beams of the Wollaston. These beams are directed onto two photodiodes.

The two photodiodes are connected to output a difference current—common mode fluctuations in the probe power are attenuated, while difference currents produced by polarisation rotation are amplified. A quarter-wave plate is used to balance the power in the two beams to zero for no T-ray power. The T-ray path is blocked and the quarter-wave plate rotated. A DC ammeter is needed to read the output of the balanced circuit for optimisation. Normally during measurement, the output of the preamplifier is attached directly to the LIA.

D.1.3 Hardware specifications

Table D.1 is the description regarding the full system layout of components. This Table lists the components of the full T-ray experiment shown in Fig. D.3, as well as the
NOTE:
This figure is included on page 305 of the print copy of the thesis held in the University of Adelaide Library.

Figure D.1. Photograph of an ultrafast laser. This photograph shows the interior of the femtosecond laser used for a number of THz experiments in this Thesis. Visible in this diagram is the Acousto-Optic Modulator (AOM unit, used to sustain mode-locking (pulsed operation) over long periods of time. The path of the laser beam is marked in red, refracting through the four dispersion-control prisms and into the AOM and output coupler (from left to right). After Mickan (2003).

A substantial amount of software was developed to support this research. Software tools were designed to control the equipment during an experiment and to process the results. This Appendix describes the major software applications used.
D.1 Ultrafast T-ray pulsed imaging

Figure D.2. Photograph of crossed polarisers in electrooptic sampling. The polarisation of the optical probe beam in EOS is detected using crossed polarisers. The probe beam is focused onto the photodiodes using a lens to minimise noise from fluctuations in the alignment of the probe beam. A quarter-wave plate is used to balance the polarisation of the probe beam equally between the s- and p-polarisations when no T-ray field is present.

MFCPentaMax

This application (written in Microsoft Visual C++) was originally developed by Paul Campbell at the Rensselaer Polytechnic Institute. It is used to control the PI Pentamax CCD camera and the motorised motion stages during 2D FSEOS THz imaging. The software is updated by Bradley Ferguson to allow it perform synchronised dynamic subtraction and to control the A200SMC stepper motor controller to allow it to be used for tomography experiments. The software sets up the CCD camera and continuously streams frames from the CCD to a memory buffer, and also controls the motion stages. The software supports CCD pixel binning and dynamic subtraction and accumulation operations. The image data are saved to a file for offline processing and reconstruction, which is performed using Matlab software (see Section 9.3). A screen shot of the MFCPentaMax application is shown in Fig. D.4(a). The code is compiled using Microsoft Visual Studio version 6.

LabView tomography application

National Instruments LabView is used to write general purpose experiment control programs due to the ease of programming and the wide support for equipment drivers. Existing programs written by members of the Department of Physics at Rensselaer
NOTE:
This figure is included on page 307 of the print copy of the thesis held in the University of Adelaide Library.

Figure D.3. Schematic of the femtosecond laser-based T-ray chirped functional imaging system based on a pump-probe configuration. The probe beam is chirped using a diffraction grating to extend its pulse width from 130 fs to 21 ps. The pump beam generates THz pulses via a photoconductive antenna (PCA) emitter. The THz pulses are focused on the sample using parabolic mirrors PM1 and PM2, the transmitted radiation is then focused on the detector using PM3 and PM4. The THz pulse is reflected by an ITO beam splitting mirror, which allows the chirped probe pulse and the THz pulse to propagate collinearly through the ZnTe detector. The wavelength components of the probe beam are then adjusted by a spectrometer and viewed on a CCD camera, revealing the THz temporal profile. The target is then raster scanned to acquire an image. P1,P2=polarizers; ITO=indium tin oxide beam splitter. This experimental setup is used for THz CT reconstruction, described in Chapter 10. After Ferguson et al. (2002a).

Polytechnic Institute are modified to provide the desired functionality. Fig. D.4(b) shows a screen shot of a Labview program designed for performing a T-ray CT experiment. The program allows three translation stages and a rotation stage to be controlled over a GPIB interface and the rotation stage is accessed through the parallel port. A lock-in amplifier is used file is saved to disk for offline processing and reconstruction, which is performed using Matlab software (see Appendix E). The Labview code was written in National Instruments Labview version 6i.
D.1 Ultrafast T-ray pulsed imaging

NOTE:
This figure is included on page 308 of the print copy of the thesis held in the University of Adelaide Library.

Figure D.4. Screen shots of the MFCPentamax software and the Labview tomography application. (a) This program is used to control 2D FSEOS THz imaging and tomography experiments. The program records images from the PI Pentamax CCD camera and controls the motorised motion stages to translate and rotate the target. The screen shot shows the CCD options setup page. (b) This program is developed to control T-ray CT experiments. The program allows three translation stages and a rotation stage to be controlled. The motion stages are controlled over a GPIB interface and the rotation stage is accessed through the parallel port. A lock-in amplifier is used to record the THz signal and is accessed over GPIB. The results of the experiment are plotted in the windows shown and may be saved to disk. After Ferguson (2004).
Table D.1. The description regarding the full system layout components, along with the hardware specifications shown in Fig. D.3.

<table>
<thead>
<tr>
<th>Number</th>
<th>Components</th>
<th>Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A Spectra-Physics Mai-Tai Ti:sapphire oscillator</td>
<td>Output power of 0.7 W at 802.3 nm, with a 1 kHz pulse repetition rate and 130 fs pulse width</td>
</tr>
<tr>
<td>2</td>
<td>Pneumatic vibration isolated optical tables</td>
<td>Separated mounting holes by 2.54 cm (1&quot;)</td>
</tr>
<tr>
<td>3</td>
<td>A lock-in amplifier</td>
<td>Stanford Research Systems SRS830 LIA. The lock-in amplifier is synchronised with an optical chopper</td>
</tr>
<tr>
<td>4</td>
<td>An optical chopper</td>
<td>Stanford Research Systems SR540, driven using synchronised dynamic subtraction</td>
</tr>
<tr>
<td>5</td>
<td>Optical mirrors</td>
<td>Newport 10D20ER.2 broadband metallic mirrors</td>
</tr>
<tr>
<td>6</td>
<td>Parabolic mirrors</td>
<td>Gold coated off-axis parabolic mirrors</td>
</tr>
<tr>
<td>7</td>
<td>Three motorised linear motion stages</td>
<td>Newport stepper motor UR73PP stages</td>
</tr>
<tr>
<td>8</td>
<td>A Newport MM3000 motion controller</td>
<td>200 mm of travel with a resolution of 1 µm</td>
</tr>
<tr>
<td>9</td>
<td>A motorised rotation stage</td>
<td>With resolution of 1.8° and a maximum speed of 120 revolutions per minute</td>
</tr>
<tr>
<td>10</td>
<td>Double side polished &lt;110&gt; oriented ZnTe crystals</td>
<td>A large 20 mm diameter crystal with the crystal thickness varied between 3 mm and 4 mm</td>
</tr>
<tr>
<td>11</td>
<td>Double side polished high-resistivity GaAs wafers</td>
<td>With a 0.6 mm thick, 3 cm diameter GaAs wafer and with metal electrodes separated by 2 cm</td>
</tr>
<tr>
<td>12</td>
<td>Quarter and half wave plates</td>
<td>Newport 10RF42-3 broadband wave plates, 1.8° resolution and a maximum speed of 120 revolutions per minute</td>
</tr>
<tr>
<td>13</td>
<td>A SPEX 500M spectrometer</td>
<td>A spectral resolution of 0.2 Å and a dispersion of 3 mm/nm</td>
</tr>
<tr>
<td>14</td>
<td>A cubic polarising beam splitter</td>
<td>Anti-reflection coated for 800 nm and its part number was Melles Griot 03 PTA 101</td>
</tr>
<tr>
<td>15</td>
<td>Diffraction grating</td>
<td>With grating constant 10 µm, the grating separation of 4mm and the angle of incidence of 51°</td>
</tr>
<tr>
<td>16</td>
<td>A Princeton Instruments EEV 576 × 384 CCD camera</td>
<td>Air-cooled to −30° with pixel size of 22 × 22 µm², 12 bit digitisation, and a frame-transfer period of 15 ms</td>
</tr>
</tbody>
</table>

**Picometrix system software**

The T-ray Picometrix 2000 system used for the experiment presented in Section 9.3 is controlled from a LabView program. The computer controls the delay stages via a General Purpose Interface Bus network. Two screen shots regarding the LabView computer interface are shown in Fig. D.5. The two screen shots are the outputs of a couple of example THz measurements in the time domain and the frequency domain, respectively. The controls for the experiment are shown on the left and the bottom and the readouts are in the middle. In the centre is a large time-domain graph (a) and frequency-domain graph (b) of a couple of sampled T-ray waveforms. The smaller graph underneath shows the transforms of the measurements in the frequency domain and the time domain, respectively. The pictures are from the Adelaide laboratory.
Figure D.5. Screen shot of control software for operating T-ray TDS experiments via a Picometrix system. These are two screen shots of the LabView computer interface used to control a Picometrix T-ray spectrometer used in Section 9.3. (a) The THz readouts in the time domain. (b) The THz readouts in the frequency domain.
Table D.2. The description regarding the full system layout components, shown in Fig. D.3, including a detailed description of the role these components play in the system, and the manufacturers. The numbers correspond to the items listed in Table D.1.

<table>
<thead>
<tr>
<th>Number</th>
<th>Functions</th>
<th>Manufacturers</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The laser is to produce the optical pump and probe beams and therefore to generate and detect THz pulses</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>The optical table is for optical posts</td>
<td>Newport</td>
</tr>
<tr>
<td>3</td>
<td>The LIA is for digitising the detected optical signal and to perform phase sensitive filtering to improve the SNR</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>The chopper is for amplitude modulated the optical signal; providing manual control of the chopping frequency and a sync output to the lock-in amplifier</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>The optical mirrors are for the optical adjustment of the NIR pump and probe beams</td>
<td>Newport</td>
</tr>
<tr>
<td>6</td>
<td>The parabolic mirrors are to focus and collimate the THz beam</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>The motion stages are for scanned THz imaging (two for raster scanning the target and a third for the optical delay line). This controller provides front panel control of the stages and GPIB control for remote access from a controller</td>
<td>Newport</td>
</tr>
<tr>
<td>8</td>
<td>The motion controller controls the three stages. This controller provided front panel control of the stages and GPIB control for remote access from a controller</td>
<td>Newport</td>
</tr>
<tr>
<td>9</td>
<td>The rotation stage is to rotate the imaging target via a NEMA23ESM stepper motor from Mil-Shaf Technologies connected to the parallel port of a computer</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>The ZnTe crystals generate THz pulses via OR and achieve the detection using EO sampling</td>
<td>eV Products</td>
</tr>
<tr>
<td>11</td>
<td>The wafers are to construct photoconductive planar strip lines by gluing two metal electrodes onto the wafer surface using conductive glue</td>
<td>University Wafer Pty. Ltd.</td>
</tr>
<tr>
<td>12</td>
<td>The wave plates are to rotate the polarisation of the NIR beam prior to splitting and photo detection</td>
<td>Newport</td>
</tr>
<tr>
<td>13</td>
<td>Spectrometer is to disperse the wavelength components of the chirped optical probe pulse for the chirped probe imaging system described in Section 9.3</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>The splitter is to split the NIR laser pulses into pump and probe beams</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>The diffraction grating chirps the optical probe pulse and extends its pulse width</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>The CCD camera is placed in the 2D FSEOS THz imaging system and in the chirped probe system. It provides a sync output signal and allowed external triggering. It is controlled using the serial port of a computer.</td>
<td></td>
</tr>
</tbody>
</table>

D.2 Continuous wave T-ray imaging via THz QCL

D.2.1 THz QCL imaging

This Section describes the experimental equipment used in T-ray CW imaging via a QCL in further detail along with the relative specifications on these components, which relates to the experiments in Chapter 13. Software tools that are designed to control the equipment during an experiment and to process the results are also described herein. Fig. D.6 is the layout of the THz QCL imaging experiment.
D.2 Continuous wave T-ray imaging via THz QCL

Figure D.6. Experimental apparatus for a terahertz QCL imaging system that is used to realise terahertz CT imaging. The pulse generator supplies pulses at a frequency of 80 kHz to the QCL. The output signal from the pulse generator is usually gated with a 15 Hz, 50% duty cycle slow modulation by an electronic chopper (a function generator of pulses) to match the detector (the Golay cell) response time, and to afford a reference frequency to the lock-in amplifier (LIA). The LIA is used to digitise signals and to significantly improve the signal-to-noise ratio by setting a time constant, over which the input signal is integrated for each data point. The optimal time constant is set to 50 ms and a threshold current density is set as 112 A/cm². The emission is collected with a 200" f/1 off-axis parabolic mirror, then focused by a 200" f/6.43 parabolic mirror onto the sample. In practice, the sample is placed on a rotational stage for multi view angles, which is mounted on a xyz linear stage to perform 3D scanning.

D.2.2 Hardware specifications

THz QCL laser illustrated in Fig. D.7(a) is the key part of the CW T-ray spectrometer. The THz QCL is used for the THz continuous wave imaging reconstruction experiment in this Thesis.

In the present experiment carried in Chapter 13, the resolution is determined by the shape of the focused THz beam at the position of the sample and is limited by the optical design. To characterise the beam, the sample is removed and a pin hole with a diameter of 0.5 mm is put in front of the Golay cell, which is moved in the x and z directions to map out the beam shape in the focal plane. The cross-section of the beam
The THz QCL used for this project is designed and fabricated by C.H. Worrall, J. Alton et al. in the Semiconductor Physics group, Cavendish Laboratory, Cambridge University, UK. The THz QCL is a GaAs-AlGaAs bound-to-continuum superlattice design, emitting at 2.9 THz (103 µm), and grown by molecular beam epitaxy, and typically operates up to 95K in pulse mode. It delivers 70 mW per facet peak power, with a threshold current density of 112 A/cm². The current THz QCL is a GaAs-AlGaAs bound-to-continuum superlattice design, emitting at 2.9 THz (103 µm), and grown by molecular beam epitaxy. After Nguyen et al. (2006). (b) Photograph showing the experimental apparatus is outlined in Fig. D.6 that is used to realise terahertz CT imaging. The numbers in the text-boxes from 1 to 5 indicate, respectively, (i) a QCL that is mounted on the cold finger of a continuous-flow helium-cooled cryostat maintaining a heat-sink temperature of 4.2 K, (ii) a pair of parabolic mirrors, (iii) the rotational stage mounted on an xyz translational stage, (iv) a Golay cell detector, and (v) a detector controller.
Figure D.8. Photograph of right part of the experimental apparatus shown in Fig. D.6. (a) Equipment in the instrument rack from top to bottom is as follows: an EG&G 5210 lock-in amplifier, a Tektronix TDS2014 digital oscilloscope, a TTi TG230 function generator, and an Agilent 8114 A high power pulse generator. (b) A CryoCon model 32 temperature controller, affording two-channel controller designed for fixed cryogenic applications.
(Fig. D.9) is found to be relatively circular, with a full-width-half-maximum (FWHM) varying between 800 $\mu$m ($x$) and 1,100 $\mu$m ($z$).

Table D.1 is the description regarding the full system layout of components associated with the terahertz QCL experiments shown in Fig. D.6.

**D.2.3 LabVIEW™ programming implement for data acquisition**

The linear and rotational stages on which the sample is mounted are connected to a motion controller (Newport, model: MM4006), which is controlled with a programme written in LabVIEW™ programming language. LabVIEW™ is an increasingly popular graphical development environment for signal acquisition, measurement analysis and data presentation (National Instruments, 2004). A LabVIEW™ programme consists of two components. One is a Front panel, which serves as user interface. The front panel is built with controls and indicators, which are interactive input and output terminals, respectively. Through the front panels of LabVIEW™ programmes, users enter operating parameters. The other is a block diagram, which contains the graphical source code. Objects on the block diagram include terminals, nodes, and functions connected with wires. The block diagram represents the electronic circuits inside physical instruments. The computer controls the delay stages and acquires data.
### Table D.3. Summary of electronic components of the THz QCL imaging system

Components shown in Fig.D.6.

<table>
<thead>
<tr>
<th>Items</th>
<th>Manufacturer/model</th>
<th>Functions/Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>A THz QCL</td>
<td>Semiconductor Physics group, Cavendish Laboratory, The University of Cambridge</td>
<td>Generation of continuous waves of terahertz radiation with high output power and sample penetration depth.</td>
</tr>
<tr>
<td>A pulse generator</td>
<td>Agilent/8144 A</td>
<td>Delivers fast, clean positive or negative pulses up to 100 V at frequencies up to 15 MHz</td>
</tr>
<tr>
<td>A cold finger of a continuous-helium-cooled flow cryostat</td>
<td>Janis Research Co. Inc/ST-300</td>
<td>Maintains a heat-sink temperature of 4.2 K.</td>
</tr>
<tr>
<td>An electronic chopper</td>
<td>Thurlby Thandar Instruments (TTI)/TG2000 Series</td>
<td>The main output with a maximum emf of 20 V pk-pk from a 50 Ohm or 600 Ohm source; A fixed 0 V to +5 V level suitable for driving both TTL and CMOS loads; Each wide sweep range at least 1000:1 either manually or via the sweep control input.</td>
</tr>
<tr>
<td>A high performance single phase analog lock-in amplifier</td>
<td>EG&amp;G Princeton Applied Research/5210</td>
<td>A sensitivity vernier control, allowing the full-scale sensitivity to be set to any value between the calibrated values; With an input impedance of down to typically only 25 W the resulting voltage generated across the source by the signal current is minimized for the very best performance.</td>
</tr>
<tr>
<td>A 100 MHz 4 channel digital real-time oscilloscope</td>
<td>Tektronix/ TDS2014</td>
<td>With up to 200 MHz bandwidth and 2 GS/s maximum sample rate; providing accurate real-time acquisition up to their full bandwidth; offering advanced pulse width triggering and line-selectable video triggering, and 11 standard automatic measurements</td>
</tr>
<tr>
<td>A temperature controller</td>
<td>Cryocon/model 32</td>
<td>Two-channel controller designed for fixed cryogenic applications. Platinum RTD sensors allows the use of built-in DIN 43760 (IEC 750) standard curves for 100 W, 1,000 W or 10 KW devices. The standard curve is used for temperatures from 70 K to 1020 K and is extended down to 30 K for cryogenic use. Operation to about 14 K is possible with user-supplied curves.</td>
</tr>
<tr>
<td>Parabolic mirrors</td>
<td>Gold coated off-axis parabolic mirrors</td>
<td>A 200” f/1 off-axis parabolic mirror for collecting the THz radiation, then a 200” f/6.43 parabolic mirror is used to focus it onto the sample</td>
</tr>
<tr>
<td>Three linear motion stages</td>
<td>Newport/VP-25XA</td>
<td>For scanned THz imaging</td>
</tr>
<tr>
<td>A motorised rotation stage</td>
<td>Newport/SR50CC</td>
<td>To rotate a sample for multi-angle imaging</td>
</tr>
<tr>
<td>A Golay cell</td>
<td>Cathodean Ltd/IR50</td>
<td>An opto-acoustic detector designed for operation in the spectral range 0.02-20 THz, equipped with a 6 mm diameter polyethylene input window that provides for high transparency at frequencies up to 20 THz. It is mounted on proprietary vibration-isolating bases.</td>
</tr>
</tbody>
</table>
via a General Purpose Interface Bus (GPIB, manufacturer: National Instruments) network. Screen shots of the LabVIEW™ application for the control of linear and rotation stages via front panels and block diagrams are shown in Fig. D.10 and the screen shots regarding front panels and block diagram for the control of a rotation stage are shown in Fig. D.11.
D.2 Continuous wave T-ray imaging via THz QCL

Figure D.10. Screen shots of LabVIEW™ programme to control motion stages via motion controllers. (a) This controller provides front panel control (a) of the stages and GPIB control via a block diagram interface (b) for remote access from a controller. A lock-in amplifier is used to record the THz signal and is accessed over a GPIB interface.
Figure D.11. Screen shots of LabVIEW™ programme to control rotation stages via motor controllers. (a) This controller provides front panel control (a) of the stage and parallel port control with application of a block diagram interface (b) for remote access of the rotation stage.
This appendix contains some of the algorithms used to achieve computer analysis of the raw T-ray waveforms. These algorithms are implemented in MATLAB (manufacturer: The MathWorks, model:7; URL: http://www.mathworks.com), and are available on the attached CD-ROM, X.X. Yin PhD Thesis/MATLAB Algorithms, as MATLAB files. Also included on the CD-ROM is a directory of raw data files, included as examples of typical T-ray experimental output, and a pdf file of this Thesis.
E.1 Implemented Matlab toolboxes

Matlab 7 is used to implement all the algorithms described in this Thesis. Matlab is an interpreted programming language with built in support for a large number of mathematical functions and data presentation tools. Mathematical derivations in this Thesis are checked using the symbolic Maple toolbox. The following Matlab toolboxes are utilised:

1. the wavelet toolbox
2. the system identification toolbox
3. the SVM and kernel methods matlab toolbox
4. the symbolic toolbox
5. the signal processing toolbox
6. the image processing toolbox.

E.2 Code listings

This Section provides code listings for the Matlab software used to implement many of the algorithms described in this Thesis. This is only a subset of the software developed to support this research. Matlab scripts with little algorithmic content, including those used to parse input data files and generate plots, have not been included. For more details on the software implementation please contact the author.

The attached CD includes copies of the Matlab files described below:

Powder recognition via frequency domain features

- WDHeuSURE.m This function implements the wavelet Soft Heuristic SURE methods to achieve preferred denoised performance of THz pulsed responses, as presented in Chapter 6 (Sec. 6.5). It also demonstrates frequency domain features that are extracted to be as inputs for classification.

- classify.m This function is to apply Mahalanobis classifier to classify different types of powders based on extracted frequency domain features in 3D. This algorithm is described in Chapter 9 (Sec. 9.2 and Section 9.4).
Cancerous cell recognition functions via auto regressive modelling over wavelet subbands

- **WPDencorreAR.m** This function implements the wavelet packet SURE methods to denoise measured cancerous cell signals *ex vivo*. Auto Regressive (AR) coefficients are calculated over wavelet subbands to extract important features, aiming to identify cancerous cell signals from normal cell THz responses. It relates to Chapter 9 (Sec. 9.4.3)

- **PronyCorreARMA.m** This function is to apply an improved Prony method to achieve Auto Regressive Moving Average (ARMA) modelling over discrete wavelet subband coefficients. It is also illustrated that the discrete wavelet based Heuristic SURE method is applied in Chapter 9 (Sec. 9.4.3) to achieve denoised THz signals of different types of powder samples.

**SVM applications for THz feature subset identification programming**

- **SVMsDualClas.m** The program is realised via applying SVM and kernel methods matlab toolbox, abbreviated as SKMT. Dual classification of THz RNA data is explored. Frequency orientation components are extracted as features to be input to SVMs. It relates to Chapter 9 (Sec. 9.5.4).

- **SVMsMulClas.m** This program is for multiclass classification, realised via applying One-Against-One algorithms for pairwise classifier designs with use of SKMT. It relates to Chapter 9 (Sec. 9.5.4).

**Wavelet Scale Correlation Based Segmentation**

WaveSegment.m This function introduces wavelet scale correlation based segmentation for a 3D classification of the nest structure of a plastic tube inserted in a glass vial. The computed tomography is also illustrated. The contents this function involves are described in Chapter 10 and Chapter 11.

**Wavelet Based Local Tomography Reconstruction of THz pulsed imaging**

- **localTomography_PulsedTHz.m** This program reconstructs the measured THz pulsed image data along the centered region of interest, via applying wavelet and scaling ramp filters. This relates to Chapter 12. Time domain parameters are extracted via applying correlation algorithms for the sinogram images.
E.2 Code listings

- FFT_scale.m This function describes the algorithm for wavelet and scaling ramp filters for local reconstruction of THz measurements.

- Sepbackproj.m This function performs the inverse Radon transform of wavelet and scaling ramp filtered projections.

Local Reconstruction via THz QCL

THz_QCL_LT.m This program reconstructs and segments the region of interest. Three algorithms are involved: global and local tomography via FBP algorithms, and wavelet based local reconstruction, all of which are applied on THz continuous wave measurements. This relates to Chapter 13.
Appendix E Matlab Code

Frequency Domain Features for THz Recognition Functions

```matlab
function [xd,fXd,aReMatrix,bReMatrix] = WDHeuSURE(inputMatrix,dTime,namePowder,numPixel,genTestPlots);

% This function is used to denoise THz raw responses via wavelet Soft
% Heuristic SURE methods. A few different wavelet families are explored to
% find which one works well on the denoising and classification of THz
% pulsed responses. The outputs include the features in the frequency domain.

% The University of Adelaide
% Xiaoxia Yin
% August 2005

%xd = wden(inputMatrix,'heursure','s','mln',3,'sym8');
xd = wden(inputMatrix,'heursure','s','one',3,'db8');
fXd = fft(xd);
aReMatrix = abs(fXd);
bReMatrix = unwrap(angle(fXd));
pReMatrix = 10*log10(aReMatrix.^2);

fx = fft(inputMatrix);
aOrigiMatrix = abs(fx);
bOrigiMatrix = unwrap(angle(fx));
pOrigiMatrix = 10*log10(aOrigiMatrix.^2);

timeOri = [0:400]*dTime;
fig = [0:400]/401/dTime;
maxFreq = 3; % THz;
maxI = ceil(maxFreq/freq(2));
time = timeOri(1:maxI);

if (nargin >= 5)
    TestPlots = genTestPlots;
else
    TestPlots = 0;
end

if (TestPlots == 1)
    % plot several of the responses to see what they look like.
    if (numPixel == 1)
        figure(gcf+1);clf;
        plot(time,inputMatrix(1: maxI, numPixel), time, xd(1: maxI, numPixel));
        formatImage(1);
        xlabel('Time (ps)');
        ylabel('THz Magnitude (a.u.)');
        legend('original signal','denoised signal');
        title('comparison of time domain spectrum ');

        figure(gcf+1);clf;
        plot(freq(1:maxI),fx(1:maxI,numPixel), freq(1:maxI),fXd(1:maxI,numPixel));
        formatImage(1);
        xlabel('Frequency (THz)');
    end
end
```

Page 325
E.2 Code listings

ylabel('THz Magnitude (a.u.)');
legend('original signal','denoised signal');
title('comparison of frequency domain spectrum ');

figure(gcf+1);clf;
plot(freq(1:maxI),pOrigimatrix(1:maxI,numPixel),freq(1:maxI),pReMatrix(1:maxI,numPixel));
formatImage(1);
xlabel('Frequency (THz)');
ylabel('THz Power(dB)');
legend('original signal','reconstructed signal');
title('the comparison of power spectrum');

figure(gcf+1);clf;
plot(freq(1:maxI),bOrigimatrix(1:maxI,numPixel),freq(1:maxI),bReMatrix(1:maxI,numPixel));
formatImage(1);
xlabel('Frequency( THz)');
ylabel('THz phase');
legend('original signal','denoised signal');
title('the comparison of phase spectrum');

end
end

function class = classify(sample,training,group)
% This function is used to achieve linear discriminant analysis.
% CLASS = CLASSIFY(SAMPLE,TRAINING,GROUP) classifies each row
% of the data in SAMPLE into one of the values of the vector
% GROUP. GROUP contains integers from one to the number of
% groups in the training set, which is the matrix, TRAINING.
% SAMPLE and TRAINING must have the same number of columns.
% TRAINING and GROUP must have the same number of rows.
% CLASS is a vector with the same number of rows as SAMPLE.

[gr,gc] = size(group);
if min(gr,gc) ~= 1
    error('Requires the third argument to be a vector.');
end

if gc ~= 1,
    group = group(:);
    gr = gc;
end

if any(group - round(group)) | any(group < 1)
    error('The third input argument must be positive integers.');
end
maxg = max(group);
[tr,tc] = size(training);
if tr ~= gr,
    error('The number of rows in the second and third input arguments must match.');
end

[sr,sc] = size(sample);
if sc ~= tc
    error('The number of columns in the first and second input arguments must match.');
end

d = zeros(sr,maxg);
for k = 1:maxg
    groupk = training(find(group == k),:);
    % d(:,k) = mahal(sample,groupk);
    d(:,k) = mahaldist(sample,groupk);
end
[tmp, class] = min(d');
class = class';

Auto Regressive Over Wavelet Subband for THz Recognition Functions

function [A1,A2,A3,xwpd,snr] = WPDen-correlationAR.m(inputMatrix,dTime);
% This function is to realize an AR modeling using autocorrelation method.
% This AR modelling coefficients are calculated on wavelet Packet subbands.
% numpix, nomainitedPowder, genScalePlots, genTestPlots, are used to control
% the plot activity, order is the order of AR models;

% Xiaoigia Yin
% January 2006

% using wavelet package to denoise T-ray signal
n=length(inputMatrix);
thr=sqrt(2*log(n*log(n)/log(2)));
xwpd=wpdencmp(inputMatrix,'s',3,'bior6.8','sure',thr,1); % 87.50%
xwpd=wpdencmp(inputMatrix,'s',3,'db4','sure',thr,1); % 93.06%
xwpd=wpdencmp(inputMatrix,'s',3,'sym4','sure',thr,1); %
freq = [0:200]/201/dTime/1e12; time = [0:200]*dTime*1e-12;
maxFreq = 2.5; % THz;
maxI = ceil(maxFreq/freq(2));
err=xwpd(1:maxI)-inputMatrix(1:maxI);

% plot on the wavelet packet tree
wpt = wpdec(xwpd,3,'sym4');
plot(wpt);

xd = wden(inputMatrix,'heursure','s','mln',3,'db1');
figure(gcf+1);clf;

% plot time series
figure(gcf+1);clf;
plot(time',inputMatrix, time', xwpd, time',xd);
formatImage(1);
xlabel('Time (ps)');
ylabel('THz Amplitude (a.u.)');
legend('original signal','WPT denoised signal','DWT denoised signal');
title('comparison of time domain spectrum');

figure(gcf+1);clf;
% plot(time(1: maxI)',inputMatrix(1: maxI), time(1: maxI)', xwpd(1: maxI));
subplot(311),plot(time',inputMatrix);
formatImage(1);
% xlabel('Time (ps)');
% ylabel('THz Amplitude (a.u.)');
title('original signal');
subplot(312),plot(time',xwpd);
formatImage(1);
% xlabel('Time (ps)');
ylabel('THz Amplitude (a.u.)');
title('WPT denosing signal');
subplot(313),plot(time',xd);
formatImage(1);
xlabel('Time (ps)');
% ylabel('THz Amplitude (a.u.)');
title('DWT denoising signal');
figure(gcf+1);clf;
% plot(time(1: maxI)',inputMatrix(1: maxI), time(1: maxI)', xwpd(1: maxI));
plot(time',inputMatrix);
formatImage(1);
% xlabel('Time (ps)');
% ylabel('THz Amplitude (a.u.)');
title('original signal');
plot(time',inputMatrix);
formatImage(1);
% xlabel('Time (ps)');
ylabel('THz Amplitude (a.u.)');
title('WPT denosing signal');
plot(time',inputMatrix);
formatImage(1);
xlabel('Time (ps)');
% ylabel('THz Amplitude (a.u.)');
title('DWT denoising signal');

err=xwpd(1:maxI)-inputMatrix(1:maxI);

end

wpt=wpdec(xwpd,3,'bior6.8');
%using wavelet package tree to decompose the x to 3 levels based on shannon entropy
% plot(wpt); %plot wavelet package tree
cfs10=wpcoef(wpt,[1 0]); % read the coefficients of wavelet package tree at a node [1 0]
cfs11=wpcoef(wpt,[1 1]);
cfs20=wpcoef(wpt,[2 0]);
cfs21=wpcoef(wpt,[2 1]);
cfs22=wpcoef(wpt,[2 2]);
Appendix E  Matlab Code

cfs23=wpcoef(wpt,[2 3]);
cfs30=wpcoef(wpt,[3 0]);
cfs31=wpcoef(wpt,[3 1]);
cfs32=wpcoef(wpt,[3 2]);
cfs33=wpcoef(wpt,[3 3]);
cfs34=wpcoef(wpt,[3 4]);
cfs35=wpcoef(wpt,[3 5]);
cfs36=wpcoef(wpt,[3 6]);
cfs37=wpcoef(wpt,[3 7]);

xdCA1=vertcat(cfs10); % the first level wavelet package coarse coefficients
xdCA2=vertcat(cfs20);
xdCA3=vertcat(cfs30,cfs31);
xdCD1=vertcat(cfs11);
xdCD2=vertcat(cfs22,cfs23);
xdCD3=vertcat(cfs34,cfs35,cfs36,cfs37);

A=zeros(9,8);
for order=2:8
  %xdCAExtend=zeros(8+xCALength,3);
  xCALength=length(xdCA1);correMatrixExch=zeros(order+1,1);
  xdCAExtend=zeros(order+xCALength,order);
  for i=0:order
    numberZero2=zeros(1,order-i)';
    numberZero1=zeros(1,i)';
    %xdCAExtend(:,i) = vertcat(numberZero1,xdCA(:,orderCA),numberZero2);
    xdCAExtend(:,i+1) = vertcat(numberZero1,xdCA1,numberZero2);
    %correlation matrix is calculated based on equation: correMatrix = X'X
  end

  X=xdCAExtend;
  correMatrix=X'*X;
  %correMatrix=X'*X./(1.0e-015);
  storVector=correMatrix(:,1);
  vecMinimumTrans=zeros(order,1);
  vecMiniSum=vertcat(-1,vecMinimumTrans);
  correMatrixExch(:,1)=vecMiniSum;
  vecMiniSumExch=-1*storVector;
  correMatrix= horzcat(correMatrixExch(:,1),correMatrix(:,2:end));
  lengthA=length(correMatrix\vecMiniSumExch); lengthAZeros=zeros(9-lengthA,1);
  A(:,order-1)=corrmatvecMiniSumExch,\vecMiniSumExch,\vecMiniSum
  %A1(1,1) is the minimum sum, A1(2:end) are the coefficients of AR
  %errorVariance(1,order-1)=(length(xdCA1)-1+1)^(-1)*A1(1,order-1); %squared error variance
end

for order=2:8
  xdCAExtend=zeros(8+xCALength,3);
  xCALength=length(xdCA2);correMatrixExch=zeros(order+1,1);
  xdCAExtend=zeros(order+xCALength,order);
  for i=0:order
    numberZero2=zeros(1,order-i)';
    numberZero1=zeros(1,i)';
%xdCAExtend(:,i) = vertcat(numberZero1,xdCA(:,orderCA),numberZero2);
xdCAExtend(:,i+1) = vertcat(numberZero1,xdCA2,numberZero2);

%correlation matrix is calculated based on equation: correMatrix = X’X
end

X=xdCAExtend;
%correMatrix=X’*X./((1.0e-015);
correMatrix=X’*X;
storVector=correMatrix(:,1);
vecMinisumTrans=zeros(order,1);
vecMiniSum=vertcat(-1,vecMinisumTrans);
correMatrixExch(:,1)=vecMiniSum;
vecMiniSumExch=1*storVector;
correMatrix(:,1)=correMatrixExch(:,1);
lengthA=length(correrMatrix)\vecMiniSumExch); lengthAZeros=zeros(9-lengthA,1);
A2(:,order-1)=vertcat(correrMatrix)\vecMiniSumExch,lengthAZeros);
%A(:,order-1)=correMatrix\vecMiniSumExch;
errorVariance2(1,order-1)=(length(xdCA2)-1+1)^(-1)*A2(1,order-1);
end

for order=2:8

%xdCAExtend=zeros(8+xdCAlength,3);
xdCAlength=length(xdCA3);correMatrixExch=zeros(order+1,1);
xdCAExtend=zeros(order+xdCAlength,order);
for i=0:order

numberZero2=zeros(1,order-i)’;
numberZero1=zeros(1,i)’;
%xdCAExtend(:,i) = vertcat(numberZero1,xdCA(:,orderCA),numberZero2);
xdCAExtend(:,i+1) = vertcat(numberZero1,xdCA3,numberZero2);
%correlation matrix is calculated based on equation: correMatrix = X’X
end

X=xdCAExtend;
%correMatrix=X’*X./((1.0e-015);
correMatrix=X’*X;
storVector=correMatrix(:,1);
vecMinisumTrans=zeros(order,1);
vecMiniSum=vertcat(-1,vecMinisumTrans);
correMatrixExch(:,1)=vecMiniSum;
vecMiniSumExch=1*storVector;
correMatrix(:,1)=correMatrixExch(:,1);
lengthA=length(correrMatrix)\vecMiniSumExch); lengthAZeros=zeros(9-lengthA,1);
A3(:,order-1)=vertcat(correrMatrix)\vecMiniSumExch,lengthAZeros);
%A(:,order-1)=correMatrix\vecMiniSumExch;
errorVariance3(1,order-1)=(length(xdCA3)-1+1)^(-1)*A3(1,order-1);
end

function [A1,A2,A3,B1,B2,B3] = PronyCorrelationARMA (inputMatrix,dTime,numPixel,nominitedPowder,...
    genScalePlots,genTestPlots);
% This function is to realize an ARMA modeling using an improved Prony method
% Variables: numpix, nominitedPowder, genScalePlots, genTestPlots, are used to control
% the plot activity, order is the order of AR models;

%denoise T-ray pulses using soft threshold
xd = wden(inputMatrix,'heursure','s','mln',3,'db8');
[c1,11]=wavedec(inputMatrix,1,'db8');
[c2,12]=wavedec(inputMatrix,2,'db8');
[c3,13]=wavedec(inputMatrix,3,'db8');

%DWT denoised T-ray pulses into 3 levels
[CA1,CD1] = dwt (inputMatrix,'db8');
[CA2,CD2] = dwt (CA1,'db8');
[CA3,CD3] = dwt (CA2,'db8');
[xdCA1,xdCD1] = dwt (xd,'db8');
[xdCA2,xdCD2] = dwt (xdCA1,'db8');
[xdCA3,xdCD3] = dwt (xdCA2,'db8');

%the order of ARMA modeling: P,Q=2,3,4...8

%A=zeros(9,8);
% level 2 of the average coefficients of ARMA at the order from 2 to 8
%first to calculate the coefficients of AR using autocorrelation method
for order=2:8
    xdCAExtend=zeros(8+xdCA1Length,3);
    xdCA1Length=length(xdCA1);correMatrixExch=zeros(order+1,1);
    xdCAExtend=zeros(order+xdCA1Length,order+1);
    for i=0:order
        numberZero2=zeros(1,order-i)';
        numberZero1=zeros(1,i)';
        xdCAExtend(:,i) = vertcat(numberZero1,xdCA(:,orderCA) ,numberZero2);
        xdCAExtend(:,i+1) = vertcat(numberZero1,xdCA1,numberZe ro2);
    correlation matrix is calculated based on equation: correMatrix = X'X
    X=xdCAExtend;
correMatrix=X'*X;
storVector=correMatrix(:,1);
vecMinisumTrans=zeros(order,1);
vecMiniSum=vertcat(-1,vecMinisumTrans);
correMatrixExch(:,1)=vecMiniSum;
vecMiniSumExch=-1*storVector;
correMatrix= horzcat(correrMatrixExch(:,1),correMatrix(:,2:end));
lengthA=length(correrMatrix\vecMiniSumExch); lengthAZeros=zeros(9-lengthA,1);
A1(:,order-1)=vertcat(correrMatrix\vecMiniSumExch,lengthAZeros);
%A1(1,1)is the minimum sum, A1(2:end)are the coefficients of AR
%A(:,1)correMatrix\vecMiniSumExch;
errorVariance1(1,order-1)=(length(xdCA1)-1+1)^(-1)*A1(1,order-1); %squared error variance

%The coefficients for MA modeling
%The first stage to use autocorrelation methods to calculate AR coefficients by choosing
%an order L, which is the five times the order of the desired MA modeling
L=5*order; xdCA1length=length(xdCA1); correMatrixExchARMA=zeros(L+1,1);
xdCAExtend=zeros(L+xdCA1length,L+1);G1=zeros(41,7);
for i=0:L
    numberZero2=zeros(1,L-i)';
    numberZero1=zeros(1,i)';
    %xdCAExtend(:,i) = vertcat(numberZero1,xdCA(:,orderCA),numberZero2);
    xdCAExtend(:,i+1) = vertcat(numberZero1,xdCA1,numberZero2);
    %correlation matrix is calculated based on equation: correMatrix = X'X
end
X=xdCAExtend;
correMatrix=X'*X;
%storVector=correMatrix(:,1);
vecMinisumTrans=zeros(L,1);
vecMiniSum=vertcat(1,vecMinisumTrans);
%correMatrixExchARMA(:,1)=vecMiniSum;
%vecMiniSumExch=-1*storVector;
%correMatrix= horzcat(correMatrixExchARMA(:,1),correMatrix(:,2:end));
lengthG=length(correMatrix\vecMiniSum); lengthGZeros=zeros(41-lengthG,1);
G1(:,L/5-1)=vertcat(correMatrix\vecMiniSum,lengthGZeros);
%G1 is the coefficient vector of first stage of AR modeling
%correlation Matrix for the second stage of AR model (of order Q)
The value gain is extracted from the first value of vector G1
gain1=G1(1,L/L-1)^(1/2); % L/L-1=order-1 with the AR order starting from 2

correMatrixExch=zeros(order+1,1);
xdCAExtend=zeros(order+L+1,order+1);
for i=0:order
    numberZero2=zeros(1,order-i)';
    numberZero1=zeros(1,i)';
    %xdCAExtend(:,i) = vertcat(numberZero1,xdCA(:,orderCA),numberZero2);
    xdCAExtend(:,i+1) = vertcat(numberZero1,G1(1:L+1,L/5-1),numberZero2);
    %correlation matrix is calculated based on equation: correMatrix = X'X
end
Xg=xdCAExtend.*(1/gain1);
%Xg is the data matrix of G1 with the order of this stage of Q
%correMatrixMA=Xg'*Xg./(1.0e-015);
correMatrixMA=Xg'*Xg;
%storVector=correMatrixMA(:,1);
vecMinisumTrans=zeros(order,1);
vecMiniSum=vertcat(1,vecMinisumTrans);
%correMatrixExch(:,1)=vecMiniSum;
%vecMiniSumExch=-1*storVector;
%correMatrixMA= horzcat(correMatrixExch(:,1),correMatrixMA(:,2:end));
lengthB=length(correMatrixMA\vecMiniSum); lengthBZeros=zeros(9-lengthB,1);
B1(:,order-1)=vertcat(correMatrixMA\vecMiniSum,lengthBZeros);
%AR and ARMA have the same error covariance

% level 2 of the average coefficients of ARMA at the order from 2 to 8
for order=2:8
%xdCAExtend=zeros(8+xdCAlength,3);

xdCAlength=length(xdCA2);correMatrixExch=zeros(order+1,1);
 xdCAExtend=zeros(order+xdCAlength,order);
 for i=0:order
     numberZero2=zeros(1,order-i)';
     numberZero1=zeros(1,i)';
     %xdCAExtend(:,i) = vertcat(numberZero1,xdCA(:,orderCA),numberZero2);
     xdCAExtend(:,i+1) = vertcat(numberZero1,xdCA2,numberZero2);
     %correlation matrix is calculated based on equation: correMatrix = X'X
 end

X=xdCAExtend;
correMatrix=X'*X;
storVector=correMatrix(:,1);
vecMinisumTrans=zeros(order,1);
vecMiniSum=vertcat(-1,vecMinisumTrans);
correMatrixExch(:,1)=vecMiniSum;
vecMiniSumExch=-1*storVector;
correMatrix(:,1)=correMatrixExch(:,1);
lengthA=length(corr}); lengthAZero3=zeros(9-lengthA,1);
A2(:,order-1)=vertcat(corr}); lengthGZeros=zeros(41-lengthG,1);
G2(:,L/5-1)=vertcat(corr}); lengthGZeros,
%The coefficients for MA modeling
%the first stage to use autocorrelation methods to calculate AR coefficients by choosing an order L,
%which is the five times the order of the desired MA modeling

L=5*order; xdCAlength=length(xdCA2);correMatrixExchARMA=zeros(L+1,1);
xdCAExtend=zeros(L+xdCAlength,L+1); G2=zeros(41,7);
for i=0:L
     numberZero2=zeros(1,L-i)';
     numberZero1=zeros(1,i)';
     %xdCAExtend(:,i) = vertcat(numberZero1,xdCA(:,orderCA),numberZero2);
     xdCAExtend(:,i+1) = vertcat(numberZero1,xdCA2,numberZero2);
     %correlation matrix is calculated based on equation: correMatrix = X'X
 end

X=xdCAExtend;
correMatrix=X'*X;
%storVector=correMatrix(:,1);
vecMinisumTrans=zeros(L,1);
vecMiniSum=vertcat(1,vecMinisumTrans);
%correMatrixExchARMA(:,1)=vecMiniSum;
%vecMiniSumExch=-1*storVector;
%correMatrix= horzcat(corr}); vecMiniSumExch, lengthGZeros);
%The values of the second stage of AR model (of order Q)
%The value gain is extracted from the first value of vector G1
%gain2=G2(1,1/L-1)^(1/2);
% $X_g$ is the data matrix of $G_1$; the order of this stage is $Q$

correMatrixExch = zeros(order+1,1);
xdCAExtend = zeros(order+L+1,order+1);
for i = 0:order
    numberZero2 = zeros(1,order-i)';
    numberZero1 = zeros(1,i)';
    %xdCAExtend(:,i) = vertcat(numberZero1,xdCA(:,orderCA),numberZero2);
    %xdCAExtend(:,i+1) = vertcat(numberZero1,G2(1:L+1,L/5-1),numberZero2);
end

% correlation matrix is calculated based on equation: correMatrix = $X'X$

X = xdCAExtend.*(1/gain2);
%correMatrixMA=Xg'*Xg./(1.0e-015);
correMatrixMA = Xg'*Xg;
%storVector=correMatrixMA(:,1);
vecMinisumTrans = zeros(order,1);
vecMiniSum = vertcat(1,vecMinisumTrans);
%correMatrixExch(:,1)=vecMiniSum;
%vecMiniSumExch=-1*storVector;
%correMatrixMA= horzcat(correMatrixExch(:,1),correMatrixMA(:,2:end));
lengthB = length(correMatrixMA \ vecMiniSum); lengthBZeros = zeros(9-lengthB,1);
B2(:,order-1) = vertcat(correMatrixMA \ vecMiniSum, lengthBZeros);
%AR and ARMA have the same error covariance
end

% level 3 of the average coefficients of ARMA at the order from 2 to 8
for order = 2:8
    xdCAExtend = zeros(8+xdCALength,3);
    xdCALength = length(xdCA3); correMatrixExch = zeros(order+1,1);
    xdCAExtend = zeros(order+xdCALength,order);
    for i = 0:order
        numberZero2 = zeros(1,order-i)';
        numberZero1 = zeros(1,i)';
        %xdCAExtend(:,i) = vertcat(numberZero1,xdCA(:,orderCA),numberZero2);
        %xdCAExtend(:,i+1) = vertcat(numberZero1,xdCA3,numberZero2);
        %correlation matrix is calculated based on equation: correMatrix = $X'X$
    end

    X = xdCAExtend;
correMatrix = X'*X;
    storVector = correMatrix(:,1);
    vecMinisumTrans = zeros(order,1);
    vecMiniSum = vertcat(1,vecMinisumTrans);
correMatrixExch(:,1) = vecMiniSum;
vecMiniSumExch = -1*storVector;
correMatrix(:,1) = correMatrixExch(:,1);
lengthA = length(correMatrix \ vecMiniSumExch); lengthAZeros = zeros(9-lengthA,1);
A3(:,order-1) = vertcat(correMatrix \ vecMiniSumExch, lengthAZeros);
%A(:,order-1) = correMatrix \ vecMiniSumExch;
errorVarianceA(1,order-1) = (length(xdCA3)-1+1)^(-1)*A3(1,order-1);
%The coefficients for MA modeling
%the first stage to use autocorrelation methods to calculate AR coefficients by choosing an order L,
%which is the five times the order of the desired MA modeling

L=5*order; xdCAlength=length(xdCA3);correMatrixExchARMA=zeros(L+1,1);
xdCAExtend=zeros(L+xdCAlength,L+1); G3=zeros(41,7);
for i=0:L
    numberZero2=zeros(1,L-i)';
    numberZero1=zeros(1,i)';
    xdCAExtend(:,i) = vertcat(numberZero1,xdCA(:,orderCA),numberZero2);
    xdCAExtend(:,i+1) = vertcat(numberZero1,xdCA3,numberZero2);
    %correlation matrix is calculated based on equation: correMatrix = X'X
end

X=xdCAExtend;
correMatrix=X'*X;
%storVector=correMatrix(:,1);
vecMinisumTrans=zeros(L,1);
vecMiniSum=vertcat(1,vecMinisumTrans);
%correMatrixExchARMA(:,1)=vecMiniSum;
%vecMiniSumExch=-1*storVector;
%correMatrix= horzcat((correMatrixExchARMA(:,1),correMatrix(:,2:end)));
lengthG=length(correMatrix\vecMiniSum); lengthGZeros=zeros(41-lengthG,1);
G3(:,L/5-1)=vertcat(correMatrix\vecMiniSum,lengthGZeros);
%G1 is the coefficient vector of first stage of AR modeling
%correlation Matrix for the second stage of AR model (of order Q)
%the value gain is extracted from the first value of vector G1
gain3=G3(1,L/5-1)^(1/2);
correMatrixExch=zeros(order+1,1);
xdCAExtend=zeros(order+L+1,order+1);
for i=0:order
    numberZero2=zeros(1,order-i)';
    numberZero1=zeros(1,i)';
    xdCAExtend(:,i) = vertcat(numberZero1,xdCA(:,orderCA),numberZero2);
    xdCAExtend(:,i+1) = vertcat(numberZero1,G3(1:L+1,L/5-1),numberZero2);
    %correlation matrix is calculated based on equation: correMatrix = X'X
end

Xg=xdCAExtend.*(1/gain3);
correMatrixMA=Xg'*Xg./(1.0e-015);
correMatrixMA=Xg'*Xg;
%storVector=correMatrixMA(:,1);
vecMinisumTrans=zeros(order,1);
vecMiniSum=vertcat(1,vecMinisumTrans);
%correMatrixExch(:,1)=vecMiniSum;
%vecMiniSumExch=-1*storVector;
%correMatrixMA= horzcat((correMatrixExch(:,1),correMatrixMA(:,2:end)));
lengthB=length(correMatrixMA\vecMiniSum); lengthBZeros=zeros(9-lengthB,1);
B3(:,order-1)=vertcat(correMatrixMA\vecMiniSum,lengthBZeros);
%AR and ARMA have the same error covariance
SVMs for THz Recognition Programming

% The THz classification system was used to classify RNA sample: Poly-A and Poly-C
% dual classification is finished via svms

% Xiaoxia Yin
% June 2006
clear; tic
load testamp_matrix.dat;
imagesc(testamp_matrix);
grid;
figure(gcf+1);grid;imagesc(testamp_matrix);
axis([7 13 15 21]);

filePath = 'D:\MATLAB6p5\work\Data\Burnd data\biochip';

[mpolyC1,bpolyC1,polyC1] = openLIACTFile_yxx('biochip1_10x34c','biochip1_1_1x1c');
% omitting the reading of the remaining RNA data files.

% the feature matrix from measured image data corresponding to each pixel of PolyA
        mpolyA13,mpolyA14,mpolyA15,mpolyA16,mpolyA17,mpolyA18,mpolyA19,mpolyA20,mpolyA21,mpolyA22,mpolyA23,mpolyA24,...
        mpolyA37,mpolyA38,mpolyA39,mpolyA40,mpolyA41,mpolyA42,mpolyA43,mpolyA44,mpolyA45,mpolyA46,mpolyA47,mpolyA48];
bpolyA=[bpolyA1,bpolyA2,bpolyA3,bpolyA4,bpolyA5,bpolyA6,bpolyA7,bpolyA8,bpolyA9,bpolyA10,bpolyA11,bpolyA12,...
        bpolyA13,bpolyA14,bpolyA15,bpolyA16,bpolyA17,bpolyA18,bpolyA19,bpolyA20,bpolyA21,bpolyA22,bpolyA23,"
        bpolyA24,...
        bpolyA25,bpolyA26,bpolyA27,bpolyA28,bpolyA29,bpolyA30,bpolyA31,bpolyA32,bpolyA33,bpolyA34,bpolyA35,"
        bpolyA36,...
        bpolyA37,bpolyA38,bpolyA39,bpolyA40,bpolyA41,bpolyA42,bpolyA43,bpolyA44,bpolyA45,bpolyA46,bpolyA47,bpolyA48];

polyA=[polyA1,polyA2,polyA3,polyA4,polyA5,polyA6,polyA7,polyA8,polyA9,polyA10,polyA11,polyA12,...
        polyA13,polyA14,polyA15,polyA16,polyA17,polyA18,polyA19,polyA20,polyA21,polyA22,polyA23,polyA24,...
        polyA25,polyA26,polyA27,polyA28,polyA29,polyA30,polyA31,polyA32,polyA33,polyA34,polyA35,"
        polyA36,...
        polyA37,polyA38,polyA39,polyA40,polyA41,polyA42,polyA43,polyA44,polyA45,polyA46,polyA47,polyA48];
% omitting the feature matrices regarding polyC

mpolyC=mpolyC(:,1:48);
bpolyC=bpolyC(:,1:48);
polyC=polyC(:,1:48);

[larpolyC,C] = max(polyC,[],1); length(larpolyC)
[larpolyA,A] = max(polyA,[],1); length(larpolyA)
LmpolyC=mpolyC(335,:);%C=335;
LbpolyC=bpolyC(335,:);
LmpolyA=mpolyA(335,:);%C=335;
LbpolyA=bpolyA(335,:);

figure(11);clf;
plot(larpolyC,'b.');hold on;
plot(larpolyA,'r.');

if 1==0
    figure(11);clf;
    plot(LmpolyC,LbpolyC,'b.');hold on;
    plot(LmpolyA,LbpolyA,'r.');
end

timestep=0.0667*10^(-12);
normalizeSpectra = 0;
maxFreqI = 70;

lambda = 1e-7;
C =10;
C =1;
kernel='gaussian';
kernelselection=30;

l=size(mpolyC,1);
if 1==1
    t1=cputime;
    %-----train svm------
    wrong=[]; corre=[];
    % for u=2:120
    u=335;
    % holdout method for classification design
    for k=12
        Xapp=[mpolyA(u,1:24+k)',bpolyA(u,1:24+k)';mpolyC(u,1:24+k)',bpolyC(u,1:24+k)'
        % 2 features and 51 vectors for each of two classes before denoi sing
        yapp=[-ones(24+k,1);ones(24+k,1)];
        [xsup,w,w0,pos,tps,alpha] = svmclass(Xapp,yapp,C,lambda,kernel,kernelselection);
        %[xsup,w,w0,pos,tps,alpha] = svmclassLS(Xapp,yapp,C,lambda,kernel,kernelselection,1,1,100);
    %-----test svm------
        xtesta1=mpolyA(u,24+k+1:end)';
        xtesta2=bpolyA(u,24+k+1:end)';
        xtest=[xtesta1, xtesta2];
        ypred = svmval(xtest,xsup,w,0,kernel,kernelselection);
        ypredmat=ypred;
        % ypredapp = svmval(Xapp,xsup,w,,kernel,kernelselection,1);
    %-----------------Classification accuracy--------------------------
        % ind1=find(ypred>0); wrong(u,k)=length(ind1)/(24-k);
        % ind1=find(ypred<0);corre(u,k)=length(ind1)/(24-k);
E.2 Code listings

```matlab
ind1 = find(ypred > 0); wrong(1, k/2) = length(ind1) / (24-k);
indm1 = find(ypred < 0); corre(1, k/2) = length(indm1) / (24-k);
length(ind1)
ind1C = find(ypred < 0); wrongC(1, k/2) = length(ind1C) / (24-k);
indm1C = find(ypred > 0); correC(1, k/2) = length(indm1C) / (24-k);
length(ind1C)
end
end
toc
size(pos)
```

__________

**SVMsMulClas.m**

```matlab
% Powder MultiClass SVM Classification
% "One against One"
%
close all
clear all
tic
%--------------------------------------------------
% The THz imaging system was used to obtain the THz response of
% several different powders with thickness of 2mm.
% We then attempted to classify the images.
%
% Xiaoxia (Sunny) Yin
% May 2006

% 1D images of the powders.
%
[mFree2mm, nX, nY, nTime, nAngle, dX, dY, dTime, dAngle] = openLIACTFile('THzFreeSpace2mm1529');
[mTalc2mm, nX, nY, nTime, nAngle, dX, dY, dTime, dAngle] = openLIACTFile('THzFreeSpace2mm1529');

mSand2mm = openLIACTFile('ChinesePowder2mm1755');
mTalc2mm = openLIACTFile('TalcumPowder2mm1939');
mSalt2mm = openLIACTFile('SaltPowder2mm2127');
mPowSugar2mm = openLIACTFile('PowderSuger2mm2237');
mSugar2mm = openLIACTFile('Suger2mm2337');

% mRef2 = openLIACTFile('THzRef1356');

mFlour2mm = openLIACTFile('2mmflour2207');
mSoda2mm = openLIACTFile('2mmSoda2313');
mSugar2mm = openLIACTFile('2mmSugar2037');

% normalization:
mRef2_avg = mean(mRef2, 2);
mFree2mm_avg = mean(mFree2mm, 2);
```
Appendix E

Matlab Code

```matlab
[ref1Max, ref1Timing] = max(mFree2mm_avg);
[ref2Max, ref2Timing] = max(mRef2_avg);
ampFactor = ref1Max/ref2Max;
timingFactor = ref1Timing-ref2Timing;

% truncate all the signals so that they are all the same size.
mFree2mm = mFree2mm(1:401,:);
mSand2mm = mSand2mm(1:401,:);
mTalc2mm = mTalc2mm(1:401,:);
mSalt2mm = mSalt2mm(1:401,:);
mSalt4mm = mSalt4mm(1:401,:);

time = [0:400]*dT;

freq = [0:400]/401/3/dT;
maxFreq = 3; % THz;
maxI = ceil(maxFreq/freq(2));

% first normalize the responses.
% scale the amplitude in the time domain
normalizeResponses = 0;
if (normalizeResponses == 1)
mFree2mm = normalizeTHz(mFree2mm, mFree2mm(:,2));
mSalt2mm = normalizeTHz(mSalt2mm, mFree2mm(:,2));
mSand2mm = normalizeTHz(mSand2mm, mFree2mm(:,2));
mTalc2mm = normalizeTHz(mTalc2mm, mFree2mm(:,2));
mPowSugar2mm = normalizeTHz(mPowSugar2mm, mFree2mm(:,2));
mSugar2mm = normalizeTHz(mSalt2mm, mFree2mm(:,2));
end

% now deconvolve.
% construct a symmetric structure, same as m*m, to plot
numPix=size(mFree2mm,2);
for i = 1:numPix
  [%aSoda3mm(:,i), pSoda3mm(:,i), c, d]=deconvolve(mSoda3mm(:,i), mRef3(:,25), dTime*1e-12, 3, 0, 0, 0, 1, 0);
  [%aFree2mm(:,i), pFree2mm(:,i), c, d]=deconvolveBrad(mFree2mm(:,i), mFree2mm(:,1), dTime*1e-12, 3, 0, 0, 0, 1, 0);
  [%aSalt2mm(:,i), pSalt2mm(:,i), c, d]=deconvolveBrad(mSalt2mm(:,i), mFree2mm(:,1), dTime*1e-12, 3, 0, 0, 0, 1, 0);
  [%aSand2mm(:,i), pSand2mm(:,i), c, d]=deconvolveBrad(mSand2mm(:,i), mFree2mm(:,1), dTime*1e-12, 3, 0, 0, 0, 1, 0);
  [%aTalc2mm(:,i), pTalc2mm(:,i), c, d]=deconvolveBrad(mTalc2mm(:,i), mFree2mm(:,1), dTime*1e-12, 3, 0, 0, 0, 1, 0);
  [%aPowSugar2mm(:,i), pPowSugar2mm(:,i), c, d]=deconvolveBrad(mPowSugar2mm(:,i), mFree2mm(:,1), ...]
               dTime*1e-12, 3, 0, 0, 0, 1, 0);
  [%aSugar2mm(:,i), pSugar2mm(:,i), c, d]=deconvolveBrad(mSugar2mm(:,i), mRef3(:,25), dTime*1e-12, 3, 0, 0, 0, 1, 0);
  [%aFlour2mm(:,i), pFlour2mm(:,i), c, d]=deconvolveBrad(mFlour2mm(:,i), mRef3(:,25), dTime*1e-12, 3, 0, 0, 0, 1, 0);
  [%aSoda2mm(:,i), pSoda2mm(:,i), c, d]=deconvolveBrad(mSoda2mm(:,i), mRef3(:,25), dTime*1e-12, 3, 0, 0, 0, 1, 0);
end

toc
normalizeSpectra = 0;
maxFreq1 = 70;
maxFreq2 = 70;
```

```
%-----------------------------------------------------
% Learning and Learning Parameters

c = 1000;
lambda = 1e-7;
kernelloption=4;
% kernel='gaussian';
kernel='poly';
verbose = 0;
%--------------powder at 2mm to test svms using LOO--------

if 1==1
  tt13=cputime;
  leng1=[]; leng2=[]; leng3=[]; leng4=[]; leng5=[]; leng6=[];

  n= numPix-2 ;
  for i=2: numPix-1 % LOO
    for i=30: 50
      posFre=[1:i-1,i+1:numPix];
      xapp=[aSalt3mm(2,posFre) aSand3mm(2,posFre) aTalc3mm(2,posFre) aSugar3mm(2,posFre) ...
           aFlour3mm(2,posFre) aSalt3mm(2,posFre) aSand3mm(2,posFre) aTalc3mm(2,posFre) ...
           pTalc3mm(2,posFre) pSalt3mm(2,posFre) pSand3mm(2,posFre) pFlour3mm(2,posFre) ...
           pSalt3mm(2,posFre) pSand3mm(2,posFre) pTalc3mm(2,posFre) pSugar3mm(2,posFre))];
      yapp=[1*ones(1,numPix-1) 2*ones(1,numPix-1) 3*ones(1,numPix-1) 4*ones(1,numPix-1) ...
            5*ones(1,numPix-1) 6*ones(1,numPix-1)]';
      nbclass=6;

      [n1, n2]=size(xapp);
      nbclass=6;

      [xsup,w,b,nbsv,classifier]=svmmulticlassoneagainstone(xapp,yapp,nbclass,c,lambda,...
                                                  kernel,kernelloption,verbose);
size(xsup)
  xtest1=aSalt3mm(2,i);
  xtest2=pSalt3mm(2,i);
  xtest=[xtest1; xtest2]';
  [ypred,maxi] = svmmultivaloneagainstone(xtest,xsup,w,b,nbsv,kernel,kernelloption);
  leng1(i)=ypred;

  xtest1=aSand3mm(2,i);% 100% before denoising 100% after denoising
  xtest2=pSand3mm(2,i);
  xtest=[xtest1; xtest2]';
  [ypred,maxi] = svmmultivaloneagainstone(xtest,xsup,w,b,nbsv,kernel,kernelloption);
  leng2(i)=ypred;
endif
```matlab
% Appendix E

xtesta1 = aTalc3mm(2,i); % 92% before denoising 96% after denoising
xtesta2 = pTalc3mm(2,i);
xtest = [xtesta1; xtesta2]';
[ypred, maxi] = svmmultivaloneagainstone(xtest, xsup, w, b, nbsv, kernel, kerneloption);
leng3(i) = ypred;

xtesta1 = aSugar3mm(2,i); % 96% before denoising 96% after denoising
xtesta2 = pSugar3mm(2,i);
xtest = [xtesta1; xtesta2]';
[ypred, maxi] = svmmultivaloneagainstone(xtest, xsup, w, b, nbsv, kernel, kerneloption);
leng4(i) = ypred;

xtesta1 = aFlour3mm(2,i); % 92% before denoising 84% after denoising
xtesta2 = pFlour3mm(2,i);
xtest = [xtesta1; xtesta2]';
[ypred, maxi] = svmmultivaloneagainstone(xtest, xsup, w, b, nbsv, kernel, kerneloption);
leng5(i) = ypred;

xtesta1 = aSoda3mm(2,i); % 100% before denoising 100% after denoising
xtesta2 = pSoda3mm(2,i);
xtest = [xtesta1; xtesta2]';
[ypred, maxi] = svmmultivaloneagainstone(xtest, xsup, w, b, nbsv, kernel, kerneloption);
leng6(i) = ypred;

% end

ACla1 = length(find(leng1 == 1))/n; ACla2 = length(find(leng2 == 2))/n; ACla3 = length(find(leng3 == 3))/n;
ACla4 = length(find(leng4 == 4))/n; ACla5 = length(find(leng5 == 5))/n; ACla6 = length(find(leng6 == 6))/n;
accuracy = [ACla1 ACla2 ACla3 ACla4 ACla5 ACla6];
% tt23 = cputime;

toc

%-----------------------------------------plot-----------------------------------------

[xtest1, xtest2] = meshgrid([0:0.25:5], [-2:0.1:0]);
[na, nb] = size(xtest1);
xtest1 = reshape(xtest1, 1, na*nb);
xtest2 = reshape(xtest2, 1, na*nb);
xtest = [xtest1; xtest2]';

[ypred, maxi] = svmmultivaloneagainstone(xtest, xsup, w, b, nbsv, kernel, kerneloption);
ypredmat = reshape(ypred, na, nb);

figure(1); clf;
contourf(xtest1, xtest2, ypredmat*3, 30); shading flat
hold on;
[cs, h] = contour(xtest1, xtest2, ypredmat, [1 2 3 4 5 6]);
clabel(cs, h); hold on;

style = 'x**x+o';
```
E.2 Code listings

color=['kcgrbm'];
for i=0:nbclass-1
    h=plot(xapp(i*n+1:(i+1)*n,1),xapp(i*n+1:(i+1)*n,2),[style(i+1) color(i+1)]);
    set(h,'LineWidth',2);
    hold on
end;
%grid off;
%formatImage(1);
xlabel('THz Amplitude (a.u.)');
ylabel('THz Phase (radians)');
hold on;
h=plot(xsup(:,1),xsup(:,2),'.r');
set(h,'LineWidth',2);
axis([0 5 -2 0]);
legend('Salt2mm','Sand2mm','Talc2mm', 'Sugar2mm','Flour2mm','Soda2mm','Support Vector');
hold off
end

Wavelet Scale Correlation Based Segmentation

function [numXVial1, numYVial1, numXtube1, numYtube1, numXair1, numYair1] = WaveSegment(...
    (allData,refPulse,stepAngle,methodIn,debugPlotin,savePathin)
% THZ CT Computes the inverse radon transform for the slice of data files
%
% This function opens the THz files CT angle files and
% computes the inverse Radon Transform to obtain an image of
% the slice.
%
% Wavelet based segmentation is used to find the position of each pixels for classification application
%
% Sunny Yin
% November 2006

useVial = 1;
figure(1);clf;

if (nargin >= 4)
    method = methodIn;
else
    method = 1;
end

if (nargin >= 5)
    debugPlot = debugPlotin;
else
    debugPlot = 0;
end

if (nargin >= 6)
Appendix E Matlab Code

printPlots = 1;
savePath = savePathin;
else
    printPlots = 0;
savePath = ' ';
end

methodIndex=1;

printOpts = '-deps2c';

nX = size(allData,2);
nAngle = size(allData,4);

nY = size(allData,3);

% refPulse = allData(:,1,1,1);

useVial = 1;
refPulse = allData(:,2,2,2);

% use only y = 2
y = floor(nY/2)+1;

for y=5:9
    %for y=7

        pad = 2;
        fallData = fft(allData(:,:,y,:),size(allData,1)*pad);
        refFFT = fallData(:,2,2);

        imagData = imag(fallData);
        absData = abs(fallData);

        imagRef = ones(size(refFFT)); %imag(refFFT);
        absRef = ones(size(refFFT));
        if 1==1
            imagRef = imag(refFFT);
            absRef = abs(refFFT);
        end

    % image fusion and 2D wavelet energy for segment detection
    % Sunny Yin
    % Feb.2006

    figure(gcf+1);
    nF= size(fallData,1);
    nF=200;
    f=1:nF;
    for nf =1:nF

        R = reshape(fallData(f(nf),:,:),nX,nAngle);
        R = -log(R / refFFT(f(nf)));
        a = iradon(real(R),stepAngle,45,'spline','Hamming');%,'Hamming'),%interp2(b,1)

        % invert and threshold a
        b = normMatrix(a);
b = threshold(b, 0.1, 0.7);
b = wiener2(b, [3 3]);
bI(:, :, nf) = interp2(b, 1);
end

figure(gcf+1); fusbIclsSum1 = bI(:, 2)/15+bI(:, 3)/15+bI(:, 4)/15+bI(:, 5)/15+bI(:, 6)/15*...
bI(:, 7)*2/15+bI(:, 8)*2/15+bI(:, 9)*2/15+bI(:, 10)*2/15+bI(:, 11)*2/15; fusbIclsSum(:, :, nf) = fusbIclsSum1;
imagesc([size(fusbIclsSum1, 2) :-1:0], [0: size(fusbIclsSum1, 1)], interp2(fusbIclsSum1, 1));
colormap('default'); formatImage(2); grid off;
colormap('default'); xlabel('mm'); ylabel('mm');

[cA, cH, cV, cD] = dwt2(fusbIclsSum1, 'db4'); % 2D wavelet decomposition to 2 levels;
[cA1, cH1, cV1, cD1] = dwt2(cA, 'db4');

[siz = s(size(s, 1), :);]
ca1 = appcoef2(c, s, 'db4', 1); % subtract the low pass coeffi;
a1 = upcoef2('a', ca1, 'db4', 1, siz); % reconstruction the low pass

figure (gcf+1); clf;
subplot(2, 2, 1); subplot(2, 2, 1); subplot(2, 2, 1);
imagesc([size(a1, 2) :-1:0], [0: size(a1, 1)], interp2(a1, 1)); formatImage(2); grid off;
colormap('default');
title('approximation coef. recons.'); xlabel('mm'); ylabel('mm');

chd1 = detcoef2('h', c, s, 1);
hd1 = upcoef2('h', chd1, 'db4', 1, siz);

subplot(2, 2, 2)
imagesc([size(hd1, 2) :-1:0], [0: size(hd1, 1)], interp2(hd1, 1)); formatImage(2); grid off;
colormap('default');
title('horizontal detail coef. recons.);
xlabel('mm'); ylabel('mm');

cvd1 = detcoef2('v', c, s, 1);
vdi = upcoef2('h', cvd1, 'db4', 1, siz);

subplot(2, 2, 3)
imagesc([size(vd1, 2) :-1:0], [0: size(vd1, 1)], interp2(vd1, 1)); formatImage(2); grid off;
colormap('default');
title('vertical detail coef. recons.'); xlabel('mm'); ylabel('mm');

cdd1 = detcoef2('d', c, s, 1);
ddi = upcoef2('d', cdd1, 'db4', 1, siz);

subplot(2, 2, 4)
imagesc([size(dd1, 2) :-1:0], [0: size(dd1, 1)], interp2(dd1, 1)); formatImage(2); grid off;
colormap('default');
title('diagonal detail coef. recons.); xlabel('mm'); ylabel('mm');

% second level of reconstruction
Appendix E Matlab Code

ca2=appcoef2(c,s,'db4',2); % subtract the low pass coeffi;
a2=upcoef2('a',ca2,'db4',2,siz); % reconstruction the low pass

figure (gcf+1);
subplot(2,2,1); % plot
imagesc([size(a2,2):-1:0],[0:size(a2,2)],interp2(a2,2)); formatImage(2); grid off;
colormap('default');
title('approximation coef. recons.'); xlabel('mm'); ylabel('mm');

chd2=detcoef2('h',c,s,2);
hd2=upcoef2('h',chd2,'db4',2,siz);

subplot(2,2,2)
imagesc([size(hd2,2):-1:0],[0:size(hd2,1)],interp2(hd2,1)); formatImage(2); grid off;
colormap('default')
title('horizontal detail coef. recons.'); xlabel('mm'); ylabel('mm');

cvd2=detcoef2('v',c,s,2);
v2=upcoef2('v',cvd2,'db4',2,siz);

subplot(2,2,3)
imagesc([size(vd2,2):-1:0],[0:size(vd2,1)],interp2(vd2,1)); formatImage(2); grid off;
colormap('default')
title('vertical detail coef. recons.'); xlabel('mm'); ylabel('mm');

cdd2=detcoef2('d',c,s,2);
d2=upcoef2('d',cdd2,'db4',2,siz);

subplot(2,2,4)
imagesc([size(dd2,2):-1:0],[0:size(dd2,1)],interp2(dd2,1)); formatImage(2); grid off;
colormap('default')
title('diagonal detail coef. recons.'); xlabel('mm'); ylabel('mm');

ca3=appcoef2(c,s,'db4',3); % subtract the low pass coeffi;
a3=upcoef2('a',ca3,'db4',3,siz); % reconstruction the low pass

figure (gcf+1);
subplot(2,2,1); % plot
imagesc([size(a3,2):-1:0],[0:size(a3,2)],interp2(a3,2)); formatImage(2); grid off;
colormap('default');
title('approximation coef. recons.'); xlabel('mm'); ylabel('mm');

chd3=detcoef2('h',c,s,3);
hd3=upcoef2('h',chd3,'db4',3,siz);

subplot(2,2,2)
imagesc([size(hd3,2):-1:0],[0:size(hd3,1)],interp2(hd3,1)); formatImage(2); grid off;
colormap('default')
title('horizontal detail coef. recons.'); xlabel('mm'); ylabel('mm');

cvd3=detcoef2('v',c,s,3);
v3=upcoef2('v',cvd3,'db4',3,siz);

vd3=upcoef2('h',cvd3,'db4',3,siz);
subplot(2,2,3)
imagesc([size(vd3,2):-1:0], [0:size(vd3,1)], interp2(vd3,1)); formatImage(2); grid off;
colormap('default')
title('vertical detail coef. recons.'); xlabel('mm'); ylabel('mm');

cdd3=detcoef2('d',c,s,3);
dd3=upcoef2('d',cdd3,'db4',3,siz);

subplot(2,2,4)
imagesc([size(dd3,2):-1:0], [0:size(dd3,1)], interp2(dd3,1)); formatImage(2); grid off;
colormap('default')
title('diagonal detail coef. recons.'); xlabel('mm'); ylabel('mm');

[c,s]=wavedec2(fushIclsSum1,4,'db4');
siz=s(size(s,1),:);

cal=appcoef2(c,s,'db4',1); % subtract the low pass coeffi;
a1=upcoef2('a',cal,'db4',1,siz);

cal2=appcoef2(c,s,'db4',2); % subtract the low pass coeffi;
a2=upcoef2('a',cal2,'db4',2,siz);

cal3=appcoef2(c,s,'db4',3); % subtract the low pass coeffi;
a3=upcoef2('a',cal3,'db4',3,siz);

------- Wavelet scale correlation based segmentation -------

[Ea,Eh,Ev,Ed] = wenergy2(c,s);

coeCor12=a1.*a2;
coeCor23=a2.*a3;

coePW1=sum(sum(a1.^2));
coePW2=sum(sum(a2.^2));

coePC1=sum(sum(coeCor12.^2));
coePC2=sum(sum(coeCor23.^2));

coeRWCor1=(coePW1/coePC1)^(1/2);
coeRWCor2=(coePW2/coePC2)^(1/2);

coeNCor1=coeCor12 * coeRWCor1;
coeNCor2=coeCor23 * coeRWCor2;

coeNCor1Den=(abs(coeNCor1)>>a1).*coeNCor1;
coeNCor2Den=(abs(coeNCor2)>>a2).*coeNCor2;

figure (gcf+1); clf;
subplot(2,2,1)
imagesc([size(coeNCor1Den,2):-1:0], [0:size(coeNCor1Den,1)], interp2(coeNCor1Den,1));
formatImage(2); grid off; colormap('default')
Appendix E Matlab Code

```matlab
title('ampl. covolu. between levels 1 and 2'); xlabel('mm'); ylabel('mm');

subplot(2,2,2)
imagesc([size(coeNCor2Den,2):-1:0],[0:size(coeNCor2Den,1)],interp2(coeNCor2Den,1));
formatImage(2); grid off; colormap('default')
title('ampl. covolu. between levels 2 and 3'); xlabel('mm'); ylabel('mm');

tcoeNCor1DenThr= (coeNCor1Den>0);
copyCoeNCor1DenThr=coeNCor1DenThr;
tcoeNCor1DenThrAir= (coeNCor1Den<=0);
norModSubBIThAir=coeNCor1DenThrAir;

subplot(2,2,3)
imagesc([size(copyCoeNCor1DenThr,2):-1:0],[0:size(copyCoeNCor1DenThr,1)]);
interp2(copyCoeNCor1DenThr,1)); formatImage(2);
grid off; colormap('default')
title('subtract segments'); xlabel('mm'); ylabel('mm');

copyCoeNCor1DenThrCan =edge(copyCoeNCor1DenThr,'canny');

subplot(2,2,4)
imagesc([size(copyCoeNCor1DenThrCan,2):-1:0],[0:size(copyCoeNCor1DenThrCan,1)]);
interp2(copyCoeNCor1DenThrCan,1)); formatImage(2); grid off; colormap('default')
title('subtract segments'); xlabel('mm'); ylabel('mm');

[xSubBITh,ySubBITh] = find(copyCoeNCor1DenThrCan);
numSubBITh =length(xSubBITh);

xSubBIThMaxPosi=max(xSubBITh); xSubBIThMinPosi=min(xSubBITh);
xSubBIThCentrPosi=(xSubBIThMaxPosi-xSubBIThMinPosi)/2;
% central point coordination is very important which
t% determine the modulus and the color match for the different circle
xSubBIThCentrPosi=xSubBIThCentrPosi+xSubBIThMinPosi;
ySubBIThMaxPosi=max(ySubBITh); ySubBIThMinPosi=min(ySubBITh);
ySubBIThCentrPosi=(ySubBIThMaxPosi-ySubBIThMinPosi)/2;
ySubBIThCentrPosi=ySubBIThCentrPosi+ySubBIThMinPosi;

% coordination of central position: (47,40)
t% calculate the modulus between the central position and edge of the vial
% and the tube

[xAxisMod,yAxisMod]=size(copyCoeNCor1DenThrCan);
ModSubBITh=zeros(xAxisMod,yAxisMod);
theta=zeros(xAxisMod,yAxisMod);
for k=1:numSubBITh
    ModSubBITh(xSubBITh(k),ySubBITh(k))=((xSubBITh(k)-xSubBIThCentrPosi)^2+...
(ySubBITh(k)-ySubBIThCentrPosi)^2)^(1/2); % modulus
    theta(ySubBITh(k),ySubBITh(k))=atan((ySubBITh(k)-ySubBIThCentrPosi)/...
(xSubBITh(k)-xSubBIThCentrPosi)); % angle between the
```
%calculated modulus vector and the x axis

end

figure(gcf+1)
subplot(2,2,1);
imagesc([size(ModSubBITh,2):-1:0],[size(ModSubBITh,1):-1:0],interp2(ModSubBITh,2));
set(gca,'YDir','normal');
formatImage(2); grid off;
colormap('default')
xlabel('mm'); ylabel('mm')
norModSubBITh=ModSubBITh./max(max(ModSubBITh));

subplot(2,2,2);
%use histogram to find the threshold
h=imhist(norModSubBITh);
%p=h/numel(fusbIclsSum2);
h1=h(1:255);
horz=1:255;
bar(horz,h1)
stem(horz,h1,'b--','fill')
axis([0 255 0 30])
set(gca,'xtick',0:30:255)
set(gca,'ytick',0:10:30)

TTube=graythresh(norModSubBITh);
% a little adjustment of threshold for the target image with blurred boundary.
%if y==7
% TTube=TTube-0.2;
% end

norModSubBIThVial=(norModSubBITh >TTube+0.04*(y-5)); % test if there are some other points left
if y==5
 norModSubBIThVial=(norModSubBITh >TTube+0.2);
end
subplot(2,2,3);
imagesc([size(norModSubBIThVial,2):-1:0],[size(norModSubBIThVial,1):-1:0],interp2(norModSubBIThVial,2));
set(gca,'YDir','normal');
formatImage(2); grid off;
colormap('default')
xlabel('mm'); ylabel('mm')
norModSubBIThVialTub=(norModSubBIThVial >0); % test if there are some other points left
norModSubBIThVialTub=norModSubBIThVialTub-norModSubBIThVial;

subplot(2,2,4);
imagesc([size(norModSubBIThVialTub,2):-1:0],[size(norModSubBIThVialTub,1):-1:0],interp2(norModSubBIThVialTub,2));
set(gca,'YDir','normal');
formatImage(2); grid off;
colormap('default')
xlabel('mm'); ylabel('mm')
copynorModSubBIThTub = norModSubBIThTub;
nYbI=size(norModSubBIThTub,2);

[xx,yy]=find(norModSubBIThTub);
leng=length(xx);
for jj=1:leng
  xx(jj) record the x axis position
  maxYY=max(yy(xx==xx(jj)));  
  minYY=min(yy(xx==xx(jj)));  
  for RR=0:nYbI
    if RR<=maxYY & RR>=minYY
      norModSubBIThTub(xx(jj),RR)=1;
    end
  end
end

figure(gcf+1)
subplot(2,2,1);  
imagesc([size(copynorModSubBIThTub,2):-1:0],[size(norModSubBIThTub,1):-1:0],...  
    interp2(copynorModSubBIThTub,2)); set(gca, 'YDir','normal');
formatImage(2); grid off;
colormap('default')
xlabel('mm'); ylabel('mm')

subplot(2,2,2);
imagesc([size(norModSubBIThTub,2):-1:0],[size(norModSubBIThTub,1):-1:0],interp2(norModSubBIThTub,2));
set(gca, 'YDir','normal');
formatImage(2); grid off;
colormap('default')
xlabel('mm'); ylabel('mm')
norModSubBIThVial=((coeNCor1DenThr- norModSubBIThTub)>0);

subplot(2,2,3);
imagesc([size(norModSubBIThVial,2):-1:0],[size(norModSubBIThVial,1):-1:0],interp2(norModSubBIThVial,2));
set(gca, 'YDir','normal');
formatImage(2); grid off;
colormap('default')
xlabel('mm'); ylabel('mm')

subplot(2,2,4);
imagesc([size(norModSubBIThAir,2):-1:0],[size(norModSubBIThAir,1):-1:0],interp2(norModSubBIThAir,2));
set(gca, 'YDir','normal');
formatImage(2); grid off;
colormap('default')
xlabel('mm'); ylabel('mm')

%  --------------record segements for classification application-----------------
claBIairSub= norModSubBIThTub;  
norModSubBIThTube= norModSubBIThTub;
segVialBITh= norModSubBIThVial;

% select the same number of pixels from each segment
[XposClaBIair,YposClaBIair]=find(claBIairSub);
XnumAir=length(XposClaBIair);% 6381 --- 5748
Wavelet Based Local Tomography via THz Pulsed Measurements

localTomography_PulsedTHz.m

% THz CT with a LIA was used to obtain a 1D image of a polystyrene sample.
% wavelet based local tomography on the centered area of interests
% Time domain parameter is extracted via applying correlation algorithms
% for the sinogram images (Radon transforms)
% The University of Adelaide
% Xiaoxia Yin
% January 2007

clear;
figure(1); clf;

% open the file and read the header information
fid = fopen(fileName,'r','n');
mHeader = fscanf(fid,'%f',8); % 6,'int32')

mDat = fscanf(fid,'%f');
fclose(fid);

nX = ceil(mHeader(1)/mHeader(2))+1;
dX = mHeader(2) / 1000; % mm
nY = ceil(mHeader(3)/mHeader(4))+1;
dY = mHeader(4) / 1000; % mm
nTime = ceil(mHeader(5)/mHeader(6))+1;
dTime = mHeader(6) / 1.5e8 * 1e6; % ps
time = [0:nTime-1]*dT ime;

nAngle = ceil(mHeader(7)/mHeader(8));
dAngle = mHeader(8); % degrees

mDat = reshape(mDat,nAngle,nTime,nX*nY);

truncateData = 0;
if (truncateData == 1)
    mDat = mDat(:,1:29,:);
    nTime = size(mDat,2);
    time = [0:nTime-1]*dT ime;
end

[amp,delay]=max(mDat(:,1:end,:),[],2);
amp = reshape(amp,nAngle,nX*nY);
delay = reshape(delay,nAngle,nX*nY);
end

if (crossCorr == 1)
    % use cross-correlation to get the delay.
    interpFactor = 10;
    subsampleTime = 2; % set to 2 to skip every second sample
    % first we interpolate the signals to improve the resolution
    refInterp = interpft(mDat(2,1:subsampleTime:end,2),nTime*interpFactor);
    interpDelay = zeros(nAngle,nX);
    for i = 1:nAngle
        g = interpft(mDat(i,1:subsampleTime:end,:),nTime*interpFactor);
        % note xcorr2 is much slower than this loop
        for j = 1:nX*nY
            h = xcorr(g(:,j),refInterp);
            % h = abs(h);
            % apply some smoothing to h
            h = boxfilt(h,60);
            [a,b] = max(h);
            interpDelay(i,j) = b;
        end
    end
    figure(gcf+1);clf;
    imagesc([0:nX*nY-1]*dY,[0:nAngle-1]*dAngle,interpDelay);
    formatImage(2);
    title('XCorr Timing Sinogram');
    ylabel('Angle (degrees)');
    xlabel('X (mm)');
if (printPlots)
    print(printOpts,strcat(printName,'xCorrSinogram'));
end

if 1==1
    sinogramOut = interpDelay;

    % preprocess the sinogram
    if (interpMin ~= 0)
        sinogramOut = interpDelay - min(min(interpDelay));
        sinogramOut(sinogramOut < interpMin) = 0;
    end

    filtSinogram = 1;
    if (filtSinogram == 1)
        % wrap around the top and bottom rows of the delay and wiener filter
        wrapSinogram = [sinogramOut(end,:);sinogramOut(1,:);];
        filtSinogram = wiener2(wrapSinogram,[2,2]);
        sinogramOut = filtSinogram(2:end-1,:);
    end

    debugPlot = 1;
    if (debugPlot == 1)
        figure(gcf+1);clf;
        imagesc([0:size(sinogramOut,1)-1]*dAngle,[0:size(sinogramOut,2)-1]*dY,sinogramOut');
        formatImage(2);  
        title('Sinogram');  
        ylabel('Y (mm)');
        xlabel('Angle (degrees)');
    end

    AA=zeros(34,nAngle);
    sinogramOut1=sinogramOut';
    if localCT==1;  %local tomography of the centered interest area
        AA=zeros(29,nAngle);
        sinogramOut1 = [AA;sinogramOut1(30:72,1:end);AA];
        reconCT = iradon(sinogramOut1,[],'linear','Ram-Lak',100);
    else
        reconCT = iradon(sinogramOut1,[],'linear','Ram-Lak',100);
    end
    % if (reconThresh == 1)
    %    reconCT = normMatrix(reconCT);
    %    reconCT = threshMatrix(reconCT,reconLowThresh,reconHighThresh,1);
    % end

    debugPlot = 1;
    if (debugPlot == 1)
        figure(gcf+1);clf;
        % imagesc([0:size(reconCT,1)]*dX,[0:size(reconCT,2)]*dX,interp2(abs(reconCT)));
        reconCT=interp2(reconCT,1);
        imagesc([0:size(reconCT,1)]*dX,[0:size(reconCT,2)]*dX,abs(reconCT));

xlabel('X (mm)');
end
end

p=sinogramOut1;
len=size(p,1);
order = max(64,2^nextpow2(2*len));

% First create a ramp filter - go up to the next highest
% power of 2.

filt = 2*( 0:(order/2 ) )./order;

w = 2*pi*(0:size(filt,2)-1)/order; % frequency axis up to Nyquist

d=1;

filt(w>pi*d) = 0; % Crop the frequency response
filt = [filt' ; filt(end-1:-1:2)'] ; % Symmetry of the filter

H=filt;Nfft=length(H);

p(length(H),1)=0; % Zero pad projections

% In the code below, I continuously reuse the array p so as to
% save memory. This makes it harder to read, but the comments
% explain what is going on.

pF = fft(p,Nfft); % p holds fft of projections

for i = 1:size(pF,2)
    pp(:,i) = pF(:,i).*H; % frequency domain filtering
end

figure(gcf+1); clf;
plot(real(pp(:,2)));hold on;
plot(real(pF(:,2)),'r');

if 1==1
    pt = real(ifft(pp)); % p is the filtered projections
    pt(len+1:end,:) = [] ; % Truncate the filtered projections
    figure(gcf+1);clf;
    plot(pt(:,i)); hold on;
    plot(sinogramOut1(:,i),'r');
    figure(gcf+1);clf;
    sinogramOut1 = [repmat(pt(36,1:end),35,1);pt(36:67,1:end); repmat(pt(67,1:end),34,1)];
    imagesc( sinogramOut1);
    reconCT1 = backproj1( sinogramOut1,[],'linear',100);
    if (debugPlot == 1)
        figure(gcf+1);clf;reconCT1=interp2(reconCT1,1);% keep the same size with wavelet LCT
        imagesc([0:size(reconCT1,1)]*dX,[0:size(reconCT1,2)]*dX,abs(reconCT1));
        formatImage(2);
        title('Filtered Back Projection');
        ylabel('Y (mm)');
        xlabel('X (mm)');
Nfft = length(H);
[LoD, hiD, LoR, hiR] = wfilters('bior2.2');
LoD = fft(LoD',Nfft);
HiD = fft(hiD',Nfft);

% approximation subband
SrampA = pp;
theta = 0:nAngle-1;
theta_rad = theta*dAngle*pi/180;

for itheta = 1:nAngle
    Sramp(:,itheta) = pp(:,itheta).*FFTscale(LoD,sin(theta_rad(itheta)),'Lin').*FFTscale(LoD,cos(theta_rad(itheta)),'Lin');
    Sramp(:,itheta) = boxfilt( Sramp(:,itheta),60);
end;

SrampA = real(ifft(SrampA));  % SrampA is the scaling ramp filtered projections
SrampA(len+1:end,:) = [];  % Truncate the filtered projections
figure(gcf+1);
imagesc(SrampA);
SrampA=[repmat(SrampA(36,1:end),36,1);SrampA(36:67,1:end); repmat(SrampA(67,1:end),34,1)];

figure(gcf+1);
imagesc(SrampA);grid on;

RA = backproj1(SrampA,[],'linear',100);
if (debugPlot == 1)
    figure(gcf+1);clf;
    imagesc([0:size(RA,1)]*dX,[0:size(RA,2)]*dX,abs(RA));
    formatImage(2);
    title('Approximate Subimage');
    ylabel('Y (mm)');
    xlabel('X (mm)');
end

SrampD1=p;  % the projections being filtered along horizontal direction
for itheta = 1:nAngle
    SrampD1(:,itheta) = pp(:,itheta).*FFT_scale(LoD,sin(theta_rad(itheta)),'Lin').*FFT_scale(HiD,cos(theta_rad(itheta)),'Lin');
    SrampD1(:,itheta) = boxfilt( SrampD1(:,itheta),60);
end;

SrampD1 = real(ifft( SrampD1));  % SrampD1 is the wavelet ramp filtered projections with horizontal orientations
SrampD1(len+1:end,:) = [];  % Truncate the filtered projections
SrampD1 = [repmat(SrampD1(36,1:end),35,1); SrampD1(36:67,1:end); repmat(SrampD1(67,1:end),34,1)];
% reconstructed horizontal detailed subimage
RD1 = backproj1( SrampD1,[],'linear',100);
for itheta = 1:nAngle % along vertical direction
    SrampD2(:,itheta) = pp(:,itheta).*FFT_scale(HiD,sin(theta_rad(itheta)),'Lin').*...
    FFT_scale(LoD,cos(theta_rad(itheta)),'Lin');
    SrampD2(:,itheta) = boxfilt( SrampD2(:,itheta),60);
end;

SrampD2 = real(ifft( SrampD2)); % wavelet ramp filtered projections with vertical orientations
SrampD2 = [repmat(SrampD2(36,1:end),35,1); SrampD2(36:67,1:end); repmat(SrampD2(67,1:end),34,1)];
RD2 = backproj1( SrampD2,[],'linear',100); %vertical detail subimage

for itheta = 1:nAngle % along diagonal directions
    SrampD3(:,itheta) = pp(:,itheta).*FFT_scale(HiD,sin(theta_rad(itheta)),'Lin').*...
    FFT_scale(HiD,cos(theta_rad(itheta)),'Lin');
    SrampD3(:,itheta) = boxfilt( SrampD3(:,itheta),60);
end;

SrampD3 = real(ifft( SrampD3)); % wavelet ramp filtered projections with diagonal orientations
SrampD3 = [repmat(SrampD3(36,1:end),35,1); SrampD3(36:67,1:end); repmat(SrampD3(67,1:end),34,1)];
RD3 = backproj1( SrampD3,[],'linear',100); %diagonal detail subimage

frec = idwt2(RA(1:2:end,1:2:end), RD1(1:2:end,1:2:end), RD2(1:2:end,1:2:end), RD3(1:2:end,1:2:end),loR,hiR);
figure(gcf+1);
imagesc([0:size(frec,1)]*dX,[0:size(frec,2)]*dX,abs(frec));
formatImage(2);
title('Wavelet Based Tomography');
ylabel('Y (mm)');
xlabel('X (mm)');

frec = idwt2(RA(1:end,1:end), RD1(1:end,1:end), RD2(1:end,1:end), RD3(1:end,1:end),loR,hiR);
figure(gcf+1);
imagesc([0:size(frec,1)]*dX,[0:size(frec,2)]*dX,abs(frec));
formatImage(2);
title('Wavelet Based Tomography');
ylabel('Y (mm)');
xlabel('X (mm)');
figure(gcf+1);
imagesc([0:size(frec,1)]*dX,[0:size(frec,2)]*dX,interp2(abs(frec)));
formatImage(2);
reconN = frec *dTime/interpFactor/1e12/dY/1e-3 * 3e8;

S1=size(reconCT1);
S2=size(frec);
diff1 = -(reconCT1(20:180,20:180)-frec(20:180,20:180));
figure(gcf+1);clf;
imagesc([0:size(diff1,1)]*dX,[0:size(diff1,2)]*dX,abs(diff1));
formatImage(2);
title('difference');
ylabel('Y (mm)');
xlabel('X (mm)');

function Hsc = FFT_scale(H,a,interp)
% evaluates H(aw) given H(w)
% need a <= 1
% interp = 'NN'  use nearest neighbour interpolation (default)
% = 'Lin'  use linear interpolation
%
% Xiaoxia Yin
% December 2006
%
len = length(H);
Hsc = zeros(size(H));
if (nargin < 3)
  interp = 'NN';
end;
if (a >= 0)
  if strcmp(interp, 'Lin')
    for ii = 1:len/2
      x = a*(ii-1)+1;
      if (mod(x,1) == 0)
        Hsc(ii) = H(x);
      else
        xi = floor(x);
        Hsc(ii) = H(xi) + (H(xi+1)-H(xi))*x-xi;
      end;
    end;
  elseif strcmp(interp, 'NN')
    idx = round(a*(0:(len/2-1)))+1;
    Hsc(1:len/2) = H(idx);
  end;
else
  if strcmp(interp, 'Lin')
    for ii = 1:len/2
      x = abs(a)*(ii-1)+1;
      if (mod(x,1) == 0)
        Hsc(ii) = H(x);
      else
        xi = floor(x);
        Hsc(ii) = H(xi) + (H(xi+1)-H(xi))*x-xi;
      end;
    end;
  elseif strcmp(interp, 'NN')
    idx = round(a*(0:(len/2-1)))+1;
    Hsc(1:len/2) = H(idx);
  end;
end;
else
    x1 = floor(x);
    Hsc(ii) = (H(x1) + (H(x1+1)-H(x1))*(x-x1))';
end;
end;
elseif strcmp(interp, 'NN')
    idx = round(abs(a)*(0:(len/2-1)))+1;
    Hsc(1:len/2) = H(idx)';
end;
Hsc((len/2+1):len) = Hsc(len/2:-1:1)';

function [img,tI,H] = Sepbackproj(varargin)
p = varargin{1};
theta = pi / size(p,2);
theta = (0:(size(p,2)-1))* theta;

string_args = {'nearest neighbor', 'linear', 'spline'};
arg = varargin{3};
idx = strmatch(lower(arg),string_args);
interp = string_args{idx};
N =varargin{4};
len=size(p,1);

img = zeros(N); % Allocate memory for the image.
% Define the x & y axes for the reconstructed image so that the origin
% (center) is in the spot which RADON would choose.
xax = (1:N)-ceil(N/2);
x = repmat(xax, N, 1); % x coordinates, the y coordinates are rot90(x)
y = rot90(x);
costheta = cos(theta);
sintheta = sin(theta);
ctrIdx = ceil(len/2); % index of the center of the projections

% Zero pad the projections to size 1+2*ceil(N/sqrt(2)) if this
% quantity is greater than the length of the projections
imgDiag = 2*ceil(N/sqrt(2))+1; % largest distance through image.
if size(p,1) < imgDiag
    rz = imgDiag - size(p,1); % how many rows of zeros
    p = [zeros(ceil(rz/2),size(p,2)); p; zeros(floor(rz/2),size(p,2))];
    ctrIdx = ctrIdx+ceil(rz/2);
end

% Backprojection - vectorized in (x,y), looping over theta
if strcmp(interp, 'nearest neighbor')
    for i=1:length(theta)
proj = p(:,i); 
    t = round(x.*costheta(i) + y.*sintheta(i)); 
    img = img + proj(t+ctrIdx); 
end

e
else if strcmp(interp, 'linear')
    for i=1:length(theta)
        proj = p(:,i);
        t = x.*costheta(i) + y.*sintheta(i);
        tI(:,:,i)=t; % record the intresting area
        a = floor(t);
        img = img + (t-a).*proj(a+1+ctrIdx) + (a+1-t).*proj(a+ctrIdx);
    end
else
    elseif strcmp(interp, 'spline')
        for i=1:length(theta)
            proj = p(:,i);
            taxis = (1:size(p,1)) - ctrIdx;
            t = x.*costheta(i) + y.*sintheta(i);
            projContrib = interp1(taxis,proj,t(:),'*spline');
            img = img + reshape(projContrib,N,N);
        end
    end
end

img = img*pi/(2*length(theta));

________________________

Local Reconstruction via THz QCL

THzQCL_LT.m

close all;
clear;
clc;

%--------Read data
A1=max(A)/A;
Am=max(A1)
Nor=Am./A; dx=0.2;
figure(gcf+1);
imagesc(Nor)
sinogramOut1=log(Nor);
filtSinogram = wiener2(sinogramOut1,[2,2]);
figure(gcf+1)
imagesc(filtSinogram)
figure(gcf+1)
imagesc(sinogramOut1);

%-------- global reconstruction via FBP
reconCT = iradon( filtSinogram,[],'nearest','Hann',250);

%---- Local reconstruction via FBP
if 1==0
size(A); nAngle=size(A,2)
AA=zeros(50,nAngle);
BB=zeros(50,nAngle);

% sinogramOut2 = [AA;filtSinogram(81:180,1:end);BB];
sinogramOut2 = [AA;filtSinogram(51:239,1:end);BB];
%S3: sinogramOut2 = [AA;sinogramOut1(11:240,1:end);BB];
figure(gcf+1);
imagesc(sinogramOut2)
reconCT = iradon(sinogramOut2,[],'nearest','Hann',250);
figure(gcf+1)
imagesc(reconCT);
ylabel('Y (mm)');
xlabel('X (mm)');

p=sinogramOut2;
len=size(p,1);
order = max(64,2^nextpow2(2*len));

% First create a ramp filter - go up to the next highest
% power of 2.

filt = 2*(0:(order/2))./order;
w = 2*pi*(0:size(filt,2)-1)/order; % frequency axis up to Nyquist

d=1;
filt(w>pi*d) = 0; % Crop the frequency response
filt = [filt' ; filt(end-1:-1:2)']; % Symmetry of the filter
H=filt;Nfft=length(H);

p(length(H),1)=0; % Zero pad projections

% In the code below, I continuously reuse the array p so as to
% save memory. This makes it harder to read, but the comments
% explain what is going on.

pF = fft(p,Nfft); % p holds fft of projections

for i = 1:size(pF,2)
    pp(:,i) = pF(:,i).*H; % frequency domain filtering
end

figure(gcf+1); clf;
plot(real(pp(:,2)));hold on;
plot(real(pF(:,2)),'r');
pt = real(ifft(pp)); % p is the filtered projections
pt(len+1:end,:) = []; % Truncate the filtered projections
figure(gcf+1);clf;
plot(pt(:,i)); hold on;
plot(sinogramOut2(:,i),'r');
figure(gcf+1);clf;
imagesc( sinogramOut2);
figure(gcf+1);clf;
sinogramOut3 = [repmat(pt(81,1:end),80,1);pt(81:190,1:end); repmat(pt(190,1:end),99,1)];
% hole reconstruction
imagesc( sinogramOut3);
[reconCT1,tI] = backproj1( sinogramOut3,[],'linear',250);
figure(gcf+1);clf;
imagesc([0:size(reconCT1,2)]*dX,[0:size(reconCT1,1)]*dX,interp2(abs(reconCT1),2));
ylabel('Y (mm)');
xlabel('X (mm)'); formatImage(2);

figure(gcf+1);clf;% keep the same size with wavelet LCT
dX=0.2; % a small adjustment of the zoomed image based on the zoomed global data..
imagesc([0:size(reconCT1(80+2:229-35,78-4:166),2)]*dX,[0:size(reconCT1(80+2:229-35,78-4:166),1)]*dX,...
interp2(abs(reconCT1(80+2:229-35,78-4:166)),2)); formatImage(2);
title('Filtered Back Projection');
ylabel('Y (mm)');
xlabel('X (mm)');

figure(gcf+1);clf;
SegTrac=reconCT1(80+2:229-35,78-4:166);
min(min(SegTrac))
max(max(SegTrac))
SegTrac1=ceil(SegTrac.*10000+109);
min(min(SegTrac1))
max(max(SegTrac1));
figure(gcf+1);clf;
imagesc([0:size(SegTrac1,2)]*dX,[0:size(SegTrac1,1)]*dX,interp2(abs(SegTrac1),2));
formatImage(2);
title('Filtered Back Projection');
ylabel('Y (mm)');
xlabel('X (mm)');
size(SegTrac1);
figure(gcf+1);clf;

for i=1:195
    ImageHis(i)=length(find(SegTrac1==i));
end
[a,b]=max(ImageHis);
figure(gcf+1);clf;
Bar=ImageHis(1:195);
horz=1:195;
bar(horz,Bar);axis on; grid on;
axis([0 195 0 158])
" h = findobj(gca,'Type','patch');
set(h,'FaceColor','r','EdgeColor','w')"
imagesc(HoldInd);
axis on; grid on;
% chasing the edge
[imX,imY]=size(HoldInd);
coY=ceil(imY/2); % 47 pixel
coX=ceil(imX/2); % 57 pixel
if 1==1
  for Iy=1:coY-1
    for Ix=1:coX-1
      A1= HoldInd( coX-1+Ix-1,coY-1+Iy+1); % calculation start from (coX,coY)
      A2= HoldInd( coX-1+Ix, coY-1+Iy+1);
      A3= HoldInd( coX-1+Ix+1,coY-1+Iy+1);
      A4= HoldInd( coX-1+Ix-1,coY-1+Iy);
      A5= HoldInd( coX-1+Ix+1,coY-1+Iy);
      A6= HoldInd( coX-1+Ix, coY-1+Iy);
      A7= HoldInd( coX-1+Ix+1,coY-1-1+Iy);
      A8= HoldInd( coX-1+Ix, coY-1-1+Iy);
      A9= HoldInd( coX-1+Ix+1,coY-1-1+Iy);
      A=[A1,A2,A3,A4,A5,A6,A7,A8,A9];
      if length(find(A==1))<=6
        HoldInd(coX-1+Ix,coY-1+Iy)=0;
      else
        HoldInd(coX-1+Ix,coY-1+Iy)=1;
      end
    end
  end
  figure(gcf+1);clf;
  imagesc(HoldInd);
  % the second quadrant of the coordinates
end

if 1==1
  for Iy=1:coY-2
    for Ix=1:coX-2
      A1= HoldInd( coX+Ix-1,coY-Iy+1); % calculation start from (coX,coY)
      A2= HoldInd( coX+Ix, coY-Iy);
      A3= HoldInd( coX+Ix+1,coY-Iy+1);
      A4= HoldInd( coX+Ix-1,coY-Iy);
      A5= HoldInd( coX+Ix+1,coY-Iy);
      A6= HoldInd( coX+Ix, coY-Iy);
      A7= HoldInd( coX+Ix+1,coY-1-Iy);
      A8= HoldInd( coX+Ix, coY-1-Iy);
      A9= HoldInd( coX+Ix+1,coY-1-Iy);
      if length(find(A==1))<=6
        HoldInd(coX+Ix,coY-1+Iy)=0;
      else
        HoldInd(coX+Ix,coY-1+Iy)=1;
      end
    end
  end
end
end
figure(gcf+1);clf;
imagesc(HoldInd);
end

if 1==1
  % the third quadrant of the coordinates
  for Iy=1:coY-2
    for Ix=1:coX-2
      A1= HoldInd( coX-Ix-1,coY-Iy+1);
      A2= HoldInd( coX-Ix,coY-Iy+1);
      A3= HoldInd( coX-Ix+1,coY-Iy+1);
      A4= HoldInd( coX-Ix-1,coY-Iy);
      A5= HoldInd( coX-Ix+1,coY-Iy);
      A6= HoldInd( coX-Ix,coY-Iy);
      A7= HoldInd( coX-Ix-1,coY-1-Iy);
      A8= HoldInd( coX-Ix,coY-1-Iy);
      A9= HoldInd( coX-Ix+1,coY-1-Iy);
      
      A=[A1,A2,A3,A4,A5,A6,A7,A8,A9];
      if length(find(A==1))<=6
        HoldInd(coX-Ix,coY-Iy)=0;
      else
        HoldInd(coX-Ix,coY-Iy)=1;
      end
    end
  end
  figure(gcf+1);clf;
  imagesc(HoldInd);
end

% the fourth quadrant of the coordinates
for Iy=1:coY-2
  for Ix=1:coX-2
    A1= HoldInd( coX-Ix-1,coY+Iy+1);
    A2= HoldInd( coX-Ix,coY+Iy+1);
    A3= HoldInd( coX-Ix+1,coY+Iy+1);
    A4= HoldInd( coX-Ix-1,coY+Iy);
    A5= HoldInd( coX-Ix+1,coY+Iy);
    A6= HoldInd( coX-Ix,coY+Iy);
    A7= HoldInd( coX-Ix-1,coY-1+Iy);
    A8= HoldInd( coX-Ix,coY-1+Iy);
    A9= HoldInd( coX-Ix+1,coY-1+Iy);

    A=[A1,A2,A3,A4,A5,A6,A7,A8,A9];
    if length(find(A==1))<=6
      HoldInd(coX-Ix,coY+Iy)=0;
    else
      HoldInd(coX-Ix,coY+Iy)=1;
    end
  end
end
figure(gcf+1);clf;
imagesc(HoldInd);

\%coX(fix) coY(mov)

for Ey=1:coY
    if length(find(HoldInd(:,coY-1+Ey)==1))<=8
        [Eya,Eyb]=find(HoldInd(:,coY-1+Ey)==1);
        HoldInd(Eya,Eyb)=0;
        HoldInd(:,coY-1+Ey+1:end)=0;
    end
end
figure(gcf+1);clf;
imagesc(HoldInd);

for Ey=1:coY
    if length(find(HoldInd(:,coY-Ey)==1))<=8
        [Eya,Eyb]=find(HoldInd(:,coY-Ey)==1);
        HoldInd(Eya,Eyb)=0;
        HoldInd(:,1:coY-Ey-1)=0;
        break
    end
end
figure(gcf+1);clf;
imagesc(HoldInd);

for Ex=1:coX-1
    if length(find(HoldInd(coX-Ex,:)==1))<=8
        [Eya,Eyb]=find(HoldInd(coX-Ex,:)==1);
        HoldInd(Eya,Eyb)=0;
        HoldInd(1:coX-Ex,:)=0;
    end
end
figure(gcf+1);clf;
imagesc(HoldInd);

for Ex=1:coX-1
    if length(find(HoldInd(coX+Ex,:)==1))<=8
        [Eya,Eyb]=find(HoldInd(coX+Ex,:)==1);
        HoldInd(Eya,Eyb)=0;
        HoldInd(coX+Ex:end,:)=0;
        break
    end
end
figure(gcf+1);clf;
imagesc(HoldInd);
formatImage(2);
figure(gcf+1);
imagesc([0:size(HoldInd,1)]*dX,[0:size(HoldInd,2)]*dX,interp2(abs(HoldInd),2));
formatImage(2);
title('Traditional Back Projection');
ylabel('Y (mm)');
xlabel('X (mm)');
end
%-----------wavelet based local tomography

size(A); %(289,18)
nAngle=size(A,2);
AA=zeros(50,nAngle);
BB=zeros(50,nAngle);

% sinogramOut2 = [AA;filtSinogram(81:180,1:end);BB];
sinogramOut2 = [AA;filtSinogram(51:239,1:end);BB];
%S3: sinogramOut2 = [AA; sinogramOut1(11:240,1:end);BB];
figure(gcf+1);
imagesc(sinogramOut2)

reconCT = iradon(sinogramOut2,[],'nearest','Hann',250);

p=sinogramOut2;
len=size(p,1);
order = max(64,2^nextpow2(2*len));

% First create a ramp filter - go up to the next highest
% power of 2.

filt = 2*(0:(order/2))./order;
w = 2*pi*(0:size(filt,2)-1)/order;  % frequency axis up to Nyquist

d=1;
filt(w>pi*d) = 0;  % Crop the frequency response
filt = [filt'; filt(end-1:-1:2)'];  % Symmetry of the filter

H=filt;Nfft=length(H);

p(length(H),1)=0;  % Zero pad projections

% In the code below, I continuously reuse the array p so as to
% save memory. This makes it harder to read, but the comments
% explain what is going on.

pF = fft(p,Nfft);  % p holds fft of projections

for i = 1:size(pF,2)
    pp(:,:,i) = pF(:,:,i).*H;  % frequency domain filtering
end

Nfft=length(H);
[loD,hiD,loR,hiR] = wfilters('bior2.2');

LoD = fft(loD',Nfft);
HiD = fft(hiD',Nfft);

dAngle=18;
% approximation subband
SrampA = pp;

theta = 0:nAngle-1;
theta_rad = theta*dAngle*pi/180;
for itheta = 1:nAngle
    SrampA(:,itheta) = pp(:,itheta).*FFT_scale(LoD,sin(theta_rad(itheta)),'Lin').*...
        FFT_scale(LoD,cos(theta_rad(itheta)),'Lin');
    % SrampA(:,itheta) = boxfilt( SrampA(:,itheta),60);
end;

SrampA = real(ifft(SrampA));  \% SrampA is the approximated ramp filtered projections
SrampA(len+1:end,:) = []; cSrampA=SrampA;
% Truncate the filtered projections
figure(gcf+1);
imagesc(SrampA);
formatImage(2);grid off;
title('Wavelet ramp filtered sinogram');

SrampA =[repmat(SrampA (81,1:end),80,1);SrampA(81:190,1:end); repmat(SrampA (190,1:end),99,1)];
[RA,WtI] = backproj1(SrampA,[],'linear',250);%calculate the area of interest
figure(gcf+1);clf;
imagesc([0:size(RA(80:229-35,78-4:166),2)]*dX,[0:size(RA(80:229-35,78-4:166),1)]*dX,interp2(abs(RA...
    (80:229-35,78-4:166)),2));
grid on;
formatImage(2);
title('Approximate Subimage');
ylabel('Y (mm)');
xlabel('X (mm)');
if 1==1
    SrampD1=p;
    for itheta = 1:nAngle
        SrampD1(:,itheta) = pp(:,itheta).*FFT_scale(LoD,sin(theta_rad(itheta)),'Lin').*...
            FFT_scale(HiD,cos(theta_rad(itheta)),'Lin');
        % SrampD1(:,itheta) = boxfilt( SrampD1(:,itheta),60);
    end;

    SrampD1 = real(ifft( SrampD1));
    SrampD1(len+1:end,:) = [];
    SrampD1 =[repmat(SrampD1(81,1:end),80,1); SrampD1(81:190,1:end); repmat(SrampD1(190,1:end),99,1)];
    % maximum intensity 6.2718
    RD1 = backproj1( SrampD1,[],'linear',250);

    for itheta = 1:nAngle
        SrampD2(:,itheta) = pp(:,itheta).*FFT_scale(HiD,sin(theta_rad(itheta)),'Lin').*...
            FFT_scale(LoD,cos(theta_rad(itheta)),'Lin');
        % SrampD2(:,itheta) = boxfilt( SrampD2(:,itheta),60);
    end;

    SrampD2 = real(ifft( SrampD2));
    SrampD2(len+1:end,:) = [];
    % Truncate the filtered projections
    SrampD2 =[repmat(SrampD2(81,1:end),80,1); SrampD2(81:190,1:end); repmat(SrampD2(190,1:end),99,1)];
    % maximum intensity 6.2718
    RD2 = backproj1( SrampD2,[],'linear',250);

    for itheta = 1:nAngle
SrampD3(:,itheta) = pp(:,itheta).*FFT.scale(HiD,sin(theta_rad(itheta)),'Lin').*...FFT.scale(HiD,cos(theta_rad(itheta)),'Lin');

% SrampD3(:,itheta) = boxfilt(SrampD3(:,itheta),60);
end;

SrampD3 = real(ifft(SrampD3));
SrampD3(len+1:end,:) = [];
SrampD3 =[repmat(SrampD3(81,1:end),80,1); SrampD3(81:190,1:end); repmat(SrampD3(190,1:end),99,1)];

% maximum intensity 6.2718
RD3 = backproj1(SrampD3,[],'linear',250);

figure(gcf+1);clf;
subplot(2,2,1);
imagesc([0:size(RA,1)]*dX,[0:size(RA,2)]*dX,interp2(abs(RA),3));%imagesc(abs([RA]));
ylabel('Y (mm)');
xlabel('X (mm)');

subplot(2,2,2);
imagesc([0:size(RD1,1)]*dX,[0:size(RD1,2)]*dX,interp2(abs(RD1),3)); %imagesc(abs([RD1]));
ylabel('Y (mm)');
xlabel('X (mm)');

subplot(2,2,3);
imagesc([0:size(RD2,1)]*dX,[0:size(RD2,2)]*dX,interp2(abs(RD2),3)); %imagesc(abs([RD2]));
ylabel('Y (mm)');
xlabel('X (mm)');

subplot(2,2,4);
imagesc([0:size(RD3,1)]*dX,[0:size(RD3,2)]*dX,interp2(abs(RD3),3)); %imagesc(abs([RD3]));
ylabel('Y (mm)');
xlabel('X (mm)');

frec1= idwt2(RA(1:end,1:end), RD1(1:end,1:end), ... RD2(1:end,1:end), RD3(1:end,1:end),loR,hiR);
figure(gcf+1);
imagesc([0:size(frec1,1)]*dX,[0:size(frec1,2)]*dX,interp2(abs(frec1),2));
formatImage(2);
title('Wavelet Based Tomography');
ylabel('Y (mm)');
xlabel('X (mm)');
end

figure(gcf+1);clf;
SegTrac=RA(80:229-35,78-4:166);
min(min(SegTrac))
max(max(SegTrac))

SegTrac1=ceil(SegTrac.*5000+110); %
Amin=max(min(SegTrac1)); %
Amax=max(max(SegTrac1)); %
figure(gcf+1);clf;
imagesc([0:size(SegTrac1,2)]*dX,[0:size(SegTrac1,1)]*dX,interp2(abs(SegTrac1),2));
size(SegTrac1)
figure(gcf+1);clf;

for i=1:Amax
ImageHis(i)=length(find(SegTrac1==i));
end
[a,b]=max(ImageHis);
sum(ImageHis);
figure(gcf+1);clf;
h1=ImageHis(1:Amx);
horz=1:Amx;
bar(horz,h1);
axis([0 Amx 0 a])
% h = findobj(gca,'Type','patch');
% set(h,'FaceColor','r','EdgeColor','w')

%(80-67)/5000=0.0026;
HoldInd=(SegTrac1<=95);%83
figure(gcf+1);clf;
imagesc(HoldInd);
axis on; grid on;

%chasing the edge
[imX,imY]=size(HoldInd);
coY=ceil(imY/2);% 45 pixel
coX=ceil(imX/2);% 75 pixel
if 1==1
% the first quadrant of the coordinates
    for Iy=1:coY-1
        for Ix=1:coX-1
            A1= HoldInd( coX-1+Ix-1,coY-1+Iy+1);% calculation start from (coX,coY)
            A2= HoldInd( coX-1+Ix, coY-1+Iy+1);
            A3= HoldInd( coX-1+Ix+1,coY-1+Iy+1);
            A4= HoldInd( coX-1+Ix-1,coY-1+Iy);
            A5= HoldInd( coX-1+Ix+1,coY-1+Iy);
            A6= HoldInd( coX-1+Ix, coY-1+Iy);
            A7= HoldInd( coX-1+Ix-1,coY-1-1+Iy);
            A8= HoldInd( coX-1+Ix, coY-1-1+Iy);
            A9= HoldInd( coX-1+Ix+1,coY-1-1+Iy);

            A=[A1,A2,A3,A4,A5,A6,A7,A8,A9];
            if length(find(A==1))<=6
                HoldInd(coX-1+Ix,coY-1+Iy)=0;
            else
                HoldInd(coX-1+Ix,coY-1+Iy)=1;
            end
        end
    end
    figure(gcf+1);clf;
    imagesc(HoldInd);grid on;
% the second quadrant of the coordinates
end

if 1==1
    for Iy=1:coY-2
        for Ix=1:coX-2
            A1= HoldInd( coX+Ix-1,coY-1+Iy+1);% calculation start from (coX,coY)
            A2= HoldInd( coX+Ix,coY-1+Iy+1);

            A=[A1,A2];
            if length(find(A==1))<=6
                HoldInd(coX+Ix,coY-1+Iy+1)=0;
            else
                HoldInd(coX+Ix,coY-1+Iy+1)=1;
            end
        end
    end
end
A3= HoldInd( coX+Ix+1,coY-Iy+1);
A4= HoldInd( coX+Ix-1,coY-Iy);
A5= HoldInd( coX+Ix+1,coY-Iy);
A6= HoldInd( coX+Ix,coY-Iy);
A7= HoldInd( coX+Ix-1,coY-Iy-1);
A8= HoldInd( coX+Ix,coY-Iy-1);
A9= HoldInd( coX+Ix+1,coY-Iy-1);

A=[A1,A2,A3,A4,A5,A6,A7,A8,A9];
if length(find(A==1))<=6
  HoldInd(coX+Ix,coY-Iy)=0;
else
  HoldInd(coX+Ix,coY-Iy)=1;
end
end
figure(gcf+1);clf;
imagesc(HoldInd);
end

if 1==1
  \% the third quadrant of the coordinates
  for Iy=1:coY-2
    for Ix=1:coX-2
      A1= HoldInd( coX-Ix-1,coY-Iy+1); \% calculation start from (coX,coY)
      A2= HoldInd( coX-Ix,coY-Iy+1);
      A3= HoldInd( coX-Ix+1,coY-Iy+1);
      A4= HoldInd( coX-Ix-1,coY-Iy);
      A5= HoldInd( coX-Ix+1,coY-Iy);
      A6= HoldInd( coX-Ix,coY-Iy);
      A7= HoldInd( coX-Ix-1,coY-1-Iy);
      A8= HoldInd( coX-Ix,coY-1-Iy);
      A9= HoldInd( coX-Ix+1,coY-1-Iy);

      A=[A1,A2,A3,A4,A5,A6,A7,A8,A9];
      if length(find(A==1))<=6
        HoldInd(coX-Ix,coY-Iy)=0;
      else
        HoldInd(coX-Ix,coY-Iy)=1;
      end
    end
end
figure(gcf+1);clf;
imagesc(HoldInd);
end

\% the fourth quadrant of the coordinates
for Iy=1:coY-2
  for Ix=1:coX-2
    A1= HoldInd( coX-Ix-1,coY+Iy+1); \% calculation start from (coX,coY)
    A2= HoldInd( coX-Ix,coY+Iy+1);
    A3= HoldInd( coX-Ix+1,coY+Iy+1);
    A4= HoldInd( coX-Ix-1,coY+Iy);
    A5= HoldInd( coX-Ix+1,coY+Iy);
    A6= HoldInd( coX-Ix,coY+Iy);
    A7= HoldInd( coX-Ix-1,coY+1-Iy);
    A8= HoldInd( coX-Ix,coY+1-Iy);
    A9= HoldInd( coX-Ix+1,coY+1-Iy);

    A=[A1,A2,A3,A4,A5,A6,A7,A8,A9];
    if length(find(A==1))<=6
      HoldInd(coX-Ix,coY+Iy)=0;
    else
      HoldInd(coX-Ix,coY+Iy)=1;
    end
  end
end
figure(gcf+1);clf;
imagesc(HoldInd);
end
Appendix E Matlab Code

A5 = HoldInd( coX-Ix+1,coY+Iy);
A6 = HoldInd( coX-Ix,coY+Iy);
A7 = HoldInd( coX-Ix-1,coY-Iy+Iy);
A8 = HoldInd( coX-Ix,coY-Iy+Iy);
A9 = HoldInd( coX-Ix+1,coY-Iy+Iy);

A=[A1,A2,A3,A4,A5,A6,A7,A8,A9];
if length(find(A==1))<=6
    HoldInd(coX-Ix,coY+Iy)=0;
else
    HoldInd(coX-Ix,coY+Iy)=1;
end
end

figure(gcf+1);clf;
imagesc(HoldInd);

for Ey=1:coY
    if length(find(HoldInd(:,coY-1+Ey)==1))<=8
        [Eya,Eyb]=find(HoldInd(:,coY-1+Ey)==1);
        HoldInd(Eya,Eyb)=0;
        HoldInd(:,coY-1+Ey+1:end)=0;
    end
end

figure(gcf+1);clf;
imagesc(HoldInd);

for Ey=1:coY
    if length(find(HoldInd(:,coY-Ey)==1))<=8
        [Eya,Eyb]=find(HoldInd(:,coY-Ey)==1);
        HoldInd(Eya,Eyb)=0;
        HoldInd(:,1:coY-Ey-1)=0;
        break
    end
end

figure(gcf+1);clf;
imagesc(HoldInd);

for Ex=1:coX-1
    if length(find(HoldInd(coX-Ex,:)==1))<=8
        [Eya,Eyb]=find(HoldInd(coX-Ex,:)==1);
        HoldInd(Eya,Eyb)=0;
        HoldInd(1:coX-Ex,:)=0;
    end
end

figure(gcf+1);clf;
imagesc(HoldInd);

for Ex=1:coX-1
    if length(find(HoldInd(coX+Ex,:)==1))<=8
        [Eya,Eyb]=find(HoldInd(coX+Ex,:)==1);
        HoldInd(Eya,Eyb)=0;
    end
end

figure(gcf+1);clf;
imagesc(HoldInd);
E.2 Code listings

```matlab
HoldInd(coX+Ex:end,:)=0;
break
end
end

figure(gcf+1);clf;
imagesc(HoldInd);
figure(gcf+1);
imagesc([0:size(HoldInd,1)]*dX,[0:size(HoldInd,2)]*dX,interp2(abs(HoldInd),2));
formatImage(2);
title('Wavelet Based Tomography');
ylabel('Y (mm)');
xlabel('X (mm)');
```
DVD containing additional data is included with the print copy held in the University of Adelaide Library.


Bibliography


Bibliography

WALNUT-D. F. (2001). An Introduction to Wavelet Analysis, 1st edn, Birkhäuser, Boston, USA.


<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR</td>
<td>auto regressive</td>
</tr>
<tr>
<td>ARMA</td>
<td>auto regressive moving average</td>
</tr>
<tr>
<td>ASOPS</td>
<td>asynchronous optical sampling</td>
</tr>
<tr>
<td>BCC</td>
<td>basal cell carcinoma</td>
</tr>
<tr>
<td>BTC</td>
<td>bound-to-continuum</td>
</tr>
<tr>
<td>CCD</td>
<td>charge-coupled device</td>
</tr>
<tr>
<td>CIL</td>
<td>complex insertion loss</td>
</tr>
<tr>
<td>CSL</td>
<td>chirped superlattice</td>
</tr>
<tr>
<td>CSR</td>
<td>coherent synchrotron radiation</td>
</tr>
<tr>
<td>CT</td>
<td>computed tomography</td>
</tr>
<tr>
<td>CV</td>
<td>cross-validation</td>
</tr>
<tr>
<td>CW</td>
<td>continuous wave</td>
</tr>
<tr>
<td>DWPTs</td>
<td>discrete wavelet packet transforms</td>
</tr>
<tr>
<td>DWT</td>
<td>discrete wavelet transform</td>
</tr>
<tr>
<td>EM</td>
<td>electromagnetic</td>
</tr>
<tr>
<td>EO</td>
<td>electrooptic</td>
</tr>
<tr>
<td>EOS</td>
<td>electrooptic sampling</td>
</tr>
<tr>
<td>FBP</td>
<td>filtered back projection</td>
</tr>
<tr>
<td>FCM</td>
<td>fuzzy c-means</td>
</tr>
<tr>
<td>FELs</td>
<td>free-electron lasers</td>
</tr>
<tr>
<td>FFTs</td>
<td>fast Fourier transforms</td>
</tr>
<tr>
<td>FIR</td>
<td>far-infrared</td>
</tr>
<tr>
<td>FIR</td>
<td>finite impulse response</td>
</tr>
<tr>
<td>FWT</td>
<td>fast wavelet transform</td>
</tr>
<tr>
<td>FWHM</td>
<td>full-width-half-maximum</td>
</tr>
<tr>
<td>GCT</td>
<td>global computed tomography</td>
</tr>
<tr>
<td>GCV</td>
<td>generalized cross-validation</td>
</tr>
<tr>
<td>GPIB</td>
<td>General Purpose Interface Bus</td>
</tr>
<tr>
<td>GVD</td>
<td>group velocity dispersion</td>
</tr>
<tr>
<td>HBO</td>
<td>human bone osteoblasts</td>
</tr>
<tr>
<td>HOS</td>
<td>human osteosarcoma</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>IDWPT</td>
<td>discrete wavelet packet transform, 173</td>
</tr>
<tr>
<td>IDWT</td>
<td>inverse discrete wavelet transform, 173</td>
</tr>
<tr>
<td>IRT</td>
<td>inverse Radon transform, 217</td>
</tr>
<tr>
<td>ITO</td>
<td>indium tin oxide, 207</td>
</tr>
<tr>
<td>IWT</td>
<td>inverse wavelet transform, 219</td>
</tr>
<tr>
<td>LDA</td>
<td>linear discriminant analysis, 71</td>
</tr>
<tr>
<td>LCT</td>
<td>local computed tomography, 253</td>
</tr>
<tr>
<td>LIA</td>
<td>lock in amplifier, 252</td>
</tr>
<tr>
<td>LOO</td>
<td>leave-one-out, 148</td>
</tr>
<tr>
<td>MA</td>
<td>moving average, 113</td>
</tr>
<tr>
<td>MQW</td>
<td>multiple-quantum-well, 250</td>
</tr>
<tr>
<td>MRI</td>
<td>magnetic resonance imaging, 4</td>
</tr>
<tr>
<td>MRA</td>
<td>multiresolution analysis, 83</td>
</tr>
<tr>
<td>MSA</td>
<td>multiscale approximation, 84</td>
</tr>
<tr>
<td>MSE</td>
<td>mean-square error, 130</td>
</tr>
<tr>
<td>NF</td>
<td>near-field, 50</td>
</tr>
<tr>
<td>NHB</td>
<td>normal human bone, 171</td>
</tr>
<tr>
<td>OAP</td>
<td>off-axis paraboloid, 48</td>
</tr>
<tr>
<td>OR</td>
<td>optical rectification, 23</td>
</tr>
<tr>
<td>PET</td>
<td>positron emission tomography, 4</td>
</tr>
<tr>
<td>PC</td>
<td>photoconductive, 12</td>
</tr>
<tr>
<td>PC</td>
<td>Pockel’s cell, 17</td>
</tr>
<tr>
<td>PCA</td>
<td>principal components analysis, 71</td>
</tr>
<tr>
<td>PCA</td>
<td>photoconductive antenna, 28</td>
</tr>
<tr>
<td>PE</td>
<td>polyethylene, 161</td>
</tr>
<tr>
<td>PEV</td>
<td>prediction error variance, 114</td>
</tr>
<tr>
<td>poly-A</td>
<td>polyadenylic acid, 557</td>
</tr>
<tr>
<td>poly-C</td>
<td>polycytidylic acid, 55</td>
</tr>
<tr>
<td>QCL</td>
<td>quantum cascade laser, 5</td>
</tr>
<tr>
<td>QWP</td>
<td>quarter-wave plate, 28</td>
</tr>
<tr>
<td>RCS</td>
<td>radar cross-section, 44</td>
</tr>
<tr>
<td>RNA</td>
<td>ribonucleic acid, 201</td>
</tr>
<tr>
<td>ROA</td>
<td>region of artifacts, 234</td>
</tr>
<tr>
<td>ROC</td>
<td>receiver operating characteristic, 147</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>ROE</td>
<td>region of exposure, 232</td>
</tr>
<tr>
<td>ROI</td>
<td>region of interest, 9, 225</td>
</tr>
<tr>
<td>RP</td>
<td>resonant-phonon, 19</td>
</tr>
<tr>
<td>SCT</td>
<td>spiral computed tomography, 285</td>
</tr>
<tr>
<td>SL</td>
<td>superlattice, 18</td>
</tr>
<tr>
<td>S/N</td>
<td>signal-to-noise ratio, 36</td>
</tr>
<tr>
<td>SNR</td>
<td>signal-to-noise ratio, 250</td>
</tr>
<tr>
<td>SNIM</td>
<td>scanning near-field imaging, 52</td>
</tr>
<tr>
<td>SURE</td>
<td>Stein’s unbiased risk estimate, 105</td>
</tr>
<tr>
<td>SVD</td>
<td>singular value decomposition, 121</td>
</tr>
<tr>
<td>SVM</td>
<td>support vector machine, 141</td>
</tr>
<tr>
<td>TDS</td>
<td>time domain spectrometry, 12</td>
</tr>
<tr>
<td>TE</td>
<td>transverse-electric, 25</td>
</tr>
<tr>
<td>TFP</td>
<td>thin-film polarizing, 17</td>
</tr>
<tr>
<td>THz</td>
<td>terahertz, 3</td>
</tr>
<tr>
<td>T-ray</td>
<td>THz, 3</td>
</tr>
<tr>
<td>TM</td>
<td>transverse-magnetic, 25</td>
</tr>
<tr>
<td>TPI</td>
<td>terahertz pulsed imaging, 13</td>
</tr>
<tr>
<td>TPO</td>
<td>THz-wave parametric oscillator, 71</td>
</tr>
<tr>
<td>VC</td>
<td>Vapnik-Chervonenkis, 141</td>
</tr>
<tr>
<td>WART</td>
<td>wide aperture reflection tomography, 43</td>
</tr>
<tr>
<td>WT</td>
<td>wavelet transform, 61</td>
</tr>
<tr>
<td>ZnTe</td>
<td>Zinc Telluride, 22</td>
</tr>
</tbody>
</table>
Index

2-channel filter bank, 93

absorption coefficient, 61, 199, 217
approximate sub-images, 218

basal cell carcinoma, 63, 281
basic Prony method, 114, 178
best wavelet packet tree, 130, 135
binary classifiers, 141
(bi)orthogonal wavelets, 83, 172
BioSplines2.2, 230
Bhattacharya divergence, 111
bound-to-continuum transitions, 14, 251

Canny edge detector, 221
complex refractive index, 37, 211
cross-sectional image, 233
data preprocessing, 79, 279
Daubechies 4 wavelet, 220
deterministic-stochastic models, 123
dichotomy, 144
discrete wavelet transform, 7, 83, 106, 218
Durbin’s method, 114, 178
ductal carcinoma, 65
detection, 216
eight neighboring pixels, 260
Euclidean distance classifier, 68
Euclidean discrimination metric, 140, 166
extrapolation, 190, 233

fast discrete wavelet transform, 83
feature vectors, 115, 156, 188
feature extraction, 7, 68, 109
Fourier spectrum, 191
frequency orientation components, 186
function spaces, 84
fuzzy c-means thresholding, 256

Gaussian noise, 160, 222
Grey-level intensity, 257

Haar transform, 111
Hankel matrices, 119, 160
highpass filter, 86
‘heuristic’ SURE, 104, 153
holdout method, 147

image segmentation, 83
image enhancement, 84, 287
information cost function, 101
ISODATA, 69, 111

k-fold cross-validation, 147, 188
k-means, 69, 198
Karhunen-Loève transform, 111
leave-one-out cross-validation, 148
least-squares procedure, 121
local tomography, 8, 213, 251
lowpass filter, 85
Mahalanobis distance, 81, 138, 176
mandelic acid, 151
Mallat’s pyramid algorithm, 82
melanoma, 61
Mertz apodization, 109, 200, 281
molecular beam epitaxy, 14, 251
morphological operations, 259
multiresolution theory, 83
multispectral classification, 69
multispectral clustering, 68

N4SID, 160
oblique projection, 7, 120, 289
one-against-one decomposition, 144
optical image, 217
Otsu’s threshold, 221
osteosarcoma cells, 8, 170, 281
overfitting, 71, 149

Pairwise classification, 144, 183
parzen estimator, 111
pattern recognition, 1, 75, 111, 185
perfect reconstruction filter banks, 102
per-pixel classification, 257
photomicrograph, 64

quantum cascade lasers, 5, 14, 250
quadrature mirror filtering, 83
quadratic programming problem, 134

Radon transform, 6, 205
ratioing procedure, 168, 200
Rayleigh scattering, 1, 62, 280
reconstruction filters, 95
region-based segmentation, 259
scaling filter, 85, 234
segmentation-by-fusion, 216
skin cancer, 63, 170
signal compression, 84, 112
Sigma-Aldrich, 184
sinogram, 205, 257
squared error variances, 175
state-space model, 118
subband system identification, 125, 279
subspace system identification, 7, 118, 289
system identification, 7, 68, 118, 151
support vector machines, 68, 111, 182, 281
time-frequency localisation, 84, 101, 287
terahertz pulsed image, 63
THz-transient spectrometer, 160
Toeplitz matrix, 160
Toeplitz covariance matrix, 94
traditional back projection, 225
wavelet and scaling ramp filters, 225, 265
wavelet filter, 6, 86, 226
Wavelet-packet identification, 109, 164, 288
wavelet preprocessing, 153
wavelet scale correlation, 213, 221
white Gaussian process, 113
Wiener-Hopf equation, 78
Xiaoxia (Sunny) Yin was born in Dalian, China. She received the BEng Degree in Industrial Electronics from Dalian University, Liaoning, P. R. China, and is currently working toward the PhD degree on three dimensional terahertz computed tomography under Derek Abbott and Brian Ng. Her research interests include multiresolution analysis, segmentation, image reconstruction & classification, and their applications to T-ray imaging. In 2007, she was a visiting scholar at the University of Reading, UK, under Dr. Silas Hadjiloucas and at the University of Cambridge, UK, under Prof. Lynn F. Gladden as well as at Rensselaer Polytechnic Institute in Troy, NY, USA, under Prof. X.-C. Zhang. She has authored and co-authored more than 18 peer-reviewed publications.

Ms Yin received a series of awards, including the PhD Scholarship of The University of Adelaide (2005), WavE Travel Fellowship from Switzerland (2006), The International Association for Pattern Recognition (IAPR) travel stipend from Hong Kong (2006), Research Abroad Scholarship from The University of Adelaide (2006), the Roger Pysden Memorial Fellowship from Australia Business, ABL State Chamber (2006), a student scholarship from the 2007 IEEE International Symposium on Industrial Electronics in Vigo, Spain (2007), a Mutual Community Postgraduate Travel Grant from the University of Adelaide (2007), an Overseas Travel Fellowship from The Australian Research Council Nanotechnology Network (ARCNN) (2007), a D. R. Stranks Traveling Fellowship from the University of Adelaide (2007), and an Australian Postdoctoral (APD) Fellowship from the Australian Research Council (2008).
Scientific Genealogy

My scientific genealogy via my primary supervisor, Derek Abbott

<table>
<thead>
<tr>
<th>Year</th>
<th>Degree</th>
<th>Institution</th>
<th>Supervisor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1774</td>
<td>MA</td>
<td>University of Cambridge</td>
<td>John Cranke</td>
</tr>
<tr>
<td>1782</td>
<td>MA</td>
<td>University of Cambridge</td>
<td>Thomas Jones</td>
</tr>
<tr>
<td>1811</td>
<td>MA</td>
<td>University of Cambridge</td>
<td>Adam Sedgwick</td>
</tr>
<tr>
<td>1830</td>
<td>MA</td>
<td>University of Cambridge</td>
<td>William Hopkins</td>
</tr>
<tr>
<td>1857</td>
<td>MA</td>
<td>University of Cambridge</td>
<td>Edward John Routh</td>
</tr>
<tr>
<td>1868</td>
<td>MA</td>
<td>University of Cambridge</td>
<td>John William Strutt (Lord Rayleigh)</td>
</tr>
<tr>
<td>1883</td>
<td>MA</td>
<td>University of Cambridge</td>
<td>Joseph John Thomson</td>
</tr>
<tr>
<td>1903</td>
<td>MA</td>
<td>University of Cambridge</td>
<td>John Sealy Townsend</td>
</tr>
<tr>
<td>1923</td>
<td>DPhil</td>
<td>University of Oxford</td>
<td>Victor Albert Bailey</td>
</tr>
<tr>
<td>1948</td>
<td>MSc</td>
<td>University of Sydney</td>
<td>Ronald Ernest Aitchison</td>
</tr>
<tr>
<td>1964</td>
<td>PhD</td>
<td>University of Sydney</td>
<td>Peter Harold Cole</td>
</tr>
<tr>
<td>1980</td>
<td>PhD</td>
<td>University of Adelaide</td>
<td>Kamran Eshraghian</td>
</tr>
<tr>
<td>1995</td>
<td>PhD</td>
<td>University of Adelaide</td>
<td>Derek Abbott</td>
</tr>
<tr>
<td>2008</td>
<td>PhD submitted</td>
<td>University of Adelaide</td>
<td>Xiaoxia (Sunny) Yin</td>
</tr>
</tbody>
</table>

My scientific genealogy via my co-supervisor, Brian Wai-Him Ng

<table>
<thead>
<tr>
<th>Year</th>
<th>Degree</th>
<th>Institution</th>
<th>Supervisor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1865</td>
<td>PhD</td>
<td>University of Jena</td>
<td>Johannes Conrad</td>
</tr>
<tr>
<td>1875</td>
<td>PhD</td>
<td>University of Halle</td>
<td>Hermann Paasche</td>
</tr>
<tr>
<td>1892</td>
<td>PhD</td>
<td>University of Marburg</td>
<td>Karl Eugen Guthe</td>
</tr>
<tr>
<td>1912</td>
<td>PhD</td>
<td>University of Michigan</td>
<td>Neal Hooker Williams</td>
</tr>
<tr>
<td>1928</td>
<td>PhD</td>
<td>University of Michigan</td>
<td>Walter Scott Huxford</td>
</tr>
<tr>
<td>1941</td>
<td>MS</td>
<td>Northwestern University</td>
<td>Richard Ward Jones</td>
</tr>
<tr>
<td>1964</td>
<td>PhD</td>
<td>Northwestern University</td>
<td>Robert Bartholomew Pinter</td>
</tr>
<tr>
<td>1991</td>
<td>PhD</td>
<td>University of Washington</td>
<td>Abdesselam (Salim) Bouzerdoum</td>
</tr>
<tr>
<td>2003</td>
<td>PhD</td>
<td>University of Adelaide</td>
<td>Brian Wai-Him Ng</td>
</tr>
<tr>
<td>2008</td>
<td>PhD submitted</td>
<td>University of Adelaide</td>
<td>Xiaoxia (Sunny) Yin</td>
</tr>
</tbody>
</table>