ABSTRACT

A New Modality for Microwave Tomographic Imaging: Transit Time Tomography

Matthew Trumbo, M.S.E.C.E.

Dr. Robert Marks Ph.D.

A new method for tomographic imaging using time delay of microwaves as they pass through objects of varying dielectric is presented. To obtain this delay measurement, recording of transmitted waveforms is performed when broadband microwave signals are sent through objects. The experimental setup consists of a circular table with vertically polarized bi-conic horns as transmitting and receiving antennas, with a Vector Network Analyzer as the excitation and measurement device. By obtaining multiple point tomographic projections at multiple angles, a reconstruction of the classic X-ray Radon transform is possible. Results obtained show accuracy and resolution limits to be within acceptable ranges and offer many new, less expensive possibilities for tomographic imaging.

A New Modality for Microwave Tomographic Imaging: Transit Time Tomography

by

Matthew Trumbo, B.S.E.C.E.

Approved by the Department of Electrical and Computer Engineering

James B. Farison

James B. Farison, Ph.D., Chairperson

Submitted to the Graduate Faculty of
Baylor University in Partial Fulfillment of the
Requirements for the Degree
of
Master of Science in Electrical and Computer Engineering

Approved by the Thesis committee:

Robert J. Marks III, Ph.D., Chairperson

B. Randall Jean, Ph.D.

Byron Newberry, Ph.D.

Accepted by the Graduate School
May 2006

J. Larry Lyon, Ph.D., Dean

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

I would like to thank my parents and family, for their infinite love, patience, and support. I love you all, and I would not be here without the gift of such an incredible family. Thanks to Dr. Marks and Dr. Jean, for teaching me the joy of research and how to love God with your work. You're incredible examples to follow. Unending thanks and love to Catherine for her unconditional love and support.

Most of all, thanks to Jesus for being with me every step of the way.

CHAPTER ONE

Introduction

In recent years, tomographic imaging has formed a crucial area of advancement for modern society. By non-intrusively providing imagery of object interiors, many fields have been able to provide increased efficiency and more accuracy to medical diagnosis, construction, and manufacturing. However, some methods for tomographic imaging are cost prohibitive for frequent use. Others do not provide the resolution necessary for a given application. For each of these reasons, constant research goes into finding new ways to perform tomographic imaging in hopes of developing a better solution for classes of imaging problems.

Many microwave tomographic modalities exist already, including microwave backscatter techniques, in which reflections and scattering effects are used to obtain images of objects [1]. Still other techniques focus exclusively on the energy transmitted through an object to perform image reconstruction [2]. All of these modalities have provided good resolution, but most require use of expensive saline tanks to slow the signals down, artificially increase resolution, and eliminate scattering effects. Most of them do not use broadband transmission and receiving techniques as well, choosing instead to go with single frequency systems [1].

In this paper, we outline a modality for microwave tomography; one that uses the transit time of microwaves as they pass through objects of varying permittivity. We demonstrate this new modality and its benefits without use of a saline tank, and using simple, easy to construct components. It is hoped that the ease of use and ease of

construction for this system will set it apart from its contemporaries, and prove useful to the medical and industrial imaging sectors in the future.

CHAPTER TWO

Principles of Tomographic Imaging

To fully understand the requirements of a tomographic system, one must be familiar with the basics of computerized tomography. This chapter will outline the mathematical derivations of the tomographic algorithms used in this project

Consider the object represented below in Figure 1, with some property non-

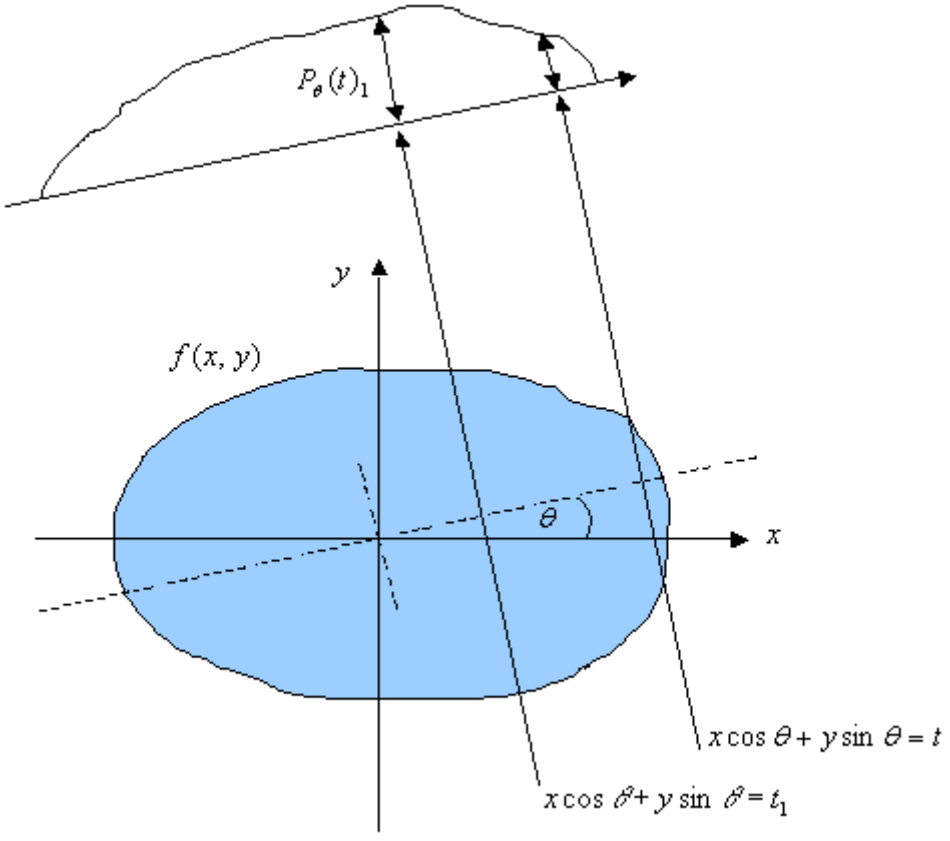

Figure 1: Line Integrals

invasively measured parameter *f*(*x,y*). Let *t* represent the distance along the rotated x-axis, and θ the angle of rotation. Equation

$$t = x \cdot \cos(\theta) + y \cdot \sin(\theta)$$

defines each ray trace through *f(x,y)*. Using this relationship, we may then define each line integral as the equation

$$P_\theta(t) = \int_t f(x, y)ds\,.$$

Converting the equation to delta function, we then have the equation

$$P_\theta(t) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x, y)\delta(x\cos(\theta) + y\sin(\theta) - t)dxdy\,,$$

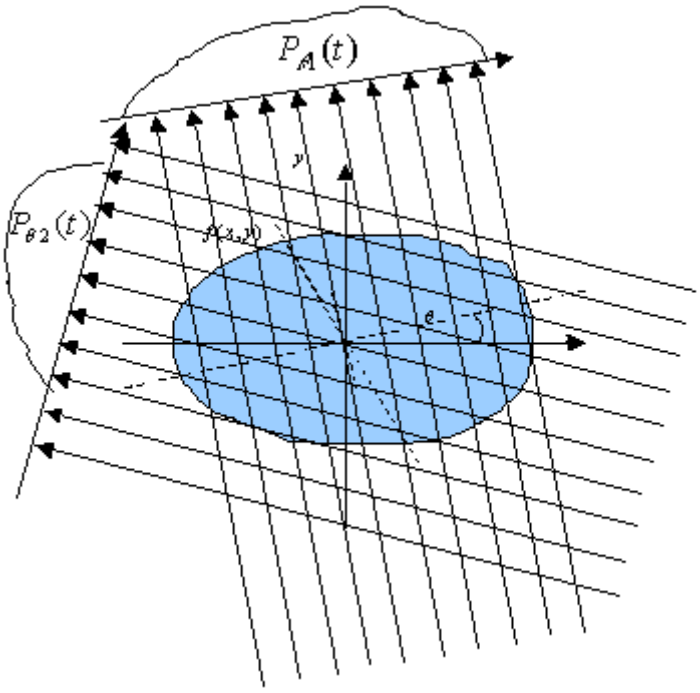

Figure 2: Parallel Projections

which is the formal definition of the Radon transform. This simply states that a projection of an object is formed by combining a set of line integrals as shown in Figure 2.

These projections are the simplest form of tomographic projections, known as the parallel projections of the object [3]. In parallel tomographic systems, these projections are obtained by moving a source and a detector to obtain multiple ray traces of an object. This method was originally demonstrated by Hounsfield in his Nobel Prize winning X-ray Computerized Tomography (CT) scanner in 1972 [4]. Today, a more popular method of obtaining tomographic projections is the fan-beam projection, in which each line integral originates from a single source for each angle θ. Fan-beam systems were developed later in 1977 by Wang, and offer reduced projection acquisition times, due to the source position being fixed [5]. As shown later in this paper, the Transit Time tomography experimental setup uses simple parallel projections as well, for both simplicity of derivation and production, so all derivations to follow will assume these types of projections.

Fourier Slice Theorem

To begin, we define the two-dimensional Fourier transform of the object as

$$F(u,v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,y) e^{-j2\pi(ux+vy)} dxdy .$$

and the Fourier transform of a projection at angle θ as

$$S_\theta(w) = \int_{-\infty}^{\infty} P_{\theta(t)} e^{-j2\pi wt} dt .$$

We desire to relate the Fourier transform of each projection to the complete two-dimensional Fourier transform of our object. If this is possible, it would follow that a reconstruction of the two-dimensional Fourier transform is possible from the Fourier transforms of the projections of the object. To illustrate this relationship between the transforms, consider the slice of the two-dimensional Fourier transform along $v = 0$,

$$F(u,0) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,y) e^{-j2\pi ux} dxdy .$$

With the phase no longer dependent on y, the integral can be separated quite easily as

$$F(u,0) = \int_{-\infty}^{\infty} [\int_{-\infty}^{\infty} f(x,y)dy] e^{-j2\pi ux} dx .$$

If we then notice that the center integral is merely the definition for the parallel projection at $\theta = 0$, the equation then becomes

$$F(u,0) = \int_{-\infty}^{\infty} P_{\theta=0}(x) e^{-j2\pi ux} dx .$$

This is easily recognizable as the above formula for the one-dimensional Fourier transform of the projection at $\theta = 0$, and we discover that the following relationship between the vertical projection and the two-dimensional Fourier transform of the objects exists

$$F(u,0) = S_{\theta=0}(u) .$$

Since this result is independent of the orientation between the object and the coordinate system, this result holds for any $P_{\Theta}(t)$. This result is known as the Fourier Slice theorem, and is stated as follows:

> The Fourier Transform of a parallel projection of an image $f(x,y)$ taken at an angle θ gives a slice of the two-dimensional Fourier transform $F(u,v)$, subtending an angle θ with the u-axis.

Therefore, to form a two-dimensional Fourier transform of an object, each slice can be found from its corresponding projection. However, to reconstruct a continuous two-dimensional Fourier transform as shown here, projections must be found at continuous angles [3].

Since projections cannot be taken at infinite angles in practice, the reconstruction of the Fourier Transform of the object is quite difficult from the radial distribution of known points provided by the Fourier Slice Theorem. To use numerical Fast Fourier Transform (FFT) techniques, the radial distribution can be interpolated to a square grid of values. However, this interpolation induces more error as one gets farther from the center of the Fourier transform, as the density of the points becomes smaller. The effect of this density reduction materializes in the form of image degradation of high frequency components in the reconstructed image [3].

Filtered Backprojection

As stated above, an algorithm is necessary to interpolate the limited number of projections onto a square grid. While many possible ways exist to perform this operation, by far the most widely used method for reconstruction of parallel projections is the filtered backprojection technique.

Consider the radial spoke approximation of the Fourier transform shown below in Figure 3. As can be seen, the Fourier transforms of each projection S_Θ form the values of

the two-dimensional Fourier transform along multiple slices centered at the origin of the *u*-*v* plane.

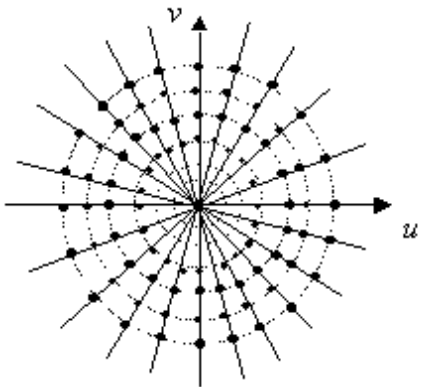

Figure 4: Radial approximation of Fourier Transform

We desire to interpolate each slice, over two pie shaped wedges shown below on the left of Figure 4. The area of each wedge is described by the equations $\frac{2\pi|w|}{K}$, where K is the number of projections taken in a 180 degree arc. This interpolation allows us to create an approximation of the two-dimensional Fourier transform that is evenly interpolated from the radial slices known from the projections [3].

To accomplish this interpolation, we multiply each point of the projection by a weighting factor equal to the area of the interpolation wedge, $\frac{2\pi|w|}{K}$, thus increasing the contribution of each projection to the two-dimensional transform reconstruction as we move to the less sampled higher frequency components, shown on the right. This operation constitutes an approximation of the ideal interpolation, and is analogous to a

filtering operation in practice, hence the “filtered” part of the filtered backprojection algorithm [3].

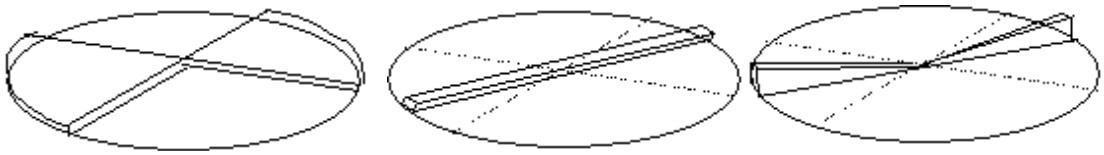

Figure 4: Filtered Backprojection

After weighting each slice, each slice undergoes an inverse two-dimensional Fourier transform, and is summed across the image domain to obtain our reconstruction. It is important to note this summation in the image domain is possible due to linearity of the Fourier transform, and is equally possible in the Fourier domain. The image domain is chosen for computational simplicity. The techniques derived in this chapter are used as the basis for all of the image reconstruction done in the subsequent chapter.

Sinograms

When viewing tomographic images and the projections acquired by a given Radon transform approximation, the projections are often lined up and placed side by side for viewing purposes. These images are called sinograms, so called because of the sinusoidal pattern a point would make on one. These images are useful for evaluation of the results of a Radon transform approximation, and will be used frequently in the subsequent chapter.

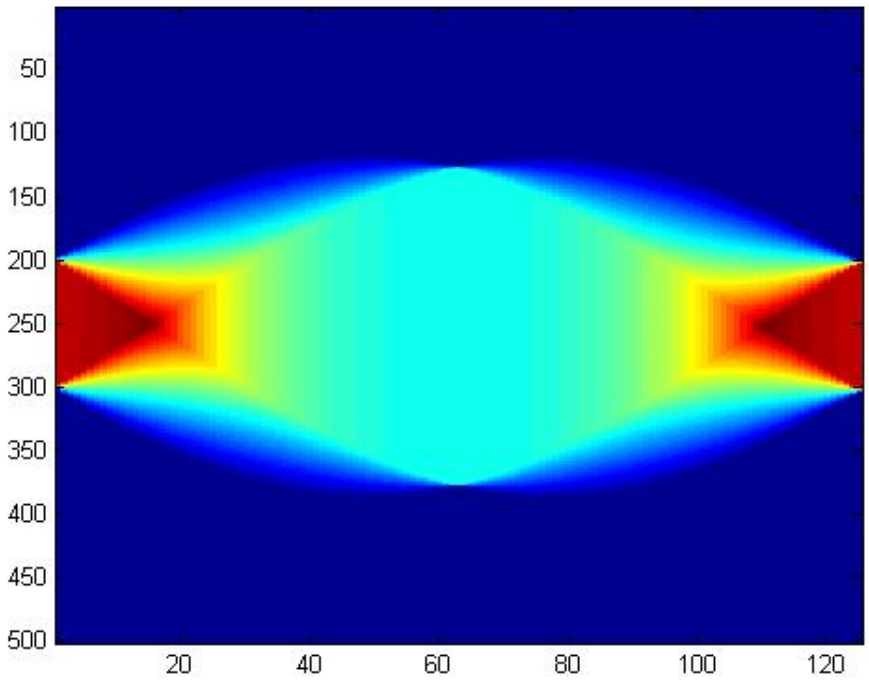

Figure 5: Example Sinogram

CHAPTER THREE

Transit Time Tomography

From the previous chapter, it is known that generating an approximation of a Radon transform requires some way of creating a line integral. This chapter outlines our technique for approximation of the Radon transform using time delay of microwaves and the specifications of our experimental setup. It also covers the challenges and problems faced as the tomographic system was constructed.

Consider an object with a two dimensional permittivity distribution $\varepsilon_r(x, y)$. According to Maxwell's equations, the real part of the permittivity ε_r or, as it is more commonly referred to, the dielectric constant of our object determines the velocity of propagation at point (x,y) as $v(x,y)$ in Figure 6. Ignore the effects of scattering and reflection for this derivation.

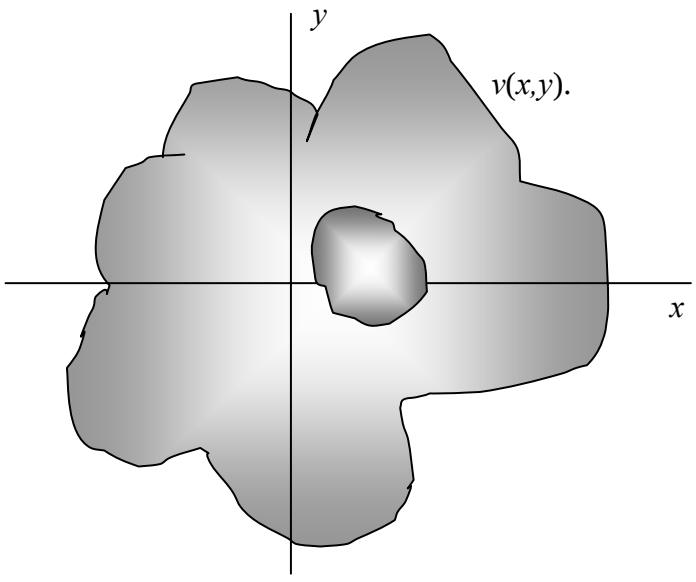

Figure 6: Velocity Distribution

Consider motion of a microwave pulse in 1-D at a positional dependent velocity $u(x)$.

Then

$$\frac{dx}{dt} = u(x)$$

or

$$dt = \frac{dx}{u(x)}.$$

In a nonmagnetic medium with permittivity ε, the E&M speed of propagation is given by

$$u = \frac{1}{\sqrt{\mu_0 \varepsilon}}.$$

Substituting from (3) into (1),

$$dt = \sqrt{\mu_0 \varepsilon}\, dx$$

where ε_r is the relative permittivity. Since $\sqrt{\mu_0 \varepsilon} = \sqrt{\varepsilon_r}$, we may now freely substitute the $\sqrt{\varepsilon_r(x, y)}$ distribution for the $v(x,y)$ distribution.

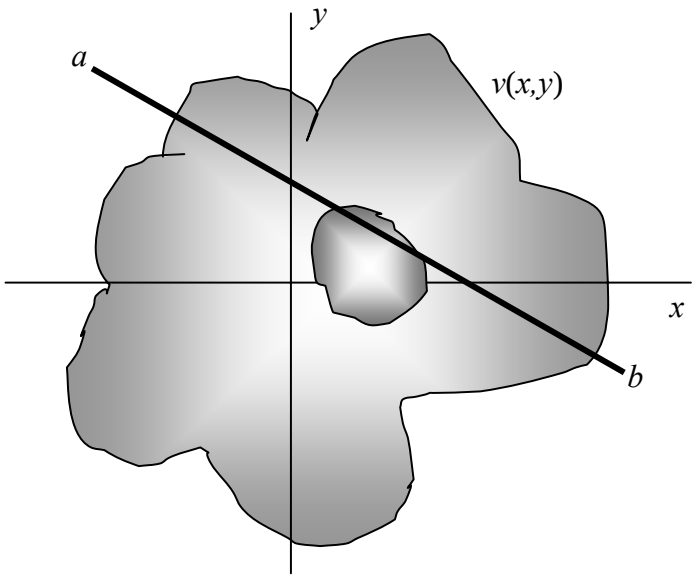

Figure 7: Ray Trace Through the Distribution

The time for a ray to propagate from *a* to *b* is simply

$$t_{a\to b} = \int dt = \int_a^b \sqrt{\varepsilon_r(x,y)}\ \mathrm{d}\ell \,.$$

This result is a point tomographic projection, or line integral as shown in Chapter 2. The concept for Transit Time Tomography then calls for transmitting microwave energy through an object with an unknown dielectric distribution and measuring the transit time of that energy through the object. Using the transit time as our measurement of the line integral for the Radon transform, we could reconstruct an image of the $\sqrt{\varepsilon_r(x,y)}$ distribution using a standard inverse Radon transform with filtered backprojection. All that is required is a practical way of measuring the transit time, or delay, of microwaves as they pass through the object being imaged.

Transit Time tomography requires several assumptions. Chief among them is the assumption that microwaves will propagate through an object in a straight line. Obviously, in practice microwaves do not behave in this manner. When striking a dielectric interface, Snell's law must be obeyed for transmitted microwaves, and large reflections can occur when going from a medium of low dielectric constant to a medium of high dielectric constant. For the initial test results here, we compensate for the issue of refraction is several ways.

First, vertically polarized transmitters and receivers limit the effects of reflected and scattered transmission paths. Direct ray traces through an object will retain the vertical polarization and thus will offer the most contribution to the resulting received

waveform. In addition, isolating the object being imaged using raised antennas and electromagnetically invisible “boosters” allows for signal processing techniques to eliminate the alternate path parts of received signals with a high degree of effectiveness.

Tomography Table Construction

The table was designed to implement the Radon transform parallel projection process as closely as possible. A transmitter and receiver pair of antennas are mounted on a single bracket, which moves back and forth on a Velmex Bislide 3MN10, controlled by a stepper motor. A Vexta CFK II 5-phase stepper motor rotates the object platform.

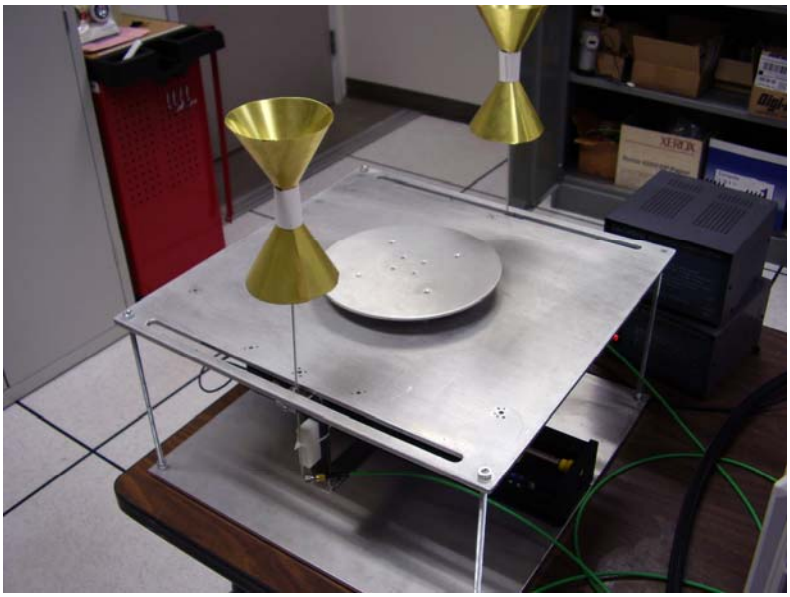

Figure 8: Tomography Table

Control for the data collection and platform/antenna motion is provided by a Visual Basic program running on a laptop computer, shown in Figure 7. Code for this program is provided in the Appendix. This program has two functions, to control the acquisition of microwave data from the antennas, and to tell the antennas/table to move in

the appropriate manner after data has been collected. Direct Motor control from the laptop is not possible, so an RS-232 interface controls a Basic Stamp 2E microcontroller, mounted on BS2E Board of Education development board for access to pins and power. This microcontroller is responsible for sending the appropriate pulses to the Velmex Bislide and the Vexta rotational motor.

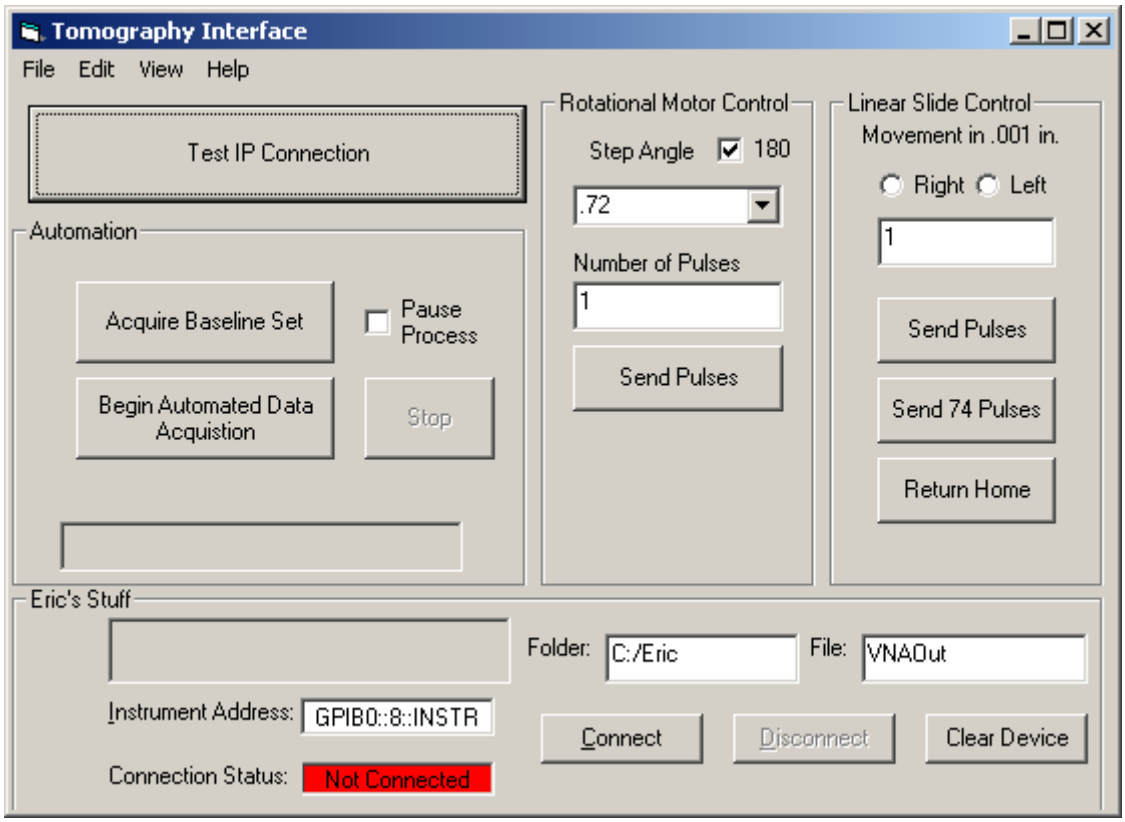

Figure 9: VB Control Program

Microwave data is taken and recorded by the Hewlett-Packard VNA 8720ET. It generates chirp microwave pulses ranging from 1GHZ to 20GHZ in frequency, and records the response. The VNA interfaces with the Visual Basic control program via a GPIB interface.

Figure 10: Bi-conic Antenna

The antennas used for the transmitter and receiver are simple bi-conic circular horn antennas. They were selected for their uniform frequency transmission pattern, vertical polarization, and ease of construction.

Typical operation of the Transit Time Tomography system follows a simple algorithm. To start, the user initiates automated data acquisition by clicking the appropriate button in the control program. The Visual Basic program requests to the BS2E that the antennas be reset to their starting positions. The BS2E sends a constant stream of pulses to the linear slide motor, until a limit switch is activated on the slide. At this point, response of a sweep is recorded and saved in a Microsoft Excel file on the laptop. Next, the Visual Basic program requests an antenna position increment, and the BS2E sends the appropriate number of pulses to increment the antennas $1/500^{th}$ of a full sweep. Measurements are then repeated 500 times for the entire parallel projection. After acquisition of this full projection, the Visual Basic program requests a rotational

increment of the object to acquire the next projection. The BS2E sends the appropriate number of pulses to increment the rotational position 18/25th of a degree. At his point, 500 responses are measured to form another projection, and the process is repeated for 250 different projections spanning 180 degrees. Waveforms are saved for each of the 500 points in a tomographic projection at each of 250 rotations of an object. 125,000 waveforms are acquired each image acquisition, each with 394 samples present. A typical time for an image acquisition is 40 hours.

Many changes to the construction of the tomography table have been made over the course of this project. Most were trivial changes to improve reliability of the table, but one did have a dramatic impact on the quality of image acquisition. As mentioned previously, in the original design the antennas did not consist of a full bi-conic. Instead, a single cone was placed directly touching the plane of the top table. By using the electromagnetic mirror effect provided by the metal conductor, we emulated the effect of true bi-conic antennas.

Unfortunately, keeping the antennas that close to the table had an unforeseen side effect. When images were acquired, they proved to have a large contribution from the rotational assembly itself, and signal processing techniques struggled to deal with reflections off the table. The solution proved to be new, true bi-conic antennas that were constructed and mounted 25cm above the surface of the table and the rotational assembly. While the table now requires use of an electromagnetically invisible object booster (in our case a simple cardboard box) to keep our objects in the plane of the antennas, the signals no longer exhibit contribution from the rotational assembly and reflections are far easier to eliminate in signal processing.

Transit Time Acquisition

The primary difficulty in implementing a Transit Time tomography system is the acquisition of delay from the received waveform. Consider the figure 11.

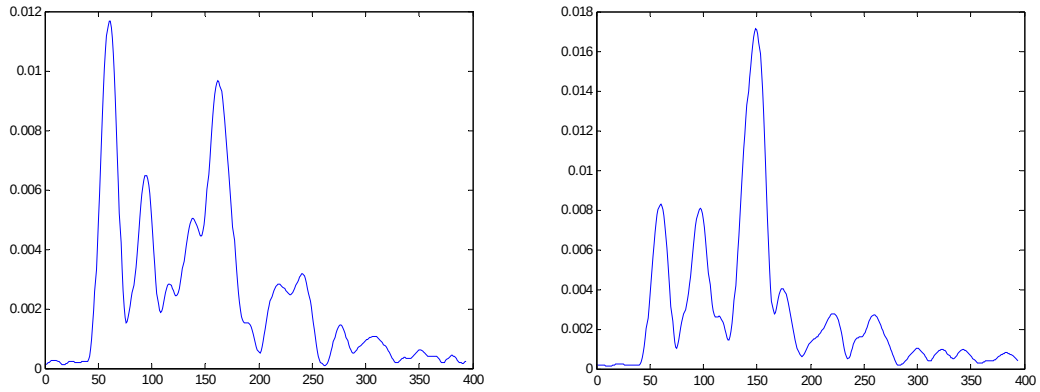

Figure 11: Example waveforms

On the left of Figure 11 is a received waveform with no object on the table; on the right is through a circular jar of cooking oil. The x-axis denotes samples and the y-axis voltage. While there are clear differences between the two waveforms, reliable delay acquisition is less apparent. Signal processing techniques to interpret these waveforms is necessary.

Initially, the algorithm to obtain the delay was to simply track the peaks of the received signal as they moved and determine which of them corresponded to the straight ray path though the object being imaged. This process proved problematic for several reasons. Consider the figure 12. This is a compilation of the received waveforms for a single projection of the same jar of oil, with the amplitude of the signals represented in color, the position of the antennas along the x-axis, and the time axis along the y-axis.

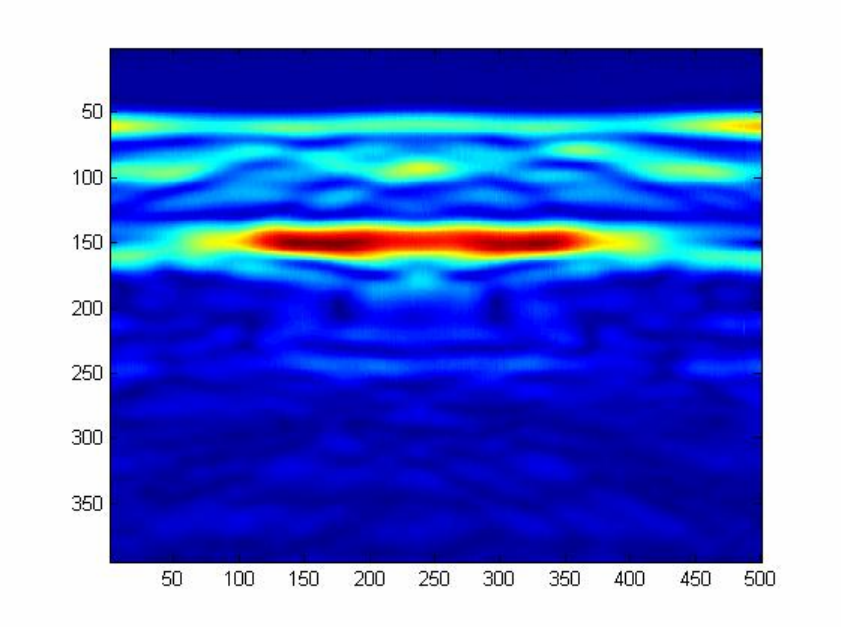

Figure 12: Example compilation of waveforms

Combined with the initial problems related to having the antennas directly on the table, delay acquisition proved essentially impossible using peak tracking. While it might seem intuitive that peaks might shift in time according to the delay generated by higher dielectric constant, the actual result proved much different. Usually most of the peaks remained essentially stationary from one waveform to the next, and energy simply shifted from one peak to the next. Attempting to track the maximums of these peaks merely resulted in essentially flat projections with abrupt jumps in measured delay as the max shifted, shown below.

After repeated attempts at solutions based on peak tracking, a new idea emerged as the best way of obtaining delay. Using calculated information on where the maximum delay might occur for the range of dielectric constant in the objects being imaged, large amounts of each waveform were eliminated from the calculation of delay. Operating just

in this region of interest (ROI), a simple center of mass calculation is performed on the energy contained in the ROI. With t as the delay and $y(n)$ as the sample in the waveform,

$$t = \frac{\sum_{n=ROI} n \cdot y(n)}{\sum_{n=ROI} n}$$

The value of this delay is stored for the image reconstruction in an Inverse Radon transform, as noted above.

Results

For our test suite, two classes of objects were considered, circles and rectangles. Each of these objects presents different challenges for a microwave tomography system, for system geometries affect reflection and scattering dramatically. Using the center of mass delay calculations and ROIs based on reasonable delay values for our objects – 20 samples, or one nanosecond of real time of delay generated by the objects of higher dielectric – the following images were obtained.

First, circles were imaged. Small jars of cooking oil (about 6cm in diameter), with a dielectric constant of approximately 2.5 in the frequency bands used were placed in an asymmetric pattern on the table and moved incrementally closer, providing results on the viability of imaging circles, multiple objects, and resolution of objects close together. The object's actual position in reality is indicated by the black outline.

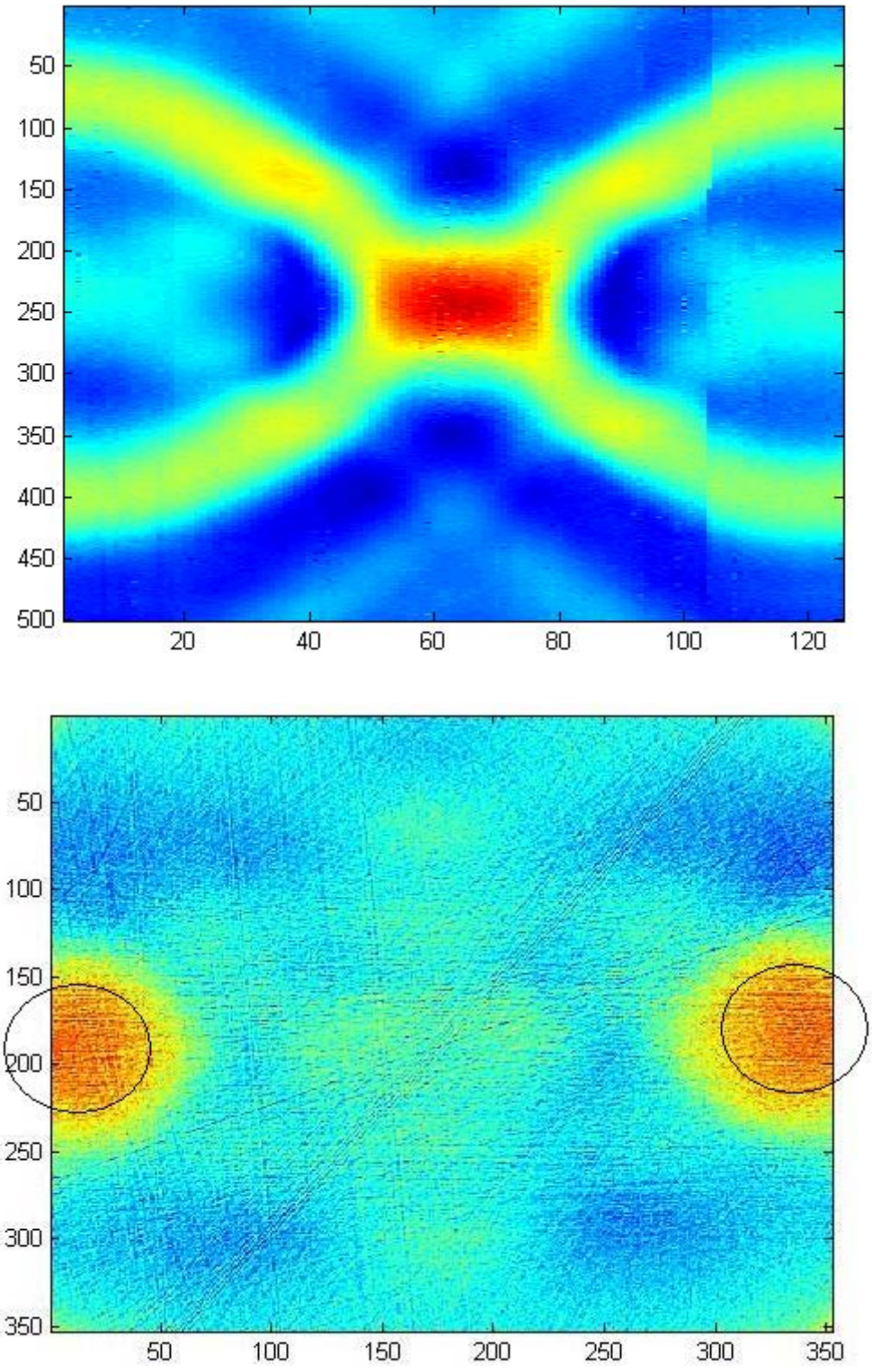

Figure 13: Image and sinogram of two jars of oil, centers 6” apart

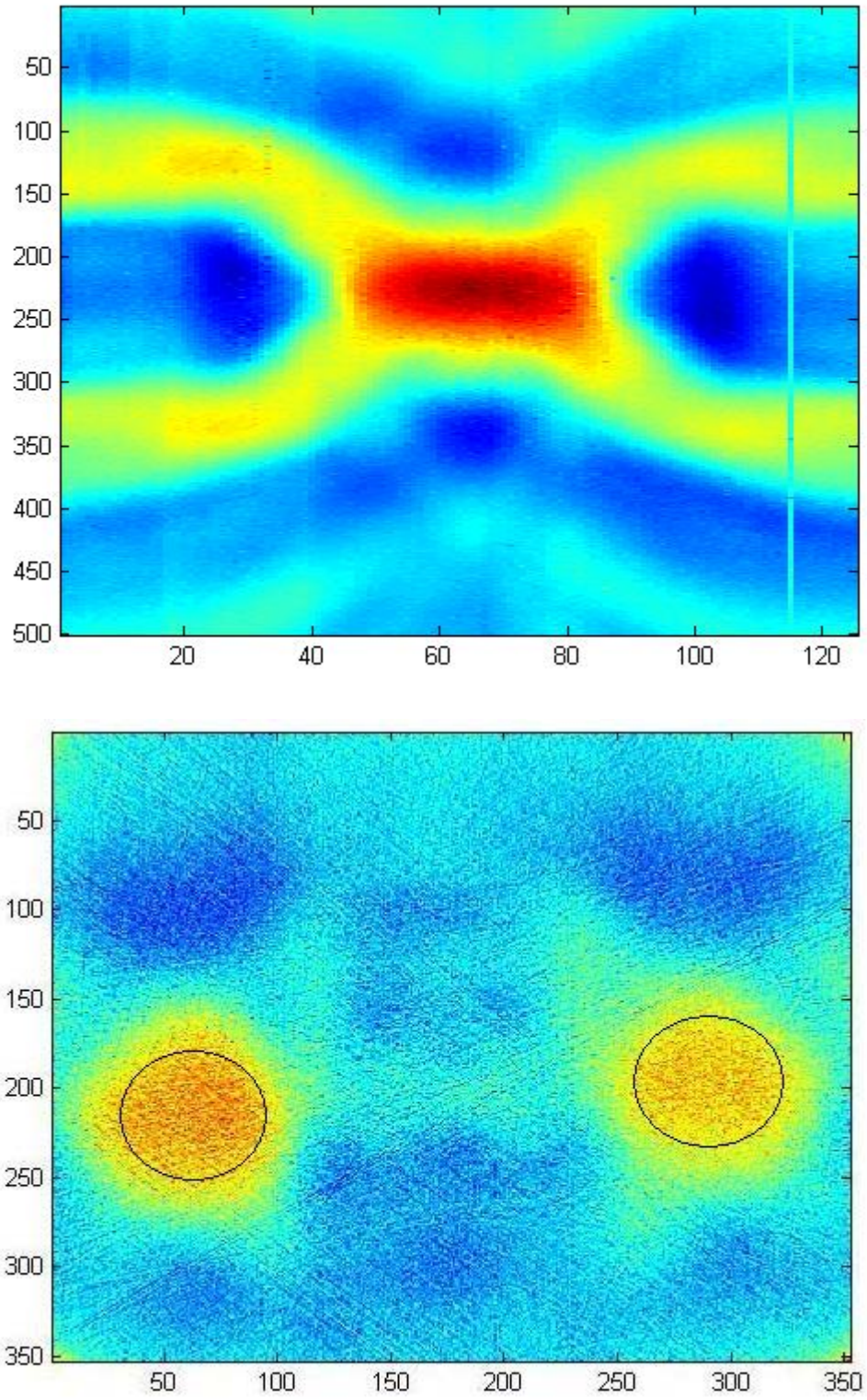

Figure 14: Image and sinogram of two jars of oil, centers 4” apart

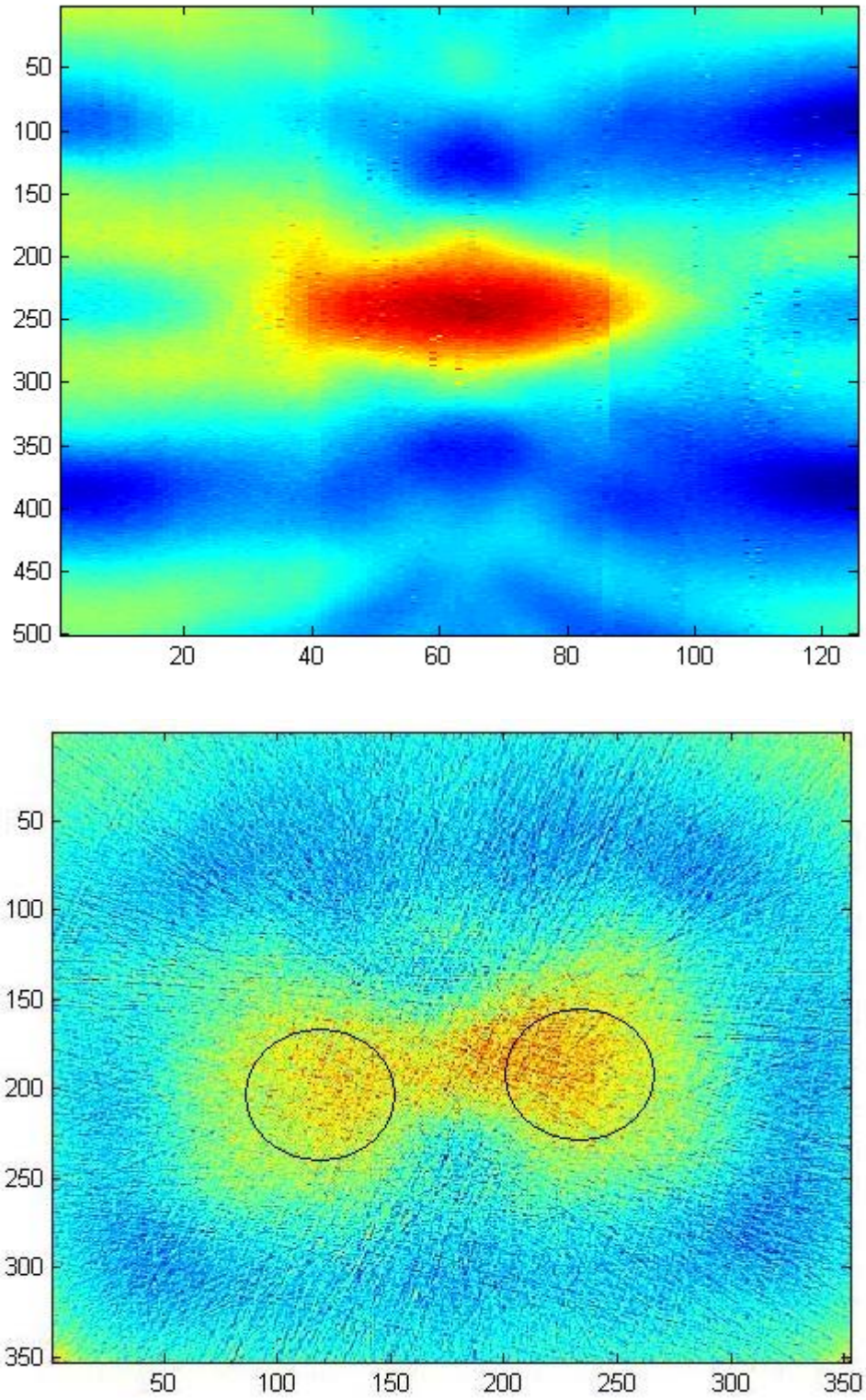

Figure 15: Image and sinogram of two jars of oil, centers 2.5” apart

The other class of image reconstructions attempted was the class of rectangles. The rectangle used in the image below was 6.5” x .75”, at a dielectric constant of about 2.

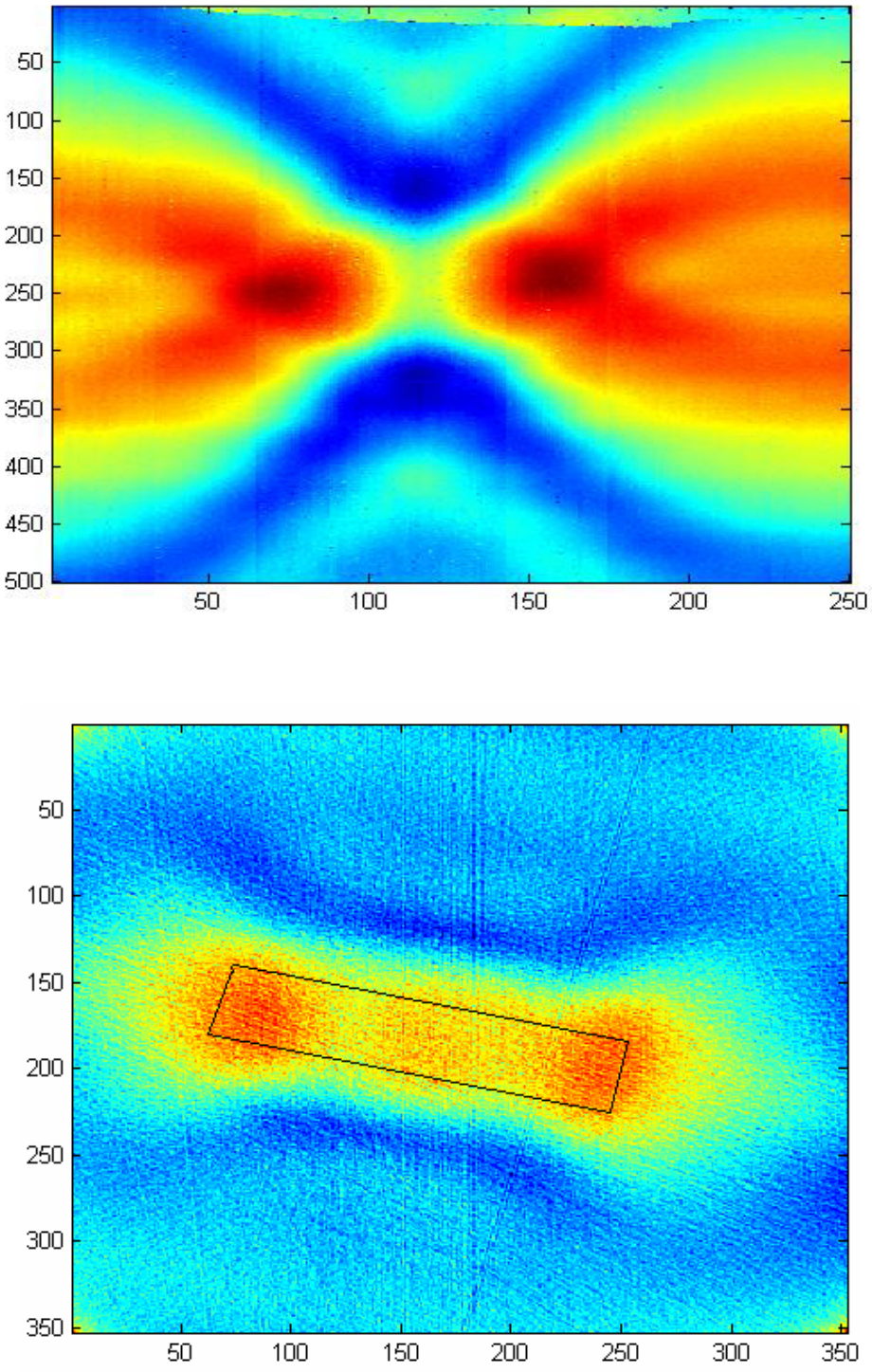

Figure 16: Image and sinogram of rectangle

CHAPTER FOUR

Discussion and Conclusions

From the results, it is clear that we have accomplished what we set out to do: prove that an imaging system based on acquisition of microwave frequency Transit Time through an object of some dielectric constant is possible. Circles are particularly well reconstructed, and their placements in the image correspond very well to the actual positions of the jars of oil. Our resolution limit is reached when the objects are placed .5" from each other, which again fits what we would expect to see: our theoretical resolution limit is half our minimum wavelength, which is .75 cm at 20 GHz.

The reconstruction of the rectangle proved far more difficult. Appropriate values for the delay are acquired for almost all possible angles, but when the object presents a vertical profile to the antennas, suddenly our delay calculations break down – giving too small of a delay over too wide a dimension. This effect is seen in the image by the uneven value of intensity within the homogeneous rectangle. Possible causes of this effect are the greater attenuation experienced by the transmitted signals as they attempt to pass though the entire object, or the waveguide effect the geometry of the system presents for this projection. Further investigation is necessary to ascertain the cause of these issues and to improve the results for this class of objects. Despite these shortcomings, the profile of the rectangle image matches the positioning of the actual rectangle, and remains within our resolution limit of .75 cm.

The list of novel accomplishments that these results validate is quite large: this project has reconstructed images of objects using a Transit Time system of approximating

the Radon transform. This was also accomplished using the air as our transmission medium – without the use of expensive saline tanks to slow the waves down and artificially increase resolution and limit reflection/scattering effects. Finally, this project has accomplished these tasks using broadband microwave pulses – a feat only Miyakawa has matched, and then only in a saline tank [3].

While other imaging techniques offer more resolution or more tolerance to various geometries of objects, this new technique offers its own set of unique benefits. Looking ahead to the future, it would be inexpensive to manufacture and offers potential for quick imagery. The ease of manufacturing broadband antennas – indeed, these antennas were made by the author, who is not a man known for his craftsmanship, out of simple brass sheets and solder – would make a low cost version of the device very attractive. The utter simplicity of the calculations required to reconstruct the images makes the computational hardware and time requirements that much less. It is easy to envision a sub $1000 device that is practical for use in doctor's offices for easy and reliable breast cancer detection, or in factories for rapid mixing verifications of sealed vats.

The first steps towards this goal are to continue to refine and explore different geometries of objects and how the system handles reconstruction of those images. Research should also be started on switching the system to a fan-beam projection technique, to begin cutting down on image acquisition times. Smaller broadband antennas to allow for arrays of receivers and ultra-wideband pulse generators to drive them should also be explored, rather than the chirp pulses and large antennas used in this proof of concept.

APPENDIX

Code

Visual Basic Code:

```
Dim stopHit As Boolean
Private Sub cboStep_Angle_Click()
Dim A As Integer
' Dim Counter2 As Integer
Dim Counter As Integer
Dim DriverSelection As String
Dim Stepcount As Integer


Select Case cboStep_Angle.Text
  Case "500"

               DriverSelection = "0"
  Case ".36"

               DriverSelection = "1"
  Case ".288"

               DriverSelection = "2"
  Case ".18"

               DriverSelection = "3"
  Case ".144"

               DriverSelection = "4"
  Case ".09":

               DriverSelection = "5"
  Case ".072":

                DriverSelection = "6"
  Case ".036":

                 DriverSelection = "7"
  Case ".0288":

                 DriverSelection = "8"
  Case ".018"

                 DriverSelection = "9"
  Case ".0144"
```

```
                DriverSelection = "A"
  Case ".009"

                DriverSelection = "B"
  Case ".0072"

                DriverSelection = "C"
  Case ".00576"

                DriverSelection = "D0"
  Case ".0036"

                DriverSelection = "E"
  Case ".00288"

                DriverSelection = "F"
End Select
Dim Temp As String
Temp = "Make sure to change the Value on the Driver to: " &
DriverSelection
A = MsgBox(Temp, 0, "Change Hardware Step Angle on Driver")

End Sub


Private Sub cmd_Send_LS_Pulses_Click()
'Loop for pulse count
For Counter = 1 To txtNumLSPulses.Text

    'Test for direction
    If Option1(1).Value = True Then
        MSComm1.Output = Chr$(3)  'If left is checked
(option 1), send C
    Else
        MSComm1.Output = Chr$(2)  'If right is checked
(option 0), send b
    End If
Pause_Run 10
Next Counter
End Sub

Sub cmdAutoAcq_Click()

Dim Counter2 As Integer
Dim txt As String
Dim Str_inx1 As Integer
Counter2 = 0
```

```
cmdAutoAcq.Enabled = False

For Module1.Counter1 = 1 To 250

MSComm1.Output = Chr$(5)  'move left
Wait2:
If (MSComm1.Input = Chr$(1)) Then
Else
    Pause_Run 1
    lbl_Pause.Caption = "Resetting"
    GoTo Wait2
End If
' ---------------------------------
    flag1 = False
' Create File with name given in the file name boxes
    Set Module1.fso =
CreateObject("Scripting.FileSystemObject")
    Set Module1.textOut =
Module1.fso.CreateTextFile(folderName.Text + "/" +
fileName.Text + CStr(Module1.Counter1) + ".csv", True)
    Module1.textOut.Close
    Progress.Value = 0
    stopHit = False
    StopBut.Enabled = True
' ---------------------------------
For Counter2 = 1 To 500
' ---------------------------------

        Progress.Max = 500

        Pause_Run 1000
        Call MeasureScope(Counter2, folderName.Text,
fileName.Text)
        Progress.Value = Counter2
        If stopHit = True Then
            lbl_Pause.Visible = False
            stopHit = False
            Exit Sub
        End If

' ---------------------------------
LBLB:
    If (Cbx_Pause.Value = 1) Then
        lbl_Pause.Visible = True
        lbl_Pause.Caption = "Paused"
        Pause_Run 1
        GoTo LBLB
```

```
    End If
    lbl_Pause.Caption = "Running"
    MSComm1.Output = Chr$(4)             'increment slide
position right
Next Counter2
Pause_Run 500
MSComm1.Output = Chr$(1)
Pause_Run 100
Next Module1.Counter1
Close

StopBut.Enabled = False
lbl_Pause.Visible = False
cmdAutoAcq.Enabled = True
End Sub


Private Sub cmdBslnSet_Click()
Dim Counter2 As Integer
Dim txt As String
Dim Str_inx1 As Integer


Counter2 = 0



MSComm1.Output = Chr$(5)  'move left
Wait2:
If (MSComm1.Input = Chr$(1)) Then
Else
    Pause_Run 1
    lbl_Pause.Caption = "Resetting"
    GoTo Wait2
End If
' ---------------------------------
    flag1 = False
' Create File with name given in the file name boxes
    Set Module1.fso =
CreateObject("Scripting.FileSystemObject")
    Set Module1.textOut =
Module1.fso.CreateTextFile(folderName.Text + "/" +
fileName.Text + CStr(Module1.Counter1) + ".csv", True)
    Module1.textOut.Close
    Progress.Value = 0
    stopHit = False
    StopBut.Enabled = True
```

```
' ---------------------------------
For Counter2 = 1 To 500
' ---------------------------------

        Progress.Max = 500

        Pause_Run 1000
        Call MeasureScope(Counter2, folderName.Text,
fileName.Text)
        Progress.Value = Counter2
        If stopHit = True Then
            lbl_Pause.Visible = False
            stopHit = False
            Exit Sub
        End If


' ---------------------------------
LBLB:
    If (Cbx_Pause.Value = 1) Then
        lbl_Pause.Visible = True
        lbl_Pause.Caption = "Paused"
        Pause_Run 1
        GoTo LBLB
    End If
    lbl_Pause.Caption = "Running"
    MSComm1.Output = Chr$(4)            'increment slide
position right
Next Counter2
Close

StopBut.Enabled = False
lbl_Pause.Visible = False
End Sub

Private Sub cmdConnectionBeep_Click()
Dim o As Object
    Dim txt As String

    Set o = CreateObject("LeCroy.ActiveDSOCtrl.1")

    Dim deviceAddress As String
    deviceAddress = "IP: 192.168.77.12"

    Dim success As Boolean
    success = o.MakeConnection(deviceAddress)
    If (success = False) Then
```

```
        MsgBox "DSO not found ! Address may be wrong ..."
        GoTo 999
    End If

    Call o.SetRemoteLocal(1)
    Call o.WriteString("buzz beep;*idn?", 1)
    txt = o.ReadString(500)

    ' check accumulated error status
    If (o.ErrorFlag = True) Then
        MsgBox o.ErrorString
    End If

    ' release the control
999 Set o = Nothing
End Sub
Private Sub Command2_Click()
MSComm1.Output = Chr$(5)
End Sub

Private Sub cmdSendPulse_Click()
Dim Counter As Integer
For Counter = 1 To txtNumPulses.Text
    'Output that number of Pulses
    MSComm1.Output = Chr$(1)
    Pause_Run 100
Next Counter
End Sub

Private Sub Command1_Click()
MSComm1.Output = Chr$(4)
End Sub


Private Sub Form_Load()
    Me.Left = GetSetting(App.Title, "Settings", "MainLeft",
1000)
    Me.Top = GetSetting(App.Title, "Settings", "MainTop",
1000)
    'Me.Width = GetSetting(App.Title, "Settings",
"MainWidth", 10000)
    'Me.Height = GetSetting(App.Title, "Settings",
"MainHeight", 6500)
    stopHit = False
    StopBut.Enabled = False

    '  Use COM1
```

```
    MSComm1.CommPort = 1
    ' 2400 baud, no parity, 8 data bits, 1 stop bit
    MSComm1.Settings = "2400,N,8,1"
    ' Open the port
    MSComm1.PortOpen = True

End Sub


Private Sub Form_Unload(Cancel As Integer)
    Dim i As Integer


    'close all sub forms
    For i = Forms.Count - 1 To 1 Step -1
        Unload Forms(i)
    Next
    If Me.WindowState <> vbMinimized Then
        SaveSetting App.Title, "Settings", "MainLeft",
Me.Left
        SaveSetting App.Title, "Settings", "MainTop",
Me.Top
        SaveSetting App.Title, "Settings", "MainWidth",
Me.Width
        SaveSetting App.Title, "Settings", "MainHeight",
Me.Height
    End If
End Sub

Private Sub ProgressBar1_MouseDown(Button As Integer, Shift
As Integer, X As Single, Y As Single)
pgbAutomation.Max = NumSteps
pgbAutomation.Value = NumSteps



End Sub



Private Sub MSComm1_OnComm()
   Select Case MSComm1.CommEvent
   ' Errors
      Case comEventBreak   ' A Break was received.
        MsgBox "A Break was received"
      Case comEventFrame   ' Framing Error
        MsgBox "Framing Error"
```

```
      Case comEventOverrun   ' Data Lost.
        MsgBox "Data Lost"
      Case comEventRxOver   ' Receive buffer overflow.
        MsgBox "Receive buffer overflow"
      Case comEventRxParity   ' Parity Error.
        MsgBox "Parity Error"
      Case comEventTxFull   ' Transmit buffer full.
        MsgBox "Transmit buffer full"
      Case comEventDCB   ' Unexpected error retrieving DCB]
        MsgBox "Unexpected error retrieving DCB"
   ' Events
      Case comEvCD   ' Change in the CD line.
        'MsgBox "Ervt 1 Something Happened"
      Case comEvCTS   ' Change in the CTS line.
        'MsgBox "Evt 2 Something Happened"
      Case comEvDSR   ' Change in the DSR line.
        MsgBox "Evt 3 Something Happened"
      Case comEvRing   ' Change in the Ring Indicator.
        MsgBox "Evt 4  Something Happened"
      Case comEvReceive   ' Received RThreshold # of
                          ' chars.
      Case comEvSend   ' There are SThreshold number of
                       ' characters in the transmit
                       ' buffer.
      Case comEvEOF   ' An EOF charater was found in
                       ' the input stream
        MsgBox "Evt 5 Something Happened"
   End Select
End Sub

Private Sub pgbAutomation_MouseDown(Button As Integer,
Shift As Integer, X As Single, Y As Single)
pgbAutomation.Max = NumSteps
End Sub

Private Sub tbToolBar_ButtonClick(ByVal Button As
MSComctlLib.Button)
    On Error Resume Next
    Select Case Button.Key
        Case "Print"
            mnuFilePrint_Click
        Case "Cut"
            mnuEditCut_Click
        Case "Copy"
            mnuEditCopy_Click
        Case "Paste"
            mnuEditPaste_Click
```

```
    End Select
End Sub

Private Sub mnuHelpAbout_Click()
    MsgBox "Version " & App.Major & "." & App.Minor & "." &
App.Revision
End Sub

Private Sub mnuHelpSearchForHelpOn_Click()
    Dim nRet As Integer


    'if there is no helpfile for this project display a
message to the user
    'you can set the HelpFile for your application in the
    'Project Properties dialog
    If Len(App.HelpFile) = 0 Then
        MsgBox "Unable to display Help Contents. There is
no Help associated with this project.", vbInformation,
Me.Caption
    Else
        On Error Resume Next
        nRet = OSWinHelp(Me.hWnd, App.HelpFile, 261, 0)
        If Err Then
            MsgBox Err.Description
        End If
    End If

End Sub

Private Sub mnuHelpContents_Click()
    Dim nRet As Integer


    'if there is no helpfile for this project display a
message to the user
    'you can set the HelpFile for your application in the
    'Project Properties dialog
    If Len(App.HelpFile) = 0 Then
        MsgBox "Unable to display Help Contents. There is
no Help associated with this project.", vbInformation,
Me.Caption
    Else
        On Error Resume Next
        'nRet = OSWinHelp(Me.hwnd, App.HelpFile, 3, 0)
        If Err Then
            MsgBox Err.Description
```

```
        End If
    End If

End Sub


Private Sub mnuViewWebBrowser_Click()
    'ToDo: Add 'mnuViewWebBrowser_Click' code.
    MsgBox "Add 'mnuViewWebBrowser_Click' code."
End Sub

Private Sub mnuViewOptions_Click()
    'ToDo: Add 'mnuViewOptions_Click' code.
    MsgBox "Add 'mnuViewOptions_Click' code."
End Sub

Private Sub mnuViewRefresh_Click()
    'ToDo: Add 'mnuViewRefresh_Click' code.
    MsgBox "Add 'mnuViewRefresh_Click' code."
End Sub

Private Sub mnuViewStatusBar_Click()
    mnuViewStatusBar.Checked = Not mnuViewStatusBar.Checked
    sbStatusBar.Visible = mnuViewStatusBar.Checked
End Sub

Private Sub mnuViewToolbar_Click()
    mnuViewToolbar.Checked = Not mnuViewToolbar.Checked
    tbToolBar.Visible = mnuViewToolbar.Checked
End Sub

Private Sub mnuEditPasteSpecial_Click()
    'ToDo: Add 'mnuEditPasteSpecial_Click' code.
    MsgBox "Add 'mnuEditPasteSpecial_Click' code."
End Sub

Private Sub mnuEditPaste_Click()
    'ToDo: Add 'mnuEditPaste_Click' code.
    MsgBox "Add 'mnuEditPaste_Click' code."
End Sub

Private Sub mnuEditCopy_Click()
    'ToDo: Add 'mnuEditCopy_Click' code.
    MsgBox "Add 'mnuEditCopy_Click' code."
End Sub

Private Sub mnuEditCut_Click()
```

```
    'ToDo: Add 'mnuEditCut_Click' code.
    MsgBox "Add 'mnuEditCut_Click' code."
End Sub

Private Sub mnuEditUndo_Click()
    'ToDo: Add 'mnuEditUndo_Click' code.
    MsgBox "Add 'mnuEditUndo_Click' code."
End Sub

Private Sub mnuFileExit_Click()
    'unload the form
    Unload Me

End Sub

Private Sub mnuFilePrint_Click()
    'ToDo: Add 'mnuFilePrint_Click' code.
    MsgBox "Add 'mnuFilePrint_Click' code."
End Sub

Private Sub mnuFileProperties_Click()
    'ToDo: Add 'mnuFileProperties_Click' code.
    MsgBox "Add 'mnuFileProperties_Click' code."
End Sub

Private Sub mnuFilePrintPreview_Click()
    'ToDo: Add 'mnuFilePrintPreview_Click' code.
    MsgBox "Add 'mnuFilePrintPreview_Click' code."
End Sub

Private Sub mnuFileClose_Click()
    'ToDo: Add 'mnuFileClose_Click' code.
    MsgBox "Add 'mnuFileClose_Click' code."
End Sub

Private Sub mnuFileOpen_Click()
    Dim sFile As String


    With dlgCommonDialog
        .DialogTitle = "Open"
        .CancelError = False
        'ToDo: set the flags and attributes of the common
dialog control
        .Filter = "All Files (*.*)|*.*"
        .ShowOpen
        If Len(.fileName) = 0 Then
```

```
            Exit Sub
        End If
        sFile = .fileName
    End With
    'ToDo: add code to process the opened file

End Sub

Private Sub mnuFileNew_Click()
    'ToDo: Add 'mnuFileNew_Click' code.
    MsgBox "Add 'mnuFileNew_Click' code."
End Sub


Private Sub txt_LS_Steps_Change()

End Sub

Private Sub sbStatusBar_PanelClick(ByVal Panel As
MSComctlLib.Panel)

End Sub

' ------------------------------------------------
' Sub        : CancelPause
' Parameters :
' Created    : n - 01/29/2004 19:52:15
' Modified   :
' ------------------------------------------------
' Comments
' winds up CPU
'
' ------------------------------------------------
Private Sub CancelPause()
    m_blnCancel = True
End Sub

'*************************************************************
****************************
'This subroutine supports the runtime conversion of PAUSE.
Private Sub vbPause(iPause As Integer)
    On Error GoTo PROC_ERR
    'Debug.Print "vbPause - Entering"
    Dim CurDateTime As Date
    'Get Current Time
    CurDateTime = Now()
    'Loop until Pause achieved
```

```
    While (CurDateTime + iPause / (86400000)) > Now()
        DoEvents
    Wend
    Exit Sub
PROC_EXIT:
    On Error Resume Next
    'Debug.Print "vbPause - Exiting"
    Exit Sub
PROC_ERR:
    Debug.Print "vbPause - Error : " & Err.Number & " (" &
Err.Description & ")"
    Resume PROC_EXIT
    'Resume Next
End Sub

Private Sub cmdConnect_Click()
    Dim success As Boolean
    Call cmdConnect2_Click(txtAddress, lblStatus, success)
    If success = True Then
        cmdConnect.Enabled = False
        cmdDisconnect.Enabled = True
    End If

End Sub
Private Sub cmdDisconnect_Click()
    Dim success As Boolean
    Call cmdDisconnect2_Click(txtAddress, lblStatus,
success)
    If success = True Then
        cmdConnect.Enabled = True
        cmdDisconnect.Enabled = False
    End If
End Sub
Private Sub deviceClear_Click()
    Call DeviceClear1
End Sub

Private Sub StopBut_Click()
    stopHit = True
    StopBut.Enabled = False
End Sub

Public textOut, fso, tempLine, textTemp, textTemp2, flag1
As Boolean
Public Counter1 As Integer
    Public fMainForm As frmMain
```

```
Global NumSteps As Double
Public Sub Pause(NbSec As Single)
 Dim Finish As Single
 Finish = Timer + NbSec
 DoEvents
 Do Until Timer >= Finish
 Loop
End Sub

Sub Main()
    Set fMainForm = New frmMain
    fMainForm.Show


End Sub


'Private Declare Sub sleep Lib "kernel32" (ByVal
dwMilliseconds As Long)
'End Sub


'Public Sub Pause(ByVal Seconds As Single)
'IngMilliseconds = Seconds * 1000
'sleep IngMilliseconds
'End Sub


Option Explicit

Dim VNA As New AgtServer8714            'automation server
variable
Dim ActiveChannel As Agt_AnalogChannels 'global variable to
hold active channel
Dim StateItems As New Collection        'global collection
of instrument state objects
Dim units As String

Public Sub cmdConnect2_Click(txtAddress As TextBox,
lblStatus As Label, success As Boolean)
On Error GoTo CaptureError
    Dim szInstrumentModel As String     'variable to hold
instrument model string
    Dim szInstrumentVersion As String   'variable to hold
instrument firmware version string
    Dim szConnName As String            'variable to hold
instrument address stringz
```

```
    Dim eChannel As Agt_AnalogChannels  'enumeration
variable to find active channel

    If txtAddress.Text = "" Then 'there is an address in
the address field
        MsgBox "You must enter an address in the Instrument
Address field." ', _
        '    vbOKOnly vbExclamation, "No Address"
        GoTo End_Connect
    End If 'txtAddress.Text = ""

    'frmMain.MousePointer = vbHourglass

    'The following demonstrates how to connect to the
instrument at the given address
    VNA.Connect txtAddress.Text

    If (VNA.ConnectedStatus) Then 'an instrument connection
was made

        'The following is an example of how to get
instrument characteristics
        szConnName = VNA.ConnectionName



            'The following is an example of how to send
SCPI commands to the instrument
            Dim szReply As String           'variable to
hold string returned from instrument

            VNA.Output "*idn?"
            VNA.Enter szReply
            MsgBox "Connected to " & szReply,
vbInformation, "Connection"

            'Overhead tasks
            'cmdConnect = False '.Enabled = False
            'cmdDisconnect = True '.Enabled = True
            txtAddress.Enabled = False
            lblStatus.Caption = "Connected"
            lblStatus.BackColor = &HFF00&   'green
            success = True

    Else 'the connection was not made
        MsgBox "Could not connect to specified address!"
',_
```

```
        '    vbOKOnly vbExclamation, "Connection"
    End If '(VNA.ConnectedStatus)
        'The following demonstrates how to find the
instrument's active channel
    'programatically using the Agt_AnalogChannels
enumeration. The global variable
    'ActiveChannel will be set to the active channel for
later use
    For eChannel = Agt_AnalogChannel_1 To
Agt_AnalogChannel_4
        If VNA.Measure.SenseActive(eChannel) Then
            'frmMain.lblActiveChannel.Caption = "Channel "
& CStr(eChannel)
            ActiveChannel = eChannel
        End If 'VNA.Measure.SenseActive(eChannel)
    Next eChannel

End_Connect:
    'frmMain.MousePointer = vbDefault
    Exit Sub
CaptureError:
    Dim szErrstring As String           'variable to hold
error string

    'frmMain.MousePointer = vbDefault
    szErrstring = Err.Description
    MsgBox "Connect failed" & vbCrLf & "Error Details:" &
vbCrLf & szErrstring, _
        vbCritical, "Connect Error"

End Sub
Public Sub cmdDisconnect2_Click(txtAddress As TextBox,
lblStatus As Label, success As Boolean)
On Error GoTo CaptureError
    'The following demonstrates a way to disconnect from
the connected instrument
    VNA.Close

    If (VNA.ConnectedStatus) Then 'the instrument is still
connected
        MsgBox "Could not disconnect from instrument!" ', _
            'vbOKOnly vbExclamation, "Disconnect Error"
    Else 'the instrument is no longer connected
        'Overhead tasks
        'cmdConnect = True '.Enabled = True
        'cmdDisconnect = False '.Enabled = False
        'fraChart.Caption = "Chart"
```

```
        'frmMain.lblActiveChannel.Caption = ""
        lblStatus.BackColor = &HFF&     'red
        lblStatus.Caption = "Not Connected"

        txtAddress.Enabled = True
        'MSChart1.Visible = False
    End If '(VNA.ConnectedStatus)
    success = True
    Exit Sub

CaptureError:
    Dim szErrstring As String           'variable to hold
error string

    'frmMain.MousePointer = vbDefault
    szErrstring = Err.Description
    MsgBox "Disconnect failed" & vbCrLf & "Error Details:"
& vbCrLf & szErrstring, _
        vbCritical, "Disconnect Error"

End Sub
'-------------------------------------------------------------
-------------------
'-------------------------------------------------------------
-------------------
'-------------------------------------------------------------
-------------------
'-------------------------------------------------------------
-------------------



Public Sub DeviceClear1()
On Error GoTo CaptureError
    'The following demonstrates how to send a Device Clear
to the instrument
    'frmMain.MousePointer = vbHourglass
    VNA.Utilities.ClearDevice
    MsgBox "Device cleared successfully", vbInformation,
"Device Clear Status"
    'frmMain.MousePointer = vbDefault
    Exit Sub

CaptureError:
    Dim szErrstring As String           'variable to hold
error string
    'frmMain.MousePointer = vbDefault
```

```
    szErrstring = Err.Description
    MsgBox "Device clear failed" & vbCrLf & "Error
Details:" & vbCrLf & szErrstring, _
        vbCritical, "Device Clear Error"

End Sub



Public Sub MeasureScope(K As Integer, folderName As String,
fileName As String)
On Error GoTo CaptureError
    Dim szTime As String                'variable to hold
time string returned from instrument
    Dim szDate As String                'variable to hold
date string returned from instrument
    Dim vFrequency As Variant           'variant to hold
frequency data returned from instrument
    Dim vData As Variant                'variant to hold
measurement data returned from instrument
    Dim complexMatrix() As Double       'array in which to
place vData data points for charting
    Dim i As Integer                    'counter variable
    Dim iType As Integer                'variable integer
flag noting whether data is an array
                                        'of real numbers or
complex pairs

    'frmMain.MousePointer = vbHourglass
    units = "Frequency"
    'The following two lines demonstrate how to get the
instrument time and date
    szTime = VNA.Measure.SenseCurrentTime
    szDate = VNA.Measure.SenseCurrentDate

    'The following 6 lines demonstrate how to obtain
various instrument measures; all
    'use the currently active channel as their only
parameter
    'frmMain.lblCharImpedance.Caption =
CStr(VNA.Measure.SenseZ0(ActiveChannel))
    'frmMain.lblDisplayForm.Caption =
VNA.Measure.CalculateForm(ActiveChannel)
    'frmMain.lblFunction.Caption =
VNA.Measure.SenseFunction(ActiveChannel)
```

```
    'frmMain.lblSweepPoints.Caption =
CStr(VNA.Measure.SenseSweepPoints(ActiveChannel))
    'frmMain.lblSweepTime.Caption =
CStr(VNA.Measure.SenseSweepTime(ActiveChannel))
    'frmMain.lblXAxisDataType.Caption =
VNA.Measure.SenseXAxisScaleType(ActiveChannel)

    'frmMain.lblTimestamp.Caption = szTime & "   " & szDate


    'The following IF-THEN block is necessary in order to
determine whether the obtained
    'values should be frequencies(Hz), time(s), or
distance; if distance, whether the
    'values are in feet or meters




        If VNA.Measure.SenseTimeDomain(ActiveChannel) Then
'instrument is in time domain mode
            'The following 4 lines change the frame
captions on the form to appropriate labels
            'frmMain.fraCenterFreq.Caption = "Center Time
(Seconds)"
            'frmMain.fraStartFreq.Caption = "Start Time
(Seconds)"
            'frmMain.fraStopFreq.Caption = "Stop Time
(Seconds)"
            'frmMain.fraSpanFreq.Caption = "Span Time
(Seconds)"
            'label for output
            units = "Time (S)"

        Else 'instrument is in frequency mode
            'The following 4 lines change the frame
captions on the form to appropriate labels
            'frmMain.fraCenterFreq.Caption = "Center
Frequency (Hz)"
            'frmMain.fraStartFreq.Caption = "Start
Frequency (Hz)"
            'frmMain.fraStopFreq.Caption = "Stop Frequency
(Hz)"
            'frmMain.fraSpanFreq.Caption = "Span Frequency
(Hz)"
            'label for output
            units = "Frequency (Hz)"
```

```
        End If 'VNA.Measure.SenseTimeDomain(ActiveChannel)

        'The following 4 lines demonstrate how to retrieve
instrument measures
        'Note that whether the instrument is in frequency
or time domain mode, the
        'method for getting the values is the same
        'frmMain.lblCenterFreq.Caption =
CStr(VNA.Measure.SenseFrequencyCenter(ActiveChannel))
        'frmMain.lblSpanFreq.Caption =
CStr(VNA.Measure.SenseFrequencySpan(ActiveChannel))
        'frmMain.lblStartFreq.Caption =
CStr(VNA.Measure.SenseFrequencyStart(ActiveChannel))
        'frmMain.lblStopFreq.Caption =
CStr(VNA.Measure.SenseFrequencyStop(ActiveChannel))


    'The following line demonstrates how to retrieve the
data points from the instrument
    VNA.GetFormattedData ActiveChannel, vFrequency, vData

    'The following line demonstrates how to determine
whether the data returned by the
    'instrument is an array of real numbers or complex
pairs. This is necessary to determine
    'the dimensions of the array needed for plotting the
data points in a chart. The easiest
    'way to determine this is to divide the number of data
points returned by the instrument
    'by the number of sweep points the instrument reports.
A value of 2 indicates that the
    'data is in complex pairs; a value of 1 indicates that
the data is real numbers.
    iType = UBound(vData) /
VNA.Measure.SenseSweepPoints(ActiveChannel)


    ' Get a handle to the file
        Set Module1.textOut =
Module1.fso.GetFile(folderName + "/" + fileName +
CStr(Module1.Counter1) + ".csv")

        ' Move the file to \tmp directory.
           Module1.textOut.Copy (folderName + "/" +
fileName + CStr(Module1.Counter1) + "Temp.csv")
         'Module1.textOut.Close
```

```
        Set Module1.textOut =
Module1.fso.CreateTextFile(folderName + "/" + fileName +
CStr(Module1.Counter1) + ".csv", True)

        Set Module1.textTemp =
Module1.fso.GetFile(folderName + "/" + fileName +
CStr(Module1.Counter1) + "Temp.csv")

        Set Module1.textTemp2 =
Module1.textTemp.OpenAsTextStream(ForReading)

'            If Module1.flag1 Then
'                    Module1.tempLine =
Module1.textTemp2.ReadLine
'                    Module1.textOut.Write
(Module1.tempLine)
'            Else
'                    Module1.textOut.Write (units + " ")
'                    Module1.flag1 = True
'            End If
'            Module1.textOut.WriteLine ("Run " + CStr(K) +
" ")

    If iType = 2 Then 'data is in complex pairs
        ReDim complexMatrix(0 To
(VNA.Measure.SenseSweepPoints(ActiveChannel) - 1), 0 To 1)
As Double

        For i = 0 To UBound(complexMatrix, 1)
            If Not (K = 1) Then
                Module1.tempLine =
Module1.textTemp2.ReadLine
            Module1.textOut.Write (Module1.tempLine)
            End If

            If K = 1 Then

                Module1.textOut.Write (CStr(vFrequency(i))
+ " ")
            End If
            ' Read data out of array into real and
imgainary parts
            complexMatrix(i, 0) = vData((i) * 2)
            complexMatrix(i, 1) = vData((i) * 2 + 1)
            Module1.textOut.WriteLine
(CStr(complexMatrix(i, 0)) + " ")
```

```
        Next i

        Module1.textOut.Close
        Module1.textTemp2.Close
        Module1.textTemp.Delete

    Else 'data is real numbers -- PROBABLY NOT USED AT ALL
        ReDim
complexMatrix(VNA.Measure.SenseSweepPoints(ActiveChannel) -
1, 0 To 1) As Double
        For i = 0 To UBound(complexMatrix, 1)
            If (vData(i) > 1000000) Then
                complexMatrix(i, 0) = (vData(i) / 1000000)
                'frmMain.Chart.Caption = "Chart (Data in
millions)"
            Else
                complexMatrix(i, 0) = vData(i)
            End If '(vData(i) > 1000000)

            complexMatrix(i, 1) = 0
        Next i
    End If 'iType = 2



    'frmMain.MousePointer = vbDefault

    Exit Sub

CaptureError:
    Dim szErrstring As String           'variable to hold
error string

    'frmMain.MousePointer = vbDefault
    szErrstring = Err.Description
    MsgBox "Measure operation failed" & vbCrLf & "Error
Details:" & vbCrLf & szErrstring, _
        vbCritical, "Measure Operation Error"

End Sub

Private m_blnUseRTPause As Boolean
Private m_blnCancel As Boolean
Private Declare Function WaitForSingleObject Lib "kernel32"
(ByVal hHandle As Long, ByVal dwMilliseconds As Long) As
Long
```

```
Private Declare Function CreateEvent Lib "kernel32" Alias
"CreateEventA" (ByVal lpEventAttributes As Long, ByVal
bManualReset As Long, ByVal bInitialState As Long, ByVal
lpname As String) As Long
Private Declare Function CloseHandle Lib "kernel32" (ByVal
hObject As Long) As Long
Private Declare Function GetTickCount Lib "kernel32" () As
Long
Private Sub CancelPause()
    m_blnCancel = True
End Sub
Public Sub Pause_Run(ByVal lngMS As Long)
    On Error GoTo PROC_ERR
    'Debug.Print "apiPause - Entering"
    Dim lngHandle As Long   ' handle of dummy object
    Dim lngRetVal As Long   ' status of API call
    Dim lngTotalMS As Long  ' number of milliseconds we
have waited
    ' number of MS to wait each iteration
    ' the smaller the number, the smoother the UI
    ' the smaller the number, the more times this loop gets
processed
    ' somewhere between 5-10 is nirvana for my machine
    Const WAIT_INCREMEMENT = 5
    ' explicit initialization is always good
    lngTotalMS = 0
    ' create a dummy event that will never fire
    lngHandle = CreateEvent(ByVal 0&, False, False, ByVal
0&)
    ' loop in WAIT_INCREMEMENT MS increments until we
accumulate requested delay
    Do While lngTotalMS < lngMS
        ' this will wait the requested milliseconds for the
object to fire
        ' since we never fire, it will return in the number
of milliseconds
        lngRetVal = WaitForSingleObject(lngHandle,
WAIT_INCREMEMENT)
        ' keep track of how long we have waited so far
        lngTotalMS = lngTotalMS + WAIT_INCREMEMENT
        ' process UI thread, to allow for cancel events,
and other updates on UI thread
        ' this makes everthing smooth
        DoEvents
        If m_blnCancel Then
            ' I am done with THIS witness
            Exit Do
```

```
        End If
    Loop
PROC_EXIT:
    On Error Resume Next
    'Debug.Print "apiPause - Exiting"
    ' make sure we clean up
    lngRetVal = CloseHandle(lngHandle)
    Exit Sub
PROC_ERR:
    Debug.Print "apiPause - Error : " & Err.Number & " (" &
Err.Description & ")"
    Resume PROC_EXIT
End Sub
'Public Sub Pause_Run(NbSec As Single)
' Dim Finish As Single
' Finish = Timer + NbSec
' DoEvents
' Do Until Timer >= Finish
' Loop
'End Sub
```

Matlab code

```
profiles = [];
ROIstart = 50;
ROIstop = 70;
summass = 0;
center = [];
%mov = avifile('words.avi');
timesum=0;
time = 2;
L = 0;
ROIspecial = zeros(1,500);

for n=1:125
    tic
    filename = ['C:\Documents and
Settings\TRUMBO.ECS\Desktop\data\11154inRedeaux' num2str(n)
'.csv'];
    clc
    n
    timeleft = (125*(timesum/n) -(timesum))/60
    time = 0;
    timeelapsed = timesum/60
    filename
    current = importdata(filename);
```

```
    filtCurr = [];
    for i=1:500
        filtCurr = [filtCurr conv(current(:,i), [1 1 1 1
1])];
    end

    filtCurr(401:405,:) = [];
    filtCurr(1:6,:) = [];
%     imagesc(filtCurr);
%     F = getframe(gca);
%     mov = addframe(mov,F);

    for i=1:500
        for k=ROIstart:ROIstop
            summass = summass + k*filtCurr(k,i);
        end
        center = [center
summass/sum(filtCurr(ROIstart:ROIstop,i))];
        summass = 0;
    end
    time = toc;
    timesum = timesum + time;
end

%mov = close(mov);
center2 = reshape(center, 500, length(center)/500);
center3 = center2-min(min(center2));
centertrack = center3;

figure;imagesc(center3)
pic2 = iradon(center3,180/125:180/125:180);
figure;imagesc(pic2)
figure;imshow(pic2/max(max(pic2)));
```

Basic Stamp Code

```
'{$STAMP BS2}
Operation  VAR   Byte
BtnWk      VAR   Byte
Counter    VAR   Word

Main:

' Use the programming port to receive
' data at 2400 baud
' Wait for the synch byte (255) and then
' get the Operation
```

```
Start:
Counter = 0
SERIN 16,16780,[WAIT(255),Operation]
IF Operation = 1 THEN Rotate_Table
IF Operation = 2 THEN Inc_AntsR
IF Operation = 3 THEN Inc_AntsL
IF Operation = 4 THEN Auto_Sens
IF Operation = 5 THEN Ret_Home

GOTO Start

Inc_AntsR:                              'Increment antennas
right
HIGH 3
BUTTON 15,1, 255, 250, BtnWk,1,Done  'checks if LS2 (pin
15) is 1 (pressed)
HIGH 1                                 'Direction --> right
PULSOUT 0, 50                         'Increment motor 1 CW
(slide to right)
GOTO Done

Inc_AntsL:                              'Increment antennas
left
HIGH 3
BUTTON 14,1, 255, 250, BtnWk,1,Done  'checks if LS1 (pin
14) is 1 (pressed)
LOW 1                                  'Direction --> left
PULSOUT 0, 50                          'Increment motor 1 CCW
(slide to left)
GOTO Done

Rotate_Table:                           'Increment table
rotationally
HIGH 3
PULSOUT 2, 50                          'Increment motor 2 CW
GOTO Done

Auto_Sens:
HIGH 3
BUTTON 15,1, 255, 250, BtnWk,1,Done  'checks if LS2 (pin
15) is 1 (pressed)
HIGH 1                                 'Direction --> right
Loop:
PULSOUT 0, 50                         'Increment motor 1 CW
(slide to right)
IF Counter > 100 THEN Done
GOTO Loop
```

```
Ret_Home
Auto_Sens:
HIGH 3
BUTTON 14,1, 255, 250, BtnWk,1,Done  'checks if LS1 (pin
14) is 1 (pressed)
LOW 1                           'Direction --> left
Loop:
PULSOUT 0, 50                   'Increment motor 1 CW
(slide to right)
IF Counter > 1000 THEN Done
GOTO Loop

Done:
LOW 3
GOTO Start
```

BIBLIOGRAPHY

[1] Semenov S.Y., Bulyshev A.E., Abubakar A., Posukh V.G., Sizov, Y.E., Souvorov A.E., van den Berg P.M., Williams T.C., "Microwave-tomographic imaging of the high dielectric-contrast objects using different image-reconstruction approaches"
IEEE Transactions on Microwave Theory and Techniques, Volume 53, pp 2284 - 2294 Issue 7, July 2005

[2] Miyakawa M., Eiyama M., Ishii N, "An attempt of time domain microwave computed tomography for biomedical use" *Engineering in Medicine and Biology Society*, *Proceedings of the 23rd Annual International Conference of the IEEE* 2001

[3] Avinash C. Kak, Malcolm Slaney, *Principles of Computerized Tomographic Imaging*, IEEE Press, New York, 1988.

[4] G. N. Hounsfield, J. A. Ambrose. "Computerized Transverse Axial Scanning (Tomography)," *British Journal of Radiology*, 46:1016--1022, 1973

[5] L. Wang, "Cross-section reconstruction with a fan-beam scanning geometry," *IEEE Trans. Comput.*, Vol. C-26, pp 351-64, No. 7, 1977