The Ultrasound Brain Helmet:

Simultaneous Multi-transducer 3D Transcranial Ultrasound Imaging

by

Brooks D. Lindsey

Department of Biomedical Engineering
Duke University

Date:_______________________

Approved:

___________________________
Stephen W. Smith, Supervisor

___________________________
Jeremy J. Dahl

___________________________
Daniel T. Laskowitz

___________________________
Gregg E. Trahey

___________________________
Rebecca M. Willett

Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Biomedical Engineering in the Graduate School of Duke University

2012
ABSTRACT

The Ultrasound Brain Helmet:
Simultaneous Multi-transducer 3D Transcranial Ultrasound Imaging

by
Brooks D. Lindsey

Department of Biomedical Engineering
Duke University

Date:_______________________
Approved:

___________________________
Stephen W. Smith, Supervisor

___________________________
Jeremy J. Dahl

___________________________
Daniel T. Laskowitz.

___________________________
Gregg E. Trahey

___________________________
Rebecca M. Willett

An abstract of a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Biomedical Engineering in the Graduate School of Duke University

2012
Abstract

In this work, I examine the problem of rapid imaging of stroke and present ultrasound-based approaches for addressing it. Specifically, this dissertation discusses aberration and attenuation due to the skull as sources of image degradation and presents a prototype system for simultaneous 3D bilateral imaging via both temporal acoustic windows. This system uses custom sparse array transducers built on flexible multilayer circuits that can be positioned for simultaneous imaging via both temporal acoustic windows, allowing for registration and fusion of multiple real-time 3D scans of cerebral vasculature. I examine hardware considerations for new matrix arrays—transducer design and interconnects—in this application. Specifically, it is proposed that signal-to-noise ratio (SNR) may be increased by reducing the length of probe cables. This claim is evaluated as part of the presented system through simulation, experimental data, and \textit{in vivo} imaging. Ultimately, gains in SNR of 7 dB are realized by replacing a standard probe cable with a much shorter flex interconnect; higher gains may be possible using ribbon-based probe cables. \textit{In vivo} images are presented depicting cerebral arteries with and without the use of microbubble contrast agent that have been registered and fused using a search algorithm which maximizes normalized cross-correlation.

The scanning geometry of a brain helmet-type system is also utilized to allow each matrix array to serve as a correction source for the opposing array. Aberration is
estimated using cross-correlation of RF channel signals followed by least mean squares solution of the resulting overdetermined system. Delay maps are updated and real-time 3D scanning resumes. A first attempt is made at using multiple arrival time maps to correct multiple unique aberrators within a single transcranial imaging volume, i.e. several isoplanatic patches. This adaptive imaging technique, which uses steered unfocused waves transmitted by the opposing or “beacon” array, updates the transmit and receive delays of 5 isoplanatic patches within a 64°×64° volume. In phantom experiments, color flow voxels above a common threshold have increased by an average of 92% while color flow variance decreased by an average of 10%. This approach has been applied to both temporal acoustic windows of two human subjects, yielding increases in echo brightness in 5 isoplanatic patches with a mean value of 24.3 ± 9.1%, suggesting such a technique may be beneficial in the future for improving image quality in non-invasive 3D color flow imaging of cerebrovascular disease including stroke.

Acoustic window failure and the possibility of overcoming it using a low frequency, large aperture array are also examined. In performing transcranial ultrasound examinations, 8-29% of patients in a general population may present with window failure, in which it is not possible to acquire clinically useful sonographic information through the temporal acoustic window. The incidence of window failure is higher in the elderly and in populations of African descent, making window failure an important concern for stroke imaging through the intact skull. To this end, I describe
the technical considerations, design, and fabrication of low-frequency (1.2 MHz), large aperture (25.3 mm) sparse matrix array transducers for 3D imaging in the event of window failure. These transducers are integrated into the existing system for real-time 3D bilateral transcranial imaging and color flow imaging capabilities at 1.2 MHz are directly compared with arrays operating at 1.8 MHz in a flow phantom with approximately 47 dB/cm^{0.8}/MHz^{0.8} attenuators. In vivo contrast-enhanced imaging allowed visualization of the arteries of the Circle of Willis in 5 of 5 subjects and 8 of 10 sides of the head despite probe placement outside of the acoustic window. Results suggest that the decrease from approximately 2 to 1 MHz for 3D transcranial ultrasound may be sufficient to allow acquisition of useful images either in individuals with poor windows or outside of the temporal acoustic window by untrained operators in the field.
Dedication

For Ned Light, a great friend and teacher.
Contents

Abstract.................................................................................................................................iv

List of Tables ..........................................................................................................................xiv

List of Figures .........................................................................................................................xv

Acknowledgements ...............................................................................................................xxii

1. Introduction .........................................................................................................................1

1.1 Background and significance..........................................................................................1

1.1.1 Stroke .........................................................................................................................1

1.1.2 Stroke imaging .............................................................................................................3

1.1.3 Stroke treatment ..........................................................................................................6

1.2 Transcranial ultrasound imaging ..................................................................................7

1.2.1 Real-time and portable real-time ultrasound ..............................................................7

1.2.2 The temporal acoustic window and transskull imaging ..........................................8

1.2.3 Real-time 3D ultrasound ...........................................................................................10

1.2.4 Registration and rendering of 3D ultrasound ............................................................11

1.2.5 Multi-array ultrasound imaging ................................................................................13

1.2.6 Transcranial ultrasound transducers .........................................................................13

1.2.7 Phase aberration correction .......................................................................................14

1.3 Objective .........................................................................................................................15

1.4 Summary of chapters ....................................................................................................16

2. Ultrasound brain helmet: real-time 3D multi-transducer imaging system ..................22
2.1 Introduction .......................................................................................................................... 22
  2.1.1 Overview ......................................................................................................................... 22
  2.1.2 Volumetrics 3D Scanner ............................................................................................... 22
  2.1.3 Motivation and initial prototype ................................................................................. 24
  2.1.4 Prototype with custom transducers ............................................................................. 26
  2.1.5 Cabling and transducers ............................................................................................... 26
  2.1.6 Aperture design ............................................................................................................. 28
  2.1.7 Transducer fabrication and integration ..................................................................... 31
  2.1.8 Performance Testing ..................................................................................................... 33
  2.1.9 Volume registration ....................................................................................................... 34
  2.1.10 Data transfer and scan conversion ............................................................................ 35
  2.1.11 Pre-processing ............................................................................................................ 35
  2.1.12 Correlation search ...................................................................................................... 36
  2.1.13 Validation of algorithm ............................................................................................. 38
  2.1.14 Thresholding and display ......................................................................................... 40
  2.1.15 Human study ............................................................................................................... 40

2.2 Results .................................................................................................................................. 42
  2.2.1 Transducer simulation ................................................................................................. 42
  2.2.2 Transducer testing ....................................................................................................... 42
  2.2.3 Validation of registration algorithm ........................................................................... 43
  2.2.4 Human study ............................................................................................................... 44

2.3 Discussion .............................................................................................................................. 49
3. Pitch-catch phase aberration correction of two matrix arrays........................................52

3.1 Background ..................................................................................................................52

3.1.1 The isoplanatic patch ..............................................................................................52

3.1.2 Phase aberration correction techniques .................................................................54

3.1.3 Time reversal and inverse filtering .........................................................................56

3.1.4 Brain helmet pitch-catch phase aberration correction .............................................57

3.2 Methods ......................................................................................................................58

3.2.1 Scanning system ....................................................................................................58

3.2.1 Cramér-Rao lower bound ......................................................................................60

3.2.2 Aberration estimation and computation of delay updates .......................................62

3.2.3 Correction of multiple isoplanatic patches .............................................................65

3.2.4 Algorithm testing and validation ..........................................................................67

3.2.5 Evaluation of steered transmit waves as correction beacons ................................68

3.2.6 Temporal bone isoplanatic patch ..........................................................................69

3.2.7 Physical aberrator correction and color flow imaging ...........................................71

3.2.8 Offline processing, registration, and fusion ............................................................71

3.2.9 Human scanning procedure ...................................................................................72

3.3 Phase aberration correction results ............................................................................73

3.3.1 Cramér-Rao lower bound ......................................................................................73

3.3.2 Electronic aberrators .............................................................................................73

3.3.3 Evaluation of a steered unfocused wave as a transmit beacon ..............................74

3.3.4 Temporal bone isoplanatic patch ..........................................................................77
3.3.5 Physical aberrator correction and color flow imaging................................. 77
3.3.6 In vivo imaging......................................................................................... 78
3.4 Discussion.................................................................................................. 86
  3.4.1 Timing constraints.................................................................................... 86
  3.4.2 Direction of arrival estimation................................................................. 87
  3.4.3 Correction of multiple isoplanatic patches............................................. 88
  3.4.4 Summary................................................................................................ 89

4. Dual transducer imaging outside of temporal acoustic window ..................... 91
  4.1 Introduction.................................................................................................. 91
    4.1.1 Temporal acoustic window failure....................................................... 91
    4.1.2 Window failure demographics............................................................... 94
    4.1.3 Addressing window failure................................................................... 95
  4.2 Methods ..................................................................................................... 97
    4.2.1 Frequency Analysis.............................................................................. 97
    4.2.2 System SNR considerations................................................................. 99
    4.2.3 Color flow SNR.................................................................................... 101
    4.2.4 Maximization of stationary-to-blood scattering ratio model.............. 102
    4.2.5 Blood flow velocity estimation............................................................. 105
    4.2.6 Transducer design.............................................................................. 107
    4.2.7 Transducer fabrication......................................................................... 109
    4.2.8 Transducer testing............................................................................... 111
    4.2.9 Safety measurements.......................................................................... 111
4.2.10 Phantom testing ................................................................. 112
4.2.11 In vivo imaging ................................................................. 113
4.3 Results .................................................................................... 114
  4.3.1 KLM model simulations ..................................................... 114
  4.3.2 Transducer testing ............................................................ 116
  4.3.3 Safety measurements ......................................................... 117
  4.3.4 Phantom imaging .............................................................. 117
  4.3.5 In vivo imaging ................................................................. 118
  4.4 Discussion .............................................................................. 123

5. Future work ............................................................................ 126
  5.1 Introduction ........................................................................... 126
  5.2 System design considerations ............................................... 126
    5.2.1 Probe frequency ............................................................ 126
    5.2.2 Matrix array considerations ........................................... 127
    5.2.3 Alternative system configurations .................................. 128
  5.3 Phase aberration correction improvements ....................... 132
    5.3.1 Steered transmit wavefronts .......................................... 132
    5.3.2 Isoplanatic patch measurement ...................................... 134
    5.3.3 Isoplanatic patch correction performance ..................... 134
    5.3.4 Overcoming the isoplanatic patch .................................. 137
    5.3.5 Phase aberration estimation .......................................... 138
  5.4 Alternative means of transcranial ultrasound propagation .... 140
5.5 Future utilization of transcranial ultrasound for emergent stroke imaging........ 141

Appendix A: Transcranial ultrasound imaging and the KLM model............................ 144
Appendix B: Decorrelation and spatial lags in the presence of a correction beacon ...... 151
Appendix C: Aberrated and corrected beam profiles............................................. 155
Appendix D: Analysis of steered beacon aberration correction............................ 158
Appendix E: Aberration simulation code......................................................... 163
Appendix F: Phase aberration correction code.................................................. 170
Appendix G: Model 1 scanner simulation code .................................................. 244
Appendix H: Dual volume display code............................................................. 253
References........................................................................................................ 302
Biography.......................................................................................................... 321
List of Tables

Table 1: Summary of phase aberration correction data for two human subjects .......... 86

Table 2: Comparison of window failure rates in previous transcranial ultrasound studies ................................................................................................................. 93

Table 3. Materials parameters for KLM model simulations ........................................ 108

Table 4: Results of KLM modeling, 1.2 MHz excitation response .............................. 115

Table 5. Detection rates of major cerebral arteries through the acoustic windows in 3D Doppler imaging at 1.8 MHz versus outside the acoustic windows at 1.2 MHz. (ACA = anterior cerebral artery, ICA = internal carotid artery, MCA = middle cerebral artery, PCA = posterior cerebral artery.) ................................................................................................. 121

Table 6: Material constants for KLM simulations ....................................................... 146

Table 7: Results of KLM model without matching layer ............................................ 146

Table 8: Results of KLM model with single matching layer (6.44 MRayl) .................. 147

Table 9: Results of KLM model with single matching layer (7.84 MRayl) ............... 147

Table 10: Results of KLM model for ceramic and composite transducers .................. 149

Table 11: Results of KLM model with single matching layer for ceramic (6.44 MRayl) and composite (4.76 MRayl) active structures ............................................. 149

Table 12: Parameters for jitter simulations ................................................................. 153
List of Figures

Figure 1: (A) Coronal ultrasound rendering and (B) representative magnetic resonance (not the same subject) showing the area under examination. Original MRA image produced by Ofir Glazer, Biomedical Engineering Department Tel Aviv University, reproduced with permission of the author. ................................................................. 17

Figure 2: A. Experimental setup (not to scale) for flow measurements in a water tank in the presence of two physical aberrators. Actual probe-to-tube distance was approximately 6 cm. B. Registered, fused rendering without aberration correction. C. Rendering acquired when a single phase map is used to correct the entire 3D volume. D. Rendering acquired when five phase maps are used to correct appropriate regions of the 3D volume. ........................................................................................................ 19

Figure 3. Registered, fused rendering of a healthy 34-year-old male with Definity® microbubble contrast enhancement. (ACA = anterior cerebral artery, ICA = internal carotid artery, MCA = middle cerebral artery, PCA = posterior cerebral artery.) ........... 21

Figure 4: Volume interrogated during a typical transtemporal 3D ultrasound examination. Coronal and transverse plane views are acquired simultaneously. A steerable Doppler beam is shown in blue................................................................. 23

Figure 5: Experimental setup (left) and resultant 3D rendering (right) acquired using initial hand-wired prototype. Red and blue indicate two different acquiring transducers; no automated registration was used, only manual registration. ........................................ 25

Figure 6: (A) Resultant B-mode + color flow scans and (B) 3D rendering using printed circuit board prototype. Experimental setup is as shown in Fig. 5A. ........................................ 25

Figure 7: The first in vivo acquisition using the brain helmet, these images show blood flow in both internal carotid arteries (A) in a screenshot from the scanner’s real-time bilateral B-mode + color flow display, and (B) in offline, manually fused renderings of the same data. ................................................................. 26

Figure 8: (A) Designed aperture (gray elements only transmit, white elements transmit and receive) and (B) resultant on-axis beamplot simulated using Field II. Simulations of the aperture steered to (C) +16° and (D) +32° in both azimuth and elevation at 2.5 MHz and a 7 cm focus. .................................................................................................. 30

Figure 9: Photograph of an array after completion of dicing. ........................................ 32
Figure 10: Illustrative stack-up of transducer fabrication. A lossy backing is bonded to the back of the flex (not shown). ................................................. 32

Figure 11: Schematic (A) and photograph (B) of the completed system for in vivo scanning. ................................................................................... 41

Figure 12: (A) Single-element pulse echo-waveform acquired in a water tank with an aluminum reflector. (B) 128-channel beamformed reflection from target at 7 cm focus. 43

Figure 13: Real-time display view of flow in middle cerebral arteries (MCA), internal carotid arteries (ICA), and posterior cerebral arteries (PCA) using described system. Directional information of flow has been discarded. Labels have been added to indicate anatomy as well as the sectors displaying the subject’s right coronal, right transverse, left coronal, and left transverse imaging planes. ........................................................................ 44

Figure 14: (A) Coronal ultrasound rendering and (B) representative magnetic resonance (not the same subject) showing the area under examination. Original MRA image produced by Ofir Glazer, Biomedical Engineering Department Tel Aviv University, reproduced with permission of the author. .................................................................................. 46

Figure 15: Paired ultrasound rendering and dissection image indicating anatomy in vessels of the Circle of Willis. MCA=middle cerebral artery, PCA=posterior cerebral artery. Original photograph produced by Prof. John A. Beal, Department of Cellular Biology and Anatomy, Louisiana State Health Sciences Center, Shreveport. Reproduced with permission of the author. .................................................................................. 47

Figure 16: The same magnetic resonance angiogram (A) is used to indicate ultrasound field of view for the registered, rendered ultrasound data in (B) acquired without microbubble contrast agent. (C) is an enlargement of the color flow data. 48

Figure 17: Automatically registered rendering of posterior cerebral arteries from simultaneous transtemporal scans along with manually registered (yellow arrow) transforaminal scan. Vessels are paired with a dissection image (inset). No microbubble contrast agent was used. .................................................................................. 49

Figure 18: Probe placement for proposed transcranial imaging system using two matrix arrays positioned at temporal bone windows. This system has previously been used to acquire the contrast-enhanced (Definity) 3D volume shown at right [63]. 58
Figure 19: A. Volume interrogated during a typical transtemporal 3D ultrasound examination. Coronal and transverse plane views are displayed simultaneously. A steerable Doppler beam is also shown in gray. Two such volumes are acquired simultaneously with this system. B. Transmit and receive apertures used for each array in this system are shown.

Figure 20: Proposed scheme for correction of multiple isoplanatic patches in a phased array sector scan. Each arrow indicates a propagation direction of an unfocused wave aligned with the patch of the same color.

Figure 21: Definitions of update regions for single-IP (A) and multi-IP correction (B) of a single 64°×64° volume. In the multi-IP case, each color indicates the use of a unique delay map to update image lines within this portion of the volume. Traditionally, the center (red) IP has been used to update an entire phased array volume or sector scan, as in A. In this work, outer 16°×16° corners were ignored in both cases.

Figure 22: A. Electronic aberrator residual error for FWHM correlation lengths of 1.3, 2.7, and 5.4 mm for static and moving target cross-correlation, static and moving target speckle brightness [61], are compared with pitch-catch (static) cross-correlation (white bars). Mean residual for pitch-catch is 11.2 ns; the Cramér-Rao lower bound is 7.8 ns. B. Point target brightness increase in pulse-echo 3D images with varying angle between transducers with (black) and without (red) direction of arrival estimation and steering. Off-axis brightness increased by approximately 20% with D.O.A. Each data point is the average of testing 6 different aberrators.

Figure 23: Plots of inter-element correlation versus lateral distance across the receive aperture of a sparse matrix array for (A) random scatterers according to the van-Cittert-Zernicke theorem for both the aperture used in this work (solid line) and that of [59] (dashed line), (B) unsteered transmitted beacon wave measured with tissue-mimicking phantom and skull casting, and (C) transmitted beacon wave steered to 15° off-axis with tissue-mimicking phantom and skull casting. (C) is the average result of steering 15° off-axis in all four directions.

Figure 24: Maps of maximum correlation coefficient on a per-channel basis for two typical aberrators using a tissue mimicking phantom and physical aberrator (skull casting). Each ring represents a correlation on channel data received on the aperture (Fig. 20). Within each map, the center ring is always 1 (autocorrelation of unsteered beacon signal channel data), while the top, right, bottom, and left maps depict that same unsteered beacon data correlated with data acquired using a (0°,15°), (15°,0°), (0°,-15°), and (-15°,0°) steered beacon, respectively. Each channel is correlated with the same...
channel, but for a different steering angle of the transmitted wave. For these 9 aberrators, the average correlation between a channel receiving an unsteered beacon signal and one receiving a 15° steered signal is 0.788. This is well below the inter-element correlation coefficient of 0.863 at a lag of 3 mm for 15° steered data or .919 at the same lag for unsteered data (Fig. 24).

Figure 25: A. Experimental setup (not to scale) for flow measurements in a water tank in the presence of two physical aberrators. Actual probe-to-tube distance was approximately 6 cm. B. Registered, fused rendering without aberration correction. C. Rendering acquired when a single phase map is used to correct the entire 3D volume. D. Rendering acquired when five phase maps are used to correct appropriate regions of the 3D volume.

Figure 26: The first two rows show aberrated (columns A and C) and corrected (columns B and D) orthogonal echo slices acquired simultaneously by transducers positioned over the right (columns A and B) and left (columns C and D) temporal bone windows of Subject 1. Corrected volumes utilize the five-IP correction scheme; single-IP corrections were not performed on this subject. Mean brightness increases due to aberration correction were 21.2% and 24.2% for the left and right volumes, respectively. The bottom row displays the corresponding 3D rendering of echo and color flow data. Orientation for these renderings is the same as the top “axial” row of images. Blood flow shown is the C5 segment of the internal carotid artery.

Figure 27: Estimated arrival time maps for the five IPs used to perform the in vivo correction of Figure 26. Negative values indicate thick portions of the skull, as incoming signals arrived earlier than the mean arrival time (t=0).

Figure 28: Correlation lengths (top row) and RMS aberrator strength (bottom row) for the Subject 1 correction (Fig. 26) are shown for each IP.

Figure 29: Channel data from the center IP was cross-correlated with the four outer IPs to produce these maps using the in vivo data used to perform the correction of Figure 26. Mean correlation coefficients were 0.652 and 0.702 for the left and right transducers respectively.

Figure 30: Increases in mean brightness of echo data by IP for left and right transducers for the in vivo correction on Subject 1 (Fig. 26). Overall mean brightness increases were 21.2% for the left transducer and 24.2% for the right transducer.
Figure 31: Three volume renderings of echo (gray) and color flow (red) for Subject 2 demonstrating the benefits of multi-IP correction. While single-IP correction (B) increases the number of color flow voxels at the center of the scan, brightness of echo data and supra-threshold color flow voxels decrease near the edges of the volume (top and bottom of the figure). Performing a 5-IP correction (C) improves this loss of information at volume edges. In (D), an anatomical reference is shown for both the echo and CF data in the region of interest, where internal carotid arteries enter base of the skull. The top images are photographs of a human skull casting. The bottom image is coronal slice of an illustrative magnetic resonance angiogram (MRA). Original MRA image (D) produced by Ofir Glazer, Biomedical Engineering Department, Tel Aviv University, reproduced with permission of the author.

Figure 32. Three-dimensional contrast-enhanced (Definity®) renderings acquired from scanning two different healthy volunteers (both 24-year-old females) with 2.3 MHz echo and 1.8 MHz Doppler. Subject A had significantly more favorable windows than Subject B, in which the major cerebral arteries appear small and Doppler signal dropout is apparent. MCA=Middle cerebral artery, ICA=internal carotid artery.

Figure 33. Predicted system SNR as a function of skull thickness when scanning both inside and outside of the temporal acoustic window.

Figure 34: Blood to tissue backscattering ratios as a function of frequency based on the data of Shung [153].

Figure 35. Cramér-Rao lower bound (CRLB) as a function of skull thickness for two frequencies. Parameters used in these plots are: bandwidth = 50%, unattenuated SNR = 32.6 dB, \( \rho = 0.945 \), \( T = 0.65 \) µµs.

Figure 36. (A) Aperture design for 1 MHz sparse array. Gray elements only transmit; white elements both transmit and receive. Simulated two-way response of this aperture (B) on-axis, (C) steered to \( +16^\circ \) in both azimuth and elevation, and (D) steered to \( +32^\circ \) in both azimuth and elevation. These beams are virtually identical to previous 2.5 MHz simulations (Fig. 8)[63].

Figure 37: (A) Optical photograph and (B) micrograph of an array after dicing. Gold pads extending beyond the PZT in (A) are extra pads used only for alignment. Element pitch is 0.725 mm.
Figure 38. Simulated pulse-echo impulse response (A) and spectrum (B) for a single element using PiezoCAD. The units on the ordinate axes are measured output voltage ($V_o$) relative to applied input voltage ($V_i$).

Figure 39. (A) Typical single element pulse-echo response and (B) average spectrum from all single-element acquisitions. The simulated spectrum of Fig. 38 is reproduced for comparison (dashed line).

Figure 40. Results of three separate tests of water tank flow testing in the present of 3 mm Wallgone rubber using two different sets of transducers at the indicated frequencies. Attenuation is 28.8 dB at 1.2 MHz and 43.2 dB at 1.8 MHz.

Figure 41. Fused renderings of the same subject using the described 1.8 MHz (A) and 1.2 MHz (B and C) imaging system. Coronal views are shown in (A) and (B) and an axial view in (C). Both data sets were acquired outside of the temporal acoustic window. For 1.8 MHz imaging, we were unable to find a threshold setting which allowed visualization of anatomical structures. In the 1.2 MHz image, anterior horns of lateral ventricles, sphenoid bone, and contralateral skull surfaces are visible.

Figure 42. Registered, fused rendering of a healthy 34-year-old male with Definity® microbubble contrast enhancement. (ACA = anterior cerebral artery, ICA = internal carotid artery, MCA = middle cerebral artery, PCA = posterior cerebral artery.)

Figure 43. An example of unilateral imaging failure in one subject, a 32-year-old African-American male. Imaging with the low frequency array does enable imaging through the right side of the subject’s skull outside of the temporal acoustic window, but not through the left side. (ACA = anterior cerebral artery, ICA = internal carotid artery, MCA = middle cerebral artery, PCA = posterior cerebral artery.)

Figure 44: Pulse-echo on-axis beams for alternative configurations of splitting the scanner channels including (A) between three transducers without Rx switching, yielding 85-receive element transtemporal arrays, and (B) between three transducers with Rx switching, yielding 192-receive element transtemporal arrays. (C) The array used in Chapter 2 is shown as a reference.

Figure 45: Comparison between pressure fields at z=14 cm due to unfocused transmitted beacon wave used in experiments (red line) and three experimental phasings for a wave steered to -15° in azimuth. The white boxes outlined in black indicate the position of the receiving array. At this location, the pressure from the phasing represented by the black line is 6 dB higher than that of the red line.
Figure 46: Difference images—the result of subtracting the aberrated from the corrected image—for the 1 IP (left) and 5 IP (right) corrections on the same subject acquired at two different sessions. While a stronger aberrator was clearly corrected on the right, the increase in brightness at the edges of the volume on the right is of greatest interest, as shown in Fig. 47. ................................. 136

Figure 47: Increase in brightness on a per voxel basis for four 2D slices (one transverse, one coronal for each transducer) through 3D volumes. Data were acquired on the same subject but at different times. The 5-IP correction technique removes the severe drop-off in brightness increase at large lateral distances................................. 137

Figure 48: KLM circuit model of ultrasound transducer. ........................................... 145

Figure 49: Simulated jitter (markers) and CRLB (continuous line) (A) with varying SNR and (B) with varying $\rho$............................................................... 154

Figure 50: Control (top), aberrated (middle), and corrected (bottom) 1-way beam profiles are shown at left with the corresponding pulse-echo C-scan image of a point target shown for each using the head of a pin as a target. C-scans are taken from the scanner’s real-time display. Target brightness is visibly recovered in going from the aberrated to the corrected case. ......................................................................................................................... 156

Figure 51: Aberrated (top) and corrected (bottom) pitch-catch beams for a skull casting measured to induce aberration of 72 ns RMS. The clutter pedestal is reduced by approximately -4 dB and beam width narrows by approximately a factor of 2........... 157

Figure 52: In the conceptual diagram of scattering volumes (top row), the scattering volume in the spatial frequency domain is the product of three weights: the transmit field $\Psi_T$ (gray), the receive field $\Psi_R$ (white), and the axial window $w$ (black). The image scan line corresponding to each scattering volume is shown on the bottom row (2D slice shown for simplicity). Three cases are shown: (A) pulse-echo acquisition near the center of the phased array sector, (B) pulse-echo acquisition near the edge of the sector, and (C) pitch-catch acquisition. The open circle on each scan line indicates that a single scattering volume applies to specific region in an image, i.e. one scan line and focal zone. This figure is adapted from Fig. 1 of [102]................................................................................. 161
Acknowledgements

I would like to thank Steve for taking me on as a graduate student and for his mentoring and support over the time it has taken me to complete this research. I would like to thank Christine for all of her support while in graduate school. I would also like to thank parents as well as Steve and Christi Hinkle for their support during this time in my life. Ned Light was my friend, mentor, and first test subject. None of this would have happened without Ned, who taught me how to build transducers and many of the day-to-day lab skills I now take for granted.

Thank you to the other members of the Smith lab during my time here: Nik Ivancevich, who instructed me in the ways of the Volumetrics scanner and phase correction, and was an enormous help with both research and non-research issues on a daily basis; Carl Herickhoff, who helped me with transducer and circuit board fabrication questions and was extremely supportive as I tried to get through my prelim; Matt Fronheiser, who invented many of the essential scanner techniques used in my work such as the ability to scan with two transducers at once. In addition, I also owe a debt of gratitude to previous grad students whose dissertations I have depended on greatly at various times, including Stockton Miller-Jones, Rod Gauss, Anna Fernandez, David Mills, and Jesse Yen, as well as Matt, Nik, and Carl.
I also would like to thank my committee members, Gregg Trahey, Jeremy Dahl, Rebecca Willett, and Danny Laskowitz. They have each made major contributions to the content of my work, how to best present it, and how to think and formulate a plan as a scientist.

I owe a special thanks to Danny Laskowitz, Heather Nicolletto, and Ellen Bennett for all their collaboration and making clinical studies possible. Lance Boland, Karen Strub, John Toptine, and Brenda Sedberry were also of great help before and during studies.

I also received technical advice and support from a great many people. Francois Vignon and Gianmarco Pinton provided helpful suggestions and conversations related to transcranial ultrasound over the years. John Castellucci was extremely helpful for all matters relating to Volumetrics scanners and analog circuits. Matt Gasell was helpful and patient with me during PCB layout, even when I didn’t know what I wanted.

Thank you to the other members of the Trahey and Nightingale labs for their assistance and support over the years, and for continuing to put up with me, especially my officemate, Samantha. Thank you to Pat Wolf and Ellen Dixon-Tulloch for allowing me to work in their space at Teer. Thank you to Kathy Barbour, Joyce Franklin, and Susan Story for their kind and willing help in departmental matters great and small over the years. Thanks to Gerald Harris at FDA for helpful conversations involving
hydrophone calibration and safety measurement and for providing Walgone rubber attenuation data.

Finally, I would like to thank the professors, teachers, and mentors who helped me arrive at graduate school: Tim Nixon, Kerry Broyles, Jim Gravina, Phyllis Chatlos, Mike Scott, Ulas Coskun, Tyler Farabi, Prof. Brian Cunningham, and Prof. Craig Koslofsky.
1. Introduction

1.1 Background and significance

While one-dimensional transcranial Doppler (TCD) has been used to non-invasively measure blood flow velocities in cerebral arteries since the 1960s [1], transcranial imaging with color flow has emerged slowly over the following decades as a clinical diagnostic tool [2, 3]. Ultrasound is a rapid, cost-effective modality capable of assessing conditions impacting cerebral vasculature including transient ischemic attack, hemorrhage, arteriovenous malformation, and sickle-cell induced stroke. Given an aging population and the high incidence of stroke in the United States [2] and other nations [5-7], providing rapid diagnostic imaging of stroke is essential. If a stroke can be determined to arise from the presence of a thrombus or embolus within a specified time window—currently 4.5 hours in the U.S. [3]—tissue plasminogen activator (tPA) can be administered intravenously to break up the thrombus, decreasing recovery times and improving patients’ functional outcomes [9].

1.1.1 Stroke

Stroke constitutes a significant public health concern [2] and is recognized by the World Health Organization as the second-leading cause of death globally and the leading cause in middle-income nations [4]. “Stroke” is a general term for loss in brain function due to reduction in blood flow which may be broken down into two broad categories, ischemic and hemorrhagic stroke. Ischemic stroke comprises approximately
87% of all stroke cases [10] and is the type of greatest interest in this work due to its predominance. There are three important relative anatomical regions during a stroke: the core, or region of immediate tissue infarct, the penumbra, a larger region surrounding the core in which brain tissue may be saved if it can be reperfused within a few hours, and the oligemia, a region surrounding the penumbra with only mildly reduced blood flow velocity [5]. In addition to the importance of rapid stroke treatment, collateral circulation is also very important, as these vessels offer an alternate route for cerebral blood flow [12, 13]. Wessels et al. have demonstrated the ability to detect blood flow in intracranial collateral arteries with contrast-enhanced 3D ultrasound imaging using a non-real-time system, and recommend 3D ultrasound as a faster and clinically acceptable alternative to X-ray or magnetic resonance angiography in the hospital setting [14].

Before arriving at the hospital, early responders are taught to rely on one of several standard rapid assessment techniques for recognizing a stroke. These include but are not limited to FAST (face, arm, speech, time) [6], the Los Angeles Prehospital Stroke Screen (LAPSS) [7], or the Cincinnati Prehospital Stroke Scale (CPSS) [8]. Once in the emergency room, a medical history and physical examination are used to form an initial diagnosis. Imaging is used where available, especially computed tomography (CT) to rule out hemorrhage [18, 19].
1.1.2 Stroke imaging

In the case of an ischemic stroke, intravenous administration of recombinant tissue plasminogen activator (r-tPA) has been shown to improve patient outcomes [9, 20]. However, in order to administer tPA to a patient, it is essential to rule out brain hemorrhage, as tPA would exacerbate any bleed, likely resulting in fatality. This is most commonly done using a CT scan [18]. While ischemic stroke accounts for 87% of all stroke cases [10], only 1 to 7% of eligible patients actually receive tPA [21-23] primarily because the patient is not diagnosed quickly enough or because symptoms are considered to be too mild [9], opening the discussion for alternative means of determining the underlying nature of a stroke.

In addition to ultrasound, alternative modalities for cerebral functional imaging include perfusion CT, CT angiography, PET/SPECT imaging, and magnetic resonance imaging (MRI) with arterial spin labeling (ASL). While conventional CT is used to eliminate a diagnosis of hemorrhagic stroke, it may not confirm a diagnosis of ischemia. Perfusion CT seeks to rectify this with the use of contrast agent and is capable of displaying maps of cerebral blood flow, cerebral blood volume, or mean transit time. It should be noted that these maps are computationally intensive, as they require a deconvolution operation to account for the time-varying injection of the contrast agent. The clinical utility of perfusion CT has not yet been widely accepted.
CT angiography relies on contrast agent to produce a 3D image of major arteries. The pre-contrast volume may be subtracted from the post-contrast volume, resulting in images of the arteries alone. CT angiography may add only a few minutes to the duration of a standard CT exam, however interpretation of the resulting images is time-consuming and requires a specialist.

Nuclear medicine imaging techniques (especially SPECT) have been widely used in neurological exams because they provide functional information. However, they do not provide anatomical information and generally lack the resolution to definitively diagnose and localize ischemia, meaning a CT or MRI must also be performed [10]. The development of new radiotracers and the increased presence of combined SPECT/CT machines may increase the feasibility of using SPECT in cerebrovascular disease in the future.

In MRI with arterial spin labeling (ASL), water molecules in arterial blood are magnetically labeled just below the region of interest by applying a 180° inversion pulse. After flowing into the slice of interest, this blood water exchanges with tissue water, reducing MR signal in this tissue and forming the “tag” image. A control image is then acquired without arterial spin labeling. When the control image is subtracted from the ASL or “tag” image, the result is a tissue perfusion image [11]. ASL has the advantage of not requiring contrast agent. Like perfusion CT, MRI with arterial spin labeling is not widely available and a consensus has yet to be reached regarding the significance and
best use of its results. Diffusion- and perfusion-weighted MRI are more common sequences which can provide imaging of the penumbra [5]. Due to high cost, slow workflow, need for patient cooperation, and lack of availability, MRI imaging of stroke does not is not presently feasible for most hospitals [12].

Ultrasound imaging of the common carotid artery in the neck is widely used to screen for atherosclerosis—a contributing factor to ischemic stroke—and to assess atherosclerotic plaques [13], which may produce an ischemic stroke if a plaque ruptures and forms a blood clot on its surface [14]. Randomized clinical trials have demonstrated the efficacy of carotid endarterectomy in stroke prevention in both symptomatic and asymptomatic patients [15-17]. Recently, researchers have investigated moving beyond traditional B-mode or B-mode + Doppler imaging to characterize carotid plaques using acoustic radiation force, non-invasively analyzing the stiffness and heterogeneity of plaques [33-37]. Van der Steen et al. have previously demonstrated the ability characterize plaques using moderately invasive intravascular ultrasound (IVUS) imaging [38-40]. If vulnerable plaques—those likely to rupture—could be distinguished non-vulnerable plaques, then risky, unnecessary surgical procedures might be avoided.

Transcranial Doppler (TCD) examinations—typically one-dimensional, non-imaging studies—have been performed clinically since the 1980s to non-invasively measure blood flow velocities in intracranial arteries [18]. TCD is used clinically to diagnose right-to-left shunts [19], vasospasm after subarachnoid hemorrhage [43, 44],
and risk of stroke in sickle cell disease [45], among other conditions. In the United States, approximately 36,000 people with sickle cell disease between ages 2 and 16 undergo two TCD exams per year [20]. Two-dimensional transcranial ultrasound imaging examinations are not commonly performed in the United States, but are more common in other countries, especially Germany [47, 48].

1.1.3 Stroke treatment

Once a diagnosis of ischemic stroke has been made, limited treatment options are available. As mentioned previously, tissue plasminogen activator (tPA), which is manufactured by Genentech under the name Alteplase or Activase, is FDA-approved for treatment of acute ischemic stroke. In the United States, tPA may be given intravenously within the first 4.5 hours after the onset of stroke symptoms [49, 50]. tPA can be effective when hemorrhagic stroke can be ruled out [23], but also poses a risk of intracerebral hemorrhage [22].

In patients who are ineligible or have failed tPA therapy, an alternative means of stroke treatment is to physically remove the clot via surgery or with a mechanical thrombectomy device [21]. As of 2010, there were 20 such known devices [21]. While some ischemic stroke patients respond well to these treatments [52, 53], these procedures have their own complications and each patient’s outcome depends heavily on the degree of collateralization [54].
Recent research studies have investigated the use of ultrasound as a means of thrombolysis either alone [55-57], with microbubbles [58, 59], or with tPA [60, 61]. While ultrasound thrombolysis has shown promise, there are safety concerns associated with the low frequency (~300 kHz), high intensity, large focus, and stationary (non-swept) beam used in these treatments which must be overcome before this technology may be used clinically [62, 63]. It is hoped that over time, improvements in and increased access to imaging technology—ultrasound or otherwise—will provide clinicians with more information, including the presence of intracerebral hemorrhage and evaluation of collaterals, allowing for more informed treatment planning.

1.2 Transcranial ultrasound imaging

1.2.1 Real-time and portable real-time ultrasound

The first ultrasound system producing electronically steered real-time two-dimensional images, the Electrosan I, was demonstrated in 1967 in Utrecht, the Netherlands. Interestingly, one of the primary uses of early real-time 1D systems was to produce A-scans of the brain to detect midline shifts in the event of a tumor or hematoma [22]. By 1969, the Electrosan I had produced real-time 2D images of the brain and heart. The first portable ultrasound system sold in the United States was released in 1973 by Advanced Diagnostic Research Corporation. It was a 64-element linear array and used an oscilloscope for its non-imaging display [23]. In 1976, von Ramm and Thurstone demonstrated a real-time phased array system for cardiac
imaging in which a PDP-11 computer was used to switch the delay lines [24]. A real-time 3D phased array system using a matrix array was demonstrated by von Ramm and Smith in 1991 [25, 26]. The first large-production battery-powered portable ultrasound scanner, the Sonosite 180, was released in 1999, with the rival Acuson Cypress following in 2000. As of 2012, portable ultrasound imaging systems such as the SonoSite MicroMaxx, Terason t3000, GE Voluson i, Siemens Acuson P50, and Philips CX50 provide high resolution 2D imaging in a form factor just larger than a laptop computer. In July of 2012, Philips became the first manufacturer to demonstrate portable real-time 3D ultrasound, which it did using its CX50 platform and a transesophageal probe. Such a matrix array system would be able to provide the benefits described in this dissertation.

1.2.2 The temporal acoustic window and transskull imaging

The quality of transcranial ultrasound images is limited by the skull’s attenuation and aberration of ultrasound waves. The discovery [18] of a thinner [27], less aberrating region of the temporal bone measuring 2-3 cm in diameter provided a window for insonification of the internal carotid arteries and the arteries of Circle of Willis. Bone within this region—the temporal acoustic window—is comprised mainly of compact bone ($\alpha \approx 2.8 \text{ dB cm}^{-1} \text{ MHz}^{-1}$) and is more uniform in thickness and composition than the cancellous bone ($\alpha \approx 25$ to 70 dB cm$^{-1}$ MHz$^{-1}$) [28] surrounding the window. Thus imaging through the temporal acoustic window reduces the aberration and attenuation
induced by the skull, although locating and imaging via this window requires great skill. In addition, microbubble contrast agent is often needed to enhance the flow signal. Because window size and composition vary greatly among individuals [27], transducer design for transcranial imaging remains an unsolved problem which will be discussed in this dissertation.

Since aberration is quite strong even in the window, in recent years many groups have presented techniques for correcting the wavefront aberration due to the human skull. Our own group has previously used the multi-lag least-squares cross-correlation algorithm [71-73] to demonstrate the first successful in vivo transcranial phase corrections on a matrix array [74, 75]. Other techniques improving transskull focusing in diagnostic or therapeutic applications include using time-reversal focusing to correct on a point target or induced cavitation bubble [76-79], computing element delays based on MRI or CT mapping of skull structure [80-83], using shear mode conversion to lessen the effects of aberration due to the similarity between shear wave velocity in the skull (1400 m/s) and longitudinal wave velocity in brain tissue (1530 m/s) [29, 30], and making use of a contralateral source [31, 32]. These approaches improve contrast and spatial resolution in imaging through the intact skull and may be extended to transskull imaging with matrix arrays in the future.

Real-time three-dimensional ultrasound has been realized using matrix array transducers [26] and parallel receive beamforming [25] and has demonstrated diagnostic
benefits over 2D ultrasound in certain applications [33, 34]. In transcranial ultrasound imaging, vessels under examination are tortuous and prone to anatomical variants [5], making it difficult to visualize blood flow in a single imaging plane. Thus real-time three-dimensional (RT3D) ultrasound provides a means of capturing a more complete view of a vessel’s course [35].

1.2.3 Real-time 3D ultrasound

Real-time three-dimensional ultrasound using a matrix array transducer and parallel receive beamforming were invented by Smith and von Ramm [25, 26]. When using this approach, real-time 3D ultrasound is limited by physical constraints on its spatial resolution and signal-to-noise ratio (SNR) relative to two-dimensional ultrasound. Specifically, each matrix array element suffers from a poor electrical match with the cable it drives during echo reception. When electroded on two sides, each piezoelectric element becomes a capacitor having a capacitance on the order of 1-10 pF, while a typical system cable connecting the probe to the scanner is approximately 2 m in length and has a capacitance on the order of 90 pF/m. As the source (element) and load (cable) impedances are poorly matched, power transfer is poor while noise sources (thermal, preamp) are unaffected, resulting in a signal-to-noise ratio that is severely diminished from the optimum case [36]. This matter may be further complicated by the design of the preamplifier, which is in turn driven by the cable [37]. The issue of impedance mismatch has been addressed by Goldberg and Smith by fabricating an
acoustic stack comprised of elements which are in parallel electrically and in series acoustically [38]. In this work, we discuss improving impedance matching by replacing a lengthy cable with a much shorter flex interconnect (Chapter 2).

In parallel receive beamforming, multiple receive beams are formed from within a single slightly broadened transmit beam, increasing the width of the point spread function [25]. In the case of the Volumetrics Model 1, 16 receive beams are formed for each transmit beam, though other scanners have higher ratios. For example, the Duke T5 scanner uses 32:1 parallel processing [39] and the Siemens SC2000 uses 64:1 parallel processing [40]. In the Model 1, parallel processing is performed simultaneously in 256 mixed-signal application-specific integrated circuits (ASICs), which delay digitized channel signals based on 10 million values (256 elements × 256 transmit beams × 16:1 parallel processing × 10 focal zones) stored in look-up tables, then convert signals back to analog for summation. In many newer scanners, partial sums are formed in ASICs located in the probe’s handle, reducing the number of connections required in the probe cable. This approach generally uses impedance matching circuitry, preamplifiers, and analog-to-digital converters (A/Ds) in the in-handle ASICs to achieve adequate SNR [36, 41].

1.2.4 Registration and rendering of 3D ultrasound

The nature of 3D ultrasound data presents unique challenges to the visualization of volumetric data sets, as rendering—display of 3D data on a 2D screen—must be
performed in order to simultaneously view an entire data set. Tasks such as segmentation and rendering become difficult in the presence of speckle and blurred boundaries of varying intensity levels [42, 43]. Approaches to this task commonly optimize a measurement of correspondence between a reference volume and a test, or homologous, volume. Registration of multimodality scans of the brain has previously been performed using intraoperative ultrasound volumes acquired with the skull removed [99-101], however to our knowledge ours is the first attempt at registration of transcranial ultrasound volumes of the brain.

Metrics that have been used in the context of 3D ultrasound registration include normalized cross-correlation [98, 102, 103], mutual information [44, 45], and local orientation and phase [46]. Optimizing mutual information (MI) is a popular approach in the image processing community and has demonstrated rapid performance for affine and elastic transformations in medical ultrasound [45, 47, 48]. The use of normalized cross-correlation as a voxel similarity metric has been shown to provide measures of registration error in volumetric ultrasound data sets similar to MI as a function of both noise level and degree of deformation [45]. In addition, the skull-encased brain is a highly rigid organ, so the need for elastic or shearing transformations would not seem to be as great as in registration of abdominal or thoracic ultrasound images [48].
1.2.5 Multi-array ultrasound imaging

Since the invention of real-time three-dimensional ultrasound, few attempts have been made to simultaneously image using multiple transducers in real-time. Fronheiser et al. first acquired two real-time three-dimensional volumes to aid in RF ablation with a 1 second delay to switch between transducers [49]. In this work, the scanner’s channels and image lines are split to enable simultaneous imaging from two matrix array transducers.

1.2.6 Transcranial ultrasound transducers

Because transcranial ultrasound imaging is not widely used clinically at the present time, the cost of fabricating array transducers designed specifically for transcranial imaging has not been warranted. Thus the transducer array used most commonly for transcranial ultrasound imaging has been a cardiac array with an aperture designed to fit between the ribs of an adult. Conveniently, this distance is similar to the reported diameter of the temporal bone acoustic window (2-3 cm) [27].

While the potential of therapeutic transcranial ultrasound has recently provoked new arrays for high-intensity focused ultrasound [110-112], combined imaging-therapy applications [50], and sonothrombolysis [51], two-dimensional or matrix arrays for transcranial imaging have received less consideration, although some manufacturers market muti-purpose arrays that include transcranial imaging among their indicated uses.
Even linear array transducers have rarely been designed for transcranial imaging. During the course of this research, two known cases of linear array designs have been published. In the first, Azuma et al. (Hitachi Corporation) pursued a 2 MHz linear array for imaging with a 500 kHz thrombolysis array lying beneath the imaging array in the same probe housing [50]. In the second, a 128-element, 750 kHz linear array was fabricated and used to image a phantom through an ex vivo skull [52]. No known attempts at matrix arrays for transcranial imaging have been made other than our own.

1.2.7 Phase aberration correction

The degrading effect of skull aberrations on image quality has been investigated since the 1970s [53-57], with Miller-Jones the first to propose the cross-skull pitch-catch estimation technique investigated in Chapter 3 of this dissertation [31]. Real-time adaptive phase correction was first demonstrated in 1990 [58]. In the skull, time reversal and inverse filtering techniques have been investigated extensively by Fink’s group, often requiring an acoustic source inside of the head [76, 121-123]. In 2006, Vignon et al. presented the migrated spatiotemporal inverse filter, an extension of this work that no longer requires an intraskull receiver or transmitter [32]. Another approach has been computation of focal delays based on a computed tomography scan for either trans-skull imaging or therapy [80, 82, 124]. For any adaptive imaging technique, the time required to compute a complete set of beamformer updates and the bandwidth of the approach are essential considerations. Most phase correction techniques for imaging which have
been implemented in vivo have used an ultrasound-based estimation technique followed by an update of imaging delays, a broadband approach. In our lab, Ivancevich has previously implemented 3D phase correction for real-time transcranial imaging using speckle targets, showing encouraging in vivo results [59-61].

1.3 Objective

The efforts presented in this dissertation demonstrate the usefulness of real-time three-dimensional transcranial ultrasound, particularly simultaneous imaging from both temporal acoustic windows. Several prototypes of such an imaging system have been constructed and demonstrated, including a 2.3 MHz system which simultaneously scans via both temporal acoustic windows and fuses acquired color flow volumes, a system which uses each transducer as beacon to correct skull-induced imaging aberrations in the images of the opposite array, and a 1.2 MHz system for scanning outside of the temporal acoustic windows. A portable system such as one of these could potentially be used in early diagnosis of stroke patients, allowing an operator with minimal training to acquire clinically relevant 3D ultrasound volumes in a field or ambulance setting and transmit them to the hospital for review before the patient arrives. This could accelerate treatment planning, ultimately leading to increased use of tPA and increasing positive functional outcomes among stroke patients. Three significant obstacles to 3D transcranial ultrasound will be addressed in the presented research: (1) poor sensitivity
(SNR) due to attenuation, (2) degraded contrast and spatial resolution due to aberration, and (3) visualization of the major arteries of the brain in 3D.

1.4 Summary of chapters

In this work, we examine the problem of pre-hospital and emergency room imaging of stroke and present ultrasound-based approaches for addressing it. Specifically, this dissertation discusses obstacles of aberration and attenuation due to the skull and presents a prototype system for simultaneous 3D bilateral imaging via both temporal acoustic windows. In Chapter 1, we introduce the basics of stroke and transcranial ultrasound, and also discuss current diagnostic protocols and the role of imaging.

In Chapter 2, we present a system capable of offline registration and fusion of 3D transcranial ultrasound volumes acquired from two sides of the head. We examine hardware considerations for the new matrix arrays in this system—transducer design and interconnects—and propose that SNR may be increased by reducing the length of probe cables. This claim is evaluated as part of the presented system through simulation, experimental data, and in vivo imaging. Ultimately, gains in SNR of 7 dB are realized by replacing a standard probe cable with a much shorter flex circuit interconnect. In vivo images are presented depicting cerebral arteries with and without the use of microbubble contrast agent and have been registered and fused using a simple
search algorithm which maximizes normalized cross-correlation (Fig. 1). Section 2.1 of Chapter 2 was published in *Ultrasound in Medicine and Biology* under the title "The ultrasound brain helmet: Feasibility study of multiple simultaneous 3D scans of cerebral vasculature" [62]. The remaining sections of Chapter 2 were published in *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control* under the title “The ultrasound brain helmet: new transducers and volume registration for *in vivo* simultaneous multi-transducer 3-D transcranial imaging” [63].

![Coronal plane](image)

**Figure 1**: (A) Coronal ultrasound rendering and (B) representative magnetic resonance (not the same subject) showing the area under examination. Original MRA image produced by Ofir Glazer, Biomedical Engineering Department Tel Aviv University, reproduced with permission of the author.

In Chapter 3, the scanning geometry of brain helmet system is utilized to allow each matrix array to serve as a beacon to correct aberrations induced by the skull
adjacent to the opposing array. Aberration is estimated using cross-correlation of RF channel signals followed by least mean squares solution of the resulting overdetermined system. Delay maps are updated and real-time 3D scanning resumes. A first attempt is made at using multiple arrival time maps to correct multiple unique aberrators within a single transcranial imaging volume, i.e. several (five) isoplanatic patches. In phantom experiments, color flow voxels above a common threshold have increased by an average of 92% while color flow variance decreased by an average of 10% as a result of estimating and correcting five unique aberrators per volume. The result of a phantom experiment is shown in Fig. 2. This approach has been applied to both temporal acoustic windows of two human subjects, yielding increases in echo brightness in 5 isoplanatic patches, with a mean value of 24.3 ± 9.1%. The contents of this chapter have been accepted for publication in *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control* under the title “Pitch-catch phase aberration correction of multiple isoplanatic patches for 3D transcranial ultrasound imaging.”
Figure 2: A. Experimental setup (not to scale) for flow measurements in a water tank in the presence of two physical aberrators. Actual probe-to-tube distance was approximately 6 cm. B. Registered, fused rendering without aberration correction. C. Rendering acquired when a single phase map is used to correct the entire 3D volume. D. Rendering acquired when five phase maps are used to correct appropriate regions of the 3D volume.

In Chapter 4, window failure and the possibility of overcoming it using a low frequency, large aperture array are examined. Chapter 4 follows the direction of inquiry indicated by Chapter 2. In performing transcranial ultrasound examinations, 8-29% of patients in a general population present with window failure, in which it is not possible to acquire clinically useful sonographic information through the temporal acoustic window. The technical considerations, design, and fabrication of low-frequency (1.2 MHz), large aperture (25.3 mm) sparse matrix array transducers for 3D imaging in the event of window failure are described. These transducers are directly compared with arrays operating at 1.8 MHz in a flow phantom with approximately 47 dB/cm^{0.8}/MHz^{0.8} attenuators. Contrast-enhanced in vivo imaging through the intact skull outside of the
temporal acoustic window allows visualization of the Circle of Willis (Fig. 3). This relatively modest decrease in transmit frequency from 1.8 to 1.2 MHz produces appreciable improvement in both echo and color flow imaging through thick skulls. The decrease from approximately 2 to 1 MHz for 3D transcranial ultrasound may be sufficient to allow acquisition of useful images either in individuals with poor windows or outside of the temporal acoustic window by untrained operators in the field. Chapter 4 has been accepted for publication in Ultrasound in Medicine and Biology under the title “Simultaneous bilateral real-time 3D transcranial ultrasound imaging at 1 MHz through poor acoustic windows.”
Figure 3. Registered, fused rendering of a healthy 34-year-old male with Definity® microbubble contrast enhancement. (ACA = anterior cerebral artery, ICA = internal carotid artery, MCA = middle cerebral artery, PCA = posterior cerebral artery.)

Chapter 5 suggests opportunities for future work in light of the presented research in areas including transducers, system design, aberration correction, and window failure.
2. Ultrasound brain helmet: real-time 3D multi-transducer imaging system

2.1 Introduction

2.1.1 Overview

The motivation behind the brain helmet system is to provide simultaneous bilateral imaging via both temporal acoustic windows, eventually in a portable scanning system. Bilateral imaging acquires images having higher SNR at depth compared to single-transducer imaging, in which attenuation greatly reduces SNR with increasing depth. It also allows two acquired volumes to be fused and registered, allows for comparison of bilateral blood flow and computation of an asymmetry index [64], and still provides imaging in the event of unilateral window failure. In constructing the prototypes presented here, we have modified an FDA-approved clinical scanning system, the Volumetrics Model 1.

2.1.2 Volumetrics 3D Scanner

The presented experiments were performed using the Volumetrics Model 1 scanner (Volumetrics Medical Imaging, Durham, NC), a real-time 3D ultrasound system using 16:1 parallel receive processing [25, 26]. This system has 256 channels that only transmit and 256 shared transmit/receive channels. A two-dimensional phased array transducer typically transmits (256) broadened beams in succession. For each transmit beam, 16 receive beams are formed from echoes arranged in a $4 \times 4$ pattern at positions slightly off-axis from the center of the transmit beam, with a transmit beam spacing of
4° in a typical 64° × 64° scan. The scanner is capable of frame rates up to 30 volumes/sec and typically displays the following views in real-time: one slice in azimuth, one in elevation, and two C-scans parallel to the face of the transducer. It also features 3D color flow imaging, pulsed spectral Doppler, and M-mode capabilities.

**Figure 4: Volume interrogated during a typical transtemporal 3D ultrasound examination. Coronal and transverse plane views are acquired simultaneously. A steerable Doppler beam is shown in blue.**

To enable the brain helmet, we have split the scanner’s channels and also its 4096 image lines equally between two matrix arrays so that two 64° × 64° pyramidal volumes may be acquired simultaneously. Each matrix array can now have a maximum 128 elements that only transmit and 128 shared transmit/receive elements. As before, for each transmit beam, 16 receive beams are formed from one of the two transducers. However, transmit beam separation has now doubled in the elevation direction, so transmit beams in each 64° × 64° volume are now spaced at 4° in azimuth and 8° in elevation. After modifying the scanner’s real-time display, four slices from two probes
are now displayed in real-time: one azimuth and one elevation slice from each transducer, corresponding to coronal and transverse planes in transtemporal imaging (Figure 4). The operator uses a trackball control to select any four slices (two from each transducer’s volume) for display. At the push of a button, system delays can be re-loaded from memory to allow single-transducer scanning from either of the two probes, improving spatial sampling by restoring all 4096 image lines to a single transducer. 3D color flow and spectral Doppler capabilities have been retained in both single and dual-transducer scanning modes.

2.1.3 Motivation and initial prototype

Having sequentially acquired meaningful 3D volumes in previous human studies from both temporal acoustic windows and the suboccipital window, we were inspired to build a system capable of acquiring all three simultaneously. The systems presented here have not been constructed to include the third suboccipital transducer in order to avoid further splitting the image lines, though sequential suboccipital images have sometimes been acquired using all image lines on a single transducer (Fig. 18).

The first prototype used a hand-wired conversion box to map two commercial arrays (Volumetrics) to the scanner, acquiring the flow phantom images of Fig. 6. The hand wiring was then replaced with a custom printed circuit board, acquiring the phantom and contrast-enhanced (Definity, Lantheus Medical Imaging, N. Billerica, MA) \textit{in vivo} images of Fig. 7.
Figure 5: Experimental setup (left) and resultant 3D rendering (right) acquired using initial hand-wired prototype. Red and blue indicate two different acquiring transducers; no automated registration was used, only manual registration.

Figure 6: (A) Resultant B-mode + color flow scans and (B) 3D rendering using printed circuit board prototype. Experimental setup is as shown in Fig. 5A.
Figure 7: The first in vivo acquisition using the brain helmet, these images show blood flow in both internal carotid arteries (A) in a screenshot from the scanner’s real-time bilateral B-mode + color flow display, and (B) in offline, manually fused renderings of the same data.

2.1.4 Prototype with custom transducers

After acquiring initial in vivo data (Fig. 7) we sought to increase SNR and to automate the offline rendering display. The remainder of this chapter details these efforts towards building custom matrix array transducers and registering acquired volumes.

2.1.5 Cabling and transducers

The technical advantages and disadvantages provided by matrix array transducers are well known. In brief, while a 2D or “matrix” array probe is capable of acquiring an entire three-dimensional volume in real time when used with parallel
receive processing (as discussed in Section 1.2.3), each matrix array element suffers from a poor electrical match with the cable it drives during echo reception. Given the length dependence of the cable capacitance, it was proposed that for certain applications such as transcranial imaging, the lengthy cable may be replaced by a short flexible multilayer circuit, or “flex,” on which the probe is fabricated, thus improving matching and increasing receive sensitivity.

In proposing reduction of cable length as a practical modification for increasing SNR, we have thus far discussed impedance matching as though each trace were a simple circuit. While circuit analysis predicts the effect of varying the impedances of the element, cable, and preamp on the amplitude of the signal at the preamp, this model obviously oversimplifies the behavior of the cable by treating it as a lumped element rather than a distributed element, i.e. a transmission line. For this reason the software package PiezoCad (Sonic Concepts, Bothell, WA) was used to model a piezoelectric element with varying cable loads using the KLM model [65]. For the control case, these simulations were performed using the actual specifications of an existing Volumetrics 2.5 MHz sparse matrix array as input parameters [66], including its actual cable length of 2.23 m. A typical cable capacitance was determined from manufacturer’s specifications to be 88.6 pF/m (Tyco Electronics, Berwyn, PA). The actual input impedance of the scanner’s preamplifiers was also used. For the test case, the cable length was set to the length of the shortest flex circuit capable of extending from the scanner to a patient’s
temporal acoustic window, approximately 0.30 m. Capacitances for both the multilayer flex circuit and the FR-4 printed circuit board were computed using microstrip equations as in [67], and depend on the trace dimensions and dielectric materials. The total capacitance for a typical trace traversing both the flex and printed circuit board were found to be 66.4 pF. Simulated pulse-echo voltages were compared for the two cases.

We have previously examined the effect of reducing interconnect length while maintaining a common cable construction both in simulations and experimentally [68]. These results showed a simulated 16 dB and measured 14 dB increase in SNR due to decreasing the length of MicroFlats cabling (W.L. Gore, Pleinfeld, Germany) from 2.87m to 0.40 m. It should be noted that the inter-trace dielectric most commonly used in flex circuits is polyimide, which has a relative dielectric constant ($\varepsilon_r$) of approximately 3.4 [67]. Griffith and Lebender estimate that for MicroFlats-type cable, $\varepsilon_r \approx 1.75$ [69]. However, for this project the manufacturability and mechanical stability provided by the polyimide flex circuit outweighed the impedance-matching benefits of a ribbon-based probe cable.

2.1.6 Aperture design

In designing the initial transcranial probe, it was assumed that the entire footprint of the array should fit within a patient’s temporal acoustic window, generally 2-3 cm in diameter but varying widely among individuals [70]. If the aperture partially
exceeded the window, signals received on this portion of the array would experience much greater attenuation given the aforementioned differences in bone structure inside and outside of the window. This drastic signal dropout over part of the array would degrade signal-to-noise ratio of the beamformed signal. Vignon et al. underscored the importance of placing the probe within the window by demonstrating a 20 dB improvement in image brightness at 3.2 MHz when beamsummed data from a contralateral source were used to optimize probe placement on ex vivo temporal bones [71].

Because the system channels have been split to drive two transducers, the size of the aperture that can be designed is further limited; there can be at most 128 receive and 256 transmit elements per probe, where all receive elements are shared transmitters/receivers. Given the general elliptical shape of the window, a sparse phased array design with the “bull’s eye” pattern of active elements has been adopted (Figure 8A). Element pitch is 0.35 mm, which is 0.57λ at 2.5 MHz or 0.41λ at 1.8 MHz, the expected frequency range of operation. This configuration was designed using concepts of grating lobe suppression and sidelobe tradeoff presented in matrix array design methodologies [72, 73] and was selected from several simulated apertures on the basis of Field II simulations (Figure 8) [74].
Figure 8: (A) Designed aperture (gray elements only transmit, white elements transmit and receive) and (B) resultant on-axis beamplot simulated using Field II. Simulations of the aperture steered to (C) +16° and (D) +32° in both azimuth and elevation at 2.5 MHz and a 7 cm focus.

The designed sparse array having 256 transmit and 128 receive elements produces the beamplot shown in Figure 8B. Field II simulations showed −6 dB beam width of 2.1° and sidelobe level of −22.9 dB at a depth of 7 cm. In comparison, the re-configured Volumetrics cardiac array (6.3 mm aperture) used previously [62] has a −6 dB beam width of 4.5° and sidelobe level of −33.6 dB at 7 cm.
2.1.7 Transducer fabrication and integration

Constructing this system with integrated transducers required the fabrication of two types of circuit boards: the flexible circuit on which the transducers are to be built and a rigid FR-4 printed circuit board (PCB) to couple these two transducers to the scanner. In collaboration with Microconnex (Snoqualmie, WA), we designed a polyimide-substrate multilayer flexible circuit 279 mm in length. The narrow end contains gold pads measuring .35 × .35 mm and arranged as in Figure 8A. The traces for 256 active elements are brought out to two rows of gold pads at the wide end of the flex for connection to a 300-pin electronic connector (Samtec BTH-150, New Albany, IN). The fabricated circuit has two trace layers with a dedicated ground layer for each. The total thickness is 210 µm, or approximately 0.2λ at 2.5 MHz assuming \( c_{\text{polyimide}} = 2170 \) m/s. A photograph of an array after dicing is shown in Figure 9. An illustrative cross-section (not to scale) of this stack-up is shown in Figure 10.

In building a single transducer, a piece of high-dielectric PZT-5H (TRS Technologies, State College, PA) was cut to size and bonded to the flex circuit with conductive epoxy (Chomerics, Woburn, MA). Next, elements were diced at 0.35 × 0.35 mm using a programmable dicing saw (Disco DAD3220, Tokyo, Japan). Then a piece of liquid crystal polymer (LCP) with gold sputtered on two sides was bonded to the diced elements, forming the ground electrode and a shield ground. These separate grounds were maintained on the coupling board. Finally, an acoustically lossy backing was cast.
and bonded to the back of the flex. Connectors were hand-soldered to each flex to mate with the PCB.

Figure 9: Photograph of an array after completion of dicing.

Figure 10: Illustrative stack-up of transducer fabrication. A lossy backing is bonded to the back of the flex (not shown).
2.1.8 Performance Testing

Tests were performed on the completed transducers to measure the following: pitch-catch sensitivity and bandwidth, 50 Ω-insertion loss, and cross talk. The pitch-catch characteristics were tested using a function generator (Agilent 33250A) and 25-watt RF power amplifier (ENI 325) to produce a 3-cycle pulse at 2.3 MHz into an aluminum block in a water tank. Oscilloscope data were read into a personal computer at 250 MHz using LabView (NI, Austin, TX) and spectra were computed in Matlab.

50 Ω insertion loss was measured on 20 pairs of elements using a similar setup, with element input and output impedances both at 50 Ω. Peak-to-peak voltage was measured with the aluminum block positioned as close as possible to the face of the transducer to limit diffraction effects. Worst-case cross talk was similarly measured on the 20 pairs of adjacent elements with traces running in parallel over the greatest distances. For this test, the receive element was loaded to match the input impedance of the scanner’s preamplifier.

Sensitivity was measured on a per-channel basis using the same setup described for the cross talk measurement. Beamsummed sensitivity was also measured on the scanner using a Signatec PDA14 acquisition board (Corona, CA) to digitize echoes from a point reflector.
2.1.9 Volume registration

While the real-time display for dual-transducer scanning has been described, in order to realize the full potential of 3D ultrasound in imaging blood flow in cerebral arteries, acquired volumes must be displayed in three dimensions. This section describes a attempt at registering and fusing acquired volumes into three-dimensional figures. As will be discussed, the methods are simple but nonetheless address the problem of registering and fusing 3D volumes of ultrasound rapidly and with limited user intervention.

Due to the limited computational power of the scanner, acquired volumes are transferred to a personal computer adjacent to the scanner for potential 3D display at the patient’s bedside. The following scheme for pre-processing and volume rendering has been implemented in Matlab. When the scanner saves a volume, it is stored as a block of envelope-detected echo data in \( r-\theta-\phi \) coordinates followed by a block of color flow data in \( r-\theta-\phi \) coordinates. For Doppler data, both magnitude and signed velocity data are saved. A file header contains information such as scan depth and Doppler gate position which is necessary for offline reconstruction. Care was taken during our transcranial studies to set the compression curve to linear so that saved data were not log-compressed.

The basic data flow for offline processing of two 3D volumes is as follows: data transfer and scan conversion, pre-processing for correlation search algorithm, 3D
correlation search, thresholding and display of fused volumes. This flow is discussed in detail in the following subsections.

**2.1.10 Data transfer and scan conversion**

When a color flow volume is acquired, saved volumes include approximately 16 frames from each transducer and can be as large as 64 MB in size. (Note that the term “frame” is used to indicate an entire 3D volume at a single snapshot in time.) All frames are saved to the scanner’s hard drive and transferred to a PC via UDP transfer over an Ethernet cable. Upon reception by the PC, data is read into Matlab and split into two volumes, one from each acquiring transducer. In the elevation dimension—which is spatially sampled at half the frequency of the azimuth dimension—data is interpolated using lowpass interpolation to maintain symmetric voxels. Each volume of echo data is scan converted to x-y-z Cartesian coordinates with a voxel size of 1.16 × 1.16 × 0.94 mm. Doppler data are similarly scan-converted, with volume sizes varying according to the size of the region defined by the Doppler gates during scanning.

**2.1.11 Pre-processing**

Next, 3D median and low pass filters of length 3 × 3 × 3 are applied. Median filters have been commonly used to reduce speckle in registration of 3D ultrasound volumes [48], while a low pass filter improves long range correlation and helps avoid entrapment in local maxima [47].
Acquired frames of the Doppler magnitude data (backscattered Doppler intensity) have been averaged in time, increasing SNR by $\sqrt{N}$ where $N$ is number of frames. While this removes information pertaining to blood flow in time, our primary objective with the three-dimensional display is to identify the spatial course of cerebral vessels; temporal information is still available on the real-time display. This is analogous to increasing the scanner’s persistence to its maximum level.

### 2.1.12 Correlation search

A rigid transform approach to registering the two Doppler volumes from opposing transducers was implemented which maximizes the normalized cross correlation (Equation 1). This is similar to multi-modality approaches which use vessels as fiducials [102, 103].

$$\rho_{\alpha,\beta,\gamma} = \frac{\sum \sum \sum a(x, y, z) \cdot b(x - \alpha, y - \beta, z - \gamma)}{\sqrt{\sum \sum \sum a(x, y, z)^2 \cdot \sum \sum \sum b(x, y, z)^2}} \quad (1)$$

$a(x, y, z)$ and $b(x, y, z)$ are the two scan-converted Doppler volumes and $\alpha$, $\beta$, and $\gamma$ represent the shifts of volume $b$ relative to $a$ in the $x$, $y$, and $z$ directions. Translation is maximized in three dimensions by determining $\alpha_0$, $\beta_0$, and $\gamma_0$, the values of $\alpha$, $\beta$, and $\gamma$ which maximize $\rho_{\alpha,\beta,\gamma}$. Next a second correlation search maximizes rotation of the
volumes in azimuth and elevation (the probes cannot rotate about the third axis). This rotational normalized cross-correlation function is given in Equation 2:

\[
\rho_{\theta,\phi} = \frac{\sum_x \sum_y \sum_z a(x, y, z) \cdot \text{rot}[b(x - \alpha, y - \beta, z - \gamma), \theta, \phi]}{\sqrt{\sum_x \sum_y \sum_z a(x, y, z)^2 \cdot \sum_x \sum_y \sum_z \text{rot}[b(x, y, z), \theta, \phi]}}
\]  
(1)

where the operator \( \text{rot}[V, \theta, \phi] \) is used to indicate rigid rotation of some volume \( V \) about angle \( \theta \) in azimuth and \( \phi \) in elevation. Rotation is maximized in two dimensions by determining \( \theta_o \) and \( \phi_o \), the values of \( \theta \) and \( \phi \) which maximize \( \rho_{\theta,\phi} \). Because only slight tilts in the transducers are expected, the extents of \( \theta \) and \( \phi \) are set to \( \pm 20^\circ \) in \( 2^\circ \) increments.

In order to decrease the runtime of the correlation search program, a time-of-flight measurement has been used to reduce the set of search values in \( z \), the range direction. During the first image line of every frame, the central nine transmit elements of transducer A are fired in phase while all receive elements on transducer B are enabled. All remaining elements on both transducers are disabled. During the second line of each volume, the reverse occurs: transducer B fires nine central elements and transducer A receives on all elements. These two times at which the fired pulse is received are averaged and a bulk speed of sound of 1540 m/s is assumed to compute the separation between the two transducers. This method was first tested in a custom water tank having aligned, side-viewing silicone rubber panels on either side. Here the speed
of sound was assumed to be 1500 m/s and the measured distance across the tank was compared to ten time-of-flight measurements made with aligned transducers. The time-of-flight measurement was 18.12 ± 0.05 cm as compared to the dimension of the tank, which was 18.10 cm. In the correlation search described by Equation 1, the value of γ—variation in range z—is only varied within ±5.5 mm of the time-of-flight estimate. Values of α and β—shifts in x and y—include all possible overlaps between the two Doppler volumes.

Using the described procedure for maximizing normalized cross-correlation of Doppler volumes in rigid body transformations, experimental values of $\rho_0$, (the correlation coefficient after successive maximizations of translation and rotation) have been observed to be approximately 0.7 when registering flow in a tube within the polymer casting of the skull. In vivo correlation coefficients have typically been lower, in the range of 0.5 to 0.6. Runtimes of the entire described processing are <10 minutes on a Dell PC running Windows with a single 3.2 GHz Pentium 4 processor. However, transferring a volume takes approximately 10 minutes due to the Volumetrics scanner’s bus speed.

2.1.13 Validation of algorithm

Registration error was evaluated for the described algorithm in which time-of-flight data restricts the search region of correlation search. This was done in simulation and using a flow phantom. In the simulation, two volumes were created each
containing a single vessel. Velocity profiles in these vessels were governed by the following equation:

\[ v(r) = v_0 \left(1 - \left(\frac{r}{R}\right)^{p_0}\right) \quad (2) \]

where \(v_0\) is velocity in the center of the lumen, \(R\) is vessel radius, \(r\) is distance across the vessel, and \(p_0\) indicates profile order [75]. Profiles were matched to those for ideal flow in the internal carotid artery as described by Jensen [75].

Zero-mean Gaussian noise with varying amplitude was added to the volumes containing ideal flow. At signal-to-noise ratios of 20, 10, and 6 dB, the correlation search algorithm exactly matched the vessels for all (4) simulations, with decreasing correlation coefficients of 0.98, 0.97, and 0.95, respectively. At an SNR of 3 dB, the program made a 2 pixel error in one of four cases. This shows that maximizing normalized cross-correlation performs quite well in controlled cases that are free of clutter and other artifacts.

For flow phantom testing, a 5 mm-diameter latex tube was positioned in a side-viewing water tank with agitated saline pumped through at approximately 100 cm/s. Both transducers acquired color flow volumes of flow in the same tube and the tubes were registered using the described algorithm. The long axis of the tube was oriented parallel to the beam of the transducers to produce the highest possible detection of flow. Three different acquisitions were performed for each of three different transmit power levels (50%, 25%, 5%). The center of mass of each 3D tube rendering was computed in x.
and $y$ (directions perpendicular to flow) in order to create two curves in three-dimensional space having height and width of a single voxel. Registration error was taken to be the measured distance between these curves in $x$, $y$, and $z$ directions.

2.1.14 Thresholding and display

A simple segmentation is performed to prepare the data for volume rendering. In order to render, it is desirable to set thresholds for both echo and Doppler data for each volume. While segmentation of ultrasound images is a complex topic which has received much attention [139, 140], here the objective is to extract vessels and bony surfaces from noise so that they may be identified in a simple 3D display, thus thresholds are set manually at the end of the rendering process. While this requires user intervention and re-rendering, this process is analogous to adjusting the echo or color flow reject on a scanner’s real-time display. Usually no more than 5 re-renderings were required in practice to achieve a suitable image. All voxels having values equal to or greater than the selected threshold were rendered following registration, producing a single fused image in an interactive display that allows viewing from any angle.

2.1.15 Human study

Per an institutional review board-approved protocol and in collaboration with Duke University Medical Center, six healthy individuals were scanned using the brain helmet system described in this chapter after providing informed consent. The latter four of these six subjects received an intravenous infusion of Definity microbubble
contrast agent (Lantheus Medical Imaging, N. Billerica, MA) at a concentration of 1.3 mL in 50 mL saline. These early scans allowed evaluation of the suitability of the device for performing clinical scans and assessment of the quality of the resulting images and renderings. In order to scan a human, a patient bed with adjustable height was positioned adjacent to the scanner (Figure 11). The two probes were placed on the temporal acoustic windows of a supine patient while scanning in echo mode (8A). Once probes had been positioned for optimal viewing of blood flow in either the middle or posterior cerebral arteries, the volunteer was given the intravenous injection. Scans were performed under the supervision of a neurologist, a registered nurse, and a registered vascular technologist.

![Multilayer flexible circuit](image)

**Figure 11:** Schematic (A) and photograph (B) of the completed system for in vivo scanning.
2.2 Results

2.2.1 Transducer simulation

When modeling two probes which are identical except for their cable lengths (2.23 m and 0.30 m), PiezoCad simulations predicted an increase in sensitivity of 16.0 dB. However, the commercial probe used as a control has an acoustic matching layer, while the prototype probe does not. Additionally, individual conductors in the commercial probe are partially separated by air and thus have a relative permittivity ($\varepsilon_r$) close to 1. Taking these differences into account, when PiezoCad simulations were used to directly compare the two physical probes—the control probe having a long cable and an acoustic matching layer versus the proposed probe having a short flex interconnect, printed circuit board, and no matching layer—the sensitivity of the proposed probe was expected to be 7.2 dB than the control.

2.2.2 Transducer testing

The two fabricated probes used for testing and in vivo evaluation had element yields of 110/128 (86%) for receive elements and 220/256 for transmit elements (86%), and 109/128 (86%) for receive elements and 217/256 for transmit elements (85%).

Mean center frequency was measured at 2.3 MHz with a fractional bandwidth of 20%. Insertion loss was measured at -80 dB. Cross talk was measured at -30 dB and is primarily due to electrical cross talk in the printed circuit board. This was evidenced by observation of the transmit pulse bleeding across to adjacent channels and an observed
decrease in severity of cross talk in channels which do not traverse the entire length of this board.

Figure 12: (A) Single-element pulse echo-waveform acquired in a water tank with an aluminum reflector. (B) 128-channel beamformed reflection from target at 7 cm focus.

The measured gain in SNR was 6.8 dB in per-channel testing and 7.1 dB in beamformed testing, in close agreement with simulations. Typical results from these tests are shown in Figures 12A (per channel) and 12B (beamformed). The average element + cable capacitance of the probe on the 0.279 m flex circuit was 60 pF, as compared to 156 pF for the probe with the 2.23 m cable.

2.2.3 Validation of registration algorithm

In registering Doppler magnitude data acquired simultaneously by different transducers, mean errors were: 2.43 mm in $x$, 3.52 mm in $y$, and 3.89 mm in $z$. Shekhar and Zagrodsky also report error in $z$ that is approximately as large as in one lateral
dimension [48]. While registration error would ideally be smaller for clinical practice, these tests validate the suitability of the correlation coefficient as a metric and confirm the implementation of the algorithm.

2.2.4 Human study

With contrast enhancement, cerebral blood flow was imaged in color Doppler mode in 4 of 4 subjects. Flow without contrast agent has been successfully visualized in at least one cerebral artery in 4 of 6 volunteers. Previously, three-dimensional color flow could not be seen without contrast agent. Figure 13 shows a labeled screen shot from the real-time display in one subject, a 24-year-old female.

![With contrast enhancement](image)

**Figure 13:** Real-time display view of flow in middle cerebral arteries (MCA), internal carotid arteries (ICA), and posterior cerebral arteries (PCA) using described
system. Directional information of flow has been discarded. Labels have been added to indicate anatomy as well as the sectors displaying the subject’s right coronal, right transverse, left coronal, and left transverse imaging planes.

The same data may be more fully visualized in the registered renderings in both coronal (Figure 14) and transverse views (Figure 15). For these renderings, all Doppler data is shown in red and all echo data in white, with the echo data depicting the falx cerebri. The coronal view (Figure 14) is paired with a typical (not the same subject) magnetic resonance angiogram (14B) to indicate the field of view captured by the ultrasound rendering (14A). For the transverse view (Figure 15), common vessels are identified in an adjacent dissection image.

A successfully registered rendering without contrast enhancement is shown in Figure 16 for a 42-year-old male. For the Doppler data, color (either red or blue) is used to indicate the acquiring transducer (left or right), while all echo data is shown in white. The rendered Doppler and echo data (Figure 16B) is shown paired with a typical (not the same subject) magnetic resonance angiogram (Figure 16A) to indicate anatomy. An enlarged view of the vessels, identified as bilateral internal carotid arteries, is presented in Figure 16C.

Finally, a volume rendering of three different views in a single subject is presented. Microbubble contrast agent was not used in this subject. First two transtemporal volumes were acquired from this 42-year-old male subject, which were later registered using the described algorithm. In these volumes, both posterior cerebral
arteries as well as midline structures were visible. Next, the subject was rotated to lie on his side and a single transforaminal volume was acquired using all the image lines on a single transducer. The yellow arrow indicates the manual positioning of the transforaminal volume with respect to the other two volumes, as the basilar artery rises toward the posterior cerebral arteries. Integration in time of several volumes of pulsatile flow likely contributes to the bulbous appearance of vessels.

**Figure 14:** (A) Coronal ultrasound rendering and (B) representative magnetic resonance (not the same subject) showing the area under examination. Original MRA image produced by Ofir Glazer, Biomedical Engineering Department Tel Aviv University, reproduced with permission of the author.
Figure 15: Paired ultrasound rendering and dissection image indicating anatomy in vessels of the Circle of Willis. MCA=middle cerebral artery, PCA=posterior cerebral artery. Original photograph produced by Prof. John A. Beal, Department of Cellular Biology and Anatomy, Louisiana State Health Sciences Center, Shreveport. Reproduced with permission of the author.
Figure 16: The same magnetic resonance angiogram (A) is used to indicate ultrasound field of view for the registered, rendered ultrasound data in (B) acquired without microbubble contrast agent. (C) is an enlargement of the color flow data.
Figure 17: Automatically registered rendering of posterior cerebral arteries from simultaneous transtemporal scans along with manually registered (yellow arrow) transforaminal scan. Vessels are paired with a dissection image (inset). No microbubble contrast agent was used.

2.3 Discussion

Several potential improvements in materials and fabrication have been illuminated by this process. While the use of the multilayer flex circuit has reduced fabrication times and provided mechanical stability during imaging, the high $\varepsilon_r$ of the polyimide substrate limits sensitivity. Recent development of lower dielectric flex circuit materials such as DuPont Pyralux® TK ($\varepsilon_r \approx 2.3 - 2.5$) could aid in this problem in the future. Bandwidth could also be improved through the use of one or more acoustic matching layers.

While we are encouraged by the diagnostic potential of this still-experimental system, any future diagnostic value is dependent on the image quality of the two
acquired volumes, the ability to display them simultaneously, and the ability to increase
diagnostic information relative to single-transducer scanning. Several additional
improvements are necessary to extend the proof-of-concept prototype presented in this
paper to the proposed three-transducer system capable of remotely transferring 3D
volumes, though all should be technically feasible on a portable scanner. The GE
Voluson i provides 3D fetal scanning with a mechanically scanning 1D array probe, and
recently the Philips CX50 has demonstrated 3D scanning with a transesophageal probe.
In the future, pre-processing and registration might be performed in the laptop,
allowing a single fused data set to be sent over the Internet to a hospital.

Registration could be improved by using multiple iterations or increasing the
number of transformation parameters. Optimization of registration using a cost
function or a coarse-to-fine approach could improve runtimes and allow for multiple
iterations. Graphics processing units (GPUs) have also been demonstrated in processing
3D ultrasound data and could vastly improve runtimes for this application [76, 77]. In
clinical scanning of a diseased individual, registration error could impose a limitation on
the ability to detect abnormal flow, emphasizing the importance of reducing this error
and further underscoring the significance of SNR. It is also worth noting that the
aberrating and refractive properties of the skull differ on the two sides of the head,
suggesting non-rigid registration may eventually be necessary to account for the
entrance angles at which the ultrasound beams enter the brain.
A third transducer could be added by splitting the channels even further, however image quality would clearly suffer due to decreases in spatial sampling and in beamformed SNR as the number of active elements per transducer declines. This problem might be addressed by multiplexing, although high-voltage switches typically carry high capacitances which would reverse most of the gains presented here. For example, the Supertex HV20220 (Supertex, Sunnyvale, CA) carries typical “on” and “off” shunt capacitances of 48 pF and 152 pF, respectively, for a 4:1 configuration. This dilemma underscores the importance of total interconnect design for matrix arrays taking into account the electrical characteristics of the transducer element, the cable, and the preamplifier and switching electronics [36, 37].

In combination, the proposed improvements might yield several additional decibels of signal-to-noise ratio and further increase image quality. In this chapter, we phase aberration correction has not been used [78], which could also contribute to the desired gains. A phase aberration correction approach for this system will be examined in Chapter 3.

In this chapter, *in vivo* results from automated registration of two simultaneously-acquired transcranial volumes have been presented, the first known result of this kind. Such a system shows significant potential for rapid cerebrovascular imaging of stroke and related conditions, particularly if microbubble contrast agent is not required.
3. Pitch-catch phase aberration correction of two matrix arrays

3.1 Background

Several techniques have been devised for improving the image quality of transcranial ultrasound images, including imaging through known acoustic windows such as the temporal bone window [41, 134], inducing shear-mode conversions at the soft tissue-skull interfaces [84, 85], contrast-specific imaging techniques for both large artery [3, 143] and perfusion imaging [79, 80], and adaptively adjusting probe element delays and/or amplitudes [31, 32, 53, 54, 56, 57, 59-61, 78, 81]. Notably, Miller-Jones proposed using a contralateral active source as a correction beacon [31], and Ivancevich et al. corrected in vivo aberration on a matrix array using a multi-lag cross-correlation technique [60]. It is now proposed to combine these last two approaches while taking into consideration the size of the isoplanatic patch (IP) of the aberrator within the temporal bone acoustic window in order to improve image quality throughout an entire 3D transcranial volume.

3.1.1 The isoplanatic patch

The isoplanatic patch—or spatial stability—of an aberrator describes the area over which the aberrator may be corrected using a single arrival time map [82]. The skull is well modeled by the nearfield phase screen model in which the aberrating layer is assumed to be infinitesimally thin, to lie immediately adjacent to the face of the transducer, and to have an infinite isoplanatic patch. Thus the thickness of the skull at a
given transducer element may be corrected by application of the appropriate equal and opposite time delay. In actuality, the skull has a finite thickness and is separated from the transducer by a few millimeters of extracranial tissues and vessels, resulting in a finite isoplanatic patch. For a phased array scan, the nearfield phase screen assumption begins to fail as scan angle increases from broadside (0°, 0°) to approximately ±16° [61, 83], decreasing the potential benefit of phase aberration correction [84]. Applying the appropriate phase correction maps to the appropriate steering angles could improve brightness and contrast in transcranial images over the entire field-of-view. In this work, the aberrator is assumed to be temporally stable, because the skull is rigid and fixed relative to the position of the transducer.

There is not a single, standard definition for the isoplanatic patch; it has been variously defined as the positions in the field over which the point spread function (PSF) increases by 10% [85], aberrators are correlated by 70%-90% [82], or speckle/target brightness increases [61, 83]. Isoplanatic patch size varies depending on the definition used, the scanning system, the correction method used, and on the individual skull [83]. Correction of multiple isoplanatic patches has been investigated by Liu and Waag in *ex vivo* abdominal tissues [85], and by Fernandez and Trahey [86] and Dahl et al. in the breast [82]. For the cranial bone (temporal acoustic window), two known attempts have been made at measuring the size of the isoplanatic patch in angular extent and depth, both using the increasing brightness metric. In the first, Ivancevich et al. measured a
two-sided angular extent of 33° in the transverse plane in a casting of a human temporal bone [87], and an angular extent of 9.7° in vivo in 2 subjects using a correlation-based correction method. Vignon et al. measured 36 ± 18° in ex vivo skulls and 33° in vivo in a single subject, both in the transverse plane, using a time reversal correction method (FDORT) [83]. This study also suggested that aberration correction effectiveness in transcranial sector scans might be divided into 3 broad categories of “effective,” “lowly effective,” and “ineffective” as scan angle increases, where “effective” is designated as the center 30° of a transcranial sector scan. The isoplanatic patch in a 2D sector scan is often reported as an azimuthal angle and an axial depth. As both previous measurements found the axial depth of the temporal bone isoplanatic patch to extend to the entire scan depth (12 cm [87] and 15 cm [83]), an infinite patch size in the axial direction will be assumed.

3.1.2 Phase aberration correction techniques

In addition to wavefront aberration, other mechanisms of degradation in transcranial ultrasound imaging include mode conversion, [152, 153], refraction [57], multiple scattering [88], and attenuation [89]. Of these phenomena, aberration is the only one which may be compensated for in the course of conventional delay-and-sum beamforming. Because a clinical ultrasound scanning system assumes a constant longitudinal propagation velocity (typically $c_{\text{issue}} = 1540 \text{ m/s}$) when computing both transmit and receive delays for focusing and steering, the introduction of a layer having
different velocity ($C_{\text{skull}} \approx 2327 \pm 90$ m/s at 1.97 MHz [90]) results in a broadening of the imaging system’s point spread function (PSF). As signals are summed partially out of phase both at transmit foci and during receive beamforming, the wavefront coherence diminishes, resulting in decreased speckle and point target brightness, enlargement of axial and lateral resolution, and decreased contrast.

Numerous techniques for aberration estimation and correction have been proposed [59, 82, 85, 87, 91-112]. The work of Trahey et al. explores phase correction techniques for phase screen aberrators and delineates the need for two-dimensional phase aberration correction to accurately estimate and correct aberrators containing high spatial frequencies [73, 146, 157-160]. Specifically, Nock et al. describe the need for different delay update profiles at different angles due to a finite isoplanatic patch [96]. Following these efforts and those of Flax and O’Donnell [97] and Liu and Waag [98], our lab has previously implemented phase aberration correction on a matrix array using both multi-lag least-mean-squares cross-correlation [97, 98] and speckle brightness [161, 162] correction algorithms [59, 87]. Optimization of a coherence metric has also demonstrated success, especially for the case of an offset phase screen [163, 164]. Waag et al. have investigated propagation path effects in aberration estimation and correction in the absence of a point target using statistical methods [165-167].
3.1.3 Time reversal and inverse filtering

The time reversal mirror technique [105] has demonstrated success at focusing through aberrating layers [106-110] by addressing the aberration problem as a complex filtering operation, allowing variation of both amplitude and phase with frequency and more fully addressing the various degradation phenomena described above. The basic principle is that the matched filter producing optimum signal-to-noise ratio (SNR) is the time reversal of the received signal. Ng has examined the closely-related phase conjugate filters, which correct distortions in phase spectra without modifying magnitude spectra [93, 111]. Inverse filters attempt to take this one step further by eliminating both phase and magnitude distortions, but are often non-realizable and amplify noise at frequencies where the operator being inverted (i.e. aberration or other distortion operation) has a small response. This is typically addressed by regularization via singular value decomposition and by applying inverse filters only over selected bandlimited regions [122, 173]. For each individual frequency, it is possible to compute the number of physically relevant singular values given imaging system characteristics, then only these singular vectors are inverted [112].

Vignon et al. have demonstrated an inverse filtering technique suitable for correcting skull-induced distortion using a linear array [32, 112] and the pitch-catch geometry proposed by Miller-Jones [31]. However, the cost of implementing this approach on a two-dimensional array is significant, requiring a fully programmable
transmitter and access to the radiofrequency (RF) signal for each element, a difficult proposition at a time when many manufacturers are moving towards digitization and partial beamforming in the probe handle [41, 113].

3.1.4 Brain helmet pitch-catch phase aberration correction

In Chapter 2, we have demonstrated an experimental system capable of acquiring, registering, and fusing two contrast-enhanced transcranial ultrasound volumes (Fig. 18) with a 7 dB increase in SNR due to reduced cable lengths [126, 127]. The work in this chapter takes advantage of the beacon provided by this geometry to add phase aberration correction to this diagnostic ultrasound brain helmet. The existing state-of-the-art is improved upon in the following ways: 1) correcting on a coherent, high-amplitude source reduces the error in arrival time maps for 3D transcranial aberration correction as compared to speckle-based corrections, and 2) utilizing each matrix array to transmit multiple steered wavefronts allows estimation of multiple arrival time maps, where each map can be used to correct a different group of image lines within the phased array volume (i.e. a different isoplanatic patch). Previous attempts at transcranial phase correction have assumed an infinite isoplanatic patch size, causing a decrease in image brightness at the edges of a 2D sector or 3D volume [83]. By updating imaging delays to correct for skull inhomogeneities and partially restore lost resolution and contrast throughout two entire 3D volumes, we hope to aid in the
emergence of transcranial ultrasound as a rapid, effective diagnostic alternative for cerebrovascular imaging.

Figure 18: Probe placement for proposed transcranial imaging system using two matrix arrays positioned at temporal bone windows. This system has previously been used to acquire the contrast-enhanced (Definity) 3D volume shown at right [63].

3.2 Methods

3.2.1 Scanning system

Experiments were performed using the Volumetrics Model 1 scanner (Volumetrics Medical Imaging, Durham, NC), a real-time 3-D ultrasound system using 16:1 parallel receive processing [25, 26]. This system has 256 transmit channels and 256 shared transmit/receive channels. A 2-D phased array transducer on this system typically transmits 256 broadened beams in succession. For each transmit beam, 16 receive beams are formed from echoes arranged in a $4 \times 4$ pattern centered on the
transmit beam. The transmit beam spacing is 4° in a typical 64° × 64° scan, enabling frame rates of up to 30 volumes/sec.

To enable simultaneous real-time 3D imaging with two matrix arrays, the scanner’s channels and its 4096 image lines have been split equally between two matrix arrays so that two 64° × 64° pyramidal volumes may be acquired simultaneously. Each matrix array has 128 transmit elements and 128 shared transmit/receive elements. As before, for each transmit beam, 16 receive beams are formed from one of the two transducers. However, transmit beam separation has now doubled in the elevation direction, so transmit beams in each 64° × 64° volume are now spaced at 4° in azimuth and 8° in elevation. Four slices from two probes are displayed in real-time: one azimuth and one elevation slice from each transducer, corresponding to coronal and transverse planes in transtemporal imaging (Fig. 19A). The operator uses a trackball control to select any four slices in either volume for display. Although custom transducers were designed for this system [63], limitations in bandwidth and inter-element uniformity compared to commercially manufactured transducers rendered the jitter errors in aberration estimates too high for this study. For this reason, all experiments were performed using commercial (Volumetrics) sparse matrix arrays described previously [59, 62] and remapped by a custom printed circuit board to allow for dual simultaneous 3D imaging [68]. Each sparse matrix array has 128 elements with an aperture diameter of 6.6 mm (Fig. 19B); studies were performed at 1.8 MHz. For phantom studies,
transducers were aligned to face one other across the phantom or water tank with zero
tilt between them unless otherwise noted.

Figure 19: A. Volume interrogated during a typical transtemporal 3D ultrasound examination. Coronal and transverse plane views are displayed simultaneously. A steerable Doppler beam is also shown in gray. Two such volumes are acquired simultaneously with this system. B. Transmit and receive apertures used for each array in this system are shown.

3.2.1 Cramér-Rao lower bound

Phase aberration correction techniques may be divided into two categories, those that correct on a beacon signal or point target and those that do not. The beaconless techniques rely on speckle signals, backscattered echoes from sub-wavelength scatterers having random positions and amplitudes. These received signals are subject to the van Cittert-Zernicke theorem, which states that the mutual coherence $\Gamma_{AB}$ of two stationary signals $A$ and $B$ at locations $x_1$ and $x_2$ on the aperture decreases with increasing spatial distance across the aperture in a manner proportional to the autocorrelation of the aperture [175, 176]:
\[ \Gamma_{AB}(x_1, x_2) = R(x_1 - x_2), \quad (3) \]

where \( R \) is the autocorrelation of the aperture.

The success of correlation-based correction techniques, such as those proposed by Flax and O’Donnell [97] or Liu and Waag [98], depends on the amount of jitter present in the aberration estimate. The Cramér-Rao lower bound (CRLB) describes the lower bound on the jitter— or the minimum standard deviation between the true (\( \Delta t \)) and estimated (\( \Delta \tilde{t} \)) time delay— for correlation-based unbiased time delay estimators [114], and is given by

\[
\sigma(\Delta t - \Delta \tilde{t}) \geq \sqrt{\frac{3}{2 f_0^2 \pi^2 T (B^3 + 12B)}} \left( \frac{1}{\rho^2} \left(1 + \frac{1}{\text{SNR}}^2 \right) - 1 \right) \quad (4)\]

where \( f_0 \) is center frequency, \( B \) is -6 dB bandwidth, \( T \) is window length, \( \rho \) is correlation coefficient for the two signals, and \( \text{SNR} \) is signal-to-noise ratio. The correlation \( \rho \) between two partially coherent signals is less than 1, leading to an increase in jitter errors as coherence between the signals decreases from 1. In theory, echoes received from a coherent source remain correlated across the entire aperture, and therefore provide an ideal signal to estimate phase errors. The dual array configuration of the proposed system can be used to generate a coherent, high-amplitude source,
ensuring that both \( q \) and SNR remain high, and thus reducing jitter to diagnostically useful levels.

A tissue-mimicking phantom (Model 539, ATS Laboratories, Bridgeport, CT) was used to measure the mean correlation coefficient of the received channel signals from an active source. The transducer separation was 17 cm for this experiment, which is larger than the diameter of the typical human head (approximately 14 cm).

### 3.2.2 Aberration estimation and computation of delay updates

Implementing adaptive imaging on any clinical system poses a considerable challenge given frame rate considerations and limited transmit control capabilities, though some successful implementations have been achieved \([58, 59, 86, 94, 115-118]\). The presented approach uses two matrix arrays, each of which sequentially transmits a series of unfocused waves at varying steering angles to serve as beacons. These data are used to compute multiple arrival time maps, each of which is used to correct the time delays of a subset of image lines.

For each beacon wave, transmit elements were fired with the desired phasing on one array to produce either a steered or an unsteered unfocused wave. Single-channel RF data were acquired on the opposing array and digitized on an adjacent PC at 25 MHz (PDA14, Signatec, Corona, CA). These signals were filtered axially (FIR bandpass filter, 60% BW) and laterally \([119]\) (FIR lowpass, cutoff at 75% of spatial Nyquist), then the
direction of arrival (D.O.A.) was estimated by determining the look direction \((\theta, \phi)\) which maximizes received power:

\[
[\theta_0, \phi_0] = \arg \max_{\theta, \phi} \left\{ \sum_z \sum_\theta \sum_\phi f(z, \theta, \phi) \right\}^2, \tag{5}
\]

where \(f\) is the received wavefield, and \([\theta_0, \phi_0]\) is the optimum look direction.

D.O.A. estimation was performed using an iterative coarse-to-fine approach starting at -35° to 35° in 5° increments and ending with step sizes of 0.67° in both \(\theta\) and \(\phi\). Received data \(f\) was then steered to \([\theta_0, \phi_0]\). D.O.A. estimation is necessary because the orientations of the two transducers are selected based on the imaging region of interest and the individual’s temporal bone windows, and thus may be tilted with respect to one another. By steering to the direction of maximum power, large steering components are removed, allowing aberrations—which are small in magnitude (RMS) relative to steering components—to be detected and corrected. D.O.A. estimation has previously been used in speckle-based aberration correction to steer toward the dominant scatterer [119].

Phase aberrations were estimated using a multi-lag least squares estimation technique [94]. This estimation algorithm is performed by computing the normalized cross correlation between all element signals within a specified spatial lag. This
provides multiple estimates of the time delays between pairs of elements, producing an overdetermined system of equations, described by

\[ D = MT, \quad (6) \]

where \( D \) is an \( m \times 1 \) vector containing the positions of maximum correlation coefficients and \( M \) is the \( m \times n \) model matrix. This system is approximately solved using linear least squares [98], producing the \( n \times 1 \) vector of arrival time estimates:

\[ T = (M^T M)^{-1} M^T D \quad (7). \]

The least squares solution ensures phase closure, or that the arrival times on any closed path on the aperture sum to zero.

Next, the receiving transducer became the transmitter and the opposing transducer became the receiver on which aberration was estimated, producing an additional 128 delay updates which were used to update both transmit and receive delays for the shared elements of the second array. A total of 256 delay updates were computed for the two transducers to be passed to the scanner. Model matrix \( M \) is typically pre-computed for a specified spatial lag between elements (3 mm in this work).
The appropriate spatial lag in the presence of a coherent target (correction beacon) is examined by simulation in Appendix B.

### 3.2.3 Correction of multiple isoplanatic patches

In order to correct aberrations in a phased array image, it is proposed to use the contralateral matrix array to transmit a series of steered unfocused waves. Each steered unfocused wave acquisition is used to compute a unique map of delay updates for the receiving array as described. Multiple maps are then passed to the scanner, with each map updating the delays only in the section of the imaging volume corresponding to the steering angle of the transmitted wave used to generate that map (Fig. 20). Acquisition and processing is otherwise identical to that described for the case of a single IP. The assumption inherent in this correction scheme is that because the failure of a correction beyond the isoplanatic patch results from differences in the source-to-receiver path, two wavefronts traveling at different angles will encounter a different aberrator and require different time delay updates to correct their beamformer mismatch.

Another way of looking at this process is to consider the receive beamformer as a non-ideal directional filter. Assuming the transducer has a sufficiently large acceptance angle, then for a single steering direction, the set of signals falling within the pass band are passed, while off-axis signals are partially rejected. For a second steering direction, different signals fall within the pass and reject bands as compared to the first direction. Thus performing phase correction for only a single direction (typically broadside)
corrects only the signals falling within a beamformer pass band at that single direction, coinciding with the direction from which the data were acquired. This means that when correcting on speckle, a strong off-axis scatterer can cause correction to be steered towards it. This concept has been exploited by Dahl and Feehan to steer towards strong scatterers in order to reduce error described by the CRLB when performing aberration correction [119].

Obviously, correction of multiple IPs still neglects the other effects mentioned such as refraction, frequency-dependent attenuation, and mode conversion [106], which would also be expected to change slightly with angle. Of these, the greatest concern for this approach is the issue of longitudinal-to-shear mode conversion at the soft tissue-outer skull interface and shear-to-longitudinal mode conversion at the inner skull-brain interface. Mode conversion at the skull has been previously investigated as a potential improvement to transcranial ultrasound imaging [29, 120] and to explain a specific transcranial imaging artifact [153, 183], and both groups of investigators found the critical angle at which longitudinal wave transmission through the skull tended to zero to be approximately 30° [29, 121, 122]. The results of Vignon et al. indicate that approximately 40% of incident pressure will be transmitted through extracranial tissues-skull-intracranial tissues by a fully longitudinal path at 15°, as compared to approximately 45% at 0° [122]. This suggests that for a 15° steered transmit, there should not be a significant decrease in transmitted energy relative to the unsteered case,
and also that the so-called “stripe artifact” would be expected to have minimal impact in 64° sector scans (±32°) [121-123].

![Diagram of phased array sector scan](image)

Figure 20: Proposed scheme for correction of multiple isoplanatic patches in a phased array sector scan. Each arrow indicates a propagation direction of a unfocused wave aligned with the patch of the same color.

### 3.2.4 Algorithm testing and validation

The single-IP approach described in Section 3.2.2 was validated using electronic near-field aberrators. To compare pitch-catch aberrator detection with previous pulse-echo detection, six 75 ns RMS aberrators (three for each transducer) were created for each of 3 correlation lengths: 1.3, 2.7, and 5.4 mm full-width at half-maximum (FWHM). The two matrix array probes were positioned facing one another across a water tank with side-viewing membrane panels. Each of the 18 aberrators was measured 3 times using the described approach, and the detected and applied aberrators were compared for each trial.
To validate direction of arrival estimation and subsequent steering of the received wavefront \( f \), facing transducers in the water tank were again used. Six 75 ns (3 each transducer), 2.7 mm FWHM aberrators were tested for each of four relative transducer alignments: \((0^\circ, 0^\circ)\), \((0^\circ, 5^\circ)\), \((0^\circ, 10^\circ)\), and \((0^\circ, 15^\circ)\). The estimated arrival time map was used to correct the pulse-echo imaging delays, and the brightness of a 3D image of a pin in the water tank was assessed before and after correction three times for each aberrator.

### 3.2.5 Evaluation of steered transmit waves as correction beacons

The received channel data need to be partially uncorrelated between different steering angles in order to correct multiple isoplanatic patches, yet should remain highly correlated within a single steering angle for accurate estimation of arrival time using the multi-lag least-mean-square error approach. Inter-element correlation and inter-steering angle correlation were tested by placing two aligned transducers on either side of a tissue-mimicking phantom (separation of 17 cm, Model 539, ATS Laboratories, Bridgeport, CT) in the presence of an aberrating layer (skull casting). Channel data were acquired on each probe using the other as a beacon for 5 proposed steering angles: \((0^\circ, 0^\circ)\), \((0^\circ, -15^\circ)\), \((0^\circ, 15^\circ)\), \((-15^\circ, 0^\circ)\), and \((15^\circ, 0^\circ)\) for 9 unique aberrators by moving the aberrator with respect to the transducer between acquisitions. \( \rho \) was measured both as a function of spatial distance within a single steering direction and as a function of varying beacon steering angle.
3.2.6 Temporal bone isoplanatic patch

Isoplanatic patch size was estimated using the following method. One transducer was placed against a casting of a human temporal bone immediately adjacent to a water tank containing a row of 1 cm-spaced copper wires at a depth of 6 cm. The second transducer was placed on the opposite side of the water tank. The aberrator was estimated and corrected in the manner described in Section II C using an unsteered unfocused wave. The corrected volume was saved and processed offline to determine the angles over which target brightness increased. The angle of each wire target was determined within each saved volume. Two-dimensional measurements were made in the transverse direction using 4 different transducer positions on 2 unique bone castings. IP size was not measured in the coronal direction at this time, so for the purposes of this work coronal IP patch size was assumed to be the same as in the transverse direction.
Figure 21: Definitions of update regions for single-IP (A) and multi-IP correction (B) of a single 64°×64° volume. In the multi-IP case, each color indicates the use of a unique delay map to update image lines within this portion of the volume. Traditionally, the center (red) IP has been used to update an entire phased array volume or sector scan, as in A. In this work, outer 16°×16° corners were ignored in both cases.

Using measured (31.8° ± 15.9°, Section D of Results) and literature isoplanatic patch values, the 64°×64° volume acquired by each transducer was subdivided into 5 patches: a center patch measuring 32°×32°, top and bottom patches each measuring 16°×32°, and left and right patches each measuring 32°×16° (Figure 21). Each 16°×16° corner has been ignored in this work, although in previous work in which an infinite IP was assumed the entire volume has been corrected [124]. Each patch has been corrected using an unfocused wave transmitted by the opposing transducer that was steered in the direction of incident echoes contributing to image lines in that particular patch. All active elements are used to provide adequate energy in the steering direction. Steering angles were chosen to be 15°, small enough to fall well within the ~30° critical angle and
yet large enough to correct a phased array sector out to $15^\circ+32^\circ+15^\circ=62^\circ$ based on the mean measurement of the isoplanatic patch for this system.

### 3.2.7 Physical aberrator correction and color flow imaging

For flow experiments, two transducers were positioned over the side-viewing silicone rubber windows of a water tank at a separation of 18 cm. Physical aberrators were applied to the proposed system using two unique polymer castings of the human temporal bone. One bone casting was placed in front of each array and a 3 mm-diameter tube was held in a U-shape using a custom fixture located above the water tank. Seltzer water was made to flow continuously through the tube at approximately 50 cm/s, and fused flow renderings were assessed for pre- and post-correction volumes. For 7 different aberrators, 10 volumes in time were acquired for each case: aberrated, 1-IP corrected, and 5-IP corrected. The three cases were compared on the basis of the number of color flow voxels above a common threshold and on color flow variance.

### 3.2.8 Offline processing, registration, and fusion

Saved volumes were processed on an offline computer, where they were scan converted, averaged in time, registered and fused into a single 3D visualization in Matlab (The Mathworks, Natick, MA). Rigid registration was performed in the frequency domain using the FFT-based phase correlation technique [125]. Then each volume was normalized and the two optimally transformed volumes were summed in place. Rigid registration of two transcranial ultrasound volumes has previously been
presented and discussed associated challenges in Chapter 2 (2.1.12-14) and in [63].

Offline processing required approximately 2.5 minutes.

3.2.9 Human scanning procedure

The operator began by placing a matrix array probe on each temporal acoustic window of the subject. The scanner’s real-time display was used to qualitatively maximize image brightness and to locate anatomical markers, such as the lesser wing of the sphenoid bone. Once positioned, the transducers were fixed in place and two 3D volumes were acquired. A modified non-imaging transcranial Doppler head frame was used to hold transducers in place (Mark III, Spencer Technologies, Seattle, WA). Next, the pitch-catch RF data was acquired and processed for each transducer as previously described. The 256 delay updates were passed to the scanner via UDP datagram and used to update the transmit and receive delay tables of the system. The two phase-corrected 3D volumes were then acquired. For a single-IP correction, the delay between data acquisition and resumption of scanning was just under 1 minute. Of this, 23 seconds were due to updating delays, an insuperable limitation for this system’s data bus. For a 5-IP correction, the delay update period was approximately 2.5 minutes.

Two human volunteers have been scanned without contrast agent, a healthy 64-year-old male (subject 1) and a healthy 27-year-old male (subject 2). Subject 1 was scanned only before and after a 5-IP correction, while Subject 2 was scanned before correction, after a 1-IP correction, and after a 5-IP correction.
3.3 Phase aberration correction results

3.3.1 Cramér-Rao lower bound

Mean correlation between nearest neighbor elements using unaberrated, unsteered data was $\rho = 0.966 \pm 0.022$. Correlation remained sufficiently high for larger spatial lags, decreasing to only .906 for elements separated by 3 mm. Using measured data, a CRLB of 7.8 ns was computed. Previous results using this scanner reported a mean $\rho$ of $0.838 \pm 0.041$ on unaberrated speckle data, yielding a CRLB of 19.3 ns at 2.5 MHz [59].

3.3.2 Electronic aberrators

As expected, residual error for pitch-catch corrections decreased substantially compared to pulse-echo techniques at all correlation lengths (Fig. 22A) [61]. Mean residual error for pitch-catch was 11.2 ns. Additionally, steering the data to the direction of maximum power, then correcting aberration produced an increase in target brightness in pulse-echo images for all tilts up to 15°, the largest angle tested (Fig. 22B). In most cases, brightness increases by approximately 20% with D.O.A. Without D.O.A., the system confuses aberration and steering. These results indicate that the planar tilt may be removed for physical tilts between the transducers of up to 15°. These results are discussed further in Section 3.4.2.
Figure 22: A. Electronic aberrator residual error for FWHM correlation lengths of 1.3, 2.7, and 5.4 mm for static and moving target cross-correlation, static and moving target speckle brightness [61], are compared with pitch-catch (static) cross-correlation (white bars). Mean residual for pitch-catch is 11.2 ns; the Cramér-Rao lower bound is 7.8 ns. B. Point target brightness increase in pulse-echo 3D images with varying angle between transducers with (black) and without (red) direction of arrival estimation and steering. Off-axis brightness increased by approximately 20% with D.O.A. Each data point is the average of testing 6 different aberrators.

3.3.3 Evaluation of a steered unfocused wave as a transmit beacon

In Figure 23, inter-element correlation for a steered wavefront decreased more rapidly and had larger standard deviations than in the unsteered case, but for 15° steering, the mean was 0.919 for elements within 1 mm and 0.863 for elements within 3 mm, indicating that accurate arrival time estimation is possible, though perhaps it is advisable to include only shorter lags when estimating the aberration map in these
In the case where only random scatterers are available in the medium, correlation is expected to decrease rapidly with spatial distance as in Fig. 23A. In the presence of a point target in the medium, in theory correlation is expected to remain constant with distance, though in practice there is some decrease as seen in Fig. 23B and 23C. Here we have observed that in practice, correlation at a relatively large distance (3 mm) remains high even for steered unfocused waves.

**Figure 23:** Plots of inter-element correlation versus lateral distance across the receive aperture of a sparse matrix array for (A) random scatterers according to the van-Cittert-Zernicke theorem for both the aperture used in this work (solid line) and that of [59] (dashed line), (B) unsteered transmitted beacon wave measured with tissue-mimicking phantom and skull casting, and (C) transmitted beacon wave steered to 15° off-axis with tissue-mimicking phantom and skull casting. (C) is the average result of steering 15° off-axis in all four directions.

The results of a typical steering angle comparison are presented in Figure 24. They suggest that in a tissue-like medium with an aberrating layer, propagation path differences between different transmit angles are significant enough that correlation
remains below ~0.8 over most of the aperture. For 9 aberrators, mean correlation between a channel receiving an unsteered beacon signal and one receiving a 15° steered signal was 0.788. Examination of these maps reveals that in some isolated spatial locations the aberrator was highly correlated, while over most of the aperture, steering the transmit beacon from (0°, 0°) to 15° off-axis in any direction produced channel data which was poorly correlated between these steering angles, which indicates the presence of unique information for correcting multiple IPs. In instances where the steered data is highly correlated with the unsteered data, the resultant phase maps will be nearly identical and the benefit of multi-IP correction over single-IP correction will be negligible.

**Intra-element correlation across transmit beacon steering angles**

![Maps of maximum correlation coefficient on a per-channel basis for two typical aberrators using a tissue mimicking phantom and physical aberrator (skull casting). Each ring represents a correlation on channel data received on the aperture (Fig. 20). Within each map, the center ring is always 1 (autocorrelation of 1)].
unsteered beacon signal channel data), while the top, right, bottom, and left maps depict that same unsteered beacon data correlated with data acquired using a (0°,15°), (15°,0°), (0°,-15°), and (-15°,0°) steered beacon, respectively. Each channel is correlated with the same channel, but for a different steering angle of the transmitted wave. For these 9 aberrators, the average correlation between a channel receiving an unsteered beacon signal and one receiving a 15° steered signal is 0.788. This is well below the inter-element correlation coefficient of 0.863 at a lag of 3 mm for 15° steered data or .919 at the same lag for unsteered data (Fig. 24).

### 3.3.4 Temporal bone isoplanatic patch

Average measured isoplanatic patch size was 31.8° ± 15.9°, which provided us with an estimate of patch size for this specific scanner, aperture, and correction technique. This was similar to the previous measurements [83, 87] and allowed us to proceed with designing a multi-patch correction system with a 32° IP dimension in azimuth and elevation.

### 3.3.5 Physical aberrator correction and color flow imaging

Dual transducer corrections of color flow volumes for physical aberrators produced an increase in CF voxels above a common threshold for both transducers in 5 of 7 aberrators for the 1-IP correction and 6 of 7 aberrators for the 5-IP correction. Of the 6 successful 5-IP corrections, 5-IP correction produced more CF voxels above the common threshold than 1-IP correction in 5 cases. Mean increase in CF voxels above a common threshold was 25.4% for 1-IP correction and 92.0% for 5-IP correction. Mean decrease in color flow variance for all voxels inside the CF gates was 4.9% for 1-IP correction and 10.0% for 5-IP correction. The improvement in image quality for a single correction may be seen the 3D renderings of Figure 25.
Figure 25: A. Experimental setup (not to scale) for flow measurements in a water tank in the presence of two physical aberrators. Actual probe-to-tube distance was approximately 6 cm. B. Registered, fused rendering without aberration correction. C. Rendering acquired when a single phase map is used to correct the entire 3D volume. D. Rendering acquired when five phase maps are used to correct appropriate regions of the 3D volume.

3.3.6 *In vivo* imaging

Results from imaging Subject 1 are shown in Fig. 26. For each of the two transducers, an axial and a coronal echo slice and a 3D rendered view are shown both before and after performing a 5-IP correction. In the four corrected echo slices, an increase in speckle brightness may be observed, even at the volume edges, indicating an increase in coherence. Brightness increase in sphenoid bone targets is also observable, particularly in the renderings in the last row. This subject’s right window appears to be more favorable than the left. Limited success was observed in consistently visualizing
color flow on this subject’s left side. The following four figures provide additional information on this correction.

In Fig. 27, the in vivo measured arrival time maps from Subject 1 are presented for both transducers. While some correlation between maps acquired with different unfocused wave steering angles is visible, unique aberrators appear to have been measured at different steering angles. Fig. 28 shows the aberrator FWHM correlation lengths and RMS strengths for each IP. The colors indicating different IPs correspond to the pyramid in Fig. 21B. Correlation lengths were comparable to those previously reported for the skull [60, 61, 83]. Aberration was relatively strong in this subject, although within previously published ranges [83].

Fig. 29 presents intra-channel correlation between data acquired with different transmit steering angles and is analogous to the phantom correlation maps shown in Fig. 24. Mean correlation coefficients were 0.652 and 0.702 for the left and right transducers respectively, lower than the phantom measurement of 0.788. Lack of significant spatial trends in Figure 29 suggests that acquired data represent unique IPs based on the spatial correlation definition of the isoplanatic patch [82]. Finally, Fig. 30 gives the increase in brightness on a per-IP basis from this correction. In general, correction of stronger aberrators (Fig. 28) yields greater brightness improvement, as expected. Brightness increases in outer patches were not significantly below those in the
center, indicating that the decreased SNR in steered beacon acquisitions did not yield an unacceptably high number of jitter errors.

Figure 26: The first two rows show aberrated (columns A and C) and corrected (columns B and D) orthogonal echo slices acquired simultaneously by transducers positioned over the right (columns A and B) and left (columns C and D) temporal bone windows of Subject 1. Corrected volumes utilize the five-IP correction scheme; single-IP corrections were not performed on this subject. Mean brightness increases due to aberration correction were 21.2% and 24.2% for the left and right volumes, respectively. The bottom row displays the corresponding 3D rendering of echo and color flow data. Orientation for these renderings is the same as the top “axial” row of images. Blood flow shown is the C5 segment of the internal carotid artery.
Figure 27: Estimated arrival time maps for the five IPs used to perform the in vivo correction of Figure 26. Negative values indicate thick portions of the skull, as incoming signals arrived earlier than the mean arrival time (t=0).
Figure 28: Correlation lengths (top row) and RMS aberrator strength (bottom row) for the Subject 1 correction (Fig. 26) are shown for each IP.
Figure 29: Channel data from the center IP was cross-correlated with the four outer IPs to produce these maps using the in vivo data used to perform the correction of Figure 26. Mean correlation coefficients were 0.652 and 0.702 for the left and right transducers respectively.
Figure 30: Increases in mean brightness of echo data by IP for left and right transducers for the in vivo correction on Subject 1 (Fig. 26). Overall mean brightness increases were 21.2% for the left transducer and 24.2% for the right transducer.

Fig. 31 shows three volume renderings from Subject 2, which demonstrate the impact of single-IP and multi-IP correction on 3D echo + color flow transcranial imaging. These renderings depict blood flow in both internal carotid arteries near the base of the skull and were registered using common color flow voxels as described in Section G of the Methods. Table 1 shows the increases in mean echo brightness and number of visible color-flow voxels for multi-patch in vivo corrections on the two subjects. Note that when averaged over an entire volume, mean brightness may not improve for a
single-IP correction due to the limited isoplanatic patch. Mean echo brightness increase for 4 in vivo corrections (2 windows on each of 2 subjects) was 24.3 ± 9.1%. For the two subjects, mean variance of the color flow data decreased by 16.7%.

Figure 31: Three volume renderings of echo (gray) and color flow (red) for Subject 2 demonstrating the benefits of multi-IP correction. While single-IP correction (B) increases the number of color flow voxels at the center of the scan, brightness of echo data and supra-threshold color flow voxels decrease near the edges of the volume (top and bottom of the figure). Performing a 5-IP correction (C) improves this loss of information at volume edges. In (D), an anatomical reference is shown for both the echo and CF data in the region of interest, where internal carotid arteries enter base of the skull. The top images are photographs of a human skull casting. The bottom image is coronal slice of an illustrative magnetic resonance angiogram (MRA). Original MRA image (D) produced by Ofir Glazer, Biomedical Engineering Department, Tel Aviv University, reproduced with permission of the author.
Table 1: Summary of phase aberration correction data for two human subjects

<table>
<thead>
<tr>
<th>Transducer</th>
<th>Subject 1 5-IP correction</th>
<th>Subject 2 1-IP correction</th>
<th>Subject 2 5-IP correction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean echo brightness increase</td>
<td>21.2%</td>
<td>24.2%</td>
<td>7.54%</td>
</tr>
<tr>
<td>Number of Color flow voxels increase</td>
<td>90.8%</td>
<td>8.51%</td>
<td>3.85%</td>
</tr>
</tbody>
</table>

3.4 Discussion

3.4.1 Timing constraints

Pitch-catch phase aberration correction of multiple isoplanatic patches in 3D demonstrated consistent improvements in color flow imaging in a controlled experiment (tube in water tank) and has also produced improvements in vivo. For future in vivo scanning, it is important to note that the time limitation (the delay update period of 2.5 minutes) is not a fundamental limitation, but a limitation of this particular system, due primarily to the scanner’s bus speed and the time required to write the delay updates.

While this system utilizes 128 channels on each array, it is conceivable that a system with a much higher channel count might be able to reliably perform in vivo 3D color flow imaging without the use of microbubble contrast agent. However, access to digitized channel data is required to perform the proposed methods and is generally unavailable on modern 3D clinical systems due to partial beamforming in the probe
handle [41]. If access to this data were available, the requisite operations for the presented approach (cross-correlation, linear algebra, filtering) could reasonably be performed onboard a clinical scanner. The physical limit on the time for acquiring data to correct \( M \) isoplanatic patches is simply:

\[
t_{acq} = \frac{MNz}{c}, \quad (8)
\]

where \( M \) is the number of independent steering angles for which data is acquired, \( N \) is the number of averages, \( z \) is the separation between the beacon and receiving arrays, and \( c \) is medium propagation velocity. For example, acquiring sets of channel data to correct 5 isoplanatic patches with no averaging and a transducer separation of 14 cm requires 0.45 ms. Therefore, the downtime required to calibrate the scanning system for emergency patients is extremely short. If a second array were to be added to an existing system to be used solely as a transmit beacon (i.e. it would not be able to form images), it would require only triggering from the scanner and the pulsing circuitry necessary to produce a small number of steered planar wave fronts.

### 3.4.2 Direction of arrival estimation

As seen in Fig. 21B, there is a noticeable notch in the correction efficacy of the approach when the two arrays are positioned at \((0^\circ, 5^\circ)\) relative to one another. This is believed to be the result of ambiguity between aberration and a strong off-axis scatterer.
(active source in this case). This is not expected to be a significant problem in vivo, as relative probe angles are usually much larger than 5°, where steering is relatively easy to distinguish from aberration. The optimum approach would be to have transducers positioned at (0°, 0°), as this would maximize SNR and ensure no ambiguity between tilt errors and aberration, but this is difficult to achieve for in vivo imaging given human skull geometry and anatomical variations between individuals. Because both transducers are fixed against the surface of the head, it is important to note that the relative angle between the two transducers is not the same as the angle at which the transmitted wavefront is incident on the skull, which is the angle that was discussed earlier in regards to the stripe artifact and critical angle concerns. It is also worth noting that refraction of the beams at the skull plays a role in transcranial ultrasound. While this may impact image quality and registration, it should have a minor role on direction of arrival estimation for the purpose of estimating higher order aberrators.

3.4.3 Correction of multiple isoplanatic patches

For Subject 1, mean correlation coefficients were 0.652 and 0.702 for the left and right transducers, respectively. As these in vivo values were well below those for the physical aberrator with the phantom (Figure 24) and below the isoplanatic patch definition proposed by Dahl and Trahey (aberrator correlation of at least 0.7 and perhaps as high as 0.9 [82]), these results suggest that further subdivision of the volume into an increased number of smaller IPs may be desirable. However, as discussed, more
time is required as more isoplanatic patches are corrected (increasing $M$ in equation 8). Also, the isoplanatic patch was assumed to be as large in the coronal direction as in the transverse direction. It is most likely smaller in this direction given that the window is smaller in this direction [27], although this cannot be concluded from the observed results.

The aberrator RMS values presented in Figure 28 are quite high for some patches, meaning the effects of aberration correction are larger and more noticeable than in a subject with a weakly aberrating window. This makes sense given the relatively high attenuation in this subject, as attenuation and aberration are often coincident phenomena in the skull [126].

3.4.4 Summary

Having previously demonstrated simultaneous dual probe real-time 3D transcranial ultrasound imaging, proof of concept of a pitch-catch approach to correct skull-induced aberrations in these pulse-echo images has now been demonstrated, even for arrays tilted at arbitrary angles. Furthermore, the first attempt at in vivo correction of multiple isoplanatic patches in the skull has also been presented, with increases in echo brightness observed in all corrected regions of two simultaneously-acquired 3D volumes with a mean value of $24.3 \pm 9.1\%$. In phantom experiments, color flow voxels above a common threshold have also increased by an average of 92% while color flow variance
decreased by an average of 10%, suggesting such a technique may be beneficial for performing non-invasive 3D color flow imaging.
4. Dual transducer imaging outside of temporal acoustic window

4.1 Introduction

In Chapter 2, we described bilateral transcranial imaging using two transducers built on multilayer flex circuits operating at 2.3 MHz (1.8 MHz for Doppler). This chapter describes attempts to overcome poor temporal acoustic windows using a low frequency, large aperture transducer in the same bilateral imaging configuration presented in Chapter 2.

4.1.1 Temporal acoustic window failure

Aaslid et al. first presented the temporal acoustic window as a thinner, more homogeneous region of the temporal bone relative to the rest of the skull through which one-dimensional transcranial Doppler (1D TCD) examinations could be performed [18]. It was later described as a roughly circular region 2-3 cm in diameter having a thickness of 2-3 mm [27, 127]. The decreased attenuation within the window results from structural variation, as the skull within the window consists of an inner and an outer table of compact bone with little or no trabecular bone between them [127]. Grolimund reported a one-way mean attenuation of 7 dB due to transmitting through the window at 2 MHz [128].
However, the idealized view of the temporal acoustic window as a several centimeter-wide region free of trabeculae does not always hold; in many patients a suitable imaging window may not be found. Temporal bone window failure rates in the range of 8% to 29% have been reported (Table 2) [190-196]. Of individuals with window failure, 39% have bilateral window failure [129]. These previous studies were performed in the 2-3 MHz range; none used 3D ultrasound. While microbubble contrast-enhancement may reduce window failure rates, it does not eliminate window failure in all patients [192, 193, 195]. In Fig. 32, we present an example from previous results of simultaneous bilateral real-time 3D transcranial ultrasound imaging showing a favorable and a less favorable window in two adult female subjects scanned with microbubble contrast enhancement at 1.8 MHz.
Table 2: Comparison of window failure rates in previous transcranial ultrasound studies

<table>
<thead>
<tr>
<th>Study</th>
<th>Number of subjects</th>
<th>Study population</th>
<th>Transmit frequency</th>
<th>2D imaging or 1D TCD</th>
<th>Window failure rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hashimoto et al. 1992</td>
<td>423</td>
<td>Japanese</td>
<td>2.0 MHz</td>
<td>1D TCD</td>
<td>29%</td>
</tr>
<tr>
<td>Seidel et al. 1995</td>
<td>84</td>
<td>German</td>
<td>2.5 MHz</td>
<td>2D imaging</td>
<td>20%</td>
</tr>
<tr>
<td>Baumgartner et al. 1997</td>
<td>33</td>
<td>Swiss</td>
<td>2.0-2.5 MHz</td>
<td>2D imaging</td>
<td>34% (with contrast agent)</td>
</tr>
<tr>
<td>Marinoni et al. 1997</td>
<td>624</td>
<td>Italian</td>
<td>2.0 MHz</td>
<td>1D TCD</td>
<td>8%</td>
</tr>
<tr>
<td>Postert et al. 1997</td>
<td>172</td>
<td>German</td>
<td>2.25 MHz</td>
<td>2D imaging</td>
<td>12%</td>
</tr>
<tr>
<td>Hoksbergen et al. 1999</td>
<td>112</td>
<td>Dutch</td>
<td>2.0-2.5 MHz</td>
<td>2D imaging</td>
<td>11%</td>
</tr>
<tr>
<td>Gahn et al. 2000</td>
<td>171</td>
<td>German</td>
<td>2.0-2.5 MHz</td>
<td>2D imaging</td>
<td>29%</td>
</tr>
<tr>
<td>Krejza et al. 2007</td>
<td>90</td>
<td>American</td>
<td>2.0 MHz (1D TCD) 2.5 MHz (2D imaging)</td>
<td>Both</td>
<td>11% (both 1D TCD and 2D imaging)</td>
</tr>
</tbody>
</table>
Figure 32. Three-dimensional contrast-enhanced (Definity®) renderings acquired from scanning two different healthy volunteers (both 24-year-old females) with 2.3 MHz echo and 1.8 MHz Doppler. Subject A had significantly more favorable windows than Subject B, in which the major cerebral arteries appear small and Doppler signal dropout is apparent. MCA=Middle cerebral artery, ICA=internal carotid artery.

4.1.2 Window failure demographics

While presenting window failure rate as a single number as in Table 2 provides a starting point for understanding the prevalence of this problem, it is also an oversimplification in that results are averaged over gender, age groups, and ethnic groups. The success rate of transtemporal imaging remains low in certain populations, particularly elderly women and non-whites [130-138]. Halsey found window failure rates of 13% in white males, 18% in white females, 23% in black males and 50% in black females at 2.0 MHz [138]. Alternatively, Hoksbergen reported failure rates of 1% in
white men and 23% in white women over age 60 [139]. In a Japanese population, Hashimoto reported a significant increase in window failure rates beginning at age 50, especially in women [137]. Hyperostosis—thickening of the skull’s inner table—has been proposed as a possible explanation for window failure in elderly women [134]. Hyperostosis has been observed in 6% to 12% of adult women of all ages (as compared with only 1% of men) [205, 206] and in 50% of women over the age of 60 at time of death [140].

The groups having the highest incidence of window failure are some of the same populations which are at greatest risk for stroke. In the United States, the incidence of stroke among African-Americans is almost double that of white Americans, and twice as many African-Americans who have a stroke die from it compared to white Americans [141]. Women are less at risk for stroke than men, although on average women have strokes at a greater age than men and are more likely to die [141].

4.1.3 Addressing window failure

Given an uncertain rate of window failure in these at-risk populations varying from perhaps 20% to 50%, we propose performing 3D diagnostic transcranial ultrasound at lower frequencies, near 1 MHz. For most diagnostic equipment—either imaging or non-imaging transcranial Doppler—this would require new transducers capable of operating at lower frequencies. In this article we describe the design, fabrication, and testing of a new prototype 3D bilateral imaging device with matrix array transducers
operating at 1.2 MHz with the goal of performing transcranial imaging either outside of the window or in patients exhibiting window failure.

Previous efforts have attempted to address the effects of poor temporal acoustic windows by using the other window [129] or by finding the optimal probe placement [71]. Other attempts at overcoming the skull in ultrasound imaging include inducing shear-mode conversions at the soft tissue-skull interfaces [29, 30] and various phase aberration correction techniques including multi-lag least squares approaches [71, 72, 74, 75] and spatiotemporal inverse filtering [87, 122, 123]. In this chapter we will focus on attenuation rather than aberration. It is worth noting that at lower frequencies, the effect of aberration is also diminished, as the root-mean-square amplitude—or “strength”—of the aberrator constitutes a smaller fraction of a wavelength at 1 MHz relative to 2 MHz [142, 143].

Previous probe designs for transcranial ultrasound include a low-frequency 1D TCD probe [211, 212], therapeutic arrays for high-intensity focused ultrasound [144-146], a combined probe for imaging and thrombolysis [50, 147], and a 128-element, 750 kHz linear array [52]. Outside of our own previous work, no known previous attempts at matrix arrays for transcranial imaging have been made.
4.2 Methods

4.2.1 Frequency Analysis

Investigations in transcranial therapeutic ultrasound [148] and non-imaging TCD [149] also support decreasing transmit frequency from greater than 2 MHz to the 1 to 2 MHz range as a means of overcoming insufficient bone windows. In imaging, Hölscher et al. have used a 1 MHz transmit pulse in the development of a harmonic contrast-enhanced transcranial imaging technique, though the transducer arrays used in this work were 1D and 1.5D arrays centered at 2.5 MHz [150].

This evidence suggested constructing a lower frequency imaging array in order to potentially acquire images either from probes placed outside the temporal acoustic window or in patients with window failure. Seeking theoretical basis for this rationale, we have examined a simple model for answering the question of which frequency is best for transcranial ultrasound imaging.

The most straightforward model is the scattering model developed in the 1970s [151]. For the continuous wave case, a design frequency ($f$) for maximizing the backscattered echo in an attenuating medium for a given depth ($R$) and attenuation ($\beta$) can be determined by taking the product of the expression for frequency-dependent attenuation and the square of frequency (Rayleigh scattering is assumed), then setting the derivative equal to zero:
\[ p = p_0 \exp(-2\beta f R) f^2 \]  \hspace{1cm} (9)

\[ \frac{\partial p}{\partial f} = p_0 \left[ 2 f \exp(-2\beta f R) - f^2 (2\beta R) \exp(-2\beta f R) \right] = 0 \]  \hspace{1cm} (10)

\[ 2f = f^2 (2\beta R) \]  \hspace{1cm} (11)

\[ f = \frac{1}{\beta R} \]  \hspace{1cm} (12)

Using a conversion factor of 0.1151 Nepers/dB, in the case of transcranial imaging with \( R_1 = 2 \text{ mm of skull at } \beta_1 = 2.8 \text{ dB/cm/MHz} \) and \( R_2 = 7 \text{ cm of brain parenchyma} \) (approximately to the midline) at \( \beta_2 = 1.0 \text{ dB/cm/MHz} \), Equation 12 yields a design frequency of 1.1 MHz. Outside of the window \( (R_1 = 4 \text{ mm, } \beta_1 = 20 \text{ dB/cm/MHz}) \), the design equation produces a frequency of 0.6 MHz. This is far too low for our scanner, which cannot transmit below 1.2 MHz, and leads to serious concerns regarding the size and resolution of an array that can be reasonably built and coupled to a human head. The inability to transmit below 1.2 MHz also precludes harmonic imaging with a transmit frequency of 0.6 MHz and a receive frequency of 1.2 MHz. While this is a simple calculation for the continuous wave case, this corroborates the view developed by other investigators and based on our own experience scanning humans: a probe having a resonant frequency of approximately 1 MHz, which is lower than those commonly used, may be advantageous.
4.2.2 System SNR considerations

Next we estimated signal to noise ratio (SNR) for our system as a function of increasing skull thickness. Based on acquired data, we used the beamformed SNR value computed from point target echoes in a water tank as a starting point. This SNR was then two-way attenuated by 7 cm of soft tissue and 0.3 cm of skull inside the acoustic window versus 7 cm of soft tissue and 0.4 cm of skull outside of the window, while the signal amplitude was assumed to vary with frequency according to $f^2$ (Rayleigh scattering). The plots of SNR versus frequency (Fig. 33) indicate that (1) SNR falls off very rapidly for 1.8 MHz outside of the window and (2) for reasonable skull thicknesses outside of the window, 1.2 MHz would be expected to significantly outperform 1.8 MHz, generally by at least 5 dB. Under these parameters, imaging outside the window at 1.8 MHz is not possible for skull thicknesses beyond 0.25 cm.
As previously noted, the ability to perform transcranial color flow imaging is of particular interest due to its potential diagnostic utility. Given the SNR of Figure 33, this signifies that: (1) for our system, color flow imaging will be very difficult given that blood scattering generally lies 20-40 dB below tissue echoes [75], (2) in order to have a chance at imaging blood outside of the window, the design model should reflect color flow SNR rather than just echo SNR, and (3) although it is desirable to perform
transcranial color flow imaging without contrast enhancement, the necessity of using microbubble contrast agent seems likely with this system. In light of consideration (2), we will briefly consider system color flow SNR.

### 4.2.3 Color flow SNR

Color Doppler SNR is largely determined by the operation of the stationary target canceller, or “wall” filter. The most common types of wall filters are the single- and double- delay-line cancellers, which remove stationary echoes by weighted subtraction of two or three successive lines of demodulated, digitized data. While efficient to implement, these filters also have zeros in their magnitude spectra which depend on the flow velocity and pulse repetition frequency (PRF) [75].

Assuming that only the flow signal remains after wall filtering, the phase of the complex autocorrelation is estimated for each range gate of each line to produce the velocity map which is superimposed on the B-mode image. In most systems, thresholding determines the write priority of each displayed pixel, i.e. whether a pixel displays grayscale or color information. The quantity that is thresholded is not estimated velocity—which may alias if the PRF is too low—but the magnitude of the complex autocorrelation estimate [152], determined primarily by the wall filter with its PRF-dependent sampling and by system sensitivity. In addition, because the wall filter cancels the highest-amplitude signals, the remaining blood flow signals suffer from low
dynamic range since they occupy only the least significant bits during analog-to-digital conversion.

4.2.4 Maximization of stationary-to-blood scattering ratio model

While the objective of color Doppler imaging from a diagnostic perspective is to accumulate a map of blood flow velocities which may be superimposed on a B-mode image, in the present work the magnitude of the Doppler signal is of greater interest than the velocity (or Doppler frequency) being measured due to the low SNR in transcranial imaging.

In addition to SNR, dynamic range also becomes an important consideration for sensitivity in color flow imaging systems. Because the stationary echoes have a much larger amplitude than signal from blood, when the scanner digitizes the data, most of the bits of the analog-to-digital converter (A/D) are consumed by the stationary echoes, resulting in poor dynamic range of the blood signal that remains after wall filtering. For this reason, many scanners compensate for this in some manner—for instance by increasing the level of amplification prior to digitization or by utilizing a separate processing chain for Doppler data—when operating in a Doppler mode (CF, spectral, or power). The scanner used in this work, however, has no such compensation.

If it is assumed that all available dynamic range, i.e. all bits of the first A/D, are used to digitize the strongest stationary echo, then in order to preserve as much dynamic range as possible for the blood signal, the appropriate metric to be optimized
would the ratio of blood scattering to tissue scattering with respect to frequency. In this vein, the ratios of blood to tissue backscattering coefficients as a function of frequency have been plotted for three cases: unattenuated, soft tissue and compact bone attenuation (i.e. scanning in the window), and soft tissue and trabecular bone attenuation (i.e. scanning outside of the window) (Fig. 34). Tissue scattering data used in this analysis were obtained by performing a polynomial fit to the data of the Shung [153]. This figure suggests that in the absence of attenuation, a Doppler ultrasound system achieves highest Doppler dynamic range at 4.67 MHz. In the presence attenuation due to parenchymal brain tissue and skull, the highest Doppler dynamic ranges are achieved at 1.74 MHz within the acoustic window and 1.08 MHz outside of the window. This implies that the transducer design frequencies used in this work have been nearly optimal for maximizing the ratio of blood to tissue scattering assuming that the scanner response is flat with respect to frequency in all other respects (for example, applied transmit voltage, receive sensitivity).
One potential flaw in this analysis results from the fact that the highest amplitude echoes will result from the skull reflections present in the first few millimeters of the image. The preceding analysis effectively assumes that these echoes are discarded and thus do not impact system dynamic range.
4.2.5 Blood flow velocity estimation

Blood velocities are estimated using a time-domain estimator such as the 1D [152] or 2D autocorrelator [154], and therefore are subject to the Cramér-Rao lower bound (Equation 4, Section 3.2.1). Fig. 35 shows the Cramér-Rao lower bound with increasing skull thickness for both 1.2 MHz and 1.8 MHz using measured system parameters. Backscattering proportional to \( f^2 \) was assumed. Since mean temporal bone thickness is usually no greater than 0.5 cm—even outside of the window—this shows that for all realistic skull thickness, velocity estimates at 1.8 MHz are expected to have fewer jitter errors than those taken at 1.2 MHz. At a thickness of 0.4 cm, the Cramér-Rao lower bound is 58.6 ns at 1.2 MHz and 32.2 ns and 1.8 MHz. The use of broader bandwidth probes would improve these Cramér-Rao lower bounds.
Figure 35. Cramér-Rao lower bound (CRLB) as a function of skull thickness for two frequencies. Parameters used in these plots are: bandwidth = 50%, unattenuated SNR = 32.6 dB, $\rho = 0.945$, $T = 0.65 \mu$s.

While in this work we are primarily concerned with the magnitude of the Doppler signal, jitter in the velocity estimate increases with increasing skull thickness due to decreasing SNR. Color Doppler SNR can be increased by increasing the packet length (number of averages), but this comes at the cost of decreasing frame rate, which is an undesirable tradeoff in real-time 3D ultrasound. Increased jitter in the velocity estimate is potentially clinically important because erroneous velocity estimates may lead the vascular sonographer or physician to incorrectly categorize blood flow in a vessel as within normal ranges or diseased [5]. For instance, at a carrier frequency of 1.2 MHz and a PRF of 4.5 kHz, a jitter error of 50 ns between two consecutively acquired
signals produces an error of 3.3 m/s when using a 1D autocorrelation approach [152]. This error is relatively small considering expected systolic blood flow velocities of approximately 50 m/s in the internal carotid artery and 100 m/s in the middle cerebral artery [5]. The effect of this error is clearly more harmful when measuring slower flow velocities. Jitter does not affect the presence or absence of blood flow in a qualitative image; here SNR is the limiting factor.

### 4.2.6 Transducer design

In designing an approximately 1 MHz sparse matrix array, the spatial dimensions of the 2.5 MHz aperture previously designed for transcranial imaging through the window [63] were scaled by approximately 2.5 in order to maintain resolution at the lower frequency. The resulting array has an element pitch of 0.725 mm and an aperture diameter of 25.3 mm (Fig. 36A). Simulations of the two-way beam for the 1 MHz array were produced using Field II (Fig. 36B, C, D) [74]. -6 dB beam width for the unsteered response is 3.9 × 3.9 mm at a depth of 7 cm, an approximate measure of spatial resolution in the lateral dimensions.

The proposed matrix array acoustic stack—consisting of ground foil, PZT, flex circuit, and acoustic backing—was simulated using the KLM model (PiezoCAD, Sonic Concepts, Bothell, WA) [65, 155]. The transducer was simulated with water at the front acoustic port, and additionally with the appropriate loads for imaging inside and outside of the temporal acoustic window using the data of Pichardo et al. for cortical
and trabecular bone [90]. Electrical parameters were set to system values. Acoustic parameters for these simulations are given in Table 3. Transducer impulse response and the response to a 3 cycle, 1.2 MHz excitation waveform were simulated.

Table 3. Materials parameters for KLM model simulations

<table>
<thead>
<tr>
<th>Material</th>
<th>$Z_0$</th>
<th>Thickness</th>
<th>$\alpha$ (attenuation)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PZT-5H (TRS HKI-HD)</td>
<td>26.95 MRayl</td>
<td>1.37 mm</td>
<td></td>
</tr>
<tr>
<td>Extracranial Tissues</td>
<td>1.54 MRayl</td>
<td>5 mm</td>
<td>1 dB/cm/MHz</td>
</tr>
<tr>
<td>Skull (inside window)</td>
<td>4.70 MRayl at 1.2 MHz, 4.43 MRayl at 1.8 MHz</td>
<td>3 mm</td>
<td>20 dB/cm/MHz</td>
</tr>
<tr>
<td>Skull (outside of window)</td>
<td>4.38 MRayl at 1.2 MHz, 4.06 MRayl at 1.8 MHz</td>
<td>4 mm</td>
<td>30 dB/cm/MHz</td>
</tr>
<tr>
<td>Brain</td>
<td>1.54 MRayl</td>
<td>140 mm</td>
<td>1 dB/cm/MHz</td>
</tr>
</tbody>
</table>
Figure 36. (A) Aperture design for 1 MHz sparse array. Gray elements only transmit; white elements both transmit and receive. Simulated two-way response of this aperture (B) on-axis, (C) steered to +16° in both azimuth and elevation, and (D) steered to +32° in both azimuth and elevation. These beams are virtually identical to previous 2.5 MHz simulations (Fig. 8)[63].

4.2.7 Transducer fabrication

The simulated transducer described in Fig. 36 and Table 3 was built on a custom flexible multilayer circuit having a polyimide substrate, 5 µm-thick copper traces, and 9 µm copper pads with ENIG (electroless nickel, immersion gold) finish, with pads arranged as in Fig. 36A (Microconnex, Snoqualmie, WA). Total thickness of this flex
circuit was 209 μm, or 0.13λ at 1 MHz. A 27 × 27 × 1.37 mm piece of high dielectric constant PZT-5H (TRS Technologies, State College, PA) was bonded to the flex circuit by screen printing a silver-loaded anisotropic conductive epoxy (Chomerics, Woburn, MA) onto the back of the PZT. The bonded PZT was then diced using a programmable dicing saw (Disco DAD3220, Tokyo, Japan). A photograph of an array after dicing is shown in Fig. 37. Kerf widths are approximately 90 μm. Ground foil was screen printed with conductive epoxy and bonded to the diced PZT, then the transducer was sealed. Finally, a lossy acoustic backing was cast and bonded to the back of the flex circuit. Two completed arrays were connected to a custom printed circuit board (PCSM/Moog Components, Galax, VA) via 300-pin connectors (Samtec BTH/BSH, New Albany, IN). This board mates with the scanner front end.

Figure 37: (A) Optical photograph and (B) micrograph of an array after dicing. Gold pads extending beyond the PZT in (A) are extra pads used only for alignment. Element pitch is 0.725 mm.
4.2.8 Transducer testing

The following measurements were made in a water tank on fabricated arrays: pulse-echo sensitivity/bandwidth, 50 Ω insertion loss, and crosstalk. Pulse-echo testing was performed on one channel at a time by positioning an aluminum block reflector as close as possible to the face of the transducer. The transducer was excited using a pulser-receiver (Panametrics 5073PR) and power amplifier (ENI 525). Results were viewed on an oscilloscope (Tektronix 744A) and on a spectrum analyzer (HP3588A) with the use of a stepless gate (Panametrics 5065A). Insertion loss testing was performed with a 50 Ω source transmitting a 5 cycle, 1 MHz pulse. Element yields were tested during fabrication using a complex impedance analyzer (HP4194A) and after connecting the transducer to the scanner using a Signatec PDA14 (Corona, CA) to digitize single channel echoes from a point target.

4.2.9 Safety measurements

Measurements in this section were made using a calibrated membrane hydrophone (Sonic Tech, Hatboro, PA) in a water tank. The hydrophone was positioned at the transducer focus (7 cm) and peaked. Acoustic output measurements were made while scanning in spectral Doppler mode with the highest transmit power level to measure worst-case values.
Mechanical index (MI) is used to estimate the potential for mechanical bioeffects, although no adverse effects in humans have been reported due to exposure to diagnostic-level ultrasonic pressures [156]. MI is defined as:

$$MI = \frac{p_r}{C_{MI} \sqrt{f}}$$ \hspace{1cm} (13)$$

where $p_r$ is peak rarefactional pressure, $C_{MI} = 1.0$ MPa/MHz$^{1/2}$. The threshold for cavitation is reduced in the presence of microbubble contrast agents, though contrast-enhanced color Doppler sonography at approved diagnostic output levels has not shown any signs of damaging brain tissue or opening the blood-brain barrier [157]. The FDA limit for MI is 1.9.

The cranial bone thermal index (TIC) is a measure of heating at the transducer-bone interface [156]. There is no FDA limit for TIC and it is infrequently reported in literature, though it is part of the output display standard [158].

### 4.2.10 Phantom testing

Phantom testing was conducted in a water tank with side-viewing silicone rubber panels with 5 mm diameter latex tubing passing through the center of the tube in a U-shape to mimic a blood vessel. Gravity-fed seltzer water was made to flow through the tubing at approximately 50 cm/s, similar to systolic flow velocity in the internal carotid artery ($66 \pm 16$ cm/s) [5]. A 3 mm attenuator comprised of Wallgone absorbing rubber (Consumer Usage Labs, Parkville, MD) was placed in front of each transducer,
producing two-way attenuation of 45.2 dB at 1.8 MHz and 32.9 dB at 1.2 MHz. This is similar to two-way *in vivo* attenuation outside of the temporal acoustic window, which is computed to be 43.2 dB at 1.8 MHz and 28.8 dB at 1.2 MHz (assuming 0.4 cm at 30 dB/cm/MHz) [28, 87].

### 4.2.11 *In vivo* imaging

As a surrogate for window failure, we scanned all subjects outside of their temporal acoustic windows by placing the probes above the pinnae of the ears, superior to the window by approximately 2 cm. An *in vivo* comparison between dual flex transducers at 1.8 MHz and 1.2 MHz was performed in a 64-year-old male. This subject was first scanned at 1.2 MHz with the probes described in the Section 4.2.7 placed superior to the temporal acoustic windows by approximately 2 cm. On-screen image quality was optimized by adjusting the overall gain and the time gain control. Next, the subject was scanned with 1.8 MHz probes fabricated as described previously [63]. Care was taken to position the probes in the same position on the skull and to acquire the same field of view as with the lower frequency arrays.

Five additional subjects were then scanned with Definity (Lantheus Medical Imaging, North Billerica, MA) microbubble contrast agent enhancement only at 1.2 MHz after giving informed consent according to a protocol approved by the Institutional Review Board of Duke University. As there was not time to change and reposition transducers or modify scanner settings after injection, these subjects were not scanned at
1.8 MHz. The five subjects consisted of 4 males, 1 female; 4 Caucasians and 1 African-American. Data for each subject were reviewed in offline 3D renderings and in the scanner’s orthogonal 3D display to assess visibility of each of the following vessels: left and right internal carotid arteries, left and right middle cerebral arteries, left and right posterior cerebral arteries, and anterior carotid artery (treated as a single structure). The presence of anatomical landmarks such as the sphenoid bone and *falx cerebri* in B-mode echo imaging was also assessed.

For each subject, the two transducer arrays were positioned approximately 2 cm superior to the bilateral temporal acoustic windows. Imaging was performed during positioning to ensure adequate contact and determine imaging field of view. Transducers were held in place using an EEG cap (Jordan NeuroScience, San Bernardino, CA). Each subject was then given an intravenous bolus injection of 10 μL/kg followed by a 10 mL saline flush according to the Definity prescribing information. After a 30 minute delay, a second bolus injection was administered to allow acquisition of additional images.

### 4.3 Results

#### 4.3.1 KLM model simulations

Results of KLM model simulations for the designed 1 MHz transducer are presented in Table 4. According to this model, the presence of the skull is expected to decrease SNR by 24.88 dB (inside the window) and 29.83 dB (outside of the window).
For comparison, for the 1.8 MHz transducer, these figures are 38.68 (inside the window) and 43.00 dB (outside of the window). Fig. 38 shows the simulated pulse-echo waveform and spectrum showing a center frequency of 1.08 MHz produced by transmitting and receiving into water with a single element. Additional results of transducer modeling using the KLM model are given in Appendix A.

<table>
<thead>
<tr>
<th></th>
<th>Without skull</th>
<th>With skull (within window)</th>
<th>With skull (outside of window)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pulse-echo amplitude</td>
<td>0 dB</td>
<td>-29.84 dB</td>
<td>-36.96 dB</td>
</tr>
<tr>
<td>relative to no skull case</td>
<td>(reference)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Center frequency (f₀)</td>
<td>1.08 MHz</td>
<td>1.00 MHz</td>
<td>0.95 MHz</td>
</tr>
<tr>
<td>-6 dB bandwidth</td>
<td>32.89%</td>
<td>17.75%</td>
<td>19.61%</td>
</tr>
</tbody>
</table>

Figure 38. Simulated pulse-echo impulse response (A) and spectrum (B) for a single element using PiezoCAD. The units on the ordinate axes are measured output voltage (Vₒ) relative to applied input voltage (Vᵢ).
4.3.2 Transducer testing

The single-element waveform and spectrum shown in Fig. 39 are similar to the simulation results, although the strong harmonics do not occur in the simulation. Bandwidth is 46.86%, centered at 1.0 MHz. 50 Ω insertion loss was measured to be -58 dB. This is an improvement over the 2.3 MHz transducer (-80 dB), as the larger elements provide better electrical matching. Worst-case crosstalk on adjacent channels in the flex circuit was -46.90 dB. This improvement in crosstalk as compared to the previous 2.3 MHz transducers (-30 dB) is due to increased trace separation in the flex circuit allowed by a larger aperture [63]. Lower cross talk in the transducer is expected to improve imaging signal to noise ratio and angular sensitivity. Element yields of the two transducers used for simultaneous imaging were 91% and 95%.

![Figure 39](image)

Figure 39. (A) Typical single element pulse-echo response and (B) average spectrum from all single-element acquisitions. The simulated spectrum of Fig. 38 is reproduced for comparison (dashed line).
4.3.3 Safety measurements

Measured worst-case MI for this transducer is 0.33. Though operating at a lower transmit frequency increases MI, the output of these transducers is well below both the FDA limit and reported MIs for previous *in vivo* transcranial imaging studies [150, 157]. In addition, Definity® microbubbles are reported to exhibit a slight decrease in energy absorption with decreasing frequency [223, 224], decreasing the potential for cavitation as compared with a transducer having the same acoustic output but operating at a higher frequency.

Measured worst-case value for TIC for this transducer is 1.66. As a rough comparison with a different thermal index, it may be noted that soft tissue thermal indices (TIS) of greater than 2 are often reported in the literature. Measured worst-case $I_{ppa}$ is 2.4 W/cm$^2$ and $I_{pta}$ is 0.068 W/cm$^2$.

4.3.4 Phantom imaging

The results of three phantom experiments are shown in Fig. 40ABC, in which fused renderings clearly depict flow in the expected spatial pattern at 1.2 MHz, while only the outermost legs of the tube are visible when scanning at 1.8 MHz.
Figure 40. Results of three separate tests of water tank flow testing in the present of 3 mm Wallgone rubber using two different sets of transducers at the indicated frequencies. Attenuation is 28.8 dB at 1.2 MHz and 43.2 dB at 1.8 MHz.

4.3.5 In vivo imaging

As seen in Fig. 41, at 1.8 MHz it was not possible to determine a threshold at which intracranial structures could be visualized, while the reduced attenuation at 1.2 MHz allows visualization of several hyperechoic structures, including the choroid plexus in the lateral ventricles and the sphenoid bone inferiorly.

Based on these initial results, additional subjects were imaged outside the window using microbubble contrast agent according to an IRB-approved protocol as described in section 4.2.11.
Figure 41. Fused renderings of the same subject using the described 1.8 MHz (A) and 1.2 MHz (B and C) imaging system. Coronal views are shown in (A) and (B) and an axial view in (C). Both data sets were acquired outside of the temporal acoustic window.

For 1.8 MHz imaging, we were unable to find a threshold setting which allowed visualization of anatomical structures. In the 1.2 MHz image, anterior horns of lateral ventricles, sphenoid bone, and contralateral skull surfaces are visible.
Figure 42. Registered, fused rendering of a healthy 34-year-old male with Definity® microbubble contrast enhancement. (ACA = anterior cerebral artery, ICA = internal carotid artery, MCA = middle cerebral artery, PCA = posterior cerebral artery.)
Table 5. Detection rates of major cerebral arteries through the acoustic windows in 3D Doppler imaging at 1.8 MHz versus outside the acoustic windows at 1.2 MHz. (ACA = anterior cerebral artery, ICA = internal carotid artery, MCA = middle cerebral artery, PCA = posterior cerebral artery.)

<table>
<thead>
<tr>
<th>Doppler scanning frequency</th>
<th>Probe placement</th>
<th>Detection rates</th>
<th>Echo landmarks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Within temporal acoustic windows</td>
<td>1/4 8/8 8/8 6/8</td>
<td>4/4</td>
</tr>
<tr>
<td>1.8 MHz</td>
<td>Outside of acoustic windows</td>
<td>4/5 8/10 9/10 3/10</td>
<td>8/10</td>
</tr>
<tr>
<td>1.2 MHz</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

An offline 3D rendering of contrast-enhanced Doppler data in a single subject is shown in Figure 42. The vessels of the Circle of Willis can be seen in this image. Table 5 summarizes the detectability of arteries of the Circle of Willis in Doppler imaging and of anatomical landmarks in B-mode imaging in this study and compares with the previous 1.8 MHz frequency study through the temporal acoustic windows (Section 2.2.4). Bilateral anterior cerebral arteries are treated as a single structure, as they commonly appear in Doppler ultrasound images [5, 159]. Detection rates were slightly lower than with higher frequency 3D imaging through the temporal acoustic window, but as discussed in Section 4.2.2 (Fig. 33) imaging in this 1.2 MHz study was performed through a much more highly attenuating region of the skull. Two of five subjects exhibited a partial or complete unilateral failure to image, suggesting window failure will persist in some subjects even at a lower frequency; Fig. 33 suggests this occurs when
the skull becomes too thick on one side. Both of these subjects were, however, successfully imaged through a single side. An example of single-sided imaging failure is shown in Figure 43.

Figure 43. An example of unilateral imaging failure in one subject, a 32-year-old African-American male. Imaging with the low frequency array does enable imaging through the right side of the subject’s skull outside of the temporal acoustic window, but not through the left side. (ACA = anterior cerebral artery, ICA = internal carotid artery, MCA = middle cerebral artery, PCA = posterior cerebral artery.)
4.4 Discussion

By designing and fabricating lower frequency, large aperture matrix array probes, we have demonstrated the potential for performing 3D transcranial ultrasound imaging either outside of the windows or in patients with poor temporal bone windows. Improvement in 3D echo imaging was significant at 1.2 MHz as compared to 1.8 MHz. However, due to the SNR requirements for color flow imaging, microbubble contrast agent was still required.

This study merely examined feasibility and was not comprehensive given the aforementioned varying window characteristics among individuals. A comprehensive study of window failure across ethnic groups and ages would provide greater insight into the prevalence of this problem, especially a study examining several different transmit frequencies.

Transducer performance could be improved by fabricating an array with one or more matching layers or a composite transducer (either with or without a matching layer), which are standard practices among commercial probe manufacturers. A polymer-ceramic composite transducer is expected to provide higher bandwidth and improved acoustic matching to soft tissue but inferior electrical matching due to the decrease in relative dielectric constant [160]. Mills and Smith demonstrated improved electrical matching using multi-layer composite structures [161, 162]. Cost and complexity of fabrication are higher for multi-layer than single-layer composites, which
in turn are higher than single-layer ceramic transducers. However, the large size of the proposed 1 MHz array is expected to improve ease of manufacturing for dice-and-fill approaches.

Detection rates of middle cerebral and internal carotid arteries were slightly lower when imaging outside the window at 1.2 MHz as compared to imaging through the temporal acoustic window at 1.8 MHz in the previous study [63]. Nonetheless, to our knowledge transcranial ultrasound imaging outside of the temporal acoustic window has not been demonstrated previously. The presented results suggest that a large, low-frequency (~1 MHz) array may allow imaging outside of the window, which would prove helpful both for imaging subjects with poor acoustic windows and for allowing operators with minimal training to image subjects from an ambulance or other field setting. In a small study, the ability to detect blood flow in internal carotid and middle cerebral arteries is encouraging, especially when one considers that 68% of strokes affect the anterior circulation, and of these 96% occur in the middle cerebral arteries [163].

When using a large aperture, different elements are more likely to encounter different aberration and attenuation due to the inhomogeneity of the skull. However, these effects are partially counteracted by the low frequency. In the future, heating at the probe surface could become a concern due to energy deposition at the skull, however because a probe with a large surface area spreads out energy, switching to a
large array at the same power setting is thus less of a concern than increasing the transmit power when using a small array.

This chapter has presented a system for overcoming temporal acoustic window limitations in transcranial imaging which relies on a bilateral approach using low frequency, large aperture sparse arrays. It is our hope that when combined with recent developments in portable ultrasound, such an approach may be useful to allow acquisition of useful images either in individuals with poor windows or outside of the temporal acoustic window by untrained operators in the field.
5. Future work

5.1 Introduction

This chapter details potential areas for future investigation illuminated by this research. System design considerations, phase aberration correction, and transcranial ultrasound propagation are discussed. Finally, future impact and potential for this research is assessed.

5.2 System design considerations

5.2.1 Probe frequency

While there are some advantages to transmitting at a lower frequency (near 1 MHz), there are also disadvantages including increased jitter and manipulation of a larger array, which can be more difficult to position for imaging through the temporal acoustic window. To this end, commercial manufacturers have investigated creating an on-screen metric to aid sonographers in probe placement [71]. Modern commercial arrays typically boast nominal bandwidths of 1-4 MHz, for example, so it should be possible for the sonographer to simply decrease the transmit frequency for patients with difficult windows, meaning a unique low frequency probe may not be required. In practice, probe sensitivities are not flat over their bandwidths and some scanners do not support certain modes such as color flow imaging over the entire bandwidth of a probe, though this should be technically feasible.
5.2.2 Matrix array considerations

As of this writing, there are still unresolved questions pertaining to interconnect technology for matrix arrays, and the clinical future of 3D ultrasound remains in some doubt. Specifically, matrix arrays are very costly to manufacture and require from a few hundred to thousands of interconnects between the probe and the scanner. Most modern scanners have circumvented this problem by performing partial beamforming with custom application-specific integrated circuits (ASICs) in the probe handle. Because of in-handle beamforming electronics and the poor impedance match (aided by in-handle impedance matching circuits), thermal management becomes an issue for matrix arrays. Transducer heating can only increase thermal noise at the front end, the most costly location to introduce system noise because it will only be amplified by later stages. For this reason preamplifiers and cable-driving circuits have also been incorporated into the ASIC package in the handle.

In-handle beamforming also removes the ability to process channel data, eliminating many potential applications in areas such as phase aberration correction, adaptive beamforming, lateral filtering, speckle tracking, and shear wave elasticity imaging. Of course, this is of greater concern for research than for clinical imaging at this time. In designing a compact 3D imaging system only for clinical imaging, a truly cableless system would be most desirable—i.e. the scanner is in the probe (or vice versa). For instance, a transcranial imaging system could be designed with two or three arrays.
embedded in a cushion for the patient’s head [62]. Because the entire scanner is housed in the cushion, the interconnect length between the probe and the scanner is virtually zero. Each channel can have its own impedance matching circuit followed by its own preamp, both positioned immediately adjacent to the element. The problem with array elements connecting directly to the front end board is the acoustic backing. Arrays built on printed circuit boards lack adequate bandwidth and angular response, leading us back to an interconnect composed of at least one flex circuit, which has a lower acoustic impedance than FR-4 printed circuit boards and may have a thickness less than $\lambda/10$.

### 5.2.3 Alternative system configurations

In considering the performance of a 3D system, perhaps the most important design consideration is channel count. As a quick and easily implementable solution, in this work the channels have simply been split between two transducers. Other options that have been considered but not implemented are discussed in this section.

One option is a system which uses receive-mode multiplexing to allow for two transtemporal arrays with 96 transmit only and 192 transmit-receive elements each and one suboccipital array with 96 transmit only and 256 transmit-receive elements each. Each receive channel is multiplexed to receive data from one of three probes. This could allow for sequential scanning from any of the three or near-real-time by alternating between the three. A complex programmable logic device (CPLD) would be used to control the multiplexers. For comparison, this system is compared to a three-probe
system without multiplexing which has three sparse matrix arrays, two transtemporal arrays having 85 transmit only elements and 85 transmit-receive elements, and one suboccipital array having 82 transmit only elements and 85 transmit-receive elements. The proposed apertures and simulated beams for the transtemporal arrays are shown in Figure 44.

The electrical matching characteristics of receive mode multiplexing were simulated using the KLM model (PiezoCAD). This is necessary because each multiplexer carries a variable impedance (usually capacitance in parallel with the element) depending on whether it is in the “on” or “off” state. This is particularly true of the high voltage multiplexers that would be required in this case because the same channels are used to transmit and receive. For instance, using the Supertex HV202 (high-voltage) multiplexer produces a 12.86 dB loss in amplitude relative to the case without multiplexing. Thus this proposed system is not worth pursuing, as the matching loss outweighs the 11 dB beamforming gain. (This would not be true if each channel had its own impedance matching network, but this is too expensive and difficult for us to fabricate in a research setting.) To give an idea of the result of a having better matching, if a Maxim low-voltage multiplexer with 2 pF of “off” and 8 pF of “on” capacitance (MAX4052) could be used, only 3.90 dB are lost due to matching, which produces a net gain of 11 dB on-axis and breaks even at the outermost corners of the pyramid.
As an alternative to multiplexing, partial analog beamforming on the front end at the board- rather than chip-level has also been investigated. Following some of the research systems designed by the Shung group [164, 165], it seems feasible to use a summing op amp configuration to sum signals from adjacent elements. This works because resistive loads are less egregious than capacitive ones from an impedance matching perspective and capacitances could realistically be cancelled with inductors on a dedicated front end board or chip, which is not the case on a flex circuit attaching directly to the scanner, as there is not enough space on the flex circuit for the inductors. Of course, this approach carries its own difficulties related to maintaining acceptably low crosstalk levels on the board, adjusting for multiple focal zones, and maintaining acceptable bandwidth, although commercial systems have undoubtedly solved these problems.
Figure 44: Pulse-echo on-axis beams for alternative configurations of splitting the scanner channels including (A) between three transducers without Rx switching, yielding 85-receive element transtemporal arrays, and (B) between three transducers with Rx switching, yielding 192-receive element transtemporal arrays. (C) The array used in Chapter 2 is shown as a reference.
5.3 Phase aberration correction improvements

5.3.1 Steered transmit wavefronts

During work on phase aberration estimation and correction (Chapter 3), several potential areas for future work have come to light. First the transmitted pressure fields for unsteered and steered (15°, 0°) unfocused waves at a distance of 14 cm were simulated using Field II (Fig. 45) [74], indicating that SNR will be reduced in the steered cases due to significant energy away from the central axis of the receiving array, with the peak pressure at approximately 4 cm. The pressure at the receiving array for steered transmits is approximately -20 dB down in this ideal case, though aberration will lead to a broader distribution of energy. In spite of this low SNR, it has still been possible to locate a steered wave and correct aberration in both tissue-mimicking phantom and in vivo cases. In the future, this could be addressed by either transmitting a defocused rather than a flat wavefront or by steering to angles smaller than 15°. Simulations indicate that transmitting a steered, defocused wavefront increases the transmitted pressure by approximately 6 dB at the receiving array (Fig. 45, black line at 0 cm).
Figure 45: Comparison between pressure fields at $z=14$ cm due to unfocused transmitted beacon wave used in experiments (red line) and three experimental phasings for a wave steered to -15° in azimuth. The white boxes outlined in black indicate the position of the receiving array. At this location, the pressure from the phasing represented by the black line is 6 dB higher than that of the red line.
5.3.2 Isoplanatic patch measurement

Secondly, a more complete 3D measurement of the temporal bone isoplanatic patch still needs to be performed. This would require a grid of wires equally spaced in both azimuth and elevation (or a gelatin phantom with equally spaced imbedded point targets) as a measuring device and a collection of degassed, formalin-fixed skulls to test. In measuring 2D isoplanatic patch sizes, researchers at Philips used pork muscle on the outside of the skull to simulate extracranial tissues [83]. After performing many phase correction experiments, this seems to be a necessary step in order to more accurately re-create the in vivo imaging case for ex vivo experiments.

An alternative means of measuring the extent of the isoplanatic patch in both elevational and azimuthal (i.e. axial and coronal) directions is through in vivo measurement. This test would be more accurate and would require a large subject population, preferably consisting of older subjects of varying ethnicities. Some preliminary measurements of the in vivo isoplanatic patch have been made while attempting to determine the effectiveness of the 5 IP correction technique and will be presented in the next section.

5.3.3 Isoplanatic patch correction performance

In order to assess the performance of multiple isoplanatic patch correction, I examined the difference images for selected slices for a subject (Subject 1) scanned with both 1 IP and 5 IP corrections. I also computed the increase in lateral brightness as a
function of distance for this subject (i.e. an isoplanatic patch measurement for this one subject). In this case, the 1 IP correction data comes from an earlier study in which we had not yet developed the 5 IP correction technique, so the comparison is not direct since the two scans were not made at the same time with the same probe placements. For this reason, the trend of brightness versus lateral distance is more important than the magnitude of the brightness for assessing the effectiveness of this technique. The trend in Fig. 46 suggests that the 5 IP correction technique achieves its intended goal, as the dropoff in brightness increase moving away from the center of the image is no longer visible.

In examining Fig. 47, it should be noted that this is the result of averaging over all depths, including the regions of greatest brightness increase (~40%) as well as those of slight decrease, the nearfield — where brightness fluctuations are expected — and the deepest regions of the volume, where brightness may be expected to decrease. While for purposes of comparison, it may be advisable to normalize the data of Figs. 46 and 47, this also removes the quantitative meaning of the color-coded brightness increases in Fig. 46 and the ordinate in Fig. 47.
Figure 46: Difference images—the result of subtracting the aberrated from the corrected image—for the 1 IP (left) and 5 IP (right) corrections on the same subject acquired at two different sessions. While a stronger aberrator was clearly corrected on the right, the increase in brightness at the edges of the volume on the right is of greatest interest, as shown in Fig. 47.
Figure 47: Increase in brightness on a per voxel basis for four 2D slices (one transverse, one coronal for each transducer) through 3D volumes. Data were acquired on the same subject but at different times. The 5-IP correction technique removes the severe drop-off in brightness increase at large lateral distances.

5.3.4 Overcoming the isoplanatic patch

In Chapter 4, an approach for making several measurements of unique aberrators was presented. These aberrators were shown to be partially correlated, but with a mean ρ generally less than 0.70. This technique requires multiple transmit events to interrogate different propagation paths, and thus is a way of circumventing rather
than solving the isoplanatic patch issue facing all adaptive imaging systems.

Researchers in optics and astronomy have previously struggled with this problem, proposing similar solutions such as using multiple correction beacons or re-positioning a single correction beacon [166]. A similar approach is multiconjugate adaptive optics, in which multiple deformable mirrors are used to characterize multiple aberrating layers where the number of deformable mirrors is equal to the number of thin aberrating layers [167]. This is physically similar to the matrix representation of aberration developed in spatiotemporal inverse filtering techniques [122, 123] in which the entire propagation operator must be acquired, with each row of this matrix corresponding to a different layer of distortion, although the isoplanatic patch remains finite.

However, unlike in optics or astronomy, in ultrasound we often deal with distributed layers, i.e. imaging through multiple distinct tissue types of finite thickness. This question of propagation path dependency and the distributed nature of ultrasound aberration has been investigated by the Waag group [102, 104]. Using multiple measurements and backpropagation techniques, a different filter may be computed and applied at each element for each ray, yielding a tighter focus and a larger isoplanatic patch without the need for thin phase screen assumptions.

**5.3.5 Phase aberration estimation**

There are several ways in which it may be possible to improve the estimation of phase aberrations, i.e. to reduce the magnitude of the residual in Fig. 21A. However,
before these are discussed, it should be noted that this was not a limiting factor in this work, as the scanner clock operated at a 40 MHz, meaning delays required rounding to the nearest 25 ns. While modern scanners may have clock speeds as high as 100 MHz, it seems likely that the magnitude of error introduced by delay quantization will be still be larger than the residual aberrator, at least for conventional beamformers and implementations of phase correction. However, because this may not be the case in futures systems (for example, phase rotation and similar beamformers achieve sub-sample delay accuracy [168]), methods of reducing the residual in phase aberration estimation will be discussed briefly. This section examines only classical phase correction in which transmit and receive delays updates are applied, not inverse filtering or propagation path estimation techniques.

The first method of increasing accuracy of aberration maps is through the development of new time delay estimators. (Note that these generally assume beacon-less correction on speckle signals.) Significant research has been carried out in this array, as this work is motivated by several other applications, including motion tracking, blood flow imaging, and shear wave velocimetry. Proposed techniques for so-called continuous delay estimation include sub-sample interpolation via spline-based algorithms [169-172] and use of a Bayesian estimator, which uses a priori information to circumvent the Cramér-Rao lower bound [173].
The second approach is to improve estimation of the aberrator itself by use an alternative to linear least squares. Gauss has used weighted least squares in which the inverse of the standard deviation of a pulse is used as its weight [94]. Alternatively, a more computationally efficient algorithm for linear least squares has been proposed [174] which may be helpful in online implementation, especially for highly overdetermined systems (i.e. in the presence of a beacon or point target, or when a very finely sampled transducer array is used to provide numerous highly correlated signals).

5.4 Alternative means of transcranial ultrasound propagation

In the past few years, a technique has been proposed which uses ultrasound to map brain elasticity. Macé et al. used radiation force and ultrafast imaging techniques to map shear wave propagation speed in rats [175]. Alternatively, Foster et al. have mapped cerebral hemodynamics by cutting windows in the skulls of living rats and measuring changes in cerebral blood volume with nonlinear microbubble contrast enhancement at 21 MHz [176], producing a result similar to a perfusion CT. Studies such as these suggest that there may be great potential for an ultrasound-based cerebral monitoring system if some of the obstacles associated with the human skull can be overcome. In particular, the ability to produce and track shear waves transcranially may open new avenues of investigation. Two groups have recently examined the role of longitudinal to shear mode conversion in producing the so-called “stripe artifact” in standard B-mode transcranial ultrasound imaging [122, 123], while a third has used full-
wave simulation to examine mode conversions in the skull [88]. The consensus is that significant mode conversion occurs at the outer table of the skull and in the diploë. If one could devise a repeatable means of inducing mode conversions such that shear waves are generated in the brain parenchyma with sufficient amplitude to be trackable, shear wave techniques [175] could potentially be safely applied in vivo using a standard diagnostic transducer. Of course, the propagation direction of these shear waves would be less well-behaved than in the case of an acoustic radiation force push.

5.5 Future utilization of transcranial ultrasound for emergent stroke imaging

While several advancements have been described in this dissertation, the system used here was not portable. As mentioned in 1.2.1, in July of 2012, Philips became the first manufacturer to demonstrate portable real-time 3D ultrasound using its CX50 platform and a transesophageal probe. Thus only engineering obstacles remain to the creation of a bilateral transcranial 3D ultrasound imaging system. These obstacles may include increasing scan depth or sector angle, implementing color or power Doppler, optimizing an existing probe or designing a new probe, and waveform optimizing the transmit pulse and receive filters, all of which may be overcome with existing technology.

Such a portable 3D transcranial imaging system could be carried in ambulances and used by first responders as in the 2D ultrasound study of Hölscher [177]. In this
study, middle cerebral artery (MCA) blood flow was able to be assessed using portable 2D ultrasound imaging in a pre-hospital setting (patient’s home, ambulance, or helicopter) in 20 of 25 subjects. In a follow-up study of 113 patients, MCA occlusion was diagnosed in 10 subjects either with or without contrast enhancement [178]. One MCA occlusion was missed and one atypical hemorrhage was misdiagnosed. Mean time of ambulance arrival was 12.3 minutes and mean transcranial examination time was 5.6 minutes.

These times and sensitivities are encouraging, although ambulance arrival times would be expected to be longer in the United States, where regional hospitals are prevalent, as compared to a single European city. In the United States, stroke death rate is at least 10 percent higher than the national average in 11 states located primarily in the southeast (the “stroke belt”) [179]. Given that these states typically have large percentages of their populations living in rural areas and longer Emergency Medical Service response times [180], the idea of transmitting volumes of imaging data to a hospital as a means of reducing time to treatment and staying within the tPA time window remains attractive in the United States. If an occlusion is identified while the patient is en route to the hospital, tPA could be administered either in the ambulance or immediately on arrival at the hospital.

A portable 3D transcranial ultrasound bilateral imaging system is feasible in terms of technical considerations related to the scanner and probe, anatomical
considerations in overcoming the skull, and human factors such as operator usability. It also provides information which directly contributes to formation of a diagnosis [178] and in ways that 2D ultrasound cannot [181], making such a system a promising way to reduce stroke mortality in developed countries.
Appendix A: Transcranial ultrasound imaging and the KLM model

One of the first and most obvious areas for improvement is to fabricate matrix arrays having a more sophisticated acoustic stack. While fabrication of matrix arrays is still an expensive and difficult undertaking, it is quite feasible for commercial manufacturers of ultrasound systems [36]. The following subsection investigates via simulation the expected improvement in transcranial imaging at 1 MHz afforded by fabricating matrix arrays having more complicated acoustic stacks.

The model of Krimholtz, Leedom, and Matthaei [65], (hereafter referred to as the KLM model) was proposed as a circuit model for ultrasonic transducers which retains physical meaning. Acoustically, a transducer is modeled using transmission line theory, while a single coupling point—an ideal transformer with a frequency-dependent turns ratio—couples the acoustic and the electrical parts of the transducer. This allows matching networks to be designed over a given frequency range using lumped components. Desilets has thoroughly characterized this model for certain cases, especially for a transducer having single backing and single matching layer, both with thicknesses of $\lambda/4$ [155, 182]. This model will now be used to examine the case of a matrix array transducer for transcranial imaging having a lossy 10 $\lambda$-thick backing and no matching layer.
The KLM model used by Desilets is shown above. The front acoustic port consists of PZT acoustically in series with 5 mm of extracranial tissues, 2 mm of skull, and 140 mm of intracranial tissue (brain). The back acoustic port is assumed to consist of 10 $\lambda$ thick-lossy material (2.1 MRayls, 28 dB/cm). Using the parameters in Table 6, complex impedances at the center transformer were computed in the presence and absence of the skull using the following equation to transform the series acoustic impedances:
\[
Z(d) = Z_0 \left( \frac{Z_R + Z_0 \tanh(\gamma d)}{Z_R \tanh(\gamma d) + Z_0} \right)
\]  

(14)

where \(Z_R\) is the rightmost impedance and \(Z_0\) is the impedance immediately to its left, \(\gamma\) is complex propagation parameter \(\gamma = \beta - j\alpha\), and \(d\) is distance along the acoustic transmission line.

**Table 6: Material constants for KLM simulations**

<table>
<thead>
<tr>
<th>Material</th>
<th>(Z_0)</th>
<th>Thickness</th>
<th>(\alpha) (attenuation)</th>
<th>Loss exponent</th>
</tr>
</thead>
<tbody>
<tr>
<td>PZT-5H (TRS HKI-HD)</td>
<td>26.95 MRayl</td>
<td>1.52 mm</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Extracranial Tissue</td>
<td>1.54 MRayl</td>
<td>5 mm</td>
<td>1 dB/cm</td>
<td>1.0</td>
</tr>
<tr>
<td>Skull</td>
<td>7.16 MRayl</td>
<td>2 mm</td>
<td>20 dB/cm</td>
<td>1.5</td>
</tr>
<tr>
<td>Brain</td>
<td>1.53 MRayl</td>
<td>140 mm</td>
<td>1 dB/cm</td>
<td>1.0</td>
</tr>
</tbody>
</table>

The two parallel impedances (front and back) were then summed to give the complex impedance at the transformer. This model was then simulated using PiezoCAD (Sonic Concepts, Bothell, WA). Pulse-echo amplitudes and bandwidths from these simulations are given.

**Table 7: Results of KLM model without matching layer**

<table>
<thead>
<tr>
<th></th>
<th>With skull</th>
<th>Without skull</th>
</tr>
</thead>
<tbody>
<tr>
<td>(</td>
<td>Z_{in}</td>
<td>) at front port</td>
</tr>
<tr>
<td>(\text{arg}[Z_{in}]) at front port</td>
<td>81.32°</td>
<td>79.76°</td>
</tr>
<tr>
<td>Pulse-echo amplitude ((V_{out}/V_{in}))</td>
<td>-62.16 dB</td>
<td>-47.17</td>
</tr>
<tr>
<td>-6 dB bandwidth</td>
<td>30.51%</td>
<td>25.58%</td>
</tr>
</tbody>
</table>
These results indicate that a transcranial imaging transducer should be matched to 2.28 MRayl rather than 1.54 MRayl. Next the same quantities are given for $\lambda/4$ matching layers having characteristic acoustic impedances of both $Z_{ML} = \sqrt{Z_{PZT} \cdot Z_{tissue}}$

$6.44$ MRayl ($Z_{tissue} = 1.54$ MRayl) and $Z_{ML} = \sqrt{Z_{PZT} \cdot Z_{tissue}} = 7.84$ MRayl ($Z_{tissue} = 2.28$ MRayl).

**Table 8: Results of KLM model with single matching layer (6.44 MRayl)**

<table>
<thead>
<tr>
<th></th>
<th>With skull</th>
<th>Without skull</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>Z_{in}</td>
<td>$ at front port</td>
</tr>
<tr>
<td>arg[$Z_{in}$] at front port</td>
<td>78.36°</td>
<td>15.00°</td>
</tr>
<tr>
<td>Pulse-echo amplitude ($V_{out}/V_{in}$)</td>
<td>-53.49</td>
<td>-37.76</td>
</tr>
<tr>
<td>-6 dB bandwidth</td>
<td>71.25%</td>
<td>58.98%</td>
</tr>
</tbody>
</table>

**Table 9: Results of KLM model with single matching layer (7.84 MRayl)**

<table>
<thead>
<tr>
<th></th>
<th>With skull</th>
<th>Without skull</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>Z_{in}</td>
<td>$ at front port</td>
</tr>
<tr>
<td>arg[$Z_{in}$] at front port</td>
<td>79.55°</td>
<td>35.48°</td>
</tr>
<tr>
<td>Pulse-echo amplitude ($V_{out}/V_{in}$)</td>
<td>-54.05</td>
<td>-38.75</td>
</tr>
<tr>
<td>-6 dB bandwidth</td>
<td>71.14%</td>
<td>62.50%</td>
</tr>
</tbody>
</table>

In the presence of the skull, the acoustic input impedance is highly inductive. For the lossless, real load case, a $\lambda/4$ matching layer should transform the acoustic impedance seen at the front port to $Z_{ML}^2/Z_{tissue}=26.95$. Matching to the complex rather than real load may produce better results, since this approach introduces large inductive loads.
Matching to the higher impedance including the skull performs similarly but slightly worse than the typical soft tissue matching layer, indicating that there is a broad peak of acceptable matching layer impedances. In the absence of extracranial tissues, that is when the transducer is directly in contact with skull, the 7.84 MRayl matching layer performs better, producing a pulse-echo amplitude of -48.00 dB and -6 dB bandwidth of 65.24% as compared to the 6.44 MRayl matching layer, which produces a pulse-echo amplitude of -49.37 dB and -6 dB bandwidth of 34.76%. Thus due to the presence of extracranial tissues of finite thickness, the standard matching layer design approach is acceptable and even preferable for transcranial imaging arrays.

An alternative design option is to use a composite transducer (either with or without a matching layer). A composite polymer-ceramic transducers is expected to provide higher bandwidth and better acoustic matching to both soft tissue and soft tissue + skull but worse electrical matching due to the decrease in relative dielectric constant [160]. Mills and Smith demonstrated improved electrical matching using multi-layer composite structures [227, 228]. Cost and complexity of fabrication are higher for multi-layer than single-layer composites, which in turn are higher than single-layer ceramic transducers. Table 10 compares the simulations of ceramic (reproduced from Table 7) and a commercially available 1-3 composite (EBL Products, 14.7 MRayls) transducers without matching layers for an acoustic load consisting of extracranial tissue-skull-brain. Bandwidth improves significantly, but most of the gains due to
improved acoustic matching are lost due to poorer electrical matching, hence the need for the multi-layer structure.

**Table 10: Results of KLM model for ceramic and composite transducers**

<table>
<thead>
<tr>
<th></th>
<th>Ceramic</th>
<th>Composite</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pulse-echo amplitude</td>
<td>-62.16 dB</td>
<td>-60.94 dB</td>
</tr>
<tr>
<td>((V_{out}/V_{in}))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-6 dB bandwidth</td>
<td>30.51%</td>
<td>58.30%</td>
</tr>
</tbody>
</table>

Finally, simulation results for ceramic and composite transducers with single matching layers are presented in Table 11. The ceramic results are the same as in Table 10.

**Table 11: Results of KLM model with single matching layer for ceramic (6.44 MRayl) and composite (4.76 MRayl) active structures**

<table>
<thead>
<tr>
<th></th>
<th>Ceramic (26.95 MRayl)</th>
<th>Composite (14.70 MRayl)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(</td>
<td>Z_{in})</td>
<td>at front port</td>
</tr>
<tr>
<td>(\text{arg}{Z_{in}} ) at front port</td>
<td>78.36°</td>
<td>78.83°</td>
</tr>
<tr>
<td>Pulse-echo amplitude</td>
<td>-53.49 dB</td>
<td>-54.46 dB</td>
</tr>
<tr>
<td>((V_{out}/V_{in}))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-6 dB bandwidth</td>
<td>71.25%</td>
<td>92.25%</td>
</tr>
</tbody>
</table>

Sensitivity is very similar, although about 1 dB better for ceramic due to the improved electrical match. A single matching layer produces significant improvement in pulse-echo amplitude and bandwidth of the composite transducer (Table 11) as compared to a transducer without a matching layer (Table 10).

It is not possible to model multi-layer structures using PiezoCAD. As a rough estimate, Mills reported increases in SNR of 7.5 dB in simulations and 6 – 11 dB in
measurements for fabricated multi-layer composites as compared to single-layer PZT [227, 228].

To summarize, of the simulated transducers, a 1-3 composite with a single matching layer is the most favorable design for transcranial imaging, followed by PZT with a single matching layer (lose 19% of bandwidth but gain 1 dB SNR), a composite without a matching layer (lose 34% of bandwidth and 6.5 dB of SNR), and PZT without a matching layer (lose 61.75% of bandwidth and 7.7 dB of SNR).
Appendix B: Decorrelation and spatial lags in the presence of a correction beacon

Historically, most phase aberration correction techniques have relied on estimation using cross-correlation of speckle signals received from neighboring elements [71, 72, 149]. Alternatively, others have used cross-correlation between channel data and the beamsum [183]. Krishnan et al. found that inter-element cross-correlation produced an estimate with less error than element-beamsum cross-correlation (full array reference) [184]. It should also be noted that other techniques such as time reversal focusing and spatio-temporal inverse filtering do not rely on correlation at all [32, 78, 105, 112, 185]. Alternatively, more recent results from the Waag lab examine propagation path estimation [102-104].

In this work, the opposing array provides a coherent external correction beacon, however this is unrealistic in most in vivo imaging scenarios. I have demonstrated that this produces lower error in estimating electronic aberrators (Fig. 22A, Chapter 3) relative to speckle correction. A second question posed by this arrangement is how many neighboring elements should be included in cross-correlations. The number of lags used in a multi-lag aberration correction scheme affects the size of the model matrix (Equation 7). In the incoherent case, this question of how many lags to include is answered by the van Cittert-Zernicke theorem. However, it has been reported in Figure
that inter-element correlation as a function of element separation falls somewhere between the $\rho=1$ case and the case described by the van Cittert-Zernicke theorem.

This question would seem to be best answered by simulation, since the error between the applied and estimated aberrator could be precisely measured. However, aberration is not well simulated in Field II. In particular, it is difficult to model the propagation path-dependent decorrelation. One alternative approach would be to use the signal model used by Walker and Trahey in deriving the Cramér-Rao lower bound [114] for speckle targets, then decrease $\rho$ linearly with increasing separation, i.e. from 1.0 to 0.65. The increasing number of equations for an overdetermined system is weighed against the decreasing quality of the estimates in those equations. The simulations described by Walker and Trahey [114] and by Viola and Walker [169] were repeated for the 2.5 MHz system described in Chapter 3. The results of two sets of simulations (100 simulations per point) are presented in Fig. 49. The left panel shows jitter (mean standard deviation of time delay error) as a function of SNR while $\rho$ is held constant at 0.97 and SNR is varied from 0-30 dB; the right panel shows jitter as a function of correlation coefficient SNR while is held constant at 20 dB and $\rho$ is varied from 0.65 to 1. Simulation parameters are given in Table 12. The distinction between the two results lies in the fact that SNR includes only electronic noise, while decorrelation noise is accounted for in $\rho$, making it a better indicator of physical decorrelation [186]. These preliminary results indicate that for channel data having SNRs less than approximately
7.5 dB, or as $\rho$ drops below approximately 0.9, jitter overwhelms relative to the CRLB.

Cross-referencing the right panel of Figure 49 with the plots of $\rho$ versus lag (Fig. 23), this suggests that in the presence of a beacon, all lags may be helpful for the unsteered unfocused wave beacon but only lags of about 0.3 cm, or approximately half the aperture, are helpful in the case of the 15 degree steered unfocused wave beacon. However, more simulations are required to confirm this, as well as the simulation of aberrator estimation via least squares solution of the overdetermined system. One approach for testing this would be to introduce jitter at levels equal to that observed in presented simulations (Fig. 49) when estimating an aberrator with a varying number of lags.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_0$</td>
<td>2.5 MHz</td>
</tr>
<tr>
<td>BW</td>
<td>50%</td>
</tr>
<tr>
<td>$T$</td>
<td>6.1 $\mu$s</td>
</tr>
<tr>
<td>$f_s$</td>
<td>200 MHz</td>
</tr>
<tr>
<td>SNR</td>
<td>20 dB*</td>
</tr>
<tr>
<td>$\rho$</td>
<td>0.97*</td>
</tr>
<tr>
<td>Number of simulations</td>
<td>1000</td>
</tr>
</tbody>
</table>

*when not varied
Figure 49: Simulated jitter (markers) and CRLB (continuous line) (A) with varying SNR and (B) with varying ρ.
Appendix C: Aberrated and corrected beam profiles

Because of the transducer arrangement, it was possible to produce an image of the energy distribution in $x$ and $y$ (or $\theta$ and $\phi$) for each transducer before and after correction. In fact, this was done routinely as part of direction of arrival estimation, though was generally not displayed while running a correction. This section displays a few of these results, showing the improvement in the point spread function (PSF). Fig. 50 shows the pitch-catch measured pressure distributions and resulting C-scans for a 75 ns RMS, 2.7 mm correlation length electronic aberrators for control, aberrated, and corrected cases. Fig. 49 shows aberrated and corrected pitch-catch measured pressure distributions for a physical aberrator.
Figure 50: Control (top), aberrated (middle), and corrected (bottom) 1-way beam profiles are shown at left with the corresponding pulse-echo C-scan image of a point target shown for each using the head of a pin as a target. C-scans are taken from the scanner’s real-time display. Target brightness is visibly recovered in going from the aberrated to the corrected case.
Figure 51: Aberrated (top) and corrected (bottom) pitch-catch beams for a skull casting measured to induce aberration of 72 ns RMS. The clutter pedestal is reduced by approximately -4 dB and beam width narrows by approximately a factor of 2.
Appendix D: Analysis of steered beacon aberration correction

This appendix discusses the physics and mathematics of steered transmit beacon aberration correction and why it is effective for estimating multiple unique aberrators. It uses the formalism of Waag and Astheimer [102], which modifies previous statements of the pulse-echo wave field [187] to include aberration and to account for the use of an array transducer. In [102], the authors examine the scattering volume which contributes to a single region of an image at the transmit and receive foci. By acquiring data at 75 unique foci and deconvolving the water tank response from the aberrated response, they are able to estimate the magnitude and phase of the aberrator’s frequency response, compensating for the distributed nature of aberration. While the full derivation in [102] will not be repeated here, the major result is the mathematical description of the scattering volume in the presence of an inhomogeneous (ab errating) medium. In the time domain, this expression is:

\[ y(t_R, r_R, r_T) = \int \int \int \psi_T(r, t, t_T) \eta(r) \frac{1}{c^2} \frac{\partial^2}{\partial t^2} \left[ \int p(t_T) \psi_T(r, t, t_T) dt_T \right] dtd^3r, \quad (15) \]

where \( p \) is the band-limited pressure pulse, \( c \) is propagation velocity, \( \eta \) is an unknown random term describing the distribution of medium variations, and \( \psi_T \) and \( \psi_R \) are the transmit and receive fields as a function of location \( r \) and time \( t \), defined as:
\[
\psi_T(r, t, r_T, t_T) = \sum_i A_i [\alpha(r, r_i, t) * G_0(r, r_i, t)] (t - t_T + \tau_i)
\]
(16)

and

\[
\psi_R(r, t, r_R, t_R) = \sum_j B_j [\alpha(r, r_j, t) * G_0(r, r_j, t)] (t_T - t + \tau_j)
\]
(17),

where \(r_i\) and \(r_j\) are the transmit and receive aperture locations (the focus is treated as a second transmit aperture in the receive case), \(G_0\) is the Green’s function or spatial impulse response diverging from location \(r\) at time \(t_T\) to location \(r\) at time \(t\) (or from location \(r\) at time \(t\) to location \(r\) at time \(t_R\) for the receive case) in a homogeneous medium, \(A_i\) and \(B_j\) are the weights of the transmitted signals and scatterers, respectively, \(\alpha\) is the aberrator convolved in \(t\) with the homogeneous Green’s function, and \(\tau_i\) and \(\tau_j\) are the delays producing correct geometric focusing in the absence of aberration.

Provided that a finite-length signal is acquired with window \(w\), the Fourier transform of (15) is:

\[
y(\omega, r, r_T, c) = -k^2 P(\omega) \int \eta(r) \Psi_T(r, r_T, \omega, c) \Psi_R(r, r_R, \omega, c) \cdot \Psi_T(r, r_T, \omega, c) \cdot \Psi_R(r, r_R, \omega, c) \cdot \eta(r) \psi_T(r, t, r_T, t_T) \int (r - c) \cdot e^{ik u_T(r - c)} \cdot \psi_T(r, t, r_T, t_T) \cdot \psi_R(r, t, r_R, t_R) d^3 r
\]
(18)

for focus \(c\), where \(u_T = \frac{r_T - c}{\|r_T - c\|}\), \(u_R = \frac{r_R - c}{\|r_R - c\|}\), and spatial frequency \(k = \frac{\omega}{c} (u_T + u_R)\).

The isoplanatic patch appears in these equations as the dependence of \(\alpha\) on \(r\). Within an isoplanatic patch, \(\alpha\) varies very slowly with \(r\).

Examining (18), it can be seen that the scattering volume is defined by the product of three weights: \(\Psi_T, \Psi_R, \text{ and } w\), which limit the spatial frequency content in
each focused region of an image. The transmit and receive beams have directionalities given by unit vectors $u_T$ and $u_R$, which are approximately aligned in the cases of both pulse-echo imaging and acquisition of speckle data for aberration correction. However, when correcting on a transmitted beacon wave as in this work, $u_T$, the directionality of the transmit field $\Psi_T$, changes as compared to the speckle correction case, interrogating a different set of spatial frequencies and providing new information regarding the aberrator $\alpha$ (keeping in mind that $\alpha$ is contained in both $\Psi_T$ and $\Psi_R$, the Fourier transforms of Equations 16 and 17). Alternatively, consider that in correcting on a transmitted beacon as compared to a pulse-echo correction on speckle, the apertures do not change (i.e. the magnitudes of weightings $A_i$, $B_j$) but because $r$ is a different location, every function of $r$ changes. Thus by transmitting beacon waves of different trajectories, information is acquired for different spatial locations $r$, at which the inhomogeneous medium operator $\alpha$ is substantially different. Additionally, the changes in $r$ values and thus $u_T$ and $u_R$ correspond to the changes in beam steering in a phased array scan, as the presented equations describe only a single scattering volume at the focus, allowing for differential correction of regions of a 3D phased array scan as described in Chapter 3. This concept is illustrated in Figure 52.
Figure 52: In the conceptual diagram of scattering volumes (top row), the scattering volume in the spatial frequency domain is the product of three weights: the transmit field $\Psi_T$ (gray), the receive field $\Psi_R$ (white), and the axial window $w$ (black). The image scan line corresponding to each scattering volume is shown on the bottom row (2D slice shown for simplicity). Three cases are shown: (A) pulse-echo acquisition near the center of the phased array sector, (B) pulse-echo acquisition near the edge of the sector, and (C) pitch-catch acquisition. The open circle on each scan line indicates that a single scattering volume applies to specific region in an image, i.e. one scan line and focal zone. This figure is adapted from Fig. 1 of [102].

Note that the pitch-catch acquisition (C) has a slightly different weight for the spatial frequency representation of its transmit field relative to the pulse-echo acquisition (B), as the pitch-case acquisition depends on forward rather than backward scattering. Its direction is also expected to be slightly different from but similar to the acquisition at the sector edge. That is, the correction technique works because most of the same spatial frequencies are interrogated in (B) and (C).
Looking at the proposed pitch-catch correction scheme from the point of view of the receiving array, \( \Psi_R \) in Equation 17 would be re-written as follows:

\[
\Psi_R(r_{oi}, t, r_R, t_{Ro}) = \sum_i A_i[\alpha(r_{oi}, r_j, t) * G_0(r_{oi}, r_j, t)](t_{Ro} - t + \tau_i)
\]  

(19)

where subscript \( oi \) is used to indicate that the pulse originates with the opposite transmit array, \( \tau_i \) is now the beacon wave focusing pattern (unfocused or defocused), transmitted signal weight \( A_i \) replaces scatterer weight \( B_j \), and the receive aperture is now in a single location (no summation in \( j \)). This obviously affects the Green’s function since the propagation path is different (from \( r_{oi} \) to \( r_j \) at time \( t_{Ro} \)), but also affects the inhomogeneous operator \( \alpha \), however from the perspective of the receiving array in the transcranial case, aberration is largely confined to a nearfield phase screen so as long as the direction between \( r_{oi} \) and \( r_j \) is similar (19) to that between \( r \) and \( r_j \) (17), the same spatial frequencies are acquired. The requirement for how similar these directions must be depends on the aberrator and is related to the discussion on isoplanatic patch size (Section 5.2.2) [83, 87].
Appendix E: Aberration simulation code

This code produces the simulations varying spatial lags presented in Appendix B.

jitter_sim_wrapper2.m is the wrapper for running multiple jitter simulations using the Walker-Trahey signal model \[114\] with varying $\rho$. Other versions of this script allow other parameters to be varied (SNR) but are omitted here. This version allows for correlation to be varied in a manner similar to the presented experimental decorrelation curve for a beacon (Figure 23, Chapter 3).

jitter_sim.m runs a single simulation using the parameters specified in the wrapper.

**jitter_sim_wrapper2.m**

```matlab
%jitter_sim_wrapper2 - vary rho instead of SNR
num_sims=1000;
corr_length=length(xcov(arr(:,1),arr(:,2),'coef'));
error_vec30=[];
rho=.999;
for(count=1:num_sims)
    jitter_sim;
    error=abs((corr_length-1)/2-find(xcov(arr(:,1),arr(:,2),'coef')==max(xcov(arr(:,1),arr(:,2),'coef'))))*1/fs;
    if(error>T/2)
        error=T/2;
    end;
    error_vec30=[error error_vec30];
end;
error_vec25=[];
rho=.95;
for(count=1:num_sims)
    jitter_sim;
    error=abs((corr_length-1)/2-find(xcov(arr(:,1),arr(:,2),'coef')==max(xcov(arr(:,1),arr(:,2),'coef'))))*1/fs;
    if(error>T/2)
        error=T/2;
    end;
```
rho=.90;
for(count=1:num_sims)
jitter_sim;
error=abs((corr_length-1)/2-find(xcov(arr(:,1),arr(:,2),'coef')==max(xcov(arr(:,1),arr(:,2),'coef'))))*1/fs;
if(error>T/2)
    error=T/2;
end;
error_vec20=[error error_vec20];
end;
error_vec15=[ ];
rho=.875;
for(count=1:num_sims)
jitter_sim;
error=abs((corr_length-1)/2-find(xcov(arr(:,1),arr(:,2),'coef')==max(xcov(arr(:,1),arr(:,2),'coef'))))*1/fs;
if(error>T/2)
    error=T/2;
end;
error_vec15=[error error_vec15];
end;
error_vec10=[ ];
rho=.825;
for(count=1:num_sims)
jitter_sim;
error=abs((corr_length-1)/2-find(xcov(arr(:,1),arr(:,2),'coef')==max(xcov(arr(:,1),arr(:,2),'coef'))))*1/fs;
if(error>T/2)
    error=T/2;
end;
error_vec10=[error error_vec10];
end;
error_vec7=[ ];
rho=.75;
for(count=1:num_sims)
jitter_sim;
error = abs((corr_length - 1)/2) - find(xcov(arr(:,1),arr(:,2),’coef’) == max(xcov(arr(:,1),arr(:,2),’coef’))) * 1/fs;
if(error > T/2)
    error = T/2;
end;
error_vec7 = [error error_vec7];
end;
error_vec5 = [];
rho = .70;
for(count=1:num_sims)
    jitter_sim;
    error = abs((corr_length - 1)/2) - find(xcov(arr(:,1),arr(:,2),’coef’) == max(xcov(arr(:,1),arr(:,2),’coef’))) * 1/fs;
    if(error > T/2)
        error = T/2;
    end;
    error_vec5 = [error error_vec5];
end;
error_vec0 = [];
rho = .65;
for(count=1:num_sims)
    jitter_sim;
    error = abs((corr_length - 1)/2) - find(xcov(arr(:,1),arr(:,2),’coef’) == max(xcov(arr(:,1),arr(:,2),’coef’))) * 1/fs;
    if(error > T/2)
        error = T/2;
    end;
    error_vec0 = [error error_vec0];
end;
% error_vecm5 = [];
% SNR = 10^((-5/20)); % SNR = -5 dB
% for(count=1:10)
%    % jitter_sim;
%    % error = abs((corr_length - 1)/2 -
%                  find(xcov(arr(:,1),arr(:,2),’coef’) == max(xcov(arr(:,1),arr(:,2),’coef’))) * 1/fs;
%    % error_vecm5 = [error error_vecm5];
% end;
rho_vec = [.999 .95 .875 .825 .75 .70 .65];
mean_error=flip( [mean(error_vec0) mean(error_vec5) mean(error_vec7) mean(error_vec10) mean(error_vec15) mean(error_vec20) mean(error_vec25) mean(error_vec30)])
std_error=[std(error_vec0) std(error_vec5) std(error_vec7) std(error_vec10) std(error_vec15) std(error_vec20) std(error_vec25) std(error_vec30)]
figure;
plot(rho_vec(1:end),mean_error(1:end)*1e9,'x')

%plot vs sampled CRLB
clear crlb_plot;
for(s=0.60:0.01:1)
    index=round((s)*100-59);
    crlb_plot(index)=sqrt(3/(2*fc^3*pi^2*T*(band^3+12*band))*(1/s^2*(1+1/SNR^2)^2-1));
end;
hold on;
%crlb_plot(1)=sqrt(3/(2*fc^3*pi^2*T*(band^3+12*band))*(1/rho^2*(1+1/1^2)^2-1));%hard code so it's not infinite for SNR=0
plot(0.60:0.01:1,crlb_plot*1e9,'r','linewidth',2);
xlabel('rho')
ylabel('jitter [ns]')
legend('Simulation','CRLB');

jitter_sim.m
%jitter simulation
%Brooks Lindsey
%12/2010-1/2011
clear signals;
clear arr; clear arr1; clear arr2;
num_signals=2;%number of RF signals
run H:\MyDocs\PitchCatchVolumetrics\ducer_121910\vxd_coords;
coords_xd0=coors(128:end-1,);
coords_xd1=coors_xd0;
band=0.5;%bandwidth
pitch=.35e-3;
fc=2.5e6;%center frequency
%fs=fc*200;
fs=200e6;
%T=40*1/fs;%window length in sec
%T=2e-6;
rho=.97;%uncomment to hold rho constant (while varying SNR)
WWt=[1 rho; rho 1];
%WWt=rho*(1-eye(num_signals))+eye(num_signals);
%scale as move away from diagonal for more realistic cross-correlation matrix
x1=1;y1=1;
x2=size(WWt,1);y2=size(WWt,2);
normalization_factor=abs((x2-x1)*(y1-1)-(x1-size(WWt,1))*(y2-y1))/sqrt((x2-x1)^2+(y2-y1)^2);
W=chol(WWt);
%SNR=100;%estimated system SNR (linear scale)
%SNR=10^(20/20);%uncomment to hold SNR constant (while varying rho)
up_factor=1;
tc = gauspuls('cutoff',fc,band,-6,-40);
t = -tc:1/fs:tc;
g1=randn(size(sig1,1),3*size(sig1,2));
sb1=conv(sig1,g1);
g3=randn(size(sig1,1),3*size(sig1,2));
sb3=conv(sig1,g3);
sr=sb1+1/SNR*sbsb3;
signals(1,:)=interp(sr,up_factor);
for(i=2:num_signals)
  g2=randn(size(sig1,1),3*size(sig1,2));
  sb2=conv(sig1,g2);
  g4=randn(size(sig1,1),3*size(sig1,2));
  sb4=conv(sig1,g4);
  sr=sb1+1/SNR*sbsb3;
  sd=W(1,i)*sb1+W(i,i)*sb2+1/SNR*sbsb4;
  signals(i,:)=interp(sd,up_factor);
end;
signals=signals(:,1+round(.05*size(signals,2)):end-round(.05*size(signals,2)));
%compute jitter standard deviation from Cramer-Rao lower bound (Walker and Trahey 1995)
T=(size(arr,1)-80)/(fs*up_factor);
%T=80*1/(fs*up_factor);
%T=(size(arr,1))/fs;
jitter_bound=sqrt(3/(2*fc^3*pi^2*T*(band^3+12*band)))*(1/rho^2*(1+1/SNR^2)^2-1));

%jitter_bound=30e-9;%jitter standard deviation in seconds
%use this to hard code jitter in ns rather than computer
%from CRLB
jitter_bound_sam=jitter_bound*fs;

fid=fopen('coords_flex.bin','rb');
coords_flex=fread(fid,'double');
close(fid);
coords_flex=reshape(coords_flex,256,3);
count1=0;count0=0;
for(i=1:256)
    if(coords_flex(i,3)==0)
        count1=count1+1;
        arr01(count1,:)=coords_flex(i,1:2);
    else
        count0=count0+1;
        arr00(count0,:)=coords_flex(i,1:2);
    end;
end;
desired_corr_length=3e-3;%this is the parameter you change to adjust corr. length of aberrator
desired_RMS=100e-9;%set aberrator strength in sec
input_corr_length_x=(diff([min(coords_flex(:,1)),max(coords_flex(:,1))])+1)*desired_corr_length/ap_size;
ap_size=(diff([min(coords_xd0(:,1)),max(coords_xd0(:,1))])+1)*pitch;
ap_size_mm=(diff([min(coords_xd0(:,1)),max(coords_xd0(:,1))])+1)*pitch*1e3;
input_corr_length_x=(diff([min(coords_xd0(:,1)),max(coords_xd0(:,1))])+1)*desired_corr_length/ap_size;
input_corr_length_y=input_corr_length_x;%can change to be unequal if desired
aberrator01=usp_gaussian_aberrator(coords_xd0,[input_corr_length_x,input_corr_length_y]);%Uses R. Gauss code, aberrator has RMS=1
aberrator01=aberrator01*desired_RMS*fs;
aberrator02=usp_gaussian_aberrator(coords_xd1,[input_corr_length_x,input_corr_length_y]);%Uses R. Gauss code, aberrator has RMS=1
aberrator02=aberrator02*desired_RMS*fs;

count1=0;count0=0;
for(i=1:num_signals)
    if(i<=128)
        count1=count1+1;
        arr_shift(round(aberrator01(count1))+1+round(-
            min(aberrator01)):round(aberrator01(count1))+round(-
            min(aberrator01))+size(arr,1),i)=arr(:,i);
    else
        count0=count0+1;
        arr_shift(round(aberrator02(count0))+1+round(-
            min(aberrator02)):round(aberrator02(count0))+round(-
            min(aberrator02))+size(arr,1),i)=arr(:,i);
    end;
end;

% if you want to run many simulations for different aberrators, you need to
% pad with zeros axially so that all RF lines always have the same length,
% regardless of the aberrator.

pad_size=size(arr,1)+100;
arr_shift=[arr_shift' zeros(num_signals,1410-size(arr_shift,1))];
Appendix F: Phase aberration correction code

This appendix contains code for both the Volumetrics scanner and offline processing (Matlab and C) to estimate aberration and update transmit and receive delays for 5 unique aberrators (5 isoplanatic patches). The code in this appendix is divided into 3 sections: data acquisition, offline processing, and Volumetrics code.

Data acquisition (called while using acquisition data set ab_ac on scanner):

mp_acq.m – script to acquire 5 patches worth of data while scanner transmits 5 different steered plane waves

%mp_acq
%simple multi-patch acquisition script

%patch 1
%!phaseAbDaqDepthAvChoice
fid=fopen('data1.dat','rb');
patch01=fread(fid,int8);
fclose(fid);
patch01=reshape(patch01,256,256);

%patch 2
%!phaseAbDaqDepthAvChoice
fid=fopen('data1.dat','rb');
patch02=fread(fid,int8);
fclose(fid);
patch02=reshape(patch02,256,256);

%patch 3
%!phaseAbDaqDepthAvChoice
fid=fopen('data1.dat','rb');
patch03=fread(fid,int8);
fclose(fid);
patch03=reshape(patch03,256,256);
%patch 4
phaseAbDaqDepthAvChoice
fid=fopen('data1.dat','rb');
patch04=fread(fid,int8);
fclose(fid);
patch04=reshape(patch04,256,256);

%patch 5
phaseAbDaqDepthAvChoice
fid=fopen('data1.dat','rb');
patch05=fread(fid,int8);
fclose(fid);
patch05=reshape(patch05,256,256);

%once all data is read in, save wkspace using today's date
filenum='00';
savefile=strcat(datestr(date,29),'_',filenum);
while(exist(strcat(savefile,'.mat'))==0)
    if(str2num(filenum)<9)
        filenum=strcat('0',num2str(str2num(filenum)+1));
    else
        filenum=num2str(str2num(filenum)+1);
    end;
savefile=strcat(datestr(date,29),'_',filenum);
end;
save(savefile,'patch01','patch02','patch03','patch04','patch05');

%tell user to hit scanner button so it will be ready to receive updates
input('Hit Update Rx Delays on Scanner, then press Enter','s')

fs=25e6;
z=8.3e-2;
data_in=patch01;
phase_corr;
!udp2dos

data_in=patch05;
phase_corr;
!udp2dos
data_in=patch04;  
phase_corr;  
!udp2dos

data_in=patch03;  
phase_corr;  
!udp2dos

data_in=patch02;  
phase_corr;  
!udp2dos

**phaseAbDaqDepthAvChoice.c** – this is one of several versions of this program which is used to acquired averaged RF (usually channel data but it depends on the scanner’s data set) from the scanner’s detector board using the Signatec PDA14.

```c
#include "PDA14win.h"
#include <iostream.h>
#include <math.h>
#include <fstream.h>
#include <stdlib.h>
#include "stdafx.h"
#include "resource.h"
#include "P14IoCtrl.h"
#include <windows.h>
#include <conio.h>
#include "basetsd.h"

#define DEF_P1K_DEV 1

void outS( const char *name, USHORT *a, int n){

    FILE *fp;
    int count=0;
    printf("Writing data to file %s (size: %d bytes)... \n", name, sizeof(USHORT)*n);
    if( ( fp=fopen( name, "wb" ) ) !=NULL){
        count =fwrite( a, sizeof(USHORT), n, fp);
        if (count!= n)
            printf("Error: Only read %d of %d data elements.", count, n);
        fclose( fp);
        printf("closing file %s...\n", name);
    }
}
```

172
else{
    printf("error: could not open file: %s\n", name);
}
}

void main(void)
{
    p14_dma_buf_t pDmaBuf;  //DMA pointer variable
    P14ID brd;  //Board ID
    ULONG count1;
    USHORT dataout[768*256]; //bdl 2/05/11 was 256*256
    int nDev;
    int trigdel=0;
    //int trigdel=65;
    cout << "Enter depth of target in mm:\n";
    cin >> trigdel;
    trigdel=(trigdel*25000)/1540 + 100;  // delay mm *25MHz * 1m/1000mm * 1s/1540m + 100 (measured offset)
    int numavg;
    cout << "Enter number of averages to take:\n";
    cin >> numavg;

    //things i dont understand
    P14ID_CONSTRUCT(brd);  //construct board;
    pDmaBuf=NULL;
    nDev=DEF_P1K_DEV;
    InitDeviceP14(&brd,nDev);
    SetPowerupDefaultsP14(&brd);
    SetModeP14 (&brd, P14MODE_STANDBY);
    //create CMA event object
    CreateEvent(NULL, TRUE, TRUE, NULL); //?????
    ULONG memnumber=numavg*256*768; //bdl 2/05/11 was 256*256
    //Allocate DMA buffer
    AllocateDmaBufferP14(&brd, memnumber*2, &pDmaBuf);
// Setup the Board  

// brd, mode, source, slope, level  
TrigInitP14(&brd, P14TRIGMODE_SEGMENTED, P14TRIGSRC_EXT,  
TRIGSLOPE_NEG, 3144);  
SetVoltRangeCh1P14(&brd, P14VOLTRNG_1V);  
SetChannelsP14(&brd, P14CHANNEL_ONE);  
SetClockP14(&brd, P14CLKSRC_INT);  
SetClockDividerP14(&brd, P14CLKDIV_4);  
// Set clock divider to 4,  
Fsample = 100M/4 = 25M  
SetOffsetCh1P14(&brd, 2048);  
// set offset to 0V  
SetSegmentSizeP14(&brd, 768);  
// bdl Set sample size to 768  
(was 256) this is the number of samples taken on each trigger  
SetTrigDelayP14(&brd, trigdel*2);  
// set trigger delay in samples  
(old one was in samples/2)  
SetStartSampleP14(&brd, 0);  
// start at 0  
SetTotalSamplesP14(&brd, memnumber);  

// acquire data  
if (SIG_SUCCESS != AcquireToBoardP14(&brd))  
{  
    cout << "ERROR could not acquire to board \n";  
}  

// transfer data  
BoardToPcP14(&brd, 0, memnumber, pDmaBuf,  
INVALID_HANDLE_VALUE);  

for (int channelcount=0; channelcount<256; channelcount++) {  
    for (int datacount=0; datacount<numavg; datacount++) {  
        dataout[channelcount*768+datacount]=0;  
    }  
}  

// find average offset value for one frame (should be close enough for all frames)  
double offset=0.0;
USHORT offsetS=0;
for(int sampcount=0;sampcount<(256*768);sampcount++)//bdl was 256*256
    offset +=((double)pDmaBuf[sampcount])/(256.0*768.0);//bdl was 256.0*256.0
offsetS = (USHORT) offset;

//average the data
for( channelcount=0;channelcount<256;channelcount++){
    for(int avgcount=0;avgcount<numavg;avgcount++){
        for(int datacount=0;datacount<768;datacount++)//bdl was 256
            dataout[channelcount*768+datacount] +=
                (unsigned)pDmaBuf[(channelcount*768)+avgcount*256*768+datacount];
        dataout[channelcount*768+datacount] -= offsetS;//bdl all 768's in these 2 lines were 256's
    }
}

outS("data1.dat",dataout,(768*256));//bdl was 256

/*
//write data to file
ofstream outfile("data1.txt");
for(count1=0;count1<(256*256); count1++)
    outfile << dataout[count1] <<endl;
}
outfile.close();*/

FreeDmaBufferP14(&brd,pDmaBuf);
CloseDeviceP14(&brd);
return;
}
Offline processing (estimate aberrator from acquired data, package delay updates):

**phase_corr.m** – wrapper script that calls other programs for offline processing. It takes as its inputs the RF channel data, the depth at which it was acquired, and the sampling frequency at which it was acquired.

%phase_corr - wrapper that calls other programs for running a correction on %the scanner
%have to first call !phaseAbDaqDepthAvChoice to acquire gated RF
%then use the gate position as your z
%
%get coods -----------------------------------------
fid=fopen('coords_flex.bin','rb');
coords_flex=fread(fid,'double');
fclose(fid);
coords_flex=reshape(coords_flex,256,3);
c1=0; c2=0;
for(i=1:size(coords_flex,1))
    if(coords_flex(i,3)==0)
        c1=c1+1;
        coords_xd0(c1,1:2)=coords_flex(i,1:2);
        arr1(:,c1)=data_in(:,i);
    else
        c2=c2+1;
        coords_xd1(c2,1:2)=coords_flex(i,1:2);
        arr2(:,c2)=data_in(:,i);
    end;
end;
coords_xd0(:,3)=1;
coords_xd1(:,3)=1;

%apply spatial filter ---------------------------------
[arr1_spf, arr2_spf, dead_chan_xd0, dead_chan_xd1] = parse_rf_3d(arr1, arr2, coords_xd0, coords_xd1, z, fs);

%estimate direction of arrival
[input_delayed_opt , input_delayed_opt2, x_max1, y_max1, x_max2, y_max2 ] = doa(arr1, arr2, coords_xd0, coords_xd1, z, fs);
%parse for pa_ncorr

%create model matrix if necessary
%exist('mod1')
if(~exist('mod1')) %if the model matrix is not in the workspace
    within_bound=3e-3; %use this to select distance of element lags to include in model matrix
    vl_modelmatrix_vx_withinmm;
end;
posvec_remove_dead_vx; %pull out dead channels (if desired) and set detrending vectors accordingly
input_delayed_opt_copy=input_delayed_opt;
input_delayed_opt_copy2=input_delayed_opt2;
clear input_delayed_opt; clear input_delayed_opt2;
c=0;
for(i=1:size(input_delayed_opt_copy,2))
    if(isequal(input_delayed_opt_copy(:,i),zeros(size(input_delayed_opt_copy,1),1)))
    
    else
        c=c+1;
        input_delayed_opt(:,c)=input_delayed_opt_copy(:,i);
    end;
end;
c=0;
for(i=1:size(input_delayed_opt_copy2,2))
    if(isequal(input_delayed_opt_copy2(:,i),zeros(size(input_delayed_opt_copy2,1),1)))
    
    else
        c=c+1;
        input_delayed_opt2(:,c)=input_delayed_opt_copy2(:,i);
    end;
end;

fid=fopen('focused1.bin','wb');
fwrite(fid,input_delayed_opt,'double');
fclose(fid);

fid=fopen('focused2.bin','wb');
fwrite(fid,input_delayed_opt2,'double');
fclose(fid);
%run correlations and least squares multi-lag approximate solution

!pa_ncorrMod12_corn.exe
%read in outputs
fid=fopen('x.bin','rb');
x=fread(fid,'double');
fclose(fid);
fid=fopen('detx0.bin','rb');
detx0=fread(fid,'double');
fclose(fid);
fid=fopen('x2.bin','rb');
x2=fread(fid,'double');
fclose(fid);
fid=fopen('detx1.bin','rb');
detx1=fread(fid,'double');
fclose(fid);

%plot (if desired)
%find energy maximum for each xducer's data so you can plot profile through maxima
[e_max01_row,
e_max01_col]=find(abs(input_delayed_opt)==max(abs(input_delayed_opt(:))));
[e_max02_row,
e_max02_col]=find(abs(input_delayed_opt2)==max(abs(input_delayed_opt2(:))));

%uncomment to plot detected profile on RF data
% figure
% imagesc(arr1_spf)
% hold on
% plot(-detx0*1.6+e_max01_row+10,'b');
% plot(-x*1.6+e_max01_row+10,'w');
aberrator02_rms=sqrt(mean(x.^2))/fs*1e9
% title(strcat('RMS :', num2str(aberrator02_rms*1e9),' ns'));
% figure
% imagesc(arr2_spf)
% hold on
% plot(-detx1*1.6+e_max02_row+10,'b');
% plot(-x2*1.6+e_max02_row+10,'w');
aberrator01_rms=sqrt(mean(x2.^2))/fs*1e9
% title(strcat('RMS :', num2str(aberrator01_rms*1e9),' ns'));
%send to scanner (may be best to check results manually before sending) 

%!udp2dos.exe

out=1;

parse_rf_3d.m – reorder channels and apply axial and lateral filtering, then write

outputs in appropriate format

%parse_rf_3d
%arrange acquired RF data into a spatially meaningful 3d data structure
function [arr1_spf, arr2_spf, dead_chan_xd0, dead_chan_xd1] = parse_rf_3d(arr1, arr2, coords_xd0, coords_xd1, z, fs);
%setup constants
c=1500;
%fs=25e6;
f0=1.8e6;
pitch=.725e-3;
lambda=c/f0;
dead_chan_xd0=[];
dead_chan_xd1=[];

%begin axial filter
%fs=25e6;
%filter 1
flow=1.4/12.5;
fupp=2.2/12.5;
BPord=512;
b_bp = fir1(BPord,[flow fupp],window(@gausswin,BPord+1));
N=max(2^nextpow2(length(b_bp)),2^nextpow2(size(arr1,1)));
B_bp=fft(b_bp,N)';
%filter 2
flow2=1.4/12.5;
fupp2=2.2/12.5;
BPord2=512;
b_bp2 = fir1(BPord2,[flow2 fupp2],window(@gausswin,BPord2+1));
N2=max(2^nextpow2(length(b_bp2)),2^nextpow2(size(arr2,1)));
B_bp2=fft(b_bp2,N2)';
%[rbp cbp]=size(b_bp);
chf_sig=zeros(BPord+size(arr1,1),size(arr1,2));
chf_sig2=zeros(BPord2+size(arr2,1),size(arr2,2));
%f=0:128;
for(i=1:size(arr1,2))
    [hh,ff]=freqz(b_bp,1,128,fs);
mag_max=max(abs(hh));
magdB=20*log10(abs(hh));
%chf_sig(:,i)=conv(b_bp,arr1(:,i)-mean(arr1(:,i)))';
chf_sigf(:,i)=ifft(B_bp.*fft(arr1(:,i)-mean(arr1(:,i)),N));
end;
chf_sigf=circshift(chf_sigf,-round(mean([N-BPord,size(arr1,1)])));
TailLength = (N-size(arr1,1))/2;
chf_sigf=chf_sigf(TailLength+1:size(chf_sigf,1)-TailLength,:);
for(i=1:size(arr2,2))
    [hh,ff]=freqz(b_bp,1,128,fs);
mag_max=max(abs(hh));
magdB=20*log10(abs(hh));
%chf_sig2(:,i)=conv(b_bp,arr2(:,i)-mean(arr2(:,i)))';
chf_sigf2(:,i)=ifft(B_bp2.*fft(arr2(:,i)-mean(arr2(:,i)),N2));
end;
chf_sigf2=circshift(chf_sigf2,-round(mean([N2-BPord2,size(arr2,1)])));
TailLength = (N2-size(arr2,1))/2;
chf_sigf2=chf_sigf2(TailLength+1:size(chf_sigf2,1)-TailLength,:);
arr1=real(chf_sigf);
arr2=real(chf_sigf2);
%end axial filter

data3d=zeros(max(coords_xd0(:,1))-min(coords_xd0(:,1))+1,max(coords_xd0(:,2))-min(coords_xd0(:,2))+1,size(arr1,1));
data3d2=zeros(max(coords_xd1(:,1))-min(coords_xd1(:,1))+1,max(coords_xd1(:,2))-min(coords_xd1(:,2))+1,size(arr1,1));
for(i=1:size(arr1,2))
    idx_x=coords_xd0(i,1)+abs(min(coords_xd0(:,1)))+1;
    idx_y=coords_xd0(i,2)+abs(min(coords_xd0(:,2)))+1;
    data3d(idx_x,idx_y,:)=arr1(:,i);
    idx_x2=coords_xd1(i,1)+abs(min(coords_xd1(:,1)))+1;
    idx_y2=coords_xd1(i,2)+abs(min(coords_xd1(:,2)))+1;
    data3d2(idx_x2,idx_y2,:)=arr2(:,i);
end;
%create lateral filter
N=size(data3d,1)-1;%filter order
fsp=1/pitch;%spatial sampling frequency in cyc/m
spatial_Nyquist=0.5*fsp;%1/m
cutoff=0.75*spatial_Nyquist;%1/m
cutoff_scaled=cutoff/spatial_Nyquist;
b=fir1(N,cutoff_scaled);
cutoff_high=0.1*spatial_Nyquist;%1/m
cutoff_high_scaled=cutoff_high/spatial_Nyquist;
b_high=fir1(N,cutoff_high_scaled,'high');
clear B;
clear B2;
clear B_high;
clear B2_high;
B=abs(fft(b));
B_high=abs(fft(b_high));
B2=zeros(size(data3d,1),size(data3d,2));
B2_high=zeros(size(data3d,1),size(data3d,2));
for(i=1:size(data3d,1))
    B2(i,:)=B;
    B2_high(i,:)=B_high;
end;
clear test1_3d;

%apply lateral filter
for(i=1:size(data3d,3))
    test1_3d(:,:,i)=real(ifft2(B2.*B2_high.*fft2(data3d(:,:,i))));
    test2_3d(:,:,i)=real(ifft2(B2.*B2_high.*fft2(data3d2(:,:,i))));
end;
count=0;
%re-parse to 2d data structure
arr1_spf=zeros(size(arr1,1),size(arr1,2));
for(i=1:size(test1_3d,1))
    for(j=1:size(test1_3d,2))
        %check if this i,j combination is found in coords_xd0
        if(getindex3(i-(abs(min(coords_xd0(:,1)))+1),j-(abs(min(coords_xd0(:,2)))+1),1,coords_xd0)==1)
            %if it is, transfer filtered data at these coords to corresponding
            %column of output matrix
        end;
    end;
end;
clear B;
count=count+1;
arr1_spf(:,getindex3(i-(abs(min(coords_xd0(:,1)))+1),j-(abs(min(coords_xd0(:,2)))+1),1,coords_xd0))=squeeze(test1_3d(i,j,:));
end;
end;
end;
count;

%begin repeat for 2nd xducer

%re-parse to 2d data structure
arr2_spf=zeros(size(arr2,1),size(arr2,2));
for(i=1:size(test2_3d,1))
    for(j=1:size(test1_3d,2))
        %check if this i,j combination is found in coords_xd1
        if(getindex3(i-(abs(min(coords_xd1(:,1)))+1),j-(abs(min(coords_xd1(:,2)))+1),1,coords_xd1)~=

        %if it is, transfer filtered data at these coords to corresponding
        %column of output matrix
        count=count+1;
        arr2_spf(:,getindex3(i-(abs(min(coords_xd1(:,1)))+1),j-(abs(min(coords_xd1(:,2)))+1),1,coords_xd1))=squeeze(test2_3d(i,j,:));
        end;
    end;
end;
count;
%end lateral filter

fid=fopen('num_dead0.bin','wb');
fwrite(fid,length(dead_chan_xd0),'int');
fclose(fid);

fid=fopen('num_dead1.bin','wb');
fwrite(fid,length(dead_chan_xd1),'int');
fclose(fid);

fid=fopen('dead_chan_xd0.bin','wb');
fwrite(fid,dead_chan_xd0,'int');
fclose(fid);
fid=fopen('dead_chan xd1.bin','wb');
fwrite(fid,dead_chan_xd1,'int');
fclose(fid);
out=1;

doa.m – estimates direction of arrival for acquired data

%doa - direction of arrival estimation
%use SVD as in Dahl and Trahey, "Off-axis scatterer filters for improved aberration measurements," IEEE IUS 2003
function [input_delayed_opt, input_delayed_opt2, x_max1, y_max1, x_max2, y_max2] = doa(arr1_spf, arr2_spf, coords_xd0, coords_xd1, z, fs);
%establish data dimensions
num_signals=128;
num_samples=size(arr1_spf,1);
c=1500;
f0=2.5e6;
pitch=.35e-3;
%fs=25e6;
lambda=c/f0;
%create complex signals
arr1=hilbert(arr1_spf);
arr2=hilbert(arr2_spf);

%get complex autocorrelation matrices
R1_mat=zeros(num_signals,num_signals,num_samples);
for(k=1:num_samples)
    sequence1=arr1(k,:);%eqn. 4
    R1_mat(1:num_signals,1:num_signals,k)=sequence1'*sequence1;%eqn. 5
end;
R1=1/num_signals*sum(R1_mat,3);%rest of eqn. 5
clear R1_mat;
R2_mat=zeros(num_signals,num_signals,num_samples);
for(k=1:num_samples)
    sequence2=arr2(k,:);%eqn. 4
    R2_mat(1:num_signals,1:num_signals,k)=sequence2'*sequence2;% eqn. 5
end;
R2=1/num_signals*sum(R2_mat,3);%rest of eqn. 5
clear R2_mat;
%create weights
clear i; clear j; clear w; clear m;
theta_start=-35;
theta_inc=5;
theta_stop=35;
theta_ind=1;
phi_start=-35;
phi_inc=5;
phi_stop=35;
phi_ind=1;
theta_start2=-35;
theta_inc2=5;
theta_stop2=35;
theta_ind2=1;
phi_start2=-35;
phi_inc2=5;
phi_stop2=35;
phi_ind2=1;
clear w;
for(m=1:num_signals)
    for(phi=phi_start:phi_inc:phi_stop)
        for(theta=theta_start:theta_inc:theta_stop)
            w(theta_ind,phi_ind,m)=1*exp(-j*2*pi/lambda*((pitch*coords_xd0(m,1))*sin(theta*pi/180) +
            (pitch*coords_xd0(m,2))*sin(phi*pi/180) + 7e-2));%eqn. 7
            theta_ind=theta_ind+1;
        end;
        theta_ind=1;
        phi_ind=phi_ind+1;
    end;
    phi_ind=1;
end;

% find direction of max power this is broadband, uses actual delays instead of phase

% Xd1=arr1*U1(:,1)*ctranspose(U1(:,1));%use Xd1 instead of arr1_spf if you want to look
% for strongest scatterer (primary eigenvector)
% Xd2=arr2*U2(:,1)*ctranspose(U2(:,1));
shrink_factor=1.2;%factor by which coarse-to-fine proceeds
num_iterations=12;%limit on the number of iterations

for(iter=1:num_iterations)
    iter
    if(iter>1)
        clear delay; clear delay2;
        clear input_delayed; clear input_delayed2;
        clear e_step; clear e_step2;
        clear e_stepb; clear e_step2b;

        theta_interval=theta_stop-theta_start;
        theta_interval2=theta_stop2-theta_start2;
        phi_interval=phi_stop-phi_start;
        phi_interval2=phi_stop2-phi_start2;
        theta_start=theta_opt-1/2*1/shrink_factor*theta_interval;
        theta_start2=theta_opt2-1/2*1/shrink_factor*theta_interval2;
        theta_stop=theta_opt+1/2*1/shrink_factor*theta_interval;
        theta_stop2=theta_opt2+1/2*1/shrink_factor*theta_interval2;
        theta_inc=theta_inc/shrink_factor;
        theta_inc2=theta_inc2/shrink_factor;
        if(length(theta_start:theta_inc:theta_stop)<size(theta_axis,2))
            theta_stop=theta_stop+theta_inc;
        end;
        if(length(theta_start2:theta_inc2:theta_stop2)<size(theta_axis2,2))
            theta_stop2=theta_stop2+theta_inc2;
        end;
        phi_start=phi_opt-1/2*1/shrink_factor*phi_interval;
        phi_stop=phi_opt+1/2*1/shrink_factor*phi_interval;
        phi_inc=phi_inc/shrink_factor;
        phi_start2=phi_opt2-1/2*1/shrink_factor*phi_interval2;
        phi_stop2=phi_opt2+1/2*1/shrink_factor*phi_interval2;
        phi_inc2=phi_inc2/shrink_factor;
        if(length(phi_start:phi_inc:phi_stop)<size(phi_axis,2))
            phi_stop=phi_stop+phi_inc;
        end;
        if(length(phi_start2:phi_inc2:phi_stop2)<size(phi_axis2,2))
            phi_stop2=phi_stop2+phi_inc2;
        end;
    end;
end;
theta_count=0;
phi_count=0;
theta_count2=0;
phi_count2=0;
input_delayed=zeros(num_samples,num_signals);
input_delayed2=zeros(num_samples,num_signals);
for(theta_step=theta_start:theta_inc:theta_stop)
  theta_count=theta_count+1;
  phi_count=0;
  for(phi_step=phi_start:phi_inc:phi_stop)
    phi_count=phi_count+1;
    for(i=1:num_signals)
      z1=sqrt(x_step^2+y_step^2+z^2);
      delay(i,theta_count,phi_count)=sqrt((coords_xd0(i,1)*pitch-z*sin(theta_step*pi/180))^2+(coords_xd0(i,2)*pitch-z*sin(phi_step*pi/180))^2+z^2)-z;
    end;
    delay(:,theta_count,phi_count)=delay(:,theta_count,phi_count)-min(min(delay(:,theta_count,phi_count)));
  end;
end;
for(theta_step2=theta_start2:theta_inc2:theta_stop2)
  theta_count2=theta_count2+1;
  phi_count2=0;
  for(phi_step2=phi_start2:phi_inc2:phi_stop2)
    phi_count2=phi_count2+1;
    for(i=1:num_signals)
      z1=sqrt(x_step^2+y_step^2+z^2);
      delay2(i,theta_count2,phi_count2)=sqrt((coords_xd1(i,1)*pitch-z*sin(theta_step2*pi/180))^2+(coords_xd1(i,2)*pitch-z*sin(phi_step2*pi/180))^2+z^2)-z;
    end;
    delay2(:,theta_count2,phi_count2)=delay2(:,theta_count2,phi_count2)-min(min(delay2(:,theta_count2,phi_count2)));
  end;
end;
delay=delay-min(delay(:));
delay=delay/c*fs;
delay2=delay2/c*fs;
theta_count=0;
theta_count2=0;
phi_count=0;
phi_count2=0;
for(theta_step=theta_start:theta_inc:theta_stop)
    theta_count=theta_count+1;
    phi_count=0;
    for(phi_step=phi_start:phi_inc:phi_stop)
        phi_count=phi_count+1;
        input_delayed=zeros(num_samples+round(max(delay(:))),num_signals);
        for(i=1:num_signals)
            input_delayed(round(delay(i,theta_count,phi_count))+1:round(delay(i,theta_count,phi_count))+num_samples,i)=arr1_spf(:,i);
            %input_delayed(:,i)=circshift(input(:,i),round(delay(i))); 
        end;
    e_step(theta_count,phi_count)=sum(sum(input_delayed(1:num_samples,1:num_signals),2).^2)/num_samples;
end;
end;
for(theta_step2=theta_start2:theta_inc2:theta_stop2)
    theta_count2=theta_count2+1;
    phi_count2=0;
    for(phi_step2=phi_start2:phi_inc2:phi_stop2)
        phi_count2=phi_count2+1;
        input_delayed2=zeros(num_samples+round(max(delay2(:,i))),num_signals);
        for(i=1:num_signals)
            input_delayed2(round(delay2(i,theta_count2,phi_count2))+1:round(delay2(i,theta_count2,phi_count2))+num_samples,i)=arr2_spf(:,i);
            %input_delayed(:,i)=circshift(input(:,i),round(delay(i))); 
        end;
end;
e_step2(theta_count2, phi_count2) = \sum(\sum(input\_delayed2(1:num\_samples, 1:num\_signals), 2).^2) / num\_samples;

e_step2b(theta_count2, phi_count2) = \max(\sum(input\_delayed2(1:num\_samples, 1:num\_signals), 2));

end;
end;

% apply optimal steering to data and output
[theta, phi] = find(e_step == max(e_step(:))); % this gives theta_count and phi_count
[theta2, phi2] = find(e_step2 == max(e_step2(:))); % this gives theta_count2 and phi_count2
theta_opt = theta_start + theta_inc*(theta - 1);
phi_opt = phi_start + phi_inc*(phi - 1);
theta_opt2 = theta_start2 + theta_inc2*(theta2 - 1);
phi_opt2 = phi_start2 + phi_inc2*(phi2 - 1);
clear input\_delayed_opt; clear input\_delayed_opt2;
\%
for (i=1:num\_signals)
\%
\% delay_opt(i) = sqrt((coords\_xd0(i, 1) * pitch - z * sin(theta_opt*pi/180))^2 + (coords\_xd0(i, 2) * pitch - z * sin(phi_opt*pi/180))^2 + z^2) - z;
\%
input\_delayed_opt(round(delay_opt(i))+1:round(delay_opt(i))+num\_samples,i)=arr1\_spf(:,i);
\%
end;
\%
for (i=1:num\_signals)
\%
\% delay_opt2(i) = sqrt((coords\_xd1(i, 1) * pitch - z * sin(theta_opt2*pi/180))^2 + (coords\_xd1(i, 2) * pitch - z * sin(phi_opt2*pi/180))^2 + z^2) - z;
\%
input\_delayed_opt2(round(delay_opt2(i))+1:round(delay_opt2(i))+num\_samples,i)=arr2\_spf(:,i);
\%
end;
\%
store this radar image
radar\_img(iter, 1:size(e\_step, 1), 1:size(e\_step, 2)) = e\_step;
theta\_axis(iter, 1:length(theta\_start:theta\_inc:theta\_stop)) = theta\_start:theta\_inc:theta\_stop;
phi\_axis(iter, 1:length(phi\_start:phi\_inc:phi\_stop)) = phi\_start:phi\_inc:phi\_stop;
radar\_img2(iter, 1:size(e\_step2, 1), 1:size(e\_step2, 2)) = e\_step2;
theta_axis2(iter,1:length(theta_start2:theta_inc2:theta_stop2))=theta_start2:theta_inc2:theta_stop2;
phi_axis2(iter,1:length(phi_start2:phi_inc2:phi_stop2))=phi_start2:phi_inc2:phi_stop2;

%below code may be uncommented to display image of received energy vs theta and phi
%build up displayed radar image as new hi-res image becomes available
%obviously a better way to do this would be to upsample low res images to
%highest sampling frequency then composite or mosaic the data, but this is ok for now
%if(iter==1)
%    figure;
%end;
% subplot(2,1,1)
% imagesc(theta_axis(iter,:),phi_axis(iter,:),squeeze(radar_img(iter,:,:)));  
% axis square;
% drawnow;
% hold on;
% subplot(2,1,2)
% imagesc(theta_axis2(iter,:),phi_axis2(iter,:),squeeze(radar_img2(iter,:,:))); 
% axis square;
% drawnow;
% hold on;
end;%end of iterations loop
%end alternative find direction of max power
x_max1=-z*sin(theta_opt*pi/180);%-neg. fixes a sign error in here
y_max1=-z*sin(phi_opt*pi/180);
x_max2=-z*sin(theta_opt2*pi/180);
y_max2=-z*sin(phi_opt2*pi/180);

%display DOA maps
%figure;
% subplot(2,1,1)
% imagesc(-z*sin((theta_start:theta_inc:theta_stop)*pi/180),-z*sin((phi_start:phi_inc:phi_stop)*pi/180),P1) 
% axis square
% subplot(2,1,2)
% imagesc(-z*sin((theta_start:theta_inc:theta_stop)*pi/180),-z*sin((phi_start:phi_inc:phi_stop)*pi/180),P2) 
% axis square

189
%apply optimal steering to data and output
clear input_delayed_opt; clear input_delayed_opt2;
for(i=1:num_signals)
    delay_opt(i)=fs/c*sqrt((coords_xd0(i,1)*pitch-x_max1)^2+(coords_xd0(i,2)*pitch-y_max1)^2+z^2)-fs/c*z;
    delay_opt2(i)=fs/c*sqrt((coords_xd1(i,1)*pitch-x_max2)^2+(coords_xd1(i,2)*pitch-y_max2)^2+z^2)-fs/c*z;
end;
    delay_opt=delay_opt-min(delay_opt);
delay_opt2=delay_opt2-min(delay_opt2);
delay_opt=-delay_opt+abs(min(-delay_opt));
delay_opt2=-delay_opt2+abs(min(-delay_opt2));
for(i=1:num_signals)
    input_delayed_opt(round(delay_opt(i))+1:round(delay_opt(i))+num_samples,i)=arr1_spf(:,i);
    input_delayed_opt2(round(delay_opt2(i))+1:round(delay_opt2(i))+num_samples,i)=arr2_spf(:,i);
end;

%clip to maintain axial length of data
input_delayed_opt=input_delayed_opt(1:num_samples,:);
input_delayed_opt2=input_delayed_opt2(1:num_samples,:);

vlmodelmatrix_dualflex_withinmm.m - create model matrix for a specified transducer
to include all elements with an arbitrary spatial lag in mm.

%Brooks Lindsey, 1/30/11
%
fid=fopen(‘coords_flex.bin’,’rb’);
coords_flex=fread(fid,’double’);%read all coords as they appear in Both_remap.geo
coords_flex=reshape(coords_flex,256,3);%reshape into x and y coords
rx1=coords_flex;
rx2=coords_flex;
for(i=1:size(coords_flex,1))
    if(coords_flex(i,3)==1)
        rx2(i,:)=0;
    else
        rx1(i,:)=0;
    end;
end;
rx2=rx2(:,1:2);
rx1=rx1(:,1:2);
%transfer only non-zero rows into matrix to be used
cl=0;cl2=0;
for(i=1:max(size(rx1,1),size(rx2,1)))
    if(isequal(rx1(i,:),[0 0]))
    ;
    else
        cl=cl+1;
        rx1squeeze(cl,:)=rx1(i,:);
    end;
    if(isequal(rx2(i,:),[0 0]))
    ;
    else
        cl2=cl2+1;
        rx2squeeze(cl2,:)=rx2(i,:);
    end;
end;
rx=rx1squeeze;%select which array to create model matrix for, either rx1squeeze or rx2squeeze
%tx=coords_flex(1:512,:);
%rx=coords_flex(512+1:512+256,:);
fclose(fid);
%rx=coords(513:640,:);
mod1=[];
%mod1=zeros(1,128);%for 128 elements in an xducer; have to include this in case last element(s) dead
mod1(1,1)=1;
count=1;
lrcount=0;
lr2count=0;
udcount=0;
ud2count=0;
drcount=0;
dlcount=0;

%note that each 'index' below (i.e. index1 or index2) equals rx channel +1
%looking at rx channels from dual_geo
%all are off by one since matlab can't have zero as an index

pitch=0.35e-3;%element separation in meters
num_ele_x=35;
num_ele_y=35;
%within_bound=3.5e-3;
%remove dead element channels
dead_chan_xd0=[]; dead_chan_xd1=[];
%convert dead channels to element coordinates
fid=fopen('coords_flex.bin','rb');
coords_flex=fread(fid,'double');
fclose(fid);
coords_flex=reshape(coords_flex,256,3);
count1=0;
count2=0;
for(i=1:256)
if(coords_flex(i,3)==0)
    count1=count1+1;
    coords_flex_xd0(count1,1)=coords_flex(i,1);
    coords_flex_xd0(count1,2)=coords_flex(i,2);
else
    count2=count2+1;
    coords_flex_xd1(count2,1)=coords_flex(i,1);
    coords_flex_xd1(count2,2)=coords_flex(i,2);
end;
end;
clear dead_chan_xd0_coords; clear deadChan_xd1_coords;
if(length(dead_chan_xd0~)=0))
    for(i=1:length(dead_chan_xd0))
        dead_chan_xd0_coords(i,1:2)=coords_flex_xd0(dead_chan_xd0(i),1:2);
    end;
else
    dead_chan_xd0_coords=[];
end;
if(length(dead_chan_xd1~=0))
    for(i=1:length(dead_chan_xd1))
        dead_chan_xd1_coords(i,1:2)=coords_flex_xd1(dead_chan_xd1(i),1:2);
    end;
else
    dead_chan_xd1_coords=[];
end;
for(p=1:128)
    if(getindex(rx(p,1),rx(p,2),dead_chan_xd0_coords)~=1)
        rx(p,:)=[0 0];
    end;
end;
rx(~any(rx,2),:)=[];
%rx(:,~any(rx,1))=[];

%compute possibilities within this bound for a given pitch
relation=[];

m=0;
for(p=0:num_ele_x)
    for(q=0:num_ele_y)
        if(sqrt((p*pitch)^2+(q*pitch)^2)<=within_bound)
            if(p==0 & & q==0)%this would be same element, an autocorrelation
                %
            else
                m=m+1;
                relation(m,1:2)=[p,q];
            end;
        end;
    end;
end;
relation(~any(relation,2),:)=[];%remove every all-zero row
m=size(relation,1);%reset number of possible spatial relationships
%end compute possibilities

for(j=1:num_ele_x) %col
    for(i=1:num_ele_y) %row
        % debug
        %     i
        %     j
%i=i-0.5; % use these lines only for 4 els as 1
%j=j-0.5;
% only use els in relation matrix
for(k=1:m)
    if(j+relation(k,1)<num_ele_x+1 && i+relation(k,2)<num_ele_y+1 && getindex(j-18,i-18,rx)~=-1 && getindex(j-18+relation(k,1),i-18+relation(k,2),rx)~=-1)
        count=count+1;
        index1=getindex(j-18,i-18,rx);
        %mod1(count,index1)=1;
        index2=getindex(j-18+relation(k,1),i-18+relation(k,2),rx);
        if(index1<index2)
            mod1(count,index1)=1;
            mod1(count,index2)=-1;
        else
            mod1(count,index2)=1;
            mod1(count,index1)=-1;
        end;
    end;
end;
end;
end;
end;
end;
end;
end;
end;
end;
end;
end;
end;
end;
end;
end;
fid = fopen('mod1.bin','wb');
fwrite(fid, mod1, 'double');
fclose(fid);
mod1=sparse(mod1);

ata = mod1'*mod1;

% perform cholesky factorization
% lower triangular storage
atac = chol(ata);

atacb = band_storage_l(atac);

fid = fopen('atacbC.bin','wb');
fwrite(fid,atacb,'double');
fclose(fid);

[I J V] = find(mod1);
I = I-1;
J = J-1;
fid = fopen('IC.bin','wb');
fwrite(fid,I,'int');
fclose(fid);
 fid = fopen('JC.bin','wb');
fwrite(fid,J,'int');
fclose(fid);
 fid = fopen('VC.bin','wb');
fwrite(fid,V,'double');
fclose(fid);

M = (length(I)+1)/2
N = size(mod1,2)
ku = size(atacb,1)-1
lrcount;
lrcount;
udcount;

fid=fopen('modelMatC.bin','wb');
fwrite(fid,full(mod1),'double');
fclose(fid);
%[all above code is repeated for second transducer but omitted here for brevity]

%write M, N, ku, etc. to be read into pancorr for data sizes
fid = fopen('M.bin','wb');
fwrite(fid,M,'int');
fclose(fid);
fid = fopen('M2.bin','wb');
fwrite(fid,M2,'int');
fclose(fid);
fid = fopen('N.bin','wb');
fwrite(fid,N,'int');
fclose(fid);
fid = fopen('N2.bin','wb');
fwrite(fid,N2,'int');
fclose(fid);
fid = fopen('ku.bin','wb');
fwrite(fid,ku,'int');
fclose(fid);
fid = fopen('ku2.bin','wb');
fwrite(fid,ku2,'int');
fclose(fid);

pa_ncorrMod12_corn.c – takes in data and model matrix, runs correlations using a
sliding window method and outputs delay updates. This code was originally written by
Gianmarco Pinton and modified by Nik Ivancevich.

//include "fftw3.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <assert.h>
#include "inout.h"
#include <time.h>
#include "interph.h"
//include "LAPACK.lib"
#define TWOPI 6.28318530717958647692528676655

/* Gianmarco Pinton
 * last modified: 030908
 * based on pa_ncorr4.c
 * uses make_model2.m
 * optimized linear algebra
 */

double *** allocate_d_3d( int row_dim, int col_dim, int dep_dim);
double ** allocate_d_2d( int row_dim, int col_dim);
short ** allocate_s_2d( int row_dim, int col_dim);
int highest_peak ( double * vec, int len );
int nearest_peak ( double * vec, int len, int win );
void sparse_mult (int *I, int *J, double *V, double *b, double *x, int m, int n);
void ncorr ( double * ref, double * disp,
            double * mom1_ref, double * mom2_ref,
            double * mom1_disp, double * mom2_disp,
            int length, int kernel_length, int search_up, int search_down,
            double ** corr);
void mom1v( double * vec, int length, int window, double * svm );
void mom2v( double * vec, int length, int window, double * svs );
void pinterp (double y1, double y2, double y3,
              double * max_value, double * max_index);
void cholsl_band (double **a, int n, double b[], double x[], int kl);
double absD (double absin);
char round(double roundin);
int roundi(double roundin);

struct dimensions { int channels; int axial; };
struct search {
    int num_elements;
    int ** coord; /* lateral and elevation search coordinates */
    int row_reduce; /* model matrix row search to eliminate redundant rows */
};

int main ()
{

int i, k, j, side;
float edge_count, edge_count2;
int itmp;
double dtmp;
double dtmp_avg, dtmp_avg2;
double detx_mean, detx_mean2; // bd1
double x_mean, x2_mean; // bd1
double *detx_full, *detx2_full; // bd1
double *x_full, *x2_full; // bd1

struct dimensions d;
// double * frf; /* focused rf */
double * frf;
double * corr_index, * corr_index2;
struct search s; /* search */

/* ncorrv */
double ** ref, ** mom1, ** mom2;
double ** corr;
int search_up=6, search_down=6; // note: if your window > 1 lambda, you will have
problems (w/ detrending)
int search_up_new, search_down_new;
int window;
int chanref, chandisp;

/* model */
double * V, * V2;
*dead_chan_xd0, *dead_chan_xd1;
int num_dead0, num_dead1;
double *weight, *weight2;
// bd1 3/23/11 doesn’t matter what these are initialized to now since you’re reading them
in down below
int M = 0;
int M2 = 0;
int N = 0;
int N2 = 0;
int next_dead_index0, next_dead_index1, next_dead0, next_dead1;
*/ linear algebra */
double *b1, *b2, **a, **a2, *x, *x2, *dtx, *dtx2, *xrx;
int ku = 0;
int ku2 = 0;
char *xrxC;
char *dtxC, *dtxCrep;

/* detrending */
double **detata, **detata2;
short **posvec, **posvec2;
double planeCoeffs[3]={0,0,0};
double planeCoeffs2[3]={0,0,0};

/* timing */
clock_t tv1, tv2;
double time;

/*** fix some variables ***/
tv1 = clock();

/* declare dimensions */
//d.lateral = 20; /* slow */
//d.elevation = 20; /* medium */
d.channels = 128;
d.axial = 256; /* fast */

/* search */
s.num_elements = 1;

/* read rf data */
*/
*/
frf = malloc(sizeof(double)*d.channels*d.axial);
inV("testdata.bin",frf,d.channels*d.axial);
*/
frf = malloc(sizeof(double)*d.channels*d.axial);
//inV("focused.bin",frf,2*d.channels*d.axial);
/printf("%f
%f
",frf[1*d.elevation*d.axial+2*d.axial+3],frf[5*d.elevation*d.axial+6*d.axial+7]);

/* read in model */
M_pt = (int*)malloc(sizeof(int)*(1));
M2_pt = (int*)malloc(sizeof(int)*(1));
N_pt = (int*)malloc(sizeof(int)*(1));
N2_pt = (int*)malloc(sizeof(int)*(1));
k1_pt = (int*)malloc(sizeof(int)*(1));
k2_pt = (int*)malloc(sizeof(int)*(1));

inVi("M.bin",M_pt,1);
inVi("M2.bin",M2_pt,1);
inVi("N.bin",N_pt,1);
inVi("N2.bin",N2_pt,1);
inVi("ku.bin",k1_pt,1);
inVi("ku2.bin",k2_pt,1);

//copy values to actual variables
M=*M_pt;
M2=*M2_pt;
N=*N_pt;
N2=*N2_pt;
k1=*k1_pt;
k2=*k2_pt;

V = (double*)malloc(sizeof(double)*(M*2-1));
I = (int*)malloc(sizeof(int)*(M*2-1));
J = (int*)malloc(sizeof(int)*(M*2-1));
m1_map = (int*)malloc(sizeof(int)*(128)); // bd1 hard coded these, not N
m2_map = (int*)malloc(sizeof(int)*(128));
m1_map_tx = (int*)malloc(sizeof(int)*256);
m2_map_tx = (int*)malloc(sizeof(int)*256);
geo_map = (int*)malloc(sizeof(int)*256);
inV("VC.bin",V,M*2-1);
inVi("IC.bin",I,M*2-1);
inVi("JC.bin",J,M*2-1);

V2 = (double*)malloc(sizeof(double)*(M2*2-1));
I2 = (int*)malloc(sizeof(int)*(M2*2-1));
J2 = (int*)malloc(sizeof(int)*(M2*2-1));
inV("VC2.bin",V2,M2*2-1);
inVi("IC2.bin",I2,M2*2-1);
inVi("JC2.bin",J2,M2*2-1);

/* linear algebra */
b1 = (double*)malloc(sizeof(double)*N);
b2 = (double*)malloc(sizeof(double)*N2);
a = allocate_d_2d(N,ku+1);
a2 = allocate_d_2d(N2,ku2+1);
x = (double*)malloc(sizeof(double)*N);
x2 = (double*)malloc(sizeof(double)*N2);
detx = (double*)malloc(sizeof(double)*N*2);
detx2 = (double*)malloc(sizeof(double)*N2*2);
detx_full = (double*)malloc(sizeof(double)*128*2);
detx2_full = (double*)malloc(sizeof(double)*128*2);
x_full = (double*)malloc(sizeof(double)*128*2);
x2_full = (double*)malloc(sizeof(double)*128*2);
detxC = (char*)malloc(sizeof(char)*768);
detxCrep = (char*)malloc(sizeof(char)*768);
xrx = (double*)malloc(sizeof(double)*256);
xrxC = (char*)malloc(sizeof(char)*256);
weight = (double*)malloc(sizeof(double)*d.axial);
weight2 = (double*)malloc(sizeof(double)*d.axial);

/* detrending */
detata = allocate_d_2d(3,N);
detata2 = allocate_d_2d(3,N2);
posvec = allocate_s_2d(3,N);
posvec2 = allocate_s_2d(3,N2);
for(side=0;side<2;side++){
    if(side==0){
        inV("focused1.bin",frf,N*d.axial); // bdl was d.channels*d.axial
        *V = *V;
        *I = *I;
        *J = *J;
        ku = ku;
        d.channels = N;
    }
    else{
        inV("focused2.bin",frf,N2*d.axial); // bdl was d.channels*d.axial
    }
}
\*V = \*V2;
\*I = \*I2;
\*J = \*J2;

D_channels = N2;
//ku = ku2;
//N = N2;
//M = M2;

} /** normalized cross correlation **/
printf("start normalized cross correlation \n");

corr_index = malloc(sizeof(double)*M);
corr_index2 = malloc(sizeof(double)*M2);
window = d.axial-(search_up+search_down);

//ref = allocate_d_3d(d.lateral, d.elevation, d.axial);
ref = allocate_d_2d(d.channels, d.axial);
//mom1 = allocate_d_3d(d.lateral, d.elevation, d.axial - window +1);
//mom2 = allocate_d_3d(d.lateral, d.elevation, d.axial - window +1);
mom1 = allocate_d_2d(d.channels, d.axial - window +1);
mom2 = allocate_d_2d(d.channels, d.axial - window +1);
corr = allocate_d_2d(1,search_up+search_down+1);

/* find moments */
/*for(i=0; i<d.lateral; i++){
   for(j=0; j<d.elevation; j++){/* vec placement */
      /* for(k=0; k<d.axial; k++)
         ref[i][j][k] = frf[i*d.elevation + j*axial+k];
      mom1v(ref[i][j],d.axial,window,mom1[i][j]);
      mom2v(ref[i][j],d.axial,window,mom2[i][j]);
   }
*/
*/

/* find moments */
for(i=0; i<d.channels; i++){/* vec placement */
   for(k=0; k<d.axial; k++)
      ref[i][k] = frf[i*d.axial+k];
}
mom1v(ref[i],d.axial,window,mom1[i]);
mom2v(ref[i],d.axial,window,mom2[i]);
}
/*if(side==0){
   outV("mom2a.bin",*mom2,d.axial);
   //compute weights
   for(i=0; i<d.axial; i++){
      weight[i] = 1/* mom2[i];
   }
}
else{
   outV("mom2b.bin",*mom2,d.axial);
   for(i=0; i<d.axial; i++){
      weight2[i] = 1/* mom2[i];
   }
}
*/
/* perform correlations */
/* first is reference
* next are row pairs */

//printf("%u \
",M);
if(side==0){
dtmp_avg=0;
edge_count=0;
for(i=1; i<M; i++){  
   /*jref = I[2*i-1]%d.elevation;
   iref = (I[2*i-1]-jref)/d.elevation;
   jdisp = I[2*i]%d.elevation;
   idisp = (I[2*i]-jref)/d.elevation;*/

   chanref = I[2*i-1];
   chandisp = I[2*i];

   /*ncorr ( ref[iref][jref], ref[idisp][jdisp],
   mom1[iref][jref], mom2[iref][jref],
   mom1[idisp][jdisp],mom2[idisp][jdisp],
   d.axial, window, search_up, search_down,
   corr );*/
}
ncorr (ref[chanref], ref[chandisp], 
mom1[chanref], mom2[chanref], 
mom1[chandisp], mom2[chandisp], 
d.axisl, window, search_up, search_down, 
corr);

/* make sure second argument is odd */
itmp = highest_peak(corr[0], search_up+search_down+1);
    //tmp = nearest_peak(corr[0], search_up+search_down+1, 21);
dtmp = corr[0][itmp];

if(itmp==0)
    corr_index[i] = -search_up;
else if (itmp==search_up+search_down)
    corr_index[i] = search_down;
else {
    pinterp(corr[0][itmp-1],corr[0][itmp],corr[0][itmp+1],&dtmp,&corr_index[i]);
    corr_index[i] += itmp - search_up;
}

    //printf("%f,%f,%u \n",dtmp,corr_index[i],i);
dtmp_avg = dtmp_avg + dtmp;
if(corr_index[i]==search_up || corr_index[i]==-search_down){
    edge_count = edge_count + 1;
}
}
dtmp_avg = dtmp_avg/(M-1);
    //end if side==0
else {//if side==1
    dtmp_avg2=0;
    edge_count2=0;
    for(i=1; i<M2; i++)
        /*jref = I[2*i-1]%d.elevation;
        iref = (I[2*i-1]-jref)/d.elevation;
        jdisp = I[2*i]%d.elevation;*/

204
idisp = (I[2*i]-jref)/d.elevation; /*

    chanref = I2[2*i-1];
    chandisp = I2[2*i];

    /*ncorr ( ref[iref][jref], ref[idisp][jdisp],
            mom1[iref][jref], mom2[iref][jref],
            mom1[idisp][jdisp],mom2[idisp][jdisp],
            d.axial, window, search_up, search_down,
            corr );*/
    
    //printf("%u \n",i);

    ncorr ( ref[chanref], ref[chandisp],
            mom1[chanref], mom2[chanref],
            mom1[chandisp],mom2[chandisp],
            d.axial, window, search_up, search_down,
            corr );
    
    /* make sure second argument is odd */
    itmp = highest_peak(corr[0],search_up+search_down+1);
    //itmp = nearest_peak(corr[0],search_up+search_down+1, 21);
    dtmp = corr[0][itmp];

    if(itmp==0){
        corr_index2[i] = -search_up;
    } else if (itmp==search_up+search_down){
        corr_index2[i] = search_down;
    } else {
        pinterp(corr[0][itmp-1],corr[0][itmp],corr[0][itmp+1],&dtmp,&corr_index2[i]);
        corr_index2[i] += itmp - search_up;
    }
    //printf("%f,%f,%u \n",dtmp,corr_index2[i],i);
    dtmp_avg2=dtmp_avg2+dtmp;
    if(corr_index2[i]==search_up || corr_index2[i]==-search_down){
        edge_count2=edge_count2+1;
    }
    }
    }
\[ dtmp\_avg2 = \frac{dtmp\_avg2}{(M - 1)}; \]

```c
// end else side==1

//printf("\%u \n", i);
if (side==0) {
    printf("Mean max corr. value: \%f \n", dtmp\_avg);
    //printf("Edge count: \%f \n", edge\_count);
    printf("Edge percent: \%f \n", edge\_count/M);
} else { // side==1
    printf("Mean max corr. value: \%f \n", dtmp\_avg2);
    //printf("Edge count: \%f \n", edge\_count2);
    printf("Edge percent: \%f \n", edge\_count2/M2);
}

printf("End perform correlations \n");
corr\_index[0] = 0;
corr\_index2[0] = 0;
outV("correlation.bin", corr\_index, M);
outV("correlation2.bin", corr\_index2, M2);

/* read matrix into buffer then reshape into a */
if (side==0) {
    /* multiply corr\_index by transpose of model matrix */
    sparse\_mult (I, J, V, corr\_index, b1, M*2-1, N);
    outV("b1.bin", b1, N);
    inM("atacbC.bin", a, N, ku+1);
} else {
    /* multiply corr\_index by transpose of model matrix */
    sparse\_mult (I2, J2, V2, corr\_index2, b2, M2*2-1, N2);
    outV("b2.bin", b2, N2);
    inM("atacbC2.bin", a2, N2, ku2+1);
}

printf("solving system \n");
if (side==0) {
    cholsl\_band(a, N, b1, x, ku);
    outV("x.bin", x, N);
} else {
```
```
```c
  cholsl_band(a2,N2,b2,x2,ku2);
  outV("x2.bin",x2,N2);
}

/* read in detrending matrix, get plane coefficients */
if(side==0){
inM("detata_flex1.bin",detata,3,N);
inMs("posvec_flex1.bin",posvec,3,N);
  for(i=0;i<3;i++){
    for(j=0;j<N;j++){
      planeCoeffs[i] += detata[i][j]*x[j];
    }
  }
}
else{
inM("detata_flex2.bin",detata2,3,N2);
inMs("posvec_flex2.bin",posvec2,3,N2);
  for(i=0;i<3;i++){
    for(j=0;j<N2;j++){
      planeCoeffs2[i] += detata2[i][j]*x2[j];
    }
  }
}

outV("planeCoeffs.bin",planeCoeffs,3);
outV("planeCoeffs2.bin",planeCoeffs2,3);
detx_mean=0;//bdl
detx_mean2=0;
/* detrend data */
if(side==0){
  for(i=0;i<N;i++){
    detx[i]=x[i]-posvec[0][i]*planeCoeffs[0]-posvec[1][i]*planeCoeffs[1] - posvec[2][i]*planeCoeffs[2];
    detx_mean=detx_mean+detx[i];//bdl
  }
}
else{
  for(i=0;i<N2;i++){
    detx2[i]=x2[i]-posvec2[0][i]*planeCoeffs2[0]-posvec2[1][i]*planeCoeffs2[1] - posvec2[2][i]*planeCoeffs2[2];
    detx_mean2=detx_mean2+detx2[i];//bdl
  }
}
}```
detx_mean2=detx_mean2+detx2[i];  //bdl
}

//begin bdl subtract off mean
if(side==0){
    detx_mean=detx_mean/N;
    for(i=0;i<N;i++){
        detx[i]=detx[i]-detx_mean;
    }
}
else{
    detx_mean2=detx_mean2/N2;
    for(i=0;i<N2;i++){
        detx2[i]=detx2[i]-detx_mean2;
    }
}
//end bdl subtract off mean

/* interpolate data to rx positions */
/* for(i=0;i<128;i++){
    xrx[i]=0;
    for(j=0;j<interporder[i*5];j++){
        xrx[i] += ((double)detx[interporder[i*5+j+1]]) / ((double)interporder[i*5]);
    }
    xrxC[i]=round(xrx[i]*1.6);  //sample *1s/25e6samples * 1e9ns/1s * 1clock/25ns = 1.6
} */

if(side==0){
    outV("detx0.bin",detx,N);
    //outV("xrx0.bin",xrx, N);
    for(i=0; i<512; i++){
        detxC[i]=0;//bdl - write all Tx delays as zero for now
    }
    for(i=512; i<640; i++){
        detxC[i]=round(detx[i-512]*1.6);
    }
}
else{
    outV("detx1.bin",detx2,N2);
    //outV("xrx1.bin",xrx, N2);
    for(i=0; i<512;i++)  
        //need to fill outer ring of tx els with same delay update as corresponding rx el, need rx to tx map
        detxC[i]=0;//bdl - write all Tx delays as zero for now
        detxCrep[i]=0;//bdl
}  
    for(i=640; i<768; i++)  
        //detxC[i]=round(detx[i-512-128]*1.6);  
        detxC[i]=round(detx2[i-512]*1.6);
}
}  

//bdl - repackage delays to send back to scanner  
num_dead0_pt = (int*)malloc(sizeof(int)*(1));  
num_dead1_pt = (int*)malloc(sizeof(int)*(1));  
inVi("num_dead0.bin",num_dead0_pt,1);  
inVi("num_dead1.bin",num_dead1_pt,1);  
num_dead0=*num_dead0_pt;  
num_dead1=*num_dead1_pt;  
dead_chan_xd0 = (int*)malloc(sizeof(int)*num_dead0);  
dead_chan_xd1 = (int*)malloc(sizeof(int)*num_dead1);  
inVi("dead_chan_xd0.bin",dead_chan_xd0,num_dead0); //these are sorted lo to hi  
inVi("dead_chan_xd1.bin",dead_chan_xd1,num_dead1);  
inVi("m1_map.bin",m1_map,128);//have to tell these which els are dead to zero these out (or turn them off)  
inVi("m2_map.bin",m2_map,128);  
inVi("m1_map_tx.bin",m1_map_tx,256);  
inVi("m2_map_tx.bin",m2_map_tx,256);  
next_dead_index0=0;  
next_dead0= dead_chan_xd0[next_dead_index0];  
next_dead_index1=0;  
next_dead1=dead_chan_xd1[next_dead_index1];  
//first fill deads with zeros so both delay update vectors have length 128;  
if(side==0){  
    k=0;
for(i=0; i<128; i++){  
    if(i==(next_dead0-1)) { // -1 for matlab indexing  
        if(next_dead_index0<num_dead0) {  
            //printf("%i\n",next_dead0);  
            detx_full[i]=0;  
            next_dead_index0++;  
            next_dead0=dead_chan_xd0[next_dead_index0];  }
        }
     }
    else{  
        detx_full[i]=detx[k];  
        k++;
     }
}
outV("detx0_full.bin",detx_full,128);
}
else{
    k=0;
    for(i=0; i<128; i++){  
        if(i==(next_dead1-1)) { // -1 for matlab indexing  
            if(next_dead_index1<num_dead1) {  
                //printf("%i\n",next_dead1);  
                detx2_full[i]=0;  
                next_dead_index1++;  
                next_dead1=dead_chan_xd1[next_dead_index1];  }
            }
        }
        else{  
            detx2_full[i]=detx2[k];  
            k++;
        }
    }
    outV("detx1_full.bin",detx2_full,128);
}

//next repackage these delays to send back to scanner

for(i=0; i<128;i++){  
    //detxCrep[512+m1_map[i]-1]=detxC[i+512];  
    detxCrep[512+m2_map[i]-1]=round(detx_full[i]*1.6);  
}
for(i=0; i<128;i++){  
    //detxCrep[512+m2_map[i]-1]=detxC[i+640];  
detxCrep[512+m1_map[i]-1]=round(detx2_full[i]*1.6);  
}
inVi("geo_map.bin",geo_map,256);//index is Tx chan #, value is Rx chan #  
for(i=1; i<=256;i++){//all outer ring transmitters are located in first 256 transmit channels  
detxCrep[i-1]=detxCrep[512+geo_map[i-1]-1];  
}

outVc("dataIn.dat",detxCrep,768);//was xrxC instead  
tv2 = clock();  
time = (tv2 - tv1)/(CLOCKS_PER_SEC / (double) 1000.0);  
printf("Execution time : %.f ms\n",time);
}

//begin 2nd time through  
/*
for(side=0;side<2;side++){
    search_up = 5;  search_down = 5;  
    if(side==0){  
        inV("focused1.bin",frf,N*d.axial);//bdl was d.channels*d.axial  
        *V= *V;  
        *I= *I;  
        *J= *J;  
        ku=ku;  
        d.channels=N;  
    }
    else{  
        inV("focused2.bin",frf,N2*d.axial);//bdl was d.channels*d.axial  
        *V= *V2;  
        *I= *I2;  
        *J= *J2;  
        d.channels=N2;  
        //ku=ku2;  
        //N=N2;  
        //M=M2;  
    }
}
// normalized cross correlation
printf("start normalized cross correlation\n");

corr_index = malloc(sizeof(double)*M);
corr_index2 = malloc(sizeof(double)*M2);
window = d.axial-(search_up+search_down);

//ref = allocate_d_3d(d.lateral, d.elevation, d.axial);
ref = allocate_d_2d(d.channels, d.axial);
//mom1 = allocate_d_3d(d.lateral, d.elevation, d.axial-window+1);
//mom2 = allocate_d_3d(d.lateral, d.elevation, d.axial-window+1);
mom1 = allocate_d_2d(d.channels, d.axial-window+1);
mom2 = allocate_d_2d(d.channels, d.axial-window+1);
corr = allocate_d_2d(1,search_up+search_down+1);

/* find moments */
/*for(i=0; i<d.lateral; i++){
 for(j=0; j<d.elevation; j++){
  /* vec placement */
  /* for(k=0; k<d.axial; k++)
   ref[i][j][k] = frf[i*d.elevation*d.axial+j*d.axial+k];
   mom1v(ref[i][j],d.axial,window,mom1[i][j]);
   mom2v(ref[i][j],d.axial,window,mom2[i][j]);
  }
 }*/

// find moments
if(side==0){
 for(i=0; i<d.channels; i++){
  // vec placement
  for(k=0; k<d.axial; k++)
   ref[i][k] = frf[i*d.axial+k];
  if(roundi((x[i]-detx[i])<0){
   search_up_new=search_up+abs(roundi((x[i]-detx[i])));
  }
  else{ search_up_new=search_up-roundi((x[i]-detx[i]));
   if(roundi((x[i]-detx[i])<(search_up)){
    search_down_new=(search_down+roundi((x[i]-detx[i])));
   }
  else{
   212
search_down_new = search_down + roundi((x[i] - detx[i]));

} // side==1

for (i=0; i<d.channels; i++) {
    // vec placement
    for (k=0; k<d.axial; k++) {
        ref[i][k] = frf[i*d.axial+k];
    }
    if (roundi((x2[i] - detx2[i]))<0) {
        search_up_new = search_up + abs(roundi((x2[i] - detx2[i])));
    } else { search_up_new = search_up - roundi((x2[i] - detx2[i]));}
    if (roundi((x2[i] - detx2[i]))<search_up) {
        search_down_new = (search_down + roundi((x2[i] - detx2[i])));
    } else {
        search_down_new = search_down + roundi((x2[i] - detx2[i]));
    }
    window = d.axial - (search_up_new + search_down_new);
}

mom1v(ref[i],d.axial,window,mom1[i]);
mom2v(ref[i],d.axial,window,mom2[i]);
}
else {//side==1
}
} // side==1

/*if(side==0){
  outV("mom2a.bin",*mom2,d.axial);
  // compute weights
  for (i=0; i<d.axial; i++){
    weight[i] = 1/* mom2[i];
  }
}*/

213
else{
    outV("mom2b.bin",*mom2,d.axial);
    for(i=0; i<d.axial; i++){
        weight2[i] = 1/(* mom2[i]);
    }
}

// perform correlations /
// first is reference
// next are row pairs
//printf("%u \n",M);
if(side==0){
    dtmp_avg=0;
    edge_count=0;
    for(i=1; i<M; i++){ /*jref = I[2*i-1]%d.elevation;
        iref = (I[2*i-1]-jref)/d.elevation;
        jdisp = I[2*i]%d.elevation;
        idisp = (I[2*i]-jref)/d.elevation;
        chanref = I[2*i-1];
        chandisp = I[2*i];
        /*ncorr ( ref[iref][jref], ref[idisp][jdisp],
            mom1[iref][jref], mom2[iref][jref],
            mom1[idisp][jdisp],mom2[idisp][jdisp],
            d.axial, window, search_up, search_down,
            corr );

        //printf("%u \n",i);
        //printf("%i \n",roundi(x[i]-detx[i]));
        //printf("%i \n",search_down_new);
        //printf("%i \n",search_up_new);
        //printf("%i \n",search_down+roundi(x[i]-detx[i]));
        //printf("%i \n",chanref);
        //printf("%i \n",chandisp);
        //printf("%i \n",corr);
        //search_up=5; search_down=5;
if(roundi((x[chanref]-dtx[chanref])-(x[chandisp]-dtx[chandisp]))<0){
    search_up_new=search_up+abs(roundi((x[chanref]-dtx[chanref])-(x[chandisp]-dtx[chandisp])));
} else{
    search_up_new=search_up-roundi((x[chanref]-dtx[chanref])-(x[chandisp]-dtx[chandisp]));
}
if(roundi((x[chanref]-dtx[chanref])-(x[chandisp]-dtx[chandisp]))<search_up){
    search_down_new=(search_down+roundi((x[chanref]-dtx[chanref])-(x[chandisp]-dtx[chandisp])));
} else{
    search_down_new=search_down+roundi((x[chanref]-dtx[chanref])-(x[chandisp]-dtx[chandisp])));

ncorr ( ref[chanref], ref[chandisp],
    mom1[chanref], mom2[chanref],
    mom1[chandisp],mom2[chandisp],
    d.axial, window, search_up_new, search_down_new,//bdl check which should
    be +/-
    corr );

    // make sure second argument is odd
    itmp = highest_peak(corr[0],search_up_new+search_down_new+1);
    itmp = nearest_peak(corr[0],search_up_new+search_down_new+1, 21);
    dtmp = corr[0][itmp];

    if(itmp==0){
        corr_index[i] = -search_up_new;
    } else if (itmp==search_up_new+search_down_new){
        corr_index[i] = search_down_new;
    } else {
        pinterp(corr[0][itmp-1],corr[0][itmp],corr[0][itmp+1],&dtmp,&corr_index[i]);
        corr_index[i]+=itmp-search_up_new;
    }

    //printf("%f,%f,%u \n",dtmp,corr_index[i],i);

}
dtmp_avg = dtmp_avg + dtmp;
if (corr_index[i] == (search_up_new) || corr_index[i] == (-search_down_new)) {
    edge_count = edge_count + 1;
}
}
dtmp_avg = dtmp_avg / (i - 1); // was / (M - 1)
} // end if side == 0
else { // if side == 1
    dtmp_avg2 = 0;
    edge_count2 = 0;
    for (i = 1; i < M2; i++) {

    /* jref = I[2*i-1]%d.elevation;
    iref = (I[2*i-1]-jref)/d.elevation;
    jdisp = I[2*i]%d.elevation;
    idisp = (I[2*i]-jref)/d.elevation;

    chanref = I2[2*i-1];
    chandisp = I2[2*i];
    /* ncorr ( ref[iref][jref], ref[idisp][jdisp],
        mom1[iref][jref], mom2[iref][jref],
        mom1[idisp][jdisp],mom2[idisp][jdisp],
        d.axial, window, search_up, search_down,
        corr );

    // printf("%u \n", search_up_new);
    // printf("%u \n", search_down_new);
    if (roundi((x2[chanref] - detx2[chanref]) -
                (x2[chandisp] - detx2[chandisp]) < 0) {
        search_up_new = search_up + abs(roundi((x2[chanref] - detx2[chanref]) -
                                                (x2[chandisp] - detx2[chandisp])));
    } else { search_up_new = search_up - roundi((x2[chanref] - detx2[chanref]) -
                                                (x2[chandisp] - detx2[chandisp]));
    if (roundi((x2[chanref] - detx2[chanref]) -
                (x2[chandisp] - detx2[chandisp]) <=
                search_up) {
        search_down_new = -(search_down + roundi((x2[chanref] - detx2[chanref]) -
                                                (x2[chandisp] - detx2[chandisp])));
    } else {

216
search_down_new = search_down + roundi((x2[chanref] - detx2[chanref]) -
(x2[chandisp] - detx2[chandisp]));
}

// printf("%u \n",search_up_new);
// printf("%u \n",search_down_new);
n_corr ( ref[chanref], ref[chandisp],
mom1[chanref], mom2[chanref],
mom1[chandisp],mom2[chandisp],
d.axial, window, search_up_new, search_down_new,
corr );

// make sure second argument is odd
itmp = highest_peak(corr[0], search_up_new + search_down_new + 1);
//itmp = nearest_peak(corr[0], search_up + search_down + 1, 21);
dtmp = corr[0][itmp];

if (itmp == 0){
corr_index2[i] = -search_up_new;
}
else if (itmp == search_up + search_down){
corr_index2[i] = search_down_new;
}
else {
pinterp(corr[0][itmp-1],corr[0][itmp],corr[0][itmp+1],&dtmp,&corr_index2[i]);
corr_index2[i] += itmp - search_up_new;
}

// printf("%i \n",search_up_new);
// printf("%i \n",search_down_new);
// printf("%f,%f,%u \n",dtmp,corr_index2[i],i);
dtmp_avg2 = dtmp_avg2 + dtmp;
if (corr_index2[i] == (search_up_new) || corr_index2[i] == (-search_down_new)){
edge_count2 = edge_count2 + 1;
}
}
dtmp_avg2 = dtmp_avg2 / (i-1);
//end else side==1

// printf("%u \n", i);
if (side == 0){
printf("Mean max corr. value: %f \n", dtmp_avg);
}
//printf("Edge count: %f \n",edge_count);
printf("Edge percent: %f \n",edge_count/M);
} else{//side==1
printf("Mean max corr. value: %f \n",dtmp_avg2);
printf("Edge count: %f \n",edge_count2);
printf("Edge percent: %f \n",edge_count2/M2);
} printf("End perform correlations\n");
corr_index[0] = 0;
corr_index2[0] = 0;
outV("correlation.bin",corr_index,M);
outV("correlation2.bin",corr_index2,M2);

// read matrix into buffer then reshape into a
if(side==0){
    // multiply corr_index by transpose of model matrix
    sparse_mult (I,J,V,corr_index, b1, M*2-1,N);
    outV("b1.bin",b1,N);
inM("atacbC.bin",a,N,ku+1);
}
else{
    // multiply corr_index by transpose of model matrix
    sparse_mult (I2,J2,V2,corr_index2, b2, M2*2-1,N2);
    outV("b2.bin",b2,N2);
inM("atacbC2.bin",a2,N2,ku2+1);
}
printf("solving system\n");
if(side==0){
    cholsl_band(a,N,b1,x,ku);
    outV("x_narrow.bin",x,N);
}
else{
    cholsl_band(a2,N2,b2,x2,ku2);
    outV("x2_narrow.bin",x2,N2);
}
// read in detrending matrix, get plane coefficients
planeCoeffs[0]=0; planeCoeffs[1]=0; planeCoeffs[2]=0;
planeCoeffs2[0]=0; planeCoeffs2[1]=0; planeCoeffs2[2]=0;
if(side==0){
inM("detata_flex1.bin",detata,3,N);
inMs("posvec_flex1.bin",posvec,3,N);
for(i=0;i<3;i++){
    for(j=0; j<N; j++){
        planeCoeffs[i] += detata[i][j]*x[j];
    }
}
}
else{
inM("detata_flex2.bin",detata2,3,N2);
inMs("posvec_flex2.bin",posvec2,3,N2);
for(i=0;i<3;i++){
    for(j=0; j<N2; j++){
        planeCoeffs2[i] += detata2[i][j]*x2[j];
    }
}
}
outV("planeCoeffs.bin",planeCoeffs,3);
outV("planeCoeffs2.bin",planeCoeffs2,3);
detx_mean=0; // bdl
detx_mean2=0;
x_mean=0;
x2_mean=0;
// detrend data
if(side==0){
    for(i=0;i<N;i++){
        detx[i]=x[i]-posvec[0][i]*planeCoeffs[0]-posvec[1][i]*planeCoeffs[1]-posvec[2][i]*planeCoeffs[2];
        detx_mean=detx_mean+detx[i]; // bdl
        x_mean=x_mean+x[i];
    }
}
else{
    for(i=0;i<N2;i++){
detx2[i]=x2[i]-posvec2[0][i]*planeCoeffs2[0]-posvec2[1][i]*planeCoeffs2[1]-posvec2[2][i]*planeCoeffs2[2];
detx_mean2=detx_mean2+detx2[i]; //bdl
x2_mean=x2_mean+x2[i];
}
}
//begin bdl subtract off mean
if(side==0){
detx_mean=detx_mean/N;
x_mean=x_mean/N;
for(i=0;i<N;i++){
detx[i]=detx[i]-detx_mean;
x[i]=x[i]-x_mean;
}
}
else{
detx_mean2=detx_mean2/N2;
x2_mean=x2_mean/N2;
for(i=0;i<N2;i++){
detx2[i]=detx2[i]-detx_mean2;
x2[i]=x2[i]-x2_mean;
}
}
//end bdl subtract off mean
// interpolate data to rx positions
/* for(i=0;i<128;i++){

    xrx[i]=0;
    for(j=0;j<interporder[i*5];j++){

        xrx[i] += ((double)detx[interporder[i*5+j+1]]) / ((double)interporder[i*5]);

    }

    xrxC[i]=round(xrx[i]*1.6); //sample *1s/25e6samples * 1e9ns/1s * 1clock/25ns = 1.6

} */
if(side==0){
    outV("detx0_narrow.bin",detx,N);
}

//outV("xrx0.bin",xrx, N);
for(i=0; i<512;i++){
    detxC[i]=0;//bdl - write all Tx delays as zero for now
}
for(i=512; i<640;i++){
    detxC[i]=round(detx[i-512]*1.6);
}
else{
    outV("detx1_narrow.bin",detx2,N2);
    //outV("xrx1.bin",xrx, N2);

    for(i=0; i<512;i++)
//need to fill outer ring of tx els with same delay update as corresponding rx el, need rx to tx map
    detxC[i]=0;//bdl - write all Tx delays as zero for now
    detxCrep[i]=0;//bdl
}
for(i=640; i<768;i++)
    //detxC[i]=round(detx[i-512-128]*1.6);
    detxC[i]=round(detx2[i-512]*1.6);
}

next_dead_index0=0;
next_dead0= dead_chan_xd0[next_dead_index0];
next_dead_index1=0;
next_dead1=dead_chan_xd1[next_dead_index1];
//first fill deads with zeros so both delay update vectors have length 128;
if(side==0){
    k=0;
    for(i=0; i<128; i++)
        if(i==(next_dead0-1)){ // -1 for matlab indexing
            if(next_dead_index0<num_dead0){
//printf("%i\n",next_dead0);
                detx_full[i] =0;
                x_full[i]=0;
                next_dead_index0++;
                next_dead0=dead_chan_xd0[next_dead_index0];
            }
        }
}
else{
    detx_full[i]=detx[k];
    x_full[i]=x[k];
    k++;
}
}
outV("detx0_narrow_full.bin",detx_full,128);
outV("x1_narrow_full.bin",x_full,128);
}
else{
    k=0;
    for(i=0; i<128; i++){ // -1 for matlab indexing
        if(i==(next_dead1-1)){ // -1 for matlab indexing
            if(next_dead_index1<num_dead1){
                //printf("%i\n",next_dead1);
                detx2_full[i]=0;
                x2_full[i]=0;
                next_dead_index1++;
                next_dead1=dead_chan_xd1[next_dead_index1];
            }
        } else{
            detx2_full[i]=detx2[k];
            x2_full[i]=x2[k];
            k++;
        }
    }
    outV("detx1_narrow_full.bin",detx2_full,128);
    outV("x2_narrow_full.bin",x2_full,128);
} /*
//end 2nd time through
//next repackaging these delays to send back to scanner
/*
for(i=0; i<128; i++){
    detxCrep[512+m1_map[i]-1]=detxC[i+512];
}
for(i=0; i<128; i++){
    detxCrep[512+m2_map[i]-1]=detxC[i+640];
}
inVi("geo_map.bin",geo_map,256);//index is Tx chan #, value is Rx chan #
for(i=1; i<=256;i++)//all outer ring transmitters are located in first 256 transmit
channels
    detxCrep[i]=detxCrep[512+geo_map[i-1]-1];
}*/

//outVc("dataIn.dat",detxCrep,768);//bdl 04/17/11 this will overwrite original dataIn.dat
unless change file name for 1 of them
return 0;
} /*double *** allocate_d_3d( int row_dim, int col_dim, int dep_dim)
{
    /* allocate arrays as one contiguous piece of memory */
    /*
        double *** result;
        int i,j;
        result = (double ***)malloc(row_dim*sizeof(double **));
        result[0] = (double **)malloc(row_dim*col_dim*sizeof(double*));
        for(i=1; i<row_dim; i++)
            result[i] = result[i-1]+col_dim;
        result[0][0] = (double *)malloc(row_dim*col_dim*dep_dim*sizeof(double));
        for(i=1; i<row_dim; i++)
            result[i][0] = result[i-1][0]+dep_dim*col_dim;
        for(i=0; i<row_dim; i++)
            for(j=1; j<col_dim; j++)
                result[i][j] = result[i][j-1]+dep_dim;
    return result;
}
*/

char round(double roundin)
{
    if( (absD( absD(roundin) - absD(floor(roundin)))>0.5)

return ((char) ceil(roundin));
else
return ((char) floor(roundin));
}

int roundi(double roundin)
{
    if( (absD(absD(roundin) - absD(floor(roundin))))>0.5)
        return ((int) ceil(roundin));
else
    return ((int) floor(roundinn));
}

double absD( double absin)
{
    if(absin>0)
        return absin;
else
    return (-1*absin);
}

double ** allocate_d_2d( int row_dim, int col_dim)
{

    /* allocate arrays as one contiguous piece of memory */

double **result;
int i;
result = (double **)malloc(row_dim*sizeof(double *));
result[0] = (double *)malloc(row_dim*col_dim*sizeof(double));
for(i=1; i<row_dim; i++)
    result[i] = result[i-1]+col_dim;
return result;
}
short ** allocate_s_2d( int row_dim, int col_dim)
{

    /* allocate arrays as one contiguous piece of memory */

    short **result;
    int i;
    result = (short **)malloc(row_dim*sizeof(short *));
    result[0] = (short *)malloc(row_dim*col_dim*sizeof(short));
    for(i=1; i<row_dim; i++)
        result[i] = result[i-1]+col_dim;
    return result;
}

int highest_peak( double * vec, int len )
{
    int center, index, i;
    double max;
    index=0;
    max=0;
    center=(len-1)/2;
    for (i=0;i<len;i++)
    {
        if(vec[i]>max)
        {
            index=i;
            max=vec[i];
        }
    }
    return index;
}

int nearest_peak( double * vec, int len, int win )
{
    int center, index, i;
    double max;
    index=0;
    max=0;
    center=(len-1)/2;
    for (i=center-(win-1)/2;i<center+(win-1)/2;i++)
    {
        if(vec[i]>max)
        {
index=i;
    max=vec[i];
}
}
return index;
}
void sparse_mult (int *I, int *J, double *V, double *b, double *x, int m, int n)
{
    /* sparse matrix multiplication
     * I : rows
     * J : cols
     * V : values
     * b : vector being multiplied
     * x : answer
     * n : length of sparse matrix vectors
     * m : size of system
     */

    int i;

    for(i=0;i<n;i++) x[i]=0;
    for(i=0;i<m;i++){
        x[I[i]] += b[J[i]]*V[i];
    }
}

Volumetrics update code (receives updates from computer and applies them to appropriate channel within the appropriate image lines):

VM_Msg.cpp – changes to this file to allow for delay updates by patch. Code which undoes the delay updates was omitted. It is essentially the same as this with all delay updates multiplied by -1.
RC VMachine::reset_sector_tilts() { //Modified by NMI to swap transmit delays from one
dataset to another

FILE *errf;
u32 memAddr1;

u32 block; //data is split into sixteen blocks
u32 mode; //four modes: bmode, CF, PW, Mmode
u32 multi=0x1; //This value is a boolean for single (0) or multi (1) b scanning
//u32 xducer_to =0x0; //delays for first xducer
u32 xducer_from=0x1; // delays for second xducer (acq. data set)

u32 dat1;
u32 dat_orig;

u32 odd=0;//bdl for storing upper 2 bytes of dat1 (odd elements)
u32 even=0;//bdl for storing lower 2 bytes of dat1 (even elements)
double clk1=0;
double clk2=0;

int elnum2;

short line;
u32 el16;

/*RX DELAY VARIABLES */

short element;
short row=0;
int del=0;
u32 *olddelays;
u32 *newdelays;
u32 flag=0;//bdl bool for whether or not this delay needs to be written
unsigned short *olddelaysParsed;
unsigned short *newdelaysParsed;
u32 *newdelaysPairs;
unsigned short *whichchannels;

signed short *delays;
short fz;
short mb;
short chip;
short startbit;
short nfirst;
short nsecond;
u32 delayLSWbottom;
u32 delayMSWbottom;
u32 delayLSWtop;
u32 delayMSWtop;
u32 delayLSW;
u32 delayMSW;
u32 baseaddr;

unsigned short rowdef[32]={3,5,11,19,21,27,29,
  1,7,9,15,23,25,31,
  0,6,8,14,22,24,30,
  2,4,10,12,18,20,26,28};

/* ALLOCATE MEMORY */

olddelays=(u32 *)malloc(sizeof(u32)*9);
newdelays=(u32 *)malloc(sizeof(u32)*9);
olddelaysParsed=(unsigned short *)malloc(sizeof(unsigned short)*32);
newdelaysParsed=(unsigned short *)malloc(sizeof(unsigned short)*32);
newdelaysPairs=(u32 *)malloc(sizeof(u32)*16);
delays=(signed short *)malloc(sizeof(signed short)*8);

if (cmd_state_table[RESET_SECTOR_TILTS_CMD] == DISABLE) {
    return RC_OK;
}

stop();

clk1=clock();

/* RECEIVE DELAYS */
system("udp2winC.exe");
clk2=clock();

/* READ DELAYS */

inV("dataOut.dat",delaysIn,768);

//open error file
if((errf=fopen("TXRX.err","wt"))==NULL)
{
    printf("error: could not open error log");
}

//bdl - inserted Nik's bug fix 7/31/08, which replaces entire Change Transmit Delays section
/* CHANGE TRANSMIT DELAYS */

for (elnum2=0x0;elnum2<0x100;elnum2++)//bdl - have to figure out which elements to adjust for which lines
{
    // all tx els for which adjustment data is available are in first 256 of geo
    // only need to do first 128*2=first 256 tx els
    // for these, check existing delay to see if turned off before updating
    
    signed int delay1=(signed int)delaysIn[elnum2*2];
    signed int delay2=(signed int)delaysIn[elnum2*2+1];
    if(delay2<0)//check
    {
        if(delay1<0)//check
        {
            delay1=abs(delay1);
            delay2=abs(delay2);
            block=elnum2>>4;
            el16=(elnum2%16);
            for(mode=0x0;mode<0x4;mode++) //five transmit modes
            {
                if(elnum2<0x100)//bdl el 0-255
                {
                    for(line=0x0;line<0x100;line++)//bdl all lines
                        ...
if( line==36 || line==37 || line==38
  || line==39 || line==40 || line==41 || line==42 || line==43
  || line==52 || line==53 ||
  line==54 || line==55 || line==56 || line==57 || line==58 || line==59
  || line==68 || line==69 ||
  line==70 || line==71 || line==72 || line==73 || line==74 || line==75
  || line==84 || line==85 ||
  line==86 || line==87 || line==88 || line==89 || line==90 || line==91) // bdl xducer A&B, non-interleave, central patch
{
  memAddr1=0x08000000 + (multi*0x4000 + block*(0x80000) +
  mode*(0x6*0x4000)+line*0x40+el16*0x4);
  HAL_read_bus_BE(memAddr1,&dat1);
  if( ((dat1 & 0x1fff0000) !=
    0x1fff0000) ) // bdl
  (delay2<<16);/
  flag=1;
  } 
  if( ((dat1 & 0x00001fff) !=
    0x00001fff) ) // bdl
  (delay1);
  flag=1;
  } 
  if(flag==1){
    HAL_write_bus_BE(memAddr1,dat1);
    flag =0;
  }
}
}
for(line=0x0;line<0x100;line++) // bdl all lines
if( line==36 || line==37 || line==38
  || line==39 || line==40 || line==41 || line==42 || line==43
  || line==52 || line==53 ||
  line==54 || line==55 || line==56 || line==57 || line==58 || line==59
  || line==68 || line==69 ||
  line==70 || line==71 || line==72 || line==73 || line==74 || line==75
  || line==84 || line==85 ||
  line==86 || line==87 || line==88 || line==89 || line==90 || line==91) // bdl
non-interleave, central patch

```
memAddr1 = 0x08000000 + (xducer_from * 0x8000) + (multi*0x4000 +
block*(0x80000) + mode*(0x6*0x4000)+line*0x40+el16*0x4);
HAL_read_bus_BE(memAddr1,&dat1);
```

if( ((dat1 & 0x1fff0000) != 0x1fff0000) ){ //bdl
    delay2 = delay2 << 16;
    flag = 1;
}
if( ((dat1 & 0x00001fff) != 0x00001fff) ){ //bdl
    delay1 = delay1;
    flag = 1;
}
if(flag==1)
{
    HAL_write_bus_BE(memAddr1,dat1);
    flag = 0;
}
```
}
```
else{ //delay1>0
    delay2 = abs(delay2);
    block = elnum2 >> 4;
    el16 = (elnum2 % 16);
    for(mode=0x0;mode<0x4;mode++) //five transmit modes
    {
        if(elnum2<0x100){ //bdl
            for(line=0x0;line<0x100;line++) //bdl all
```
if (line==36 || line==37 || line==38
    || line==39 || line==40 || line==41 || line==42 || line==43
    || line==52 || line==53 ||
line==54 || line==55 || line==56 || line==57 || line==58 || line==59
    || line==68 || line==69 ||
line==70 || line==71 || line==72 || line==73 || line==74 || line==75
    || line==84 || line==85 ||
line==86 || line==87 || line==88 || line==89 || line==90 || line==91) //bdl xducer A&B,
non interleave, central patch
{
    memAddr1=0x08000000 + (multi*0x4000 + block*(0x80000) +
mode*(0x6*0x4000)+line*0x40+el16*0x4);
    HAL_read_bus_BE(memAddr1,&dat1);
    if( ((dat1 & 0x1fff0000) !=
0x1fff0000) )//bdl
        dat1 -=
(delay2<<16);//check
        flag=1;
    } 
    if( ((dat1 & 0x00001fff) !=
0x00001fff) )//bdl
    dat1 +=
(delay1);//check
    flag=1;
    } 
    if(flag=1){
        HAL_write_bus_BE(memAddr1,dat1);
        flag =0;
    }
}
}
} 
for(line=0x0;line<0x100;line++) //bdl
all lines 
    if (line==36 || line==37 ||
line==38 || line==39 || line==40 || line==41 || line==42 || line==43
    || line==52 || line==53 ||
line==54 || line==55 || line==56 || line==57 || line==58 || line==59

232
memAddr1 = 0x08000000 + (xducer_from * 0x8000) + (multi * 0x4000 + block * (0x8000) + mode * (0x6 * 0x4000) + line * 0x40 + el16 * 0x4);
HAL_read_bus_BE(memAddr1, &dat1);
if (((dat1 & 0x1fff0000) != 0x1fff0000)) {//bdl
dat1 -= (delay2 << 16);
flag = 1;
}
if (((dat1 & 0x00001fff) != 0x00001fff)) {//bdl
dat1 += (delay1);
flag = 1;
}
if (flag == 1) {
HAL_write_bus_BE(memAddr1, dat1);
flag = 0;
}
else {//delay2>0
if (delay1 < 0)
{
delay1 = abs(delay1);
block = elnum2 >> 4;
el16 = (elnum2 % 16);
for (mode = 0x0; mode < 0x4; mode++) //five transmit modes
{
if(elnum2<0x100){//bdl
  for(line=0x0;line<0x100;line++)//bdl lower
    lines
      if( line==36 || line==37 || line==38 ||
         line==39 || line==40 || line==41 || line==42 || line==43
             || line==52 || line==53 ||
         line==54 || line==55 || line==56 || line==57 || line==58 || line==59
             || line==68 || line==69 ||
         line==70 || line==71 || line==72 || line==73 || line==74 || line==75
             || line==84 || line==85 ||
         line==86 || line==87 || line==88 || line==89 || line==90 || line==91)//bdl xducer A&B, non interleave, central patch
          {
            memAddr1=0x08000000 + (multi*0x4000 + block*(0x80000) +
                mode*(0x6*0x4000)+line*0x40+el16*0x4);
            HAL_read_bus_BE(memAddr1,&dat1); if( ((dat1 & 0x1fff0000) !=
          (0x1fff0000) )//bdl
              dat1 +=
          (delay2<<16); flag=1;
          } if( ((dat1 & 0x00001fff) !=
          0x00001fff )//bdl
              dat1 -=
          (delay1); flag=1;
          } if(flag==1){
            HAL_write_bus_BE(memAddr1,dat1); flag =0;
          }
      }
  }
  for(line=0x0;line<0x100;line++)//bdl acq.data set
    if( line==36 || line==37 || line==38 ||
        line==39 || line==40 || line==41 || line==42 || line==43
    }
line==52 || line==53 || line==54 || line==55 || line==56 || line==57 || line==58 || line==59

|| line==68 || line==69 ||
line==70 || line==71 || line==72 || line==73 || line==74 || line==75 || line==76

|| line==84 || line==85 || line==86 || line==87 || line==88 || line==89 || line==90 || line==91
//bdl xducer A&B, non interleave, central patch
{
    memAddr1=0x08000000 + (xducer_from * 0x8000) + (multi*0x4000 + block*(0x80000) + mode*(0x6*0x4000)+line*0x40+el16*0x4);
    HAL_read_bus_BE(memAddr1,&dat1);
    if( ((dat1 & 0x1fff0000) != 0x1fff0000) ) {//bdl
        dat1 += (delay2<<16);
        flag=1;
    } //bdl
    if( ((dat1 & 0x00001fff) != 0x00001fff) ) {//bdl
        dat1 -= delay1;
        flag=1;
    } //bdl
    if(flag==1){
        HAL_write_bus_BE(memAddr1,dat1);
        flag =0;
    } //bdl
}
} //bdl
else //delay1>0
{
    block=elnum2>>4;
    el16=(elnum2%16);
    for(mode=0x0;mode<0x4;mode++) //five transmit modes
    {
}
if(elnum2<0x100){//bdl
for(line=0x0;line<0x100;line++){//bdl lower
lines
if( line==36 || line==37 || line==38
   || line==39 || line==40 || line==41 || line==42 || line==43
   || line==52 || line==53 || line==54 || line==55 || line==56 || line==57 || line==58 || line==59
   || line==68 || line==69 || line==70 || line==71 || line==72 || line==73 || line==74 || line==75
   || line==84 || line==85 || line==86 || line==87 || line==88 || line==89 || line==90 || line==91)//bdl xducer A&B, non interleave, central patch
{
    memAddr1=0x08000000 + (multi*0x4000 + block*(0x80000) +
    mode*(0x6*0x4000)+line*0x40+el16*0x4);

    HAL_read_bus_BE(memAddr1,&dat1);
    if( ((dat1 & 0x1fff0000) != 0x1fff0000) )//bdl
        (delay2<<16);//check
        dat1 +=
    (delay1);//check
    if(flag==1)
        HAL_write_bus_BE(memAddr1,dat1);
        flag =0;
    if(flag==1){

    }
}
}
for(line=0x0;line<0x100;line++)//bdl
acq.data set
if( line==36 || line==37 || line==38
   || line==39 || line==40 || line==41 || line==42 || line==43
   || line==39 || line==40 || line==41 || line==42 || line==43

236
non interleaved, central patch

\{
    \text{memAddr1}=0x08000000+(\text{ducer_from} \times 0x8000) + (\text{multi} \times 0x4000 + \text{block} \times (0x8000) + \text{mode} \times (0x6 \times 0x4000) + \text{line} \times 0x40 + \text{el16} \times 0x4); \}
\text{HAL\_read\_bus\_BE(memAddr1,&dat1);}
    \text{if}((\text{dat1} & 0x1fff0000) != 0x1fff0000) { //bdl
        \text{dat1} += (\text{delay2} \ll 16);
        \text{flag}=1;
    }
    \text{if}((\text{dat1} & 0x00001fff) != 0x00001fff) { //bdl
        \text{dat1} += \text{delay1};
        \text{flag}=1;
    }
    \text{if(flag==1)}{
        \text{HAL\_write\_bus\_BE(memAddr1,dat1);}
    }
}

/* CHANGE RECEIVE DELAYS */
\text{for(block=0;block<8;block++)} //bdl controls which image lines get updated, i.e. 0 to 3 is 1st half (left half of lower sector)
    \text{for(row=0;row<16;row++)}{
whichchannels = &rowdef[(row%4)*8];
for(del=0;del<8;del++){
    delays[del]=(signed short)delaysIn[whichchannels[del] +
    block*32 +512];
}
for (line=0;line<256;line++){//bdl lower lines 4/01/11 - if
    want to do for acq data set (slot 3), also do for lines 768 to 1024
    if( line==36 || line==37 || line==38 || line==39 ||
        line==40 || line==41 || line==42 || line==43
        || line==52 || line==53 ||
        line==54 || line==55 || line==56 || line==57 || line==58 || line==59
        || line==68 || line==69 ||
        line==70 || line==71 || line==72 || line==73 || line==74 || line==75
        || line==84 || line==85 ||
        line==86 || line==87 || line==88 || line==89 || line==90 || line==91){//bdl xducer A&B,
        non interleave, central patch
        for (fz=0;fz<10;fz++){//bdl
            baseaddr=block*BLOCKSIZE +
            MULTIBOFFSET +fz*64 + line*640 + row*4;
            for(mb=0;mb<9;mb++){//bdl
                HAL_read_bus_BE((baseaddr +
                mb*MBOFFSET),&olddelays[mb]);
                newdelays[mb]= 0;
            }  // end mb loop
        } // end fz loop
        /* PARSE THE DELAYS */
        for(chip=0;chip<8;chip++){
            startbit=2*chip;
            nfirst=16-startbit;
            nsecond=18-nfirst;
            delayLSWbottom =
            (olddelays[chip]%(1<<16))>>startbit;
            delayMSWbottom =
            olddelays[chip]>>(startbit+16);
            delayLSWtop = olddelays[chip+1] %
            (1<<nsecond);
            delayMSWtop =
            (olddelays[chip+1]>>16) % (1<<nsecond);
            delayLSW = ( delayLSWtop << (16-
            (chip*2)) ) + delayLSWbottom;
delayMSW = (delayMSWtop << (16 - (chip * 2))) + delayMSWbottom;

delayLSW % (1 << 9);

delayLSW >>= 9;

delayMSW % (1 << 9);

delayMSW >>= 9;

/* MODIFY THE DELAYS */

only modify if element is turned on

if(delays[chip] != 511) {// bdl 1/30/11
    if(delays[chip] < 0) {
        if(olddelaysParsed[chip*4] != 511 && delays[chip] != 0) {
        } else {
            newdelaysParsed[chip*4] = olddelaysParsed[chip*4];
        }
    } if(olddelaysParsed[chip*4+1] != 511 && delays[chip] != 0) {
        newdelaysParsed[chip*4+1] = olddelaysParsed[chip*4+1] - abs(delays[chip]);
    } else {
        newdelaysParsed[chip*4+1] = olddelaysParsed[chip*4+1];
    }
} if(olddelaysParsed[chip*4+2] != 511 && delays[chip] != 0) {
    newdelaysParsed[chip*4+2] = olddelaysParsed[chip*4+2] - abs(delays[chip]);
} else {
    newdelaysParsed[chip*4+2] = olddelaysParsed[chip*4+2];
}
newdelaysParsed[chip*4+2] = olddelaysParsed[chip*4+2];
    }
    if(olddelaysParsed[chip*4+3] != 511 && delays[chip] != 0){
newdelaysParsed[chip*4+3] = olddelaysParsed[chip*4+3] - abs(delays[chip]);
    } else{
newdelaysParsed[chip*4+3] = olddelaysParsed[chip*4+3];
    }

newdelaysParsed[chip*4+3] = olddelaysParsed[chip*4+3];
}

newdelaysPairs[chip*2] = newdelaysParsed[chip*4] + (newdelaysParsed[chip*4+1] << 9); // LSW part
newdelaysPairs[chip*2+1] = newdelaysParsed[chip*4+2] + (newdelaysParsed[chip*4+3] << 9); // MSW part
/* REPACK THE DELAYS */

//%bottom part starts at chip*2, ends at 16; top part starts at chip*2+16, ends at 32
newdelays[chip] =
(newdelaysPairs[chip*2] << startbit) % (1 << 16) +
(((newdelaysPairs[chip*2+1] << startbit) % (1 << 16)) << 16) + newdelays[chip]; //%both parts start at 16-chip*2, shift top up by 16
newdelays[chip+1] =
((newdelaysPairs[chip*2]) >> (16-chip*2)) + ((newdelaysPairs[chip*2+1]) >> (16-chip*2)) << 16;
}
else{
if(olddelaysParsed[chip*4] != 511 && delays[chip] != 0){
} else{
newdelaysParsed[chip*4] = olddelaysParsed[chip*4];
}
if(olddelaysParsed[chip*4+1] != 511 && delays[chip] != 0){
newdelaysParsed[chip*4+1] = olddelaysParsed[chip*4+1] + abs(delays[chip]);
} else{
newdelaysParsed[chip*4+1] = olddelaysParsed[chip*4+1];
}
if(olddelaysParsed[chip*4+2] != 511 && delays[chip] != 0){
newdelaysParsed[chip*4+2] = olddelaysParsed[chip*4+2] + abs(delays[chip]);
} else{
newdelaysParsed[chip*4+2] = olddelaysParsed[chip*4+2];
}
if(olddelaysParsed[chip*4+3] != 511 && delays[chip] != 0){
newdelaysParsed[chip*4+3] = olddelaysParsed[chip*4+3] + abs(delays[chip]);
} else{
newdelaysParsed[chip*4+3] = olddelaysParsed[chip*4+3];
}
newdelaysPairs[chip*2]=newdelaysParsed[chip*4] + (newdelaysParsed[chip*4+1]<<9); // LSW part
newdelaysPairs[chip*2+1]=newdelaysParsed[chip*4+2] + (newdelaysParsed[chip*4+3]<<9); // MSW part
/* REPACK THE DELAYS */

//%bottom part starts
at chip*2, ends at 16; top part starts at chip*2+16, ends at 32
newdelays[chip] =
(newdelaysPairs[chip*2]<<startbit)%((1<<16) +
(((newdelaysPairs[chip*2+1]<<startbit)%(1<<16)<<16) + newdelays[chip];

16-chip*2, shift top up by 16

newdelays[chip+1] =
((newdelaysPairs[chip*2])>>(16-chip*2)) + ((newdelaysPairs[chip*2+1]>>(16-
chip*2))<<16);

set chip_write flag

} // end chip loop

/* PUT DELAYS BACK */
//if(chip_write==1){
   for(mb=0;mb<9;mb++){
      HAL_write_bus_BE((baseaddr+ mb*MBOFFSET),newdelays[mb]);

   } // end mb loop

} // end fz loop

} // end if line== || line==
} // end line if
} // end line loop

free(olddelays);
free(newdelays);
free(olddelaysParsed);
free(newdelaysParsed);
free(newdelaysPairs);
free(delays);

clk1=(clock()-clk1)/CLOCKS_PER_SEC;
clk2=(clock()-clk2)/CLOCKS_PER_SEC;
fprintf(errf,"%e\n%e",clk1,clk2);
//fclose(fp);
return RC_OK;

}
Appendix G: Model 1 scanner simulation code

M1_dop_sim.m was used to simulate Doppler processing of model 1 to investigate its low frequency performance.

```matlab
%M1_dop_sim.m
% simulate Model 1 Doppler processing
clear;
f0=1.2e6;
fs=40e6;
c=1540;
cycles=8;
v=0.75;%m/s
Tprf=1/4e3;
shift_sam=(v*Tprf/c*fs);
sam_per_cyc=fs/f0;
shift_phase=shift_sam/sam_per_cyc*2*pi;
A=0.01;%amplitude - set A based on output range of AD9050
ensemble_length=7;

t=1/fs:1/fs:ncycles*1/1e6;

s1=A*sin(2*pi*t*f0);
s2=A*sin(2*pi*t*f0+shift_phase);
s3=A*sin(2*pi*t*f0+2*shift_phase);
s4=A*sin(2*pi*t*f0+3*shift_phase);
for(k=1:ensemble_length)
    s_pack(k,:)=A*sin(2*pi*t*f0+(k-1)*shift_phase);
end;

%begin halfband filter sim (TMC 2242B)
%coeffs are from TMC 2242BB data sheet

h_hb=[-0.000875473 0.0 0.001390457 0.0 -0.002265930 0.0 0.003501892 0.0 -0.006366836 0.0
0.007621765 0.0 -0.01071167 0.0 0.01483154 0.0 -0.02018738 0.0 0.02796364 0.0 -0.03949928
0.0 0.05937767 0.0 -0.1036148 0.0 0.3180542 0.5009766];
h_hb(29:55)=fliplr(h_hb(1:27));
```
% input data to this filter should be 10 bits at 40MHz
% decimate
for(k=1:ensemble_length)
    s_pack_re(k,:) = conv(resample(s_pack(k,:),1,2),h hb);
end;
s_pack = s_pack_re;
fs = fs/2;

% throw out all but 16 bits
% all data is < 1 (all bits are to right of decimal)
for(k=1:ensemble_length)
    for(i=1:size(s_pack,2))
        s_pack16(k,i,:) = ’0000000000000000’;
        s_pack16(k,i,:) = ’0000000000000000’;
        if(s_pack(k,i)<0)
            % first take magnitude
            res = dec2bin(bitcmp(abs(round(s_pack(k,i)*2^15)),15));
            s_pack16(k,i,2+15-length(res):16) = res;
            s_pack16(k,i,1) = dec2bin(1); % set sign bit to neg.
        else
            s_pack16(k,i,2:16) = dec2bin(round(s_pack(k,i)*2^15),15);
            s_pack16(k,i,1) = dec2bin(0); % pos. sign bit
        end;
    end;
end;
% convert back to dec
for(k=1:ensemble_length)
    for(i=1:size(s_pack,2))
        if(bin2dec(s_pack16(k,i,1))==1)
            s_pack_out(k,i) = -1*(bitcmp(bin2dec(squeeze(s_pack16(k,i,2:16))))*2^15);
        else
            s_pack_out(k,i) = bin2dec(squeeze(s_pack16(k,i,2:16)))*2^-15;
        end;
    end;
end;
%end halfband filter sim

%Begin filter coeff handling
%change these next 2 lines to be desired filter
I_Coeff= [27 78 -269 187 36 62 -246 346 39 40 -191 270 35 19 -119 158
25 5 -46 31 10 0 17 -92 -4 5 60 -191 -17 16 80 -252 ];
Q_Coeff= [15 28 -48 0 5 -1 49 -134 -9 -16 122 -247 -24 -18 164 -321 -
35 -9 172 -345 -40 3 152 -319 -39 14 114 -250 -33 18 69 -153 ];
I = I_Coeff(1:32);
Q = Q_Coeff(1:32);

%reshape 32 coefficient vector into a 4 x 8 matrix
rows = 4;
columns = 8;
Imatrix = reshape(I,rows,columns);
Qmatrix = reshape(Q,rows,columns);

%find time domain impulse response (I_IR and Q_IR)

%rearrange coefficients because of the backwards ordering
%in the model 1 filter text file
for i=1:rows
    I_IR( (columns*(i-1)+1):columns*i ) = fliplr(Imatrix(i,:));
    Q_IR( (columns*(i-1)+1):columns*i ) = fliplr(Qmatrix(i,:));
end

%last coefficient is doubled due to odd tap filter
%it was originally halved
I_IR(32) = 2*I_IR(32);
Q_IR(32) = 2*Q_IR(32);

%create other side (symmetric) of filter time
domain response
I_sym = fliplr(I_IR(1:31));
Q_sym = -fliplr(Q_IR(1:31));

I_IR(33:63) = I_sym;
Q_IR(33:63) = Q_sym;
%end filter coeff handling

for(k=1:ensemble_length)
    s_packf(k,:)=conv(s_pack_out(k,:),I_IR);
    s_packfh(k,:)=conv(s_pack_out(k,:),Q_IR);
end;

%what is rel. phase b/w I, Q signals?
rel_phase=((length(xcov(s1f,s1fh,'coef'))+1)/2-
find(xcov(s1f,s1fh,'coef')==max(xcov(s1f,s1fh,'coef'))))/sam_per_cyc*2*360;

%remove transient

wf
s1fw=s1f-2*s2f+s3f;

s1fhw=s1fh-2*s2fh+s3fh;
s2fw=s2f-2*s3f+s4f;
s2fhw=s2fh-2*s3fh+s4fh;

%output stage
these should all be 19 bits - then make them 16 bits
first make then 19 bits with 1 sign bit, 9 bits to left of decimal, 9 bits
to right of decimal

%sign and bits to left of decimal
for(k=1:ensemble_length)
for(i=1:size(s_packf,2))
    s_packf_bin(k,i,:)=’0000000000000000000’;
    s_packfh_bin(k,i,:)=’0000000000000000000’;
    %s2f_bin(i,:)=’0000000000000000000’;
    if(s_packf(k,i)<0)
        %first take magnitude
        res=dec2bin(bitcmp(abs(round(s_packf(k,i))),9));
        s_packf_bin(k,i,2+9-length(res):10)=res;
        s_packf_bin(k,i,1)=dec2bin(1),%set sign bit to neg.
    else
        s_packf_bin(k,i,2:10)=dec2bin(round(s_packf(k,i)),9);
    endif
endfor
endfor
s_packf_bin(k,i,1)=dec2bin(0); %pos. sign bit
end;
if(s_packfh(k,i)<0)
    resh=dec2bin(bitcmp(abs(round(s_packfh(k,i))),9));
    s_packfh_bin(k,i,2+9-length(resh):10)=resh;
    s_packfh_bin(k,i,1)=dec2bin(1); %set sign bit to neg.
else
    s_packfh_bin(k,i,2:10)=dec2bin(round(s_packfh(k,i)),9);
    s_packfh_bin(k,i,1)=dec2bin(0); %pos. sign bit
end;

% bits to right of decimal
res_dec=dec2bin(abs(round(s_packf(k,i))-s_packf(k,i))*2^9);
res_dech=dec2bin(abs(round(s_packfh(k,i))-s_packfh(k,i))*2^9);
s_packf_bin(k,i,11+9-length(res_dec):19)=res_dec(1:length(res_dec));
s_packfh_bin(k,i,11+9-length(res_dech):19)=res_dech(1:length(res_dech));
end;
end;

% Now scale all these 19 bit numbers by dividing by 1, 2, 4, or 8
scale=1;
for(k=1:ensemble_length)
    s_packf_bin(k,:,2:end)=cirqhift(s_packf_bin(k,:,2:end),[0 log2(scale)]);
    s_packfh_bin(k,:,2:end)=cirqhift(s_packfh_bin(k,:,2:end),[0 log2(scale)]);
    if(scale==2)
        s_packf_bin(k,:,2)=dec2bin(0); % set highest (non-sign) bit to zero
        s_packfh_bin(k,:,2)=dec2bin(0); % set highest (non-sign) bit to zero
    end;
    if(scale==4)
        s_packf_bin(k,:,2)=dec2bin(0); % set highest 2 (non-sign) bits to zero
        s_packf_bin(k,:,3)=dec2bin(0); % set highest 2 (non-sign) bits to zero
        s_packfh_bin(k,:,2)=dec2bin(0); % set highest 2 (non-sign) bits to zero
        s_packfh_bin(k,:,3)=dec2bin(0); % set highest 2 (non-sign) bits to zero
    end;
    if(scale==8)

end;
s_packf_bin(k,:,2)=dec2bin(0);%set highest 3 (non-sign) bits to zero
s_packf_bin(k,:,3)=dec2bin(0);%set highest 3 (non-sign) bits to zero
s_packf_bin(k,:,4)=dec2bin(0);%set highest 3 (non-sign) bits to zero
s_packfh_bin(k,:,2)=dec2bin(0);%set highest 3 (non-sign) bits to zero
s_packfh_bin(k,:,3)=dec2bin(0);%set highest 3 (non-sign) bits to zero
s_packfh_bin(k,:,4)=dec2bin(0);%set highest 3 (non-sign) bits to zero
s2f_bin(:,2)=dec2bin(0);%set highest 3 (non-sign) bits to zero
s2f_bin(:,3)=dec2bin(0);%set highest 3 (non-sign) bits to zero
s2f_bin(:,4)=dec2bin(0);%set highest 3 (non-sign) bits to zero
end;
end;

%Now convert to 16 bits, clamp to indicate overflow
%let LSB be clamp bit according to Detector spec page 7
%At each iteration, shift out the MSB (not sign bit). If it is a 1,
%overflow has occurred.
clamp_count=0;
clamp_count2=0;
for(k=1:ensemble_length)
    for(i=1:size(s_packf,2))
        s_packf_bin16(k,i,:)=’0000000000000000’;
        s_packf_bin16(k,i,1)=s_packf_bin(k,i,1);%copy sign bit
        s_packf_bin16(k,i,2:16)=s_packf_bin(k,i,5:19);
        % convert neg. numbers to pos.
        if(s_packf_bin(k,i,1)==dec2bin(1))
            temp=dec2bin(bitcmp(bin2dec(squeeze(s_packf_bin(k,i,2:10))’),9));
            if(length(temp)==1)
                temp(2)=0;temp(3)=0;temp(4)=0;
            end;
            if(length(temp)==2)
                temp(3)=0;temp(4)=0;
            end;
            if(length(temp)==3)
                temp(4)=0;
            end;
            if(~isequal(bitand(bin2dec(temp),bin2dec(’111000000’)), 0))
                s_packf_bin16(k,i,2:16)=’1111111111111111’;%don't mess w/ sign bit
                clamp_count=clamp_count+1;
            end;
        else
            ...
%if any of 3 highest bits is 1, then clamp to max value
    temp2 = squeeze(s_packf_bin(k,i,2:9));
    if(~isequal(bitand(bin2dec(temp2),bin2dec('111000000')), 0))
        s_packf_bin16(k,i,2:16) = '1111111111111111';%don't mess w/ sign bit
        clamp_count = clamp_count + 1;
    end;
end;
end;

s_packfh_bin16(k,i,:) = '0000000000000000';
s_packfh_bin16(k,i,1) = s_packfh_bin(k,i,1);%copy sign bit
s_packfh_bin16(k,i,2:16) = s_packfh_bin(k,i,5:19);
% convert neg. numbers to pos.
if(s_packfh_bin(k,i,1) == dec2bin(1))
    temph = dec2bin(bitcmp(bin2dec(squeeze(s_packfh_bin(k,i,2:10))),9));
    if(length(temph) == 1)
        temph(2) = 0; temp(3) = 0; temp(4) = 0;
    end;
    if(length(temp) == 2)
        temph(3) = 0; temp(4) = 0;
    end;
    if(length(temp) == 3)
        temph(4) = 0;
    end;
    if(~isequal(bitand(bin2dec(temph),bin2dec('111000000')), 0))
        s_packfh_bin16(k,i,2:16) = '1111111111111111';%don't mess w/ sign bit
        clamp_count2 = clamp_count2 + 1;
    end;
else
    %if any of 3 highest bits is 1, then clamp to max value
    temp2h = squeeze(s_packfh_bin(k,i,2:9));
    if(~isequal(bitand(bin2dec(temp2h),bin2dec('111000000')), 0))
        s_packfh_bin16(k,i,2:16) = '1111111111111111';%don't mess w/ sign bit
        clamp_count2 = clamp_count2 + 1;
    end;
end;
end;
end;
%get signal back (in dec)
for(k=1:ensemble_length)
    for(i=1:size(s_packf,2))
if(bin2dec(s_packf_bin16(k,i,1))==1)
    s_packf_outout(k,i)=-
    1*(bitcmp(bin2dec(squeeze(s_packf_bin16(k,i,2:10)')),9)+bin2dec(squeeze(s_packf_bin16( k,i,11:16))))*2^-9);
else
    s_packf_outout(k,i)=bin2dec(squeeze(s_packf_bin16(k,i,2:10)'))+bin2dec(squeeze(s_packf_ 
    _bin16(k,i,11:16)))*2^-9;
end;
if(bin2dec(s_packfh_bin16(k,i,1))==1)
    s_packfh_outout(k,i)=-
    1*(bitcmp(bin2dec(squeeze(s_packfh_bin16(k,i,2:10)')),9)+bin2dec(squeeze(s_packfh_bin 
    16(k,i,11:16))))*2^-9);
else
    s_packfh_outout(k,i)=bin2dec(squeeze(s_packfh_bin16(k,i,2:10)'))+bin2dec(squeeze(s_pa 
    ckfh_bin16(k,i,11:16)))*2^-9;
end;
end;
end;
end;
end;

%get signal back from 19 bit version(in dec) (debug only)
for(k=1:ensemble_length)
    for(i=1:size(s_packf,2))
        if(bin2dec(s_packf_bin(k,i,1))==1)
            s_packf_out19(k,i)=-
            1*(bitcmp(bin2dec(squeeze(s_packf_bin(k,i,2:10)')),9)+bin2dec(squeeze(s_packf_bin(k,i,1 
            1:19))))*2^-9);
        else
            s_packf_out19(k,i)=bin2dec(squeeze(s_packf_bin(k,i,2:10)'))+bin2dec(squeeze(s_packf_b 
            in(k,i,11:19)))*2^-9;
        end;
    end;
end;

251
if(bin2dec(s_packfh_bin(k,i,1))==1)
    s_packfh_out19(k,i)=-1*(bitcmp(bin2dec(squeeze(s_packfh_bin(k,i,2:10))'),9)+bin2dec(squeeze(s_packfh_bin(k,i,11:19))')*2^-9);
else
    s_packfh_out19(k,i)=bin2dec(squeeze(s_packfh_bin(k,i,2:10))')+bin2dec(squeeze(s_packfh_bin(k,i,11:19))')*2^-9;
end;
end;
end;

%velocity estimator
I=s_packf_outout;
Q=s_packfh_outout;
for(n=1:ensemble_length-1)
    v1d(n)=c/2*1/(2*pi*Tprf)*atan(sum(sum(Q(n,:))*sum(I(n+1,:))-sum(I(n,:))*sum(Q(n+1,:)))/sum(sum(I(n,:)*sum(I(n+1,:))+sum(Q(n,:)*sum(Q(n+1,:)))))/(fs);
end;
Appendix H: Dual volume display code

This appendix contains the code modifications required for simultaneous real-time display and control of two 3D volumes using the Volumetrics Model 1 scanner.

Each file changed is listed followed by the changes made in that particular file.

**VM_Msg.cpp**

```cpp
RC VMachine::force_record() {
    VMParams *params; //farfield added

    leave_current_state();
    freeze_screen();

    params->test_id=4; //farfield /// Change this to change the default ID number
    xc->xd_test_id=4; //farfield /// Ditto Search file for 30 and 40 and replace all with 31 & 41 //bdl now 50 and 60

    if (begin_record() != RC_OK) {
        stop();
    }

    vs->reset_stored_volumes();
    vs->configure_view(VIEW_VOLUME_STANDARD); //bdl try this to go to dual display
    set_command_state();
    return RC_STATE_CHANGED;
}
```

```cpp
RC VMachine::force_record_id() {//begin add farfield
    VMParams *params; //farfield

    leave_current_state();
    freeze_screen();

    xc->attached_xd_id=0; //farfield
    params->test_id=50; //farfield //bdl
    xc->xd_test_id=50; //farfield //bdl

    if (begin_record() != RC_OK) {
```
stop();
}

vs->reset_stored_volumes();
vs->configure_view(VIEW_VOLUME_STANDARD2); // bdl try this to go to standard single display
    set_command_state();
return RC_STATE_CHANGED;
} // end add farfield

RC VMachine::record_id() { // begin add farfield
if (cmd_state_table[RECORD_CMD] == DISABLE) {
    return RC_OK;
}
return force_record_id();
} // end add farfield*/

RC VMachine::record_id40() { // begin add farfield
if (cmd_state_table[RECORD_CMD] == DISABLE) {
    return RC_OK;
}
return force_record_id40();
} // end add farfield*/

RC VMachine::record_id2(int change_id) { // changed to take integer input nmi
if (cmd_state_table[RECORD_CMD] == DISABLE) {
    return RC_OK;
}
return force_record_id2(change_id); // nmi
} // end add farfield

RC VMachine::force_record_id40() { // begin add farfield nmi
VMParams *params; // farfield

leave_current_state();
freeze_screen();

// printf("%i\t%i\n",orig_id,orig_xducer);
if (begin_record() != RC_OK) {
    stop();
}

vs->reset_stored_volumes();
vs->configure_view(VIEW_VOLUME_STANDARD2);//bdl try this to go to standard single displayset_command_state();
//orig_xducer=!orig_xducer;
return RC_STATE_CHANGED;
}//end add farfield
RC VMachine::force_record_id2(int change_id2) {//begin add farfield
VMParams *params; //farfield
leave_current_state();
freeze_screen();

xc->attached_xd_id=0; //farfield
params->test_id=change_id2;//farfield bdl
xc->xd_test_id=change_id2;//farfield bdl

if (begin_record() != RC_OK) {
    stop();
}

vs->reset_stored_volumes();
set_command_state();
return RC_STATE_CHANGED;
}//end add farfield
C VMachine::select_pw_mode() {
    if (cmd_state_table[SELECT_PW_MODE_CMD] == DISABLE) {
        return RC_OK;
    }
leave_current_state();
freeze_screen();
rec_init.set_mode(PW_MODE);
if (vs->get_data_type() == VOLUME) {
    if(vs->xc->xd_test_id==4){//bdl - dual case
        rec_init.set_view_config(VIEW_VOLUME_SCROLL);//bdl
dual case
    }
    else{
        rec_init.set_view_config(VIEW_VOLUME_SCROLL2);//bdl - was
        VIEW_VOLUME_SCROLL
            //bdl dual case
    }
} else { 
    rec_init.set_view_config(VIEW_SINGLE_SCROLL);
}
last_rec_vs = NONE;

if (resume() != RC_OK) {
    stop();
}
set_command_state();
// Set to first PW trackball state (after set_control_state is done)
tb->reset_trackball_to_mode(PW_MODE);
return RC_STATE_CHANGED;
} RC VMachine::volume_oblique_select() {
    if (cmd_state_table[VOLUME_OBLIQUE_SELECT_CMD] == DISABLE) {
        return RC_OK;
    }
    leave_current_state();
    freeze_screen();

    if (vs->get_data_type() == VOLUME) {
        switch (vs->get_mode()) {
            case ECHO_MODE:
            case CF_MODE:
                vs->configure_view(VIEW_VOLUME_OBLIQUE_STANDARD2);//bdl was
                break;
            default:
                break;
        }
    }
case M_MODE:
case PW_MODE:
    vs->configure_view(VIEW_VOLUME_SCROLL2);//bd1 was
    VIEW_VOLUME_SCROLL
    break;
default:
    return RC_ERROR;
}
}
else if (sc->get_last_scan_state() == RECORDING) {
    rec_init.set_type(VOLUME);
    switch (vs->get_mode()) {
        case ECHO_MODE:
        case CF_MODE:
            rec_init.set_view_config(VIEW_VOLUME_STANDARD2);//bd1 was
            VIEW_VOLUME_OBLIQUE
            break;
        case M_MODE:
        case PW_MODE:
            rec_init.set_view_config(VIEW_VOLUME_SCROLL2);//bd1 was
            VIEW_VOLUME_SCROLL
            break;
        default:
            return RC_ERROR;
    }
    last_rec_vs = NONE;
}
if (resume() != RC_OK) {
    stop();
}
set_command_state();
return RC_STATE_CHANGED;
}
RC VMachine::change240(){      //nmi
    if(orig_xducer)
        {
            orig_xducer=!orig_xducer;
            record_id40();
        }
}
{       
    orig_xducer=!orig_xducer;
    record_id2(4);//bdl was record_id2(3); was record_id2(9); needs to
    be changed to equal IDPROM value
}

return RC_OK;
}
RC VMachine::reset_parallel_tilts() //edited by farfield
   if (cmd_state_table[RESET_PARALLEL_TILTS_CMD] == DISABLE) {
      return RC_OK;
   }
//if (vs == NULL) return RC_ERROR;
//vs->reset_parallel_tilts();
record_id2(60);
return RC_STATE_CHANGED;//RC_OK;
}

RC VMachine::reset_sector_tilts() //edited by farfield
   if (cmd_state_table[RESET_SECTOR_TILTS_CMD] == DISABLE) {
      return RC_OK;
   }
   // RC ret_code = RC_OK;
   // if (vs == NULL) return RC_ERROR;

   // Stop scanning in pw or m to change aiming
   // ModeType mode = vs->get_mode();
   // if ((mode == PW_MODE) || (mode == M_MODE)) {
   //   leave_current_state();
   //   freeze_screen();
   //   record_id();
   //   return RC_STATE_CHANGED;
   // }
   RC VMachine::shift_parallel_x_tilt(int increment) {
      if (cmd_state_table[SHIFT_PARALLEL_X_TILT_CMD] == DISABLE) {
         return RC_OK;
      }
      RC ret_code = RC_OK;//bdl
if (vs == NULL) return RC_ERROR;
    // Stop scanning in pw or m to change aiming  //bdl had to add this here
ModeType mode = vs->get_mode();
if ((mode == PW_MODE) || (mode == M_MODE)) {
    leave_current_state();
    freeze_screen();
}
// Convert increment to tilt?
if(vs->xc->xd_test_id==4){//bdl - dual case
    vs->shift_az_tilt1((double) increment);
    if ((mode == PW_MODE) || (mode == M_MODE)) {//bdl add
        if (resume() != RC_OK) {
            stop();
            ret_code = RC_ERROR;
        }
    }
}
else{//bdl single case
    vs->shift_parallel_x_tilt((double) increment);
}
return ret_code;//bdl
//return RC_OK;

RC VMachine::shift_parallel_y_tilt(int increment) {
    if (cmd_state_table[SHIFT_PARALLEL_Y_TILT_CMD]==DISABLE) {
        return RC_OK;
    }
    RC ret_code = RC_OK;//bdl
    if (vs == NULL) return RC_ERROR;
        // Stop scanning in pw or m to change aiming  //bdl had to add this here
ModeType mode = vs->get_mode();
if ((mode == PW_MODE) || (mode == M_MODE)) {
    leave_current_state();
    freeze_screen();
}
// Convert increment to tilt?
if(vs->xc->xd_test_id==4){//bdl - dual case
    vs->shift_el_tilt1((double) increment);
    if ((mode == PW_MODE) || (mode == M_MODE)) {//bdl add

if (resume() != RC_OK) {
    stop();
    ret_code = RC_ERROR;
}

else {//bdl single case
    vs->shift_parallel_y_tilt((double) increment);
}

return ret_code;//bdl

//return RC_OK;

for (i=0; i<MAX_DEPTHS; i++) {
    start_depth[i] = 40.0;//bdl was 20.0
    end_depth[i] = 130.0; // Check end depth will limit this for some depths
}

cf_velocity_scale = 8191;//bdl was 8189 01/03/11
vol_theta_span = 60.0;//bdl was 20.0
sb_theta_span = 60.0;//bdl was 20.0
phi_span = 120.0;//bdl was 20.0
cf_xmit_power = 3;//bdl was 3
packet_size_index = 0;//bdl was 1
wall_filter_index = 3;//bdl was 0 (0 = Max, 1 = High, 2 = Med, 3 = Low)
echo_min = 0xff;//bdl changed from 0x30 at farfield recommendation
cf_baseline_shift = -50;//bdl changed from 0 at farfield recommendation

CFScan.cpp
if (el_lines>1) {
    if(vs->xc->xd_test_id==4){//dual case
        bdl added conditional
    }
    double phi_lower = 0 - (vs->get_cf_phi_span()/2.00);//bdl was vs->get_cf_phi_aim() instead of 0 - (vs->get_cf_phi_span()/2.00);
    double phi_lower = vs->get_cf_phi_aim() - (vs->get_cf_phi_span()/2.00);//bdl use this line for displaying old data only
    *phi_lower_line = angle_to_line (phi_lower, el_lines, el_angle);
    if(*phi_lower_line<0){*phi_lower_line=0;}//bdl
    lines_per_degree = (el_lines - 1) / el_angle;
    int phi_span_lines = ROUND_INT(vs->get_cf_phi_span() * lines_per_degree);
2/8/10
    //double phi_lower = 0 - (vs->get_cf_phi_span()/2.00);//bdl was
    vs->get_cf_phi_aim() instead of 0 - (vs->get_cf_phi_span()/2.00);
    double phi_lower = vs->get_cf_phi_aim() - (vs->get_cf_phi_span()/2.00);//bdl use this line for displaying old data only
    *phi_lower_line = angle_to_line (phi_lower, el_lines, el_angle);
    if(*phi_lower_line<0){*phi_lower_line=0;}//bdl
    lines_per_degree = (el_lines - 1) / el_angle;
    int phi_span_lines = ROUND_INT(vs->get_cf_phi_span() * lines_per_degree);
/*phi_upper_line = 8+angle_to_line((vs->get_cf_phi_aim()-9),el_lines,el_angle);/bdl. Was: *phi_lower_line + phi_span_lines - 1;8+angle_to_line((vs->get_cf_phi_aim()-30),el_lines,el_angle);

*phi_upper_line = *phi_lower_line + phi_span_lines - 1;/bdl use this line for displaying old data only

//printf("cf_phi_aim: %i\n",vs->get_cf_phi_aim());//bdl
if(*phi_upper_line>15){*phi_upper_line=15;}//bdl
//printf("*phi_lower_line: %i\n", *phi_lower_line);//bdl
//printf("*phi_upper_line: %i\n", *phi_upper_line);//bdl
}
else{//bdl single case

double phi_lower = vs->get_cf_phi_aim();//bdl was instead of 0
0 - (vs->get Cf_phi_span()/2.00)
;
//double phi_lower = vs->get Cf_phi_aim() - (vs-
>get Cf_phi_span()/2.00);/bdl use this line for displaying old data only

*phi_lower_line = angle_to_line (phi_lower, el_lines, el_angle);
//if(*phi_lower_line<0){*phi_lower_line=0;}//bdl
lines_per_degree = (el_lines - 1) / el_angle;
int phi_span_lines = ROUND_INT(vs->get Cf_phi_span() *
lines_per_degree);

//phi_upper_line = 8+angle_to_line((vs->get Cf_phi_aim()-
9),el_lines,el_angle);//bdl Was: *phi_lower_line + phi_span_lines - 1;8+angle_to_line((vs-
>get Cf_phi_aim()-30),el_lines,el_angle);

*phi_upper_line = *phi_lower_line + phi_span_lines - 1;/bdl use this line for displaying old data only

//printf("cf_phi_aim: %i\n",vs->get Cf_phi_aim());//bdl
if(*phi_upper_line>15){*phi_upper_line=15;}//bdl
//printf("*phi_lower_line: %i\n", *phi_lower_line);//bdl
//printf("*phi_upper_line: %i\n", *phi_upper_line);//bdl
}
}
DualPlay.cpp
//bdl commented this out to allow entering dual playback with any 2 vols
/* if ((vs_left==NULL) || (vs_right==NULL)) {
    return FALSE;
}
if (!(vs_left->get_data_available()) || !(vs_right->get_data_available())) {
    return FALSE;
}
if ((vs_left->get_mode() != ECHO_MODE) || (vs_right->get_mode() != ECHO_MODE)) {
    return FALSE;
}
if ((vs_left->get_data_type() != VOLUME) || (vs_right->get_data_type() != VOLUME)) {
    return FALSE;
}
if (vs_left->get_echo_theta_upper() != vs_right->get_echo_theta_upper()) {
    return FALSE;
}
if (vs_left->get_echo_theta_lower() != vs_right->get_echo_theta_lower()) {
    return FALSE;
}
if (vs_left->get_echo_phi_upper() != vs_right->get_echo_phi_upper()) {
    return FALSE;
}
if (vs_left->get_echo_phi_lower() != vs_right->get_echo_phi_lower()) {
    return FALSE;
}
if (vs_left->get_echo_samples() != vs_right->get_echo_samples()) {
    return FALSE;
}
if (vs_left->get_echo_min_depth_samp() != vs_right->get_echo_min_depth_samp()) {
    return FALSE;
}
if (vs_left->get_sample_size() != vs_right->get_sample_size()) {
    return FALSE;
} /*
// Everything must be matched.
return TRUE;
}
RC VMachine::dual_reset_parallel_tilts(DualVSSide dual_side) {
    DualPairType dual_pair = DUAL_PLAY_SET[current_dual_set];
    EchoVolSeg *vs_side;
    if (dual_side==LEFT_DUAL_VS) {
        vs_side = v_seg[dual_pair.left_vs];
        if (vs_side == NULL) return RC_ERROR;
        vs_side->reset_parallel_tilts();
        if (vs_side->get_echo_samples() != vs_side->get_echo_min_depth_samp()) {
            return FALSE;
        }
        if (vs_side->get_sample_size() != vs_side->get_sample_size()) {
            return FALSE;
        }
    } /*
else if (dual_side==RIGHT_DUAL_VS) {
    vs_side = v_seg[dual_pair.right_vs];
    if (vs_side == NULL) return RC_ERROR;
    vs_side->reset_parallel_tilts();
}
else {
    vs_side = v_seg[dual_pair.left_vs];
    if (vs_side == NULL) return RC_ERROR;
    vs_side->reset_parallel_tilts();
    vs_side = v_seg[dual_pair.right_vs];
    if (vs_side == NULL) return RC_ERROR;
    vs_side->reset_parallel_tilts();
}
//record_id2(4);//bdl
return RC_STATE_CHANGED;//RC_OK
}

void VMachine::set_dual_cmd_state() {
    ScanStateType state = sc->get_scan_state();
    switch (state) {
    case RECORDING:
        //disable_dual_cmd_state();//bdl
        break;
    …
    PWScan.cpp – analogous changes are also made to CFScan.cpp and EScan.cpp
#include "Viewport.hpp"//bdl

    pw_theta1 = pw_init->pw_theta;//bdl
    pw_phi1 = pw_init->pw_phi;//bdl

    calc_tilt_aiming(&pw_theta, &pw_phi, &pw_sum);
    calc_tilt_aiming1(&pw_theta1, &pw_phi1, &pw_sum);//bdl
    if (vs->get_no_probe_vs()==FALSE) {
        pw_send_rt_line_table();
        pw_send_rt_line_table1();//bdl
    …
    PWScanner::set_pw_scanner_init(PWVSInit *pw_init) {
    …
    pw_init->pw_theta1 = pw_theta1;//bdl
}
pw_init->pw_phi1 = pw_phi1;//bdl
...
}
...
void PWSScanner::pw_unfreeze() {
...
set_elevation_aiming(); // Also gets azimuth
    set_elevation_aiming1();//bdl added
...
}
...
void PWSScanner::set_azimuth_aiming1() {//bdl
    if (!freeze) {
        calc_tilt_aiming1(&pw_theta, &pw_phi, &pw_sum);//bdl
        pw_send_rt_line_table();//bdl
    }
}
void PWSScanner::set_elevation_aiming1() {//bdl
    if (!freeze) {
        calc_tilt_aiming1(&pw_theta, &pw_phi, &pw_sum);//bdl
        pw_send_rt_line_table();//bdl
    }
}
...
void PWSScanner::pw_build_rt_header(PWRTTableInitType *pw_table) {
    pw_table->pw_aim_line1 = pw_phi1 * az_lines + pw_theta1;//bdl
...
}
...
void PWSScanner::pw_send_rt_line_table() {
    if (vs->get_no_probe_vs()==TRUE) {return;}

    PWRTTableInitType pw_table;
    pw_build_rt_header(&pw_table);
    HAL_send_pw_rt_header(&pw_table);
    HAL_send_pw_rt_line_table(&pw_table,1);//bdl
}

void PWSScanner::pw_send_rt_line_table1() {//bdl
    if (vs->get_no_probe_vs()==TRUE) {return;}

    }
PWRTTableInitType pw_table1;
pw_build_rt_header(&pw_table1);
HAL_send_pw_rt_header(&pw_table1);
HAL_send_pw_rt_line_table(&pw_table1, 0);//bdl
}

PW_VM.cpp – analogous changes are also made to CF_VM.cpp and
Echo_VM.cpp
PWVSInit::PWVSInit() {
...
pw_theta1 = 0;//bdl
pw_phi1 = 0;//bdl
....
}

Viewport.cpp – requires analogous changes to CFVport.cpp and PWVport.cpp
static const Color AZ_PARALLEL_MARKER_COLOR = VID_COLOR_TRANSPARENT;//bdl
static const Color EL_PARALLEL_MARKER_COLOR = VID_COLOR_TRANSPARENT;//bdl
...
void EchoSliceViewport::set_thickness(int thickness) {
   assert_void(thickness >= 0 && thickness < 100);
   // only do this for parallel vps
   /*if (type == VP_PARALLEL_FLAT || type == VP_PARALLEL_OBLIQUE) { //bdl commented out
      assert_void(echo_slice != NULL);
   }
   */
   if (type == VP_AZIMUTH || type == VP_ELEVATION) {//farfield: added if statements for VP_AZIMUTH VP_ELEVATION for thick b scan
      assert_void(echo_slice != NULL);
      echo_slice->dw = thickness;
      params_changed = TRUE;
   }
   echo_slice->dw = thickness;
   params_changed = TRUE;
}
void EchoSliceViewport::init_slice(SliceInfo *slice, SliceDataType slice_type)
{
  if (vs->get_data_type() == SINGLE_B) {
    ...
  } else {
    slice->c_scan_format = 1;
    slice->theta_offset = VP_SC_VOLUME_THETA_OFFSET;
    slice->theta_max = VP_SC_VOLUME_THETA_MAX;

    if((type== VP_ELEVATION1)){
      slice->phi_max = VP_SC_VOLUME_PHI_MAX;
      slice->phi_offset = VP_SC_VOLUME_PHI_OFFSET;//bd
    }
    else{
      if((type== VP_ELEVATION)){
        slice->phi_max = VP_SC_VOLUME_PHI_MAX;
        slice->phi_offset = VP_SC_VOLUME_PHI_OFFSET;//bd
      }
      else{
        slice->phi_max = VP_SC_VOLUME_PHI_MAX;//bd
        slice->phi_offset = VP_SC_VOLUME_PHI_OFFSET;
      }
    }
    slice->angle_scale_theta = VP_SC_VOLUME_THETA_ANGLE_SCALE;
    //slice->angle_scale_phi = VP_SC_VOLUME_PHI_ANGLE_SCALE;
    if(vs->xc->xd_test_id==4}//begin bd add
      slice->angle_scale_phi =
      VP_SC_VOLUME_PHI_ANGLE_SCALE_DUAL;
    }
    else{
      slice->angle_scale_phi =
      VP_SC_VOLUME_PHI_ANGLE_SCALE_SINGLE;
    //end bd add
    }
  }
}

EchoSliceViewport(vm_ptr, vs_ptr, view_ptr, vp_id, vp_info)
if (type == VP_AZIMUTH1) {//bdl
    data_region.clip_polygon.color = PYRAMID_AZIMUTH_COLOR;
    sector_int_color = ELEVATION_MARKER_COLOR;
    parallel_int_color = AZ_PARALLEL_MARKER_COLOR;
    pyramid_prim_color = PYRAMID_AZIMUTH_COLOR;
    init_tilt = view->get_az_tilt1();
}
if (type == VP_ELEVATION1) {//bdl
    data_region.clip_polygon.color = PYRAMID_ELEVATION_COLOR;
    sector_int_color = AZIMUTH_MARKER_COLOR;
    parallel_int_color = EL_PARALLEL_MARKER_COLOR;
    pyramid_prim_color = PYRAMID_ELEVATION_COLOR;
    init_tilt = view->get_el_tilt1();
}

void EchoSectorViewport::compute_Mt_v(Matrix M, double tilt)
{
    if (type == VP_AZIMUTH || type == VP_AZIMUTH1) {//bdl
        // U column
        M[X][U] = s;
        M[Y][U] = 0;
        M[Z][U] = 0;

        // V column
        M[X][V] = 0;
        M[Y][V] = s * sin_t;
        M[Z][V] = s * cos_t;

        // W column
        M[X][W] = 0;
        M[Y][W] = -s * cos_t;
        M[Z][W] = s * sin_t;

        // this assumes z offset has been updated if nec
        M[X][3] = -s * width/2;
        M[Y][3] = sin_t * sector_z_offset;
        M[Z][3] = cos_t * sector_z_offset;
    }
}
else {
    //assert_void(type == VP_ELEVATION);

    // U column
    M[X][U] = 0;
    M[Y][U] = s;
    M[Z][U] = 0;

    // V column
    M[X][V] = s * sin_t;
    M[Y][V] = 0;
    M[Z][V] = s * cos_t;

    // W column
    M[X][W] = s * cos_t;
    M[Y][W] = 0;
    M[Z][W] = -s * sin_t;

    // this assumes z offset has been updated if nec
    M[X][3] = sin_t * sector_z_offset;
    M[Y][3] = -s * width/2;
    M[Z][3] = cos_t * sector_z_offset;
}
}

... void EchoSectorViewport::recompute()
{
    if (type == VP_AZIMUTH) {
        update_geometry(view->get_az_tilt());
    }
    if (type == VP_AZIMUTH1) {//bdl
        update_geometry(view->get_az_tilt1());
    }
    if (type == VP_ELEVATION1) {//bdl
        update_geometry(view->get_el_tilt1());
    }
    else {
        update_geometry(view->get_el_tilt());
    }
}
void EchoSectorViewport::calculate_clip_boundary()
{
    double rx_angle_2; // bdl
    // bdl - I added this whole if-else sequence, previously:
    // rx_angle_2 = vs->get_rx_echo_theta_span()/2.0;
    if ((type == VP_ELEVATION || type == VP_ELEVATION1) && vs->xc-
        >xd_test_id==4) {// bdl only for lower vps and dual data set
        rx_angle_2 = vs->get_rx_echo_theta_span()*2/2.0; // bdl no longer mult. this by 2
    } else {
        rx_angle_2 = vs->get_rx_echo_theta_span()/2.0; // bdl DON'T mult. other vp's by 2 to
        // fix range markers in 60 x 120 scan
    }
}

... // bottom right corner of sector
    if (type == VP_ELEVATION || type == VP_ELEVATION1) {
        vert[CLIP_BOTTOM_RIGHT][U] = u_a + r_total * sin(RADIANS*vs-
            >get_rx_echo_theta_span()-0.2) * cos_tilt / scale; // bdl
        vert[CLIP_BOTTOM_RIGHT][V] = v_a + r_total * cos(RADIANS*vs-
            >get_rx_echo_theta_span()) / scale; // bdl
    } else {
        vert[CLIP_BOTTOM_RIGHT][U] = u_a + r_total * aim_u / scale;
        vert[CLIP_BOTTOM_RIGHT][V] = v_a + r_total * aim_v / scale;
    }

// bottom left: u coord is bottom right coord mirrored about Z axis;
// v coord is the same as bottom_right
    // bdl - edit 2/5/10 - conditional on single vs. dual
    if (type == VP_ELEVATION1 && vs->xc->xd_test_id==4) {// bdl
        vert[CLIP_BOTTOM_LEFT][U] = 0; // bdl
        vert[CLIP_BOTTOM_LEFT][V] = (v_a + r_total/scale)/2.0-20; // bdl
    } else {
        vert[CLIP_BOTTOM_LEFT][U] = u_a + (u_a - vert[CLIP_BOTTOM_RIGHT][U]);
        vert[CLIP_BOTTOM_LEFT][V] = vert[CLIP_BOTTOM_RIGHT][V];
    }
/*vert[CLIP_BOTTOM_LEFT][U] = u_a + (u_a - vert[CLIP_BOTTOM_RIGHT][U]);
   vert[CLIP_BOTTOM_LEFT][V] = vert[CLIP_BOTTOM_RIGHT][V];*/

// top left: see notes for bottom_left
vert[CLIP_TOP_LEFT][U] = u_a + (u_a - vert[CLIP_TOP_RIGHT][U]);
vert[CLIP_TOP_LEFT][V] = vert[CLIP_TOP_RIGHT][V];

// interior bottom edge points: 0 is on right, 1 in middle, 2 on left
right_of_center_bottom_point. cos_tilt corrects for sector
// tilt as above.
if(type==VP_AZIMUTH || type==VP_AZIMUTH1){
   vert[CLIP_BOTTOM0][U] = u_a + r_total*sin(RADIANS*rx_angle_2*0.5)*cos_tilt/scale;
   vert[CLIP_BOTTOM0][V] = v_a + r_total*cos(RADIANS*rx_angle_2*0.5)/scale+10;//bdl
}
else{
   vert[CLIP_BOTTOM0][U] = u_a + r_total*sin(RADIANS*rx_angle_2*0.5)*cos_tilt/scale;
   vert[CLIP_BOTTOM0][V] = v_a + r_total*cos(RADIANS*rx_angle_2*0.5)/scale;
}

// center bottom point
if(type==VP_AZIMUTH || type==VP_AZIMUTH1){
   vert[CLIP_BOTTOM1][U] = u_a;
   vert[CLIP_BOTTOM1][V] = v_a + r_total/scale+10; //bdl
}
else{
   vert[CLIP_BOTTOM1][U] = u_a;
   vert[CLIP_BOTTOM1][V] = v_a + r_total/scale;
}

// left-of-center bottom point: mirror bottom0
vert[CLIP_BOTTOM2][U] = u_a + (u_a - vert[CLIP_BOTTOM0][U]);
vert[CLIP_BOTTOM2][V] = vert[CLIP_BOTTOM0][V];

// set up the draw/clear region to go from the left and right edges of the
// sector to the top and bottom of the vp. this keeps the clear from
// clobbering the color ramp
bounding_box = region_ptr->bounding_box;
if(type==VP_AZIMUTH || type==VP_ELEVATION){//bdl rt. side vp's
    int u0 = origin[U];
    int v0 = origin[V];

    // top and bottom
    bounding_box[0].v = 0;
    bounding_box[1].v = v0 + height - 1;//bdl

    // right side: this is the same as bottom_right except that we divide out
    // the tilt correction factor so that we always clear the maximum width
    // (otherwise, we get pixels left behind when the tilt changes quickly)
    // + 4 is a fudge factor to make sure we can clear the orientation marker
    if(type == VP_AZIMUTH){
        bounding_box[1].u = u0 +
            ROUND_USHORT(u_a + r_total * (aim_u/cos_tilt) / scale)-20;//bdl
    }
    else{
        bounding_box[1].u = u0 +
            ROUND_USHORT(u_a + r_total * (aim_u/cos_tilt) / scale); //bdl
    }

    // left side: u coord is just right-side u coord mirrored about Z. fudge
    // factor is mirrored too. do the mirroring in screen coords since bbox.u
    // is already in screen coords.
    // (9/99) the -8 is yet another fudge factor now that tilts can go right
    // up to the edge.
    bounding_box[0].u =
        0;//bdl
}
else{//bdl left side vp's
    int u0 = origin[U];
    int v0 = origin[V];

    // top and bottom
    bounding_box[0].v = 0;
bounding_box[1].v = v0 + height - 1;//bdl

// right side: this is the same as bottom_right except that we divide out
// the tilt correction factor so that we always clear the maximum width
// (otherwise, we get pixels left behind when the tilt changes quickly)
// + 4 is a fudge factor to make sure we can clear the orientation marker
if (type == VP_AZIMUTH1) {
    bounding_box[1].u = u0 +
        ROUND_USHORT(u_a + r_total * (aim_u/cos_tilt) / scale) - 20;//bdl
} else {
    bounding_box[1].u = u0 +
        ROUND_USHORT(u_a + r_total * (aim_u/cos_tilt) / scale);//bdl
}
// left side: u coord is just right-side u coord mirrored about Z. fudge
// factor is mirrored too. do the mirroring in screen coords since bbox.u
// is already in screen coords.
// (9/99) the -8 is yet another fudge factor now that tilts can go right
// up to the edge.
bounding_box[0].u =
    0;//bdl

}
void EchoSectorViewport::calculate_intersections()
{
if (type == VP_AZIMUTH) {
    other_sector_ptr = view->get_elevation_vp();
}
else if (type == VP_AZIMUTH1) {//bdl
    other_sector_ptr = view->get_elevation_vp1();
} else if (type == VP_ELEVATION1) {//bdl
    other_sector_ptr = view->get_azimuth_vp1();
} else {
    other_sector_ptr = view->get_azimuth_vp();
}
...
void EchoSectorViewport::create_pyramid_primitives()
{
  if (type == VP_AZIMUTH1) { //bdl
    other_sector_vp = view->get_elevation_vp1();
  }
  if (type == VP_ELEVATION1) { //bdl
    other_sector_vp = view->get_azimuth_vp1();
  }
}

... void EchoSectorViewport::redraw(boolean full_clear)
{
  clear(full_clear);

  // Draw probe orientation marker if this is the az sector
  if ((type == VP_AZIMUTH || type == VP_AZIMUTH1) && (! vs->is_true_axis_on())) { //bdl
    draw_orientation_marker(rm.u0 + ORIENTATION_MARKER_OFFSET,
                           rm.v0, VID_COLOR_WHITE);
  }
  // this uses range marks stored in rm_ptr
  if (! vs->is_true_axis_on()) {
    draw_range_marks(VID_COLOR_WHITE);
    if (type == VP_ELEVATION) {
      //draw_blank(VID_COLOR_BLACK); //bdl - this works but causes very slow refresh when tilt is changed
      //draw_box2(VID_COLOR_BLACK); //bdl
    }
    if (type == VP_ELEVATION1 && vs->xc->xd_test_id == 4) {
      draw_box1(VID_COLOR_BLACK); //bdl
    }
  }
  if (vs->get_data_type() == VOLUME) {
    draw_intersections();
  }
}
if (data_region.axis_color != VID_COLOR_TRANSPARENT) {
    draw_axis_line();
}

if (highlight_enabled) {
    //draw_highlight(HIGHLIGHT_COLOR);
}

//draw_clip_boundary(VID_COLOR_GREEN);

void EchoSectorViewport::draw_blank(Color color) //begin bdl
{
    int i;
    int u;
    int v;
    short du_i, dv_i;

    //for (i = 0; i < rm.count*8; i++) {
    for(i=0; i<400;i++){
        if(type==VP_ELEVATION){
            HAL_draw_filled_GO_box(rm.u0-180, rm.v0-20+1*i, 180-i*6/10, 1,
            color,GO_BUFFER0);//bdl
        }
    }
    /*
    // compute rounded increment value for each i
    du_i = ROUND_USHORT(i * rm.du/8);
    dv_i = ROUND_USHORT(i * rm.dv/8);
    
    HAL_write_GO_pixel(rm.u0+4 + du_i+u + RANGE_MARK_U_OFFSET,
    rm.v0 + dv_i-v, color, USER_PIXEL, GO_BUFFER0); 
    HAL_write_GO_pixel(rm.u0+4 + du_i+u + 1 + RANGE_MARK_U_OFFSET,
    rm.v0 + dv_i-v, color, USER_PIXEL, GO_BUFFER0); 
    HAL_write_GO_pixel(rm.u0+4 + du_i+u + RANGE_MARK_U_OFFSET,
    rm.v0 + dv_i-v + 1, color, USER_PIXEL, GO_BUFFER0); 
    HAL_write_GO_pixel(rm.u0+4 + du_i+u + 1 + RANGE_MARK_U_OFFSET,
    rm.v0 + dv_i-v + 1, color, USER_PIXEL, GO_BUFFER0);*/
    }
void EchoSectorViewport::draw_box1(Color color) //begin bdl
{  
   int  i;
   int u;
   int v;
   short du_i, dv_i;

   //for (i = 0; i < rm.count*8; i++) {
      if(type==VP_ELEVATION1 && vs->xc->xd_test_id==4) {//bdl only for lower vps and only on dual data set
         HAL_draw_filled_GO_box(rm.u0-15, rm.v0-10, rm.u0+8, rm.v0+30, color,GO_BUFFER0);//bdl
      }
   }

   ....
   void EchoSectorViewport::draw_intersections()
   {
      ....

      // now do sector-sector intersections. compute this even if sector_markers are turned off since this is used for the bottom marker as well.
      // azimuth or elevation?
      if (type == VP_AZIMUTH) {
         other_sector_ptr = view->get_elevation_vp();
      }
      else if (type == VP_AZIMUTH1) {//bdl  
         other_sector_ptr = view->get_elevation_vp1();
      }
      else if (type == VP_ELEVATION1) { //bdl  
         other_sector_ptr = view->get_azimuth_vp1();
      }

      ....
   }

   void EchoSectorViewport::get_apex(double *u, double *v)
   {
      // assume apex is always centered in sectors
      if(vs->xc->xd_test_id==4) {
         *u = (double) origin[X] + ((double) width)/2.0; //bdl change this to translate rm's and clip bounds
      }
   
   

   275
else{
    *u = (double) origin[X] + ((double) width)/2.0;//bdl change this to translate rm's and clip bounds
    }
    *v = (double) origin[Y] - sector_z_offset/view->get_scale();
}
// same as above, but answer is in vp coords
void EchoSectorViewport::get_apex_in_vp(double *u, double *v)
{
    // assume apex is always centered in sectors
    *u = ((double) width)/2.0;//bdl change this to translate rm's and clip bound
    *v = (double) (-sector_z_offset/view->get_scale());
}
...
void EchoParallelViewport::compute_Mt_v(Matrix M, double tilt)//bdl
{
    int border_width;
    double sin_t, cos_t;

double s = view->get_scale();

border_width = view->get_border_width();

double theta_span = vs->get_rx_echo_theta_span();

// skin offset in samples
double r_so = vs->get_echo_min_depth_samp();

sector_z_offset = (r_so * cos(RADIANS*theta_span/2.0)) -
    (s * border_width);

sin_t = sin(RADIANS*tilt);
cos_t = cos(RADIANS*tilt);

if (type == VP_PARALLEL_FLAT) {
    // U column
    M[X][U] = s;
    M[Y][U] = 0;
    M[Z][U] = 0;
// V column
M[X][V] = 0;
M[Y][V] = s * sin_t;
M[Z][V] = s * cos_t;

// W column
M[X][W] = 0;
M[Y][W] = -s * cos_t;
M[Z][W] = s * sin_t;

// this assumes z offset has been updated if nec
M[X][3] = -s * width/2;
M[Y][3] = sin_t * sector_z_offset;
M[Z][3] = cos_t * sector_z_offset;
}
else {
    //assert_void(type == VP_ELEVATION);

    // U column
    M[X][U] = 0;
    M[Y][U] = s;
    M[Z][U] = 0;

    // V column
    M[X][V] = s * sin_t;
    M[Y][V] = 0;
    M[Z][V] = s * cos_t;

    // W column
    M[X][W] = s * cos_t;
    M[Y][W] = 0;
    M[Z][W] = -s * sin_t;

    // this assumes z offset has been updated if nec
    M[X][3] = sin_t * sector_z_offset;
    M[Y][3] = -s * width/2;
    M[Z][3] = cos_t * sector_z_offset;
}
}
void EchoParallelViewport::set_z_c(double new_z_c)
{
    z_c = new_z_c;

    update_geometry(view->get_parallel_x_tilt(), view->get_parallel_y_tilt()); // bdl
}

void EchoParallelViewport::calculate_clip_boundary() // bdl pasted in from sector
calculate_clip_boundary()
{
...
// update geometry on depth or tilt change. does not recompute intersections
void EchoParallelViewport::update_geometry(double x_tilt, double y_tilt) // bdl modify
{
    /*compute_matrices(x_tilt);
     calculate_clip_boundary();
    */
    // end bdl version, original is below
    compute_matrices(x_tilt, y_tilt);

    calculate_clip_boundary(); // virtual
}

// recompute matrices, etc, after some external change
void EchoParallelViewport::recompute()
{
    // update_geometry(view->get_parallel_x_tilt()); // bdl
    update_geometry(view->get_parallel_x_tilt(), view->get_parallel_y_tilt()); // bdl
}

void EchoParallelViewport::get_apex_in_vp(double *u, double *v) // bdl
{
    // assume apex is always centered in sectors
    *u = ((double) width)/2.0;
    *v = (double) (-sector_z_offset/view->get_scale());
}

char *vp_type_to_string(ViewportType type)
{
    // this is derived from the defs in viewport.hpp; the order has to match
static char vp_string[][40] = {
    "VP_AZIMUTH",
    "VP_ELEVATION",
    "VP_PARALLEL_FLAT",
    "VP_AZIMUTH1"//bdl
    "VP_ELEVATION1"//bdl
    "VP_PARALLEL_OBLIQUE",
    "VP_SCROLL",
    "VP_ECG",

    // multi-slice or 3D data viewports
    "VP_VOLUME_RENDERING",    // for Viz board
    "VP_ASSEMBLED_VIEW",    // "
    "VP_WIRE_MESH",
    "VP_COLOR_MAP",
};

static char  bogus[] = "unknown";

if (type >= VP_MAX_TYPES) {
    return bogus;
}

return vp_string[type];

EVolSeg.cpp – Analogous changes are also made to CFVolSeg.cpp and PWVolSeg.cpp

void EchoVolSeg::shift_az_tilt(double increment) {
    // Increment sector tilts in 0.5 degree steps
    double tilt_change = increment * DEGREES_PER_TILT_SHIFT_INC;
    double new_tilt = az_tilt + tilt_change;
    if (mode == PW_MODE){
        if (vm->vs->xc->xd_test_id==4)//dual case bdl
            if ((new_tilt <= 0)//bdl was get_rx_echo_phi_lower() instead of 0
                && (az_tilt == 0)) {//bdl was get_rx_echo_phi_lower()
                return;
            }
        else if ((new_tilt >= get_rx_echo_phi_upper())
                && (az_tilt == get_rx_echo_phi_upper())) {
            return;
        }
return;

// If trying to go over limits, set to limits
else if (new_tilt < 0) {//bdl was get_rx_echo_phi_lower() instead of 0
    new_tilt = 0;//bdl was get_rx_echo_phi_lower() instead of 0
}
else if (new_tilt > get_rx_echo_phi_upper()) {
    new_tilt = get_rx_echo_phi_upper();
}

az_tilt = new_tilt;
set_azimuth_tilt(new_tilt);

//end dual case
dual case bdl
else if((new_tilt <= get_rx_echo_phi_lower())
    && (az_tilt == get_rx_echo_phi_lower())) {
    return;
}
else if ((new_tilt >= get_rx_echo_phi_upper())
    && (az_tilt == get_rx_echo_phi_upper())) {
    return;
}

// If trying to go over limits, set to limits
else if (new_tilt < get_rx_echo_phi_lower()) {
    new_tilt = get_rx_echo_phi_lower();
}
else if (new_tilt > get_rx_echo_phi_upper()) {
    new_tilt = get_rx_echo_phi_upper();
}

az_tilt = new_tilt;
set_azimuth_tilt(new_tilt);

//end single case bdl
else{
    if(vm->vs->xc->xd_test_id==4)//dual case
        // Do nothing if at limits and trying to go beyond them.
        if ((new_tilt <= 0)//bdl was get_rx_echo_phi_lower()
            && (az_tilt == 0)) {//bdl was get_rx_echo_phi_lower() instead of 0
            return;
        }
}
else if ((new_tilt >= get_rx_echo_phi_upper())
    && (az_tilt == get_rx_echo_phi_upper())) {  
    return;
}
// If trying to go over limits, set to limits
else if (new_tilt < 0) {//bdl was get_rx_echo_phi_lower()
    new_tilt = 0;//bdl was get_rx_echo_phi_lower()
    instead of 0
}
else if (new_tilt > get_rx_echo_phi_upper()) {
    new_tilt = get_rx_echo_phi_upper();
}
az_tilt = new_tilt;
set_azimuth_tilt(new_tilt);
}
else{//single case
     // Do nothing if at limits and trying to go beyond them.
    if ((new_tilt <= get_rx_echo_phi_lower())//bdl was
get_rx_echo_phi_lower() instead of 0
    && (az_tilt == get_rx_echo_phi_lower())) {//bdl
    was get_rx_echo_phi_lower() instead of 0
        return;
    }
else if ((new_tilt >= get_rx_echo_phi_upper())
    && (az_tilt == get_rx_echo_phi_upper())) {  
    return;
}
// If trying to go over limits, set to limits
else if (new_tilt < get_rx_echo_phi_lower()) {//bdl was
get_rx_echo_phi_lower() instead of 0
    new_tilt = get_rx_echo_phi_lower();//bdl was
get_rx_echo_phi_lower() instead of 0
    }
else if (new_tilt > get_rx_echo_phi_upper()) {
    new_tilt = get_rx_echo_phi_upper();
}
az_tilt = new_tilt;
set_azimuth_tilt(new_tilt);
void EchoVolSeg::shift_az_tilt1(double increment) {
    // Increment sector tilts in 0.5 degree steps
    double tilt_change = increment * DEGREES_PER_TILT_SHIFT_INC;
    double new_tilt = az_tilt1 + tilt_change;
    if(mode == PW_MODE) {
        if ((new_tilt <= get_rx_echo_phi_lower())) // bdl was instead of 0
            && (az_tilt1 == get_rx_echo_phi_lower()) // bdl was
            return;
    } else if ((new_tilt >= 0)) // bdl was get_rx_echo_phi_upper() instead of 0
        && (az_tilt1 == 0) // bdl was
        get_rx_echo_phi_upper() instead of 0
            return;
    } // If trying to go over limits, set to limits
    else if (new_tilt < get_rx_echo_phi_lower()) // bdl was
        instead of 0
        new_tilt = get_rx_echo_phi_lower(); // bdl was
        instead of 0
    } else if (new_tilt > 0) // bdl was get_rx_echo_phi_upper() instead of 0
        new_tilt = 0; // bdl was get_rx_echo_phi_upper()
        instead of 0
} az_tilt1 = new_tilt;
set_azimuth_tilt1(new_tilt);
else {
    // Do nothing if at limits and trying to go beyond them.
    if ((new_tilt <= get_rx_echo_phi_lower())
        && (az_tilt1 == get_rx_echo_phi_lower()) )
        return;
} else if ((new_tilt >= 0)) // bdl was get_rx_echo_phi_upper() instead of 0

& (az_tilt1 == 0)) // bdl was
get_rx_echo_phi_upper() instead of 0
    return;
}
// If trying to go over limits, set to limits
else if (new_tilt < get_rx_echo_phi_lower()) {
    new_tilt = get_rx_echo_phi_lower();
}
else if (new_tilt > 0) {// bdl was get_rx_echo_phi_upper() instead of 0
    new_tilt = 0;// bdl was get_rx_echo_phi_upper() instead of 0
}
az_tilt1 = new_tilt;// bdl
set_azimuth_tilt1(new_tilt);// bdl
}

void EchoVolSeg::shift_el_tilt1(double increment) {
    // Increment sector tilts in 0.5 degree steps
    double tilt_change = increment * DEGREES_PER_TILT_SHIFT_INC;
    double new_tilt = el_tilt1 + tilt_change;
    if(mode == PW_MODE){
        if ((new_tilt <= get_rx_echo_theta_lower())// bdl was instead of 0
            && (el_tilt1 == get_rx_echo_theta_lower())) {// bdl
            was ) instead of 0
            return;
        }
        else if ((new_tilt >= get_rx_echo_theta_upper())
            && (el_tilt1 ==
get_rx_echo_theta_upper())) {
            return;
        }
    // If trying to go over limits, set to limits
    else if (new_tilt < get_rx_echo_theta_lower()) // bdl was
    instead of 0
        new_tilt = get_rx_echo_theta_lower(); // bdl was
    instead of 0
    }
    else if (new_tilt > get_rx_echo_theta_upper()) {
        new_tilt = get_rx_echo_theta_upper();
    }
el_tilt1 = new_tilt;
set_elevation_tilt1(new_tilt);
}
else{
    // Do nothing if at limits and trying to go beyond them.
    if ((new_tilt <= get_rx_echo_theta_lower())
        && (el_tilt1 == get_rx_echo_theta_lower())) {
        return;
    }
    else if ((new_tilt >= get_rx_echo_theta_upper())
        && (el_tilt1 == get_rx_echo_theta_upper())) {//bdl was get_rx_echo_phi_upper() instead of 0
        return;
    }
    else if (new_tilt < get_rx_echo_theta_lower()) {
        new_tilt = get_rx_echo_theta_lower();
    }
    else if (new_tilt > get_rx_echo_theta_upper()) {//bdl was get_rx_echo_phi_upper() instead of 0
        new_tilt = get_rx_echo_theta_upper();//bdl was get_rx_echo_phi_upper() instead of 0
    }
    el_tilt1 = new_tilt;//bdl
    set_elevation_tilt1(new_tilt);//bdl
}

//begin bdl add
void EchoVolSeg::set_azimuth_tilt1(double az_angle) {
    az_tilt = az_angle;
    view->set_az_tilt1(az_angle);
}

void EchoVolSeg::set_elevation_tilt1(double el_angle) {
    el_tilt = el_angle;
    view->set_el_tilt1(el_angle);
}
//end bdl add
TBState.cpp
TBState::TBState(VMachine *vm_in) {
    double machine_tb_scale = 1;
    if (sys_viper) {
        machine_tb_scale = VIPER_TB_SCALE;
    }
    ...
    trk_echo_sector_tilt.ol_text = "Right Sector \"; //bdl
    trk_echo_sector_tilt.x_scale = ROUND_INT(machine_tb_scale * 3);
    trk_echo_sector_tilt.x_work = &VMachine::shift_el_tilt;
    trk_echo_sector_tilt.y_scale = ROUND_INT(machine_tb_scale * 3);
    trk_echo_sector_tilt.y_work = &VMachine::shift_az_tilt;
    trk_echo_sector_tilt.end_work = &VMachine::null_end_work;
    ...
    //begin bdl add
    trk_CF_steering1.ol_text = "CF/P Steering 2 \"; //bdl
    trk_CF_steering1.x_scale = ROUND_INT(machine_tb_scale * 14);
    trk_CF_steering1.x_work = &VMachine::shift_cf_theta_aim; //bdl
    trk_CF_steering1.y_scale = ROUND_INT(machine_tb_scale * 14);
    trk_CF_steering1.y_work = &VMachine::shift_cf_phi_aim; //bdl
    trk_CF_steering1.end_work = &VMachine::finish_cf_aiming;
    ...
    trk_CF_width1.ol_text = "CF/P Width 2 \";
    trk_CF_width1.x_scale = ROUND_INT(machine_tb_scale * 10);
    trk_CF_width1.x_work = &VMachine::shift_cf_theta_span; //bdl
    trk_CF_width1.y_scale = ROUND_INT(machine_tb_scale * 10);
    trk_CF_width1.y_work = &VMachine::shift_cf_phi_span; //bdl
    trk_CF_width1.end_work = &VMachine::finish_cf_aiming;
    ...
    trk_CF_depth1.ol_text = "CF/P Depth 2 \";
    trk_CF_depth1.x_scale = ROUND_INT(machine_tb_scale * 20);
    trk_CF_depth1.x_work = &VMachine::shift_cf_start_depth; //bdl
    trk_CF_depth1.y_scale = ROUND_INT(machine_tb_scale * 20);
trk_CF_depth1.y_work = &VMachine::shift_cf_end_depth; // bdl
trk_CF_depth1.end_work = &VMachine::null_end_work;

trk_pw_az_steering.ol_text = "PW El R "; // bdl
trk_pw_az_steering.x_scale = ROUND_INT(machine_tb_scale * 4);
trk_pw_az_steering.x_work = &VMachine::shift_az_tilt;
trk_pw_az_steering.y_scale = ROUND_INT(machine_tb_scale * SPEC_DEPTH_SCALE_NOM);
trk_pw_az_steering.y_work = &VMachine::shift_gate_depth;
trk_pw_az_steering.end_work = &VMachine::null_end_work;

trk_pw_el_steering.ol_text = "PW Az R"; // bdl
trk_pw_el_steering.x_scale = ROUND_INT(machine_tb_scale * 4);
trk_pw_el_steering.x_work = &VMachine::shift_el_tilt;
trk_pw_el_steering.y_scale = ROUND_INT(machine_tb_scale * SPEC_DEPTH_SCALE_NOM);
trk_pw_el_steering.y_work = &VMachine::shift_gate_depth;
trk_pw_el_steering.end_work = &VMachine::null_end_work;

trk_pw_az_steering1.ol_text = "PW El L "; // bdl
trk_pw_az_steering1.x_scale = ROUND_INT(machine_tb_scale * 4);
trk_pw_az_steering1.x_work = &VMachine::shift_parallel_x_tilt;
trk_pw_az_steering1.y_scale = ROUND_INT(machine_tb_scale * SPEC_DEPTH_SCALE_NOM);
trk_pw_az_steering1.y_work = &VMachine::shift_gate_depth;
trk_pw_az_steering1.end_work = &VMachine::null_end_work;

trk_pw_el_steering1.ol_text = "PW Az L"; // bdl
trk_pw_el_steering1.x_scale = ROUND_INT(machine_tb_scale * 4);
trk_pw_el_steering1.x_work = &VMachine::shift_parallel_y_tilt;
trk_pw_el_steering1.y_scale = ROUND_INT(machine_tb_scale * SPEC_DEPTH_SCALE_NOM);
trk_pw_el_steering1.y_work = &VMachine::shift_gate_depth;
trk_pw_el_steering1.end_work = &VMachine::null_end_work;

...
if (cmd_state_table[SHIFT_AZ_TILT_CMD] == ENABLE) { //bdl
    if (vm->vs->xc->xd_test_id==4) { //dual case - bd added
        conditional 2/8/10
        track_state_id = SECTOR_TILT; //bdl
        current_trk_state = &trk_echo_sector_tilt; //bdl
    }
    else { //bdl single case
        track_state_id = PAR_DEPTH_SEP;
        current_trk_state = &trk_echo_par_depth_sep;
    }
    break; //bdl
}
...
}

void TBState::trk_change_cf_state (void) {
    const int *cmd_state_table=vm->get_cmd_state_table();
    if (vm->vs->xc->xd_test_id==4) { //bd added 8/5/10
        switch (track_state_id) { //bdl
            case TRK_DISABLE:
                return;
            // Handle cases of coming from different mode
            case TRK_OFF:
                default:
                    case CF_DEPTH:
                        if (cmd_state_table[SHIFT_CF_THETA_AIM_CMD] == ENABLE) {
                            track_mode = TRK_NOT_ECHO;
                            track_state_id = CF_AIM;
                            current_trk_state = &trk_CF_steering;
                            break;
                        }
            // Fall through if no match
            case CF_AIM:
                if (cmd_state_table[SHIFT_CF_THETA_SPAN_CMD] == ENABLE) {
                    track_mode = TRK_NOT_ECHO;
                    track_state_id = CF_SPAN;
                    current_trk_state = &trk_CF_width;
                    break;
                }
// Fall through if no match

case CF_SPAN:
    if (cmd_state_table[SHIFT_CF_START_DEPTH_CMD] == ENABLE) {
        track_mode = TRK_NOT_ECHO;
        track_state_id = CF_DEPTH;
        current_trk_state = &trk_CF_depth;
        break;
    }
    case CF_DEPTH1://begin bdl add
    if (cmd_state_table[SHIFT_CF_THETA_AIM_CMD] == ENABLE) {
        track_mode = TRK_NOT_ECHO;
        track_state_id = CF_AIM1;
        current_trk_state = &trk_CF_steering1;
        break;
    }
    // Fall through if no match

case CF_AIM1:
    if (cmd_state_table[SHIFT_CF_THETA_SPAN_CMD] == ENABLE) {
        track_mode = TRK_NOT_ECHO;
        track_state_id = CF_SPAN1;
        current_trk_state = &trk_CF_width1;
        break;
    }
    // Fall through if no match

case CF_SPAN1:
    if (cmd_state_table[SHIFT_CF_START_DEPTH_CMD] == ENABLE) {
        track_mode = TRK_NOT_ECHO;
        track_state_id = CF_DEPTH;
        current_trk_state = &trk_CF_depth;
        break;
    }//end bdl add
    // If no available state, do nothing
  //end switch
  }//end if
else{
    switch (track_state_id) {
  case TRK_DISABLE:
    return;
  // Handle cases of coming from different mode
  case TRK_OFF:
default:
case CF_DEPTH:
   if (cmd_state_table[SHIFT_CF_THETA_AIM_CMD] == ENABLE) {
      track_mode = TRK_NOT_ECHO;
      track_state_id = CF_AIM;
      current_trk_state = &trk_CF_steering;
      break;
   }
   // Fall through if no match
   case CF_AIM:
   if (cmd_state_table[SHIFT_CF_THETA_SPAN_CMD] == ENABLE) {
      track_mode = TRK_NOT_ECHO;
      track_state_id = CF_SPAN;
      current_trk_state = &trk_CF_width;
      break;
   }
   // Fall through if no match
   case CF_SPAN:
   if (cmd_state_table[SHIFT_CF_START_DEPTH_CMD] == ENABLE) {
      track_mode = TRK_NOT_ECHO;
      track_state_id = CF_DEPTH;
      current_trk_state = &trk_CF_depth;
      break;
   }
   // If no available state, do nothing
   //end switch
   } //end else

void TBState::trk_change_pw_state (void) {//bdl changed
   const int *cmd_state_table=vm->get_cmd_state_table();
   if(vm->vs->xc->xd_test_id==4){/bdl added 8/5/10
   switch (track_state_id) {
   case TRK_DISABLE:
      return;
   // Handle cases of coming from different mode
   case TRK_OFF:
      default:
      case PW_EL_AIM1:
if (cmd_state_table[SHIFT_AZ_TILT_CMD] == ENABLE) {
    track_mode = TRK_NOT_ECHO;
    track_state_id = PW_AZ_AIM;
    current_trk_state = &trk_pw_az_steering;
    break;
}
    case PW_AZ_AIM:
if (cmd_state_table[SHIFT_EL_TILT_CMD] == ENABLE) {
    track_mode = TRK_NOT_ECHO;
    track_state_id = PW_EL_AIM;
    current_trk_state = &trk_pw_el_steering;
    break;
}
    case PW_EL_AIM:
if (cmd_state_table[SHIFT_AZ_TILT_CMD1] == ENABLE) {
    track_mode = TRK_NOT_ECHO;
    track_state_id = PW_AZ_AIM1;
    current_trk_state = &trk_pw_az_steering1;
    break;
}
// Fall through if no match
    case PW_AZ_AIM1:
if (cmd_state_table[SHIFT_EL_TILT_CMD1] == ENABLE) {
    track_mode = TRK_NOT_ECHO;
    track_state_id = PW_EL_AIM1;
    current_trk_state = &trk_pw_el_steering1;
    break;
}
// Fall through if no match

    // Fall through if no match

    //end bdl
    // If no available state, do nothing
}//end switch

});//end if
else{
switch (track_state_id) {
case TRK_DISABLE:


return;
// Handle cases of coming from different mode
case TRK_OFF:
default:
    case PW_AZ_AIM:
        if (cmd_state_table[SHIFT_EL_TILT_CMD] == ENABLE) {
            track_mode = TRK_NOT_ECHO;
            track_state_id = PW_EL_AIM;
            current_trk_state = &trk_pw_el_steering;
            break;
        }
    case PW_EL_AIM:
        if (cmd_state_table[SHIFT_AZ_TILT_CMD1] == ENABLE) {
            track_mode = TRK_NOT_ECHO;
            track_state_id = PW_AZ_AIM;
            current_trk_state = &trk_pw_az_steering;
            break;
        }
        // Fall through if no match
        //end bdl
    // If no available state, do nothing
    //end switch
    //end else
}

int TBState::is_non_echo_tb_state_ok() {
    ModeType mode = vm->get_mode();
    const int *cmd_state_table = vm->get_cmd_state_table();

    switch (track_state_id) {
        case TRK_DISABLE:
            return TRUE;
        case CF_AIM:
            return ((mode == CF_MODE) && (cmd_state_table[SHIFT_CF_THETA_AIM_CMD] == ENABLE));
        case CF_SPAN:
            return ((mode == CF_MODE) && (cmd_state_table[SHIFT_CF_THETA_SPAN_CMD] == ENABLE));
        case CF_DEPTH:
            return ((mode == CF_MODE) && (cmd_state_table[SHIFT_CF_THETA_SPAN_CMD] == ENABLE));
    
    return FALSE;
    }
return
((mode==CF_MODE)&&(cmd_state_table[SHIFT_CF_START_DEPTH_CMD] ==
ENABLE));

  case CF_AIM1://bdl
    return ((mode==CF_MODE)&&(cmd_state_table[SHIFT_CF_THETA_AIM_CMD1]
    == ENABLE));
  case CF_SPAN1://bdl
    return
((mode==CF_MODE)&&(cmd_state_table[SHIFT_CF_THETA_SPAN_CMD1] ==
ENABLE));
  case CF_DEPTH1://bdl
    return
((mode==CF_MODE)&&(cmd_state_table[SHIFT_CF_START_DEPTH_CMD1] ==
ENABLE));
  case PW_EL_AIM:
    return ((mode==PW_MODE)&&(cmd_state_table[SHIFT_EL_TILT_CMD] ==
ENABLE));
  case PW_AZ_AIM:
    return ((mode==PW_MODE)&&(cmd_state_table[SHIFT_AZ_TILT_CMD] ==
ENABLE));
    case PW_EL_AIM1://bdl
      return ((mode==PW_MODE)&&(cmd_state_table[SHIFT_EL_TILT_CMD1] ==
ENABLE));
  case PW_AZ_AIM1://bdl
      return ((mode==PW_MODE)&&(cmd_state_table[SHIFT_AZ_TILT_CMD1] ==
ENABLE));
  case M_STEERING:
      return ((mode==M_MODE)&&(cmd_state_table[SHIFT_EL_TILT_CMD] ==
ENABLE));
  case TRK_OFF:
    default:
      return FALSE;
    }
  }

View_cfg.cpp
static OVDescType single_b_overlay_item[OV_SINGLE_B_LIMIT] = {
...
  OV_BANNER, 2, 4, 19, &overlay_item[OV_BANNER], FALSE,//bdl was TRUE
<table>
<thead>
<tr>
<th>Variable</th>
<th>ID</th>
<th>W</th>
<th>H</th>
<th>Overlay Item</th>
<th>Bdl Changed</th>
<th>New ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>OV_TBALL_CONTROL_STATE</td>
<td>29</td>
<td>55</td>
<td>24</td>
<td>TRUE,</td>
<td>/bdl</td>
<td>46</td>
</tr>
<tr>
<td>OV_SPECTRAL_SCALE_MAX</td>
<td>17</td>
<td>16</td>
<td>15</td>
<td>FALSE,</td>
<td>/bdl</td>
<td>16</td>
</tr>
<tr>
<td>OV_SPECTRAL_SCALE_MIN</td>
<td>28</td>
<td>16</td>
<td>15</td>
<td>FALSE,</td>
<td>/bdl</td>
<td>16</td>
</tr>
<tr>
<td>OV_SPECTRAL_SCALE_UNITS</td>
<td>16</td>
<td>16</td>
<td>15</td>
<td>FALSE,</td>
<td>/bdl</td>
<td>16</td>
</tr>
<tr>
<td>OV_SPECTRAL_MAP</td>
<td>15</td>
<td>32</td>
<td>15</td>
<td>FALSE,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OV_VOLUME</td>
<td>2</td>
<td>53</td>
<td>13</td>
<td>FALSE,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OV_AZ_TILT</td>
<td>11</td>
<td>36</td>
<td>5</td>
<td>TRUE,</td>
<td>/bdl</td>
<td>11</td>
</tr>
<tr>
<td>OV_EL_TILT</td>
<td>14</td>
<td>36</td>
<td>5</td>
<td>TRUE,</td>
<td>/bdl</td>
<td>14</td>
</tr>
</tbody>
</table>

```

static const ViewConfig view_config_defs[] = {

  { VIEW_VOLUME_STANDARD, // id //bdl changed all around
    5, // # vps
    { // viewport info
      // vp_type origin extent (w,h)
      { VP_PARALLEL_FLAT,  { 600, 600}, {10, 10} },
      { VP_ELEVATION,      {330, 300}, {400, 300} },
      { VP_ELEVATION1,     { 50, 300}, {400, 300} },
      { VP_AZIMUTH,        {400,  0}, {400, 300} },
      { VP_AZIMUTH1,       { 0,  0}, {400, 300} }
    },

    // overlay text list and size for this view
    OV_MULTI_B_LIMIT, multi_b_overlay_item,
  },

  { VIEW_VOLUME_STANDARD2, // id
```
5, // # vps
{ // viewport info
  // vp_type    origin     extent (w,h)
  { VP_PARALLEL_FLAT,     {0, 0}, {400, 300} },
  { VP_PARALLEL_FLAT,     {0, 300}, {400, 300} },
  { VP_AZIMUTH,           {400, 0}, {400, 300} },
  { VP_ELEVATION,         {400, 300}, {400, 300} },
  { VP_ECG,               {0, 559}, {400, 40} }
},

// overlay text list and size for this view
OV_MULTI_B_LIMIT,
multi_b_overlay_item,
},
...
{
  VIEW_VOLUME_SCROLL, // id
  6, // # vps
  { // viewport info
    // vp_type    origin     extent (w,h)
    { VP_PARALLEL_FLAT,     {600, 600}, {10, 10} },
    { VP_ELEVATION,         {340, 320}, {400, 300} }, // bdl was 330, 300 for origin
    { VP_ELEVATION1,        {0, 320}, {400, 300} }, // bdl was 50, 300 for origin
    { VP_AZIMUTH,           {400, 0}, {400, 300} }
  },
  { VP_AZIMUTH1,          {0, 0}, {400, 300} },
  { VP_SCROLL,            {240, 320}, {304, 250} }, // bdl was 60, 320 for origin, 304, 220 for extent
  },

// overlay text list and size for this view
OV_MULTI_B_LIMIT,
multi_b_overlay_item,
},
{
  VIEW_VOLUME_SCROLL2, // id
  4, // # vps
  { // viewport info
    // vp_type    origin     extent (w,h)
    { VP_PARALLEL_FLAT,     {0, 0}, {400, 300} },
  },
{ VP_AZIMUTH, [400, 0], [400, 300] },
{ VP_ELEVATION, [400, 300], [400, 300] } // bdl was 330, 300
for origin
{ VP_SCROLL, [60, 320], [304, 220] } // bdl was 60, 320 for origin, 304, 220 for extent
}

// overlay text list and size for this view
OV_MULTI_B_LIMIT,
multi_b_overlay_item,
},
...
}

PWView.cpp – analogous changes also made to EView.cpp and CFView.cpp
void PWView::new_az_sector_viewport1(int vp_id, const ViewportInfo *vp_info) {// bdl
    pw_az_vp1 = new PWSectorViewport(vm, vs, this, vp_id, vp_info);
    az_vp1 = pw_az_vp1;
}

void PWView::new_el_sector_viewport1(int vp_id, const ViewportInfo *vp_info) {// bdl
    pw_el_vp1 = new PW SectorViewport(vm, vs, this, vp_id, vp_info);
    el_vp1 = pw_el_vp1;
}

void PWView::new_scroll_viewport1(int vp_id, const ViewportInfo *vp_info) {// bdl
    pw_scroll_vp = new PWScrollViewport(vm, vs, this, vp_id, vp_info);
    scroll_vp1 = pw_scroll_vp;
}

... // begin bdl add
void PWView::set_az_tilt1(double az_tilt) {
    echo_set_az_tilt1(az_tilt); // bdl
    update_aiming();
}

void PWView::set_el_tilt1(double el_tilt) {
    echo_set_el_tilt1(el_tilt); // bdl
    // printf("inside set_el_tilt1, el_tilt = %d \n", el_tilt); // bdl
    update_aiming();
}
VolRend.cpp

```cpp
void VolRendViewport::compute_matrices(double x_tilt, double y_tilt)
{
    //EchoParallelViewport::compute_matrices(x_tilt);//bdl
    EchoParallelViewport::compute_matrices(x_tilt, y_tilt);//bdl

    update_top_bottom_matrices();

    // have to re-do update sc params since it uses the top/bottom matrix.
    // for that, have to recompute Mt_q.
    Matrix Mt_q;
    matrix_multiply_3x4(Mt_q, Mt_v, Mv_q);
    update_sc_params(Mt_q);
}
```

Echo_VM.cpp

```cpp
EchoVSInit::EchoVSInit() {
...
    // View Init
    view_config = VIEW_VOLUME_STANDARD; //farfield was
    VIEW_SINGLE_STANDARD
    doppler_units = FREQ_UNITS;    //bdl via farfield was VEL_UNITS
    parallel_x_tilt = 10;//bdl check
    parallel_y_tilt = 10;//bdl check
    parallel_depth = 4;
    parallel_spacing = 0;//bdl
...
}
```

CmdState.cpp

```cpp
void VMachine::set_echo_cmd_state() {
...
    if (data_type == VOLUME) {
...
        cmd_state_table[SHIFT_AZ_TILT_CMD1] = ENABLE;//bdl
        cmd_state_table[SHIFT_EL_TILT_CMD1] = ENABLE;//bdl
...
    }
```
void VMachine::set_echo_cmd_state() {
    ...
    if (data_type == VOLUME) {
        ...
        cmd_state_table[SHIFT_AZ_TILT_CMD1] = ENABLE;//bdl
        cmd_state_table[SHIFT_EL_TILT_CMD1] = ENABLE;//bdl
    } else {
        ...
        cmd_state_table[SHIFT_AZ_TILT_CMD1] = DISABLE;//bdl
        cmd_state_table[SHIFT_EL_TILT_CMD1] = DISABLE;//bdl
    }
}

void VMachine::disable_echo_cmd_state() {
    ...
    cmd_state_table[SHIFT_AZ_TILT_CMD1] = DISABLE;//bdl
    cmd_state_table[SHIFT_EL_TILT_CMD1] = DISABLE;//bdl
    ...
}

Sc_State.cpp
//farfield addition begins
RC VMachine::begin_record_id() {
    RC ret_code = RC_OK;
    clear_annotation();

    // See if the transducer has changed. Load the new transducer if it has.
    // Do this here since transducer changes while playing/paused are ignored.
    if ((int) update_xducer_status(RECORD_UPDATE) < 0) {
        return RC_ERROR;
    }
    // Only start record if a transducer is attached
    if (xc->is_xducer_attached()) {
        // If the current vs exists and was the last rec vs, it can be restored.
        // Note: Anything that destroys and recreates the last_rec_vs must
        //     set it to NONE.
        // If the current vs doesn’t exist (or is set to none), then restore_rec_vs
// will create it from rec_init and restore it.
// If the current vs wasn’t the last rec vs, it may have been used to play other data
// or values may have changed so restore_rec_vs must also be called
if ((current_vs != NONE) &&
    (current_vs == last_rec_vs) &&
    (vs != NULL)) {
    if (current_vs != last_vs_used) {
        vs->restore();
    }
}
else {
    restore_rec_vs();
}

// if vs is NULL, we have a serious failure. This also covers no-probe although
// we shouldn’t get here in that case.
if ((vs == NULL) || (vs->get_no_probe_vs()))) {
    // No vs or vs with no probe assigned
    last_rec_vs = NONE;
    last_vs_used = NONE;
    ret_code = RC_ERROR;
    ret_code = begin_stop();
    HAL_clear_slice_buffer();
} else {
    // The vs is set up so we can start recording
    ret_code = vs->start_record();
    sc->set_scan_state(RECORDING);
    last_rec_vs = current_vs;
    last_vs_used = current_vs;
}
else {
    // In the no probe case, clear flags and clear the screen. A no-probe vs
    // will have been created so that overlay text can be drawn.
    // Set the machine in stop mode.
    restore_rec_vs();
    last_rec_vs = NONE;
    last_vs_used = NONE;
    ret_code = begin_stop();
    HAL_clear_slice_buffer();
}
return ret_code;
}
//end farfield

RC VMachine::resume() {
    RC ret_code = RC_OK;

    switch (sc->get_last_scan_state()) {
        case RECORDING:
            ret_code = begin_record();
            if (ret_code == RC_ERROR)//farfield
                ret_code = begin_record_id();//farfield
            break;//farfield
        break;//farfield
        case PAUSED:
            ret_code = begin_pause();
            break;
        case PLAYING:
            ret_code = begin_play();
            break;
        case STOPPED:
            ret_code = begin_stop();
    }
    return ret_code;
}

Scr2d.cpp
RC ScreenEcho2D::annotate( void * ) {//bdl now using this to transfer a volume to PC
    if (cmd_state_table[RESET_PARALLEL_TILTS_CMD] == DISABLE) {
        return RC_OK;
    }*/
    if (vm->vs == NULL) return RC_ERROR;

    String file_desc;
    int current_frame;
    int start_frame;
    int end_frame;

    file_desc="doppler data for Tx";
    vm->vs->set_file_description(&file_desc);
String save_file_name;
save_file_name="c:\brooks\test.vol";

current_frame=vm->vs->get_current_frame();
start_frame=vm->vs->get_start_frame();
end_frame=vm->vs->get_end_frame();

vm->vs->store_file(&save_file_name, start_frame, end_frame);

system("cls");
cout << "File description registered as: " << file_desc << ".
";
cout << "The save path is " << save_file_name << ".
";
system("cd c:\brooks");
system("dostxvol.exe");

return RC_STATE_CHANGED;
//return RC_SCREEN_PUSH;
//end bdl
//bdl - was:

/*
next_screen = SCREEN_ANNOTATE;
return RC_SCREEN_TRANSPARENT_PUSH;*/
}

UserCtrl.cpp

[A_ANNOTATE, "Send Vol\nto PC", NULL_DEVICE, DISABLED, NO_REPEAT, NULL, //bdl changed text from "Annotate"
{&Screen::annotate,
 &Screen::none,
 &Screen::none}]
,
[A_MULTI_B_FLAT_MODE, "Dual\nvol. view", NULL_DEVICE, //bdl changed 2/8/10, was [A_MULTI_B_FLAT_MODE, "Multi-B\nFlat", NULLDEVICE, INVISIBLE, NO_REPEAT, NULL,
{&Screen::multi_b_flat_mode,
 &Screen::none,
 &Screen::none}]
,  
[A_MULTI_B_OBLIQUE_MODE, "Single\nvol. view", NULL_DEVICE, //bdl changed 2/8/10, was [A_MULTI_B_OBLIQUE_MODE, "Multi-B\nOblique", NULL_DEVICE,
INVISIBLE, NO_REPEAT, NULL,
{&Screen::multi_b_oblique_mode,
 &Screen::none,
 &Screen::none}
},
Scr3d.cpp
RC ScreenEcho3D::define_true_axis(void *data) //nmi
{
    vm->change240();
    return RC_OK;
}

Vidinit.c
HAL_set_doppler_echo_min(0xff);//bdl was 2e
References


Biography

Name: Brooks D. Lindsey

Date of Birth: April 27, 1984

Place of Birth: Liberty, MO

Education: Ph.D., Biomedical Engineering, 2012
Duke University, Durham, NC

B.S., Electrical Engineering, 2007
University of Illinois, Urbana-Champaign

Peer-reviewed publications

“Simultaneous bilateral real-time 3D transcranial ultrasound imaging at 1 MHz through poor acoustic windows,” Lindsey BD, Nicoletto HA, Bennett ER, Laskowitz DT, Smith SW (in review).


Patent Application (PCT)
Smith SW, Ivancevich NM, Lindsey BD, Light ED, Frohneiser MP, and Whitman JJ.

Awards
NIH Pre-doctoral fellow, 2009-2011