UNIVERSITY OF CALIFORNIA SAN DIEGO

3D Reconstruction and Segmentation of Barely Visible Impact Damage in Composites from Pulse-Echo Ultrasonic C-Scans

A Thesis submitted in partial satisfaction of the requirements
for the degree Master of Science

in

Structural Engineering

by

Jessica Yeu Mao Chan

Committee in charge:

Professor Hyonny Kim, Chair
Professor Veronica Eliasson
Professor Francesco Lanza Di Scalea
Professor Michael D. Todd

2023

University of California San Diego

2023

DEDICATION

In memory of my friend, Ayesha Mohammed

May she rest in peace.

TABLE OF CONTENTS

Thesis Approval Page iii

Dedication iv

Table of Contents v

List of Figures vii

List of Tables xiii

List of Abbreviations xiv

Acknowledgements xv

Abstract of the Thesis xvii

CHAPTER 1: Introduction 1

1.1 Scope 1
1.2 Motivation 2
1.3 Background 2
1.4 Previous Work 6
1.5 Literature Review 7
1.6 Acknowledgements 9

CHAPTER 2: Experimental Dataset 10

2.1 25-Ply UT C-scan Samples 10
2.1.1 Sample Information 10
2.1.2 Pulse-Echo UT C-scan Information 12
2.2 24-ply UT C-scan and CT Scanned Sample 13
2.2.1 Sample Information 14
2.2.2 UT C-scan and CT Scan Information 14

CHAPTER 3: Code Development 15

3.1 Code Overview 15
3.2 File Input Requirements 17
3.3 Primary Script – main.m 19
3.4 Read C-scan – readcscan 22
3.5 Process C-scan – processcscan 23
3.5.1 Damage Bounding Box 23
3.5.2 Calculate TOF – calctof 27
3.5.3 Figures and Output Variables 29
3.6 Segment C-scan – segcscan 31

3.6.1 Change in Peak Labels – labelpeaks 32
3.6.2 Peaks in Second Peak Magnitude – magpeaks 35
3.6.3 Mask of Undamaged Region 37
3.6.4 Inflection Points Cleanup 39
3.6.5 Figures and Output Variables 44
3.7 Plot Test Figures – plottest 45
3.8 Plot Figures – plotfig 46
3.9 Merge C-scan – mergecscan 47
3.10 Plot Custom Figures – plotcustom 48

CHAPTER 4: Results and Discussion 49

4.1 25-ply Damage Depth Maps 49
4.2 25-ply Hybrid C-scans and 3D C-scan Reconstruction 55
4.3 24-ply Damage Depth Maps 60
4.4 24-ply Hybrid C-scans and 3D C-scan Reconstruction 62
4.5 Acknowledgements 62

CHAPTER 5: Conclusion and Future Work 63

5.1 Conclusion 63
5.2 Future Work 64

Appendix A : Links to Code and Data 65

Appendix B : Damage Depth Maps 66

Appendix C : 3D Reconstructions of C-scans 69

Appendix D : Morphological TOF Processing Steps 73

References 99

LIST OF FIGURES

Figure 1.1: Visual summary of C-scan processing code steps. 1

Figure 1.2: (A) Example through-thickness X-ray CT scan slice showing barely visible impact damage labeled with impact direction and front/back side with respect to impact direction. Reprinted with permission [4]. (B) Processed UT C-scan of the same sample as in (A) showing front side damage and representative location for section cut. 3

Figure 1.3: (A) Photo of a sample with manually mapped damage region outline, (B) UT C-scan time-of-flight map, (C) C-scan process and calculating TOF for undamaged scan point (right) and damaged scan point (left) using the A-scan signal at each scan point. 5

Figure 1.4: A-scan gating procedure. 6

Figure 1.5: A-scan pulses showing low sampling frequency (top) creating asymmetrical peaks compared to high sampling frequency (bottom). 7

Figure 2.1: Pendulum Impactor. 11

Figure 2.2: Sample in immersion tank and C-scan system. 13

Figure 3.1: Code overview diagram. 16

Figure 3.2: File header information. 18

Figure 3.3: Input file format. 18

Figure 3.4: main.m script section structure. 20

Figure 3.5: main.m Section iii, function options. 21

Figure 3.6: Run Function section example, Section 1. 21

Figure 3.7: Example progress message from Section 1. 21

Figure 3.8: Raw and absolute value of A-scan signal 23

Figure 3.9: Searching for damage bounding box example. (A) Photo of sample showing manually mapped damage, (B) C-scan of damage, (C) Damage bounding box search input parameters. 24

Figure 3.10: Damage bounding box search process. (A) Search along columns, (B) picking start row and end row indices, (C) search along rows, (D) picking start and end column indices. 26

Figure 3.11: TOF calculating process. (A) Finding peaks, (B) Fitting curve to peaks, (C) Calculating TOF. 28

Figure 3.12: (A) Time of first peak, (B) raw TOF. 29

Figure 3.13: (A) Queryable raw TOF plot, (B) Non-queryable raw TOF plot. 30

Figure 3.14: **segcscan** code diagram. 32

Figure 3.15: Peak label process example. (A) Example path direction along Row 215 from Column 95 to 100, (B) A-scan signals along path direction showing changes in peak labels and detected peak changes. 33

Figure 3.16: Inflection points found via (A) peak labels changes along rows, (B) peak label changes along columns, (C) peaks in second peak magnitude along rows, (D) peaks in second peak magnitude along columns. (E) Inflection points from all methods (A)-(D). (F) Unprocessed TOF for reference. 34

Figure 3.17: Second peak magnitude plot example. 35

Figure 3.18: Second peak magnitude layer change detection method example. (A) Example path direction along Row 171 from Column 115 to 121, (B) A-scan signals along path direction showing changes in peak magnitude and detected layer change, (C) simplified example of damage layer and UT C-scan transducer progression.. 36

Figure 3.19: Examples of morphological operations (A) clean, (B) spur, operated once, (C) spur, operating with option ‘inf’ (until image no longer changes). (D) Examples of regions in inflection point map that will be cleaned from clean (blue) and spur operations (orange)**.** 38

Figure 3.20: (A) Inflection points map before morphological operation cleanup, (B) inflection point map after morphological operation cleanup for creating damage area mask, (C) damage area mask, (D) damage region boundary. 39

Figure 3.21: (A) Example of a structuring element. Examples of morphological operations (B) dilation, (C) erosion, and (D) closing. 40

Figure 3.22: **bwmorph** bridge operation. 42

Figure 3.23: (A) Inflection point map before morphological operation cleanup, (B) inflection point map after morphological operation cleanup for segmentation, (C) labeled connected regions, (D) processed TOF using mode value of respective connected regions. 42

Figure 3.24: (A) Raw TOF compared to (B) processed TOF. 43

Figure 3.25: Example of **plotascan** output, where (A) are the A-scan plots with smoothing spline fit, peaks, peak width/prominence, and *noiseThresh* shown, (B) tabular data displayed in command window about the peak magnitude, location, width, and prominence for each A-scan. 45

Figure 3.26: Example of **plotpeak2** output, where (A) is the row-wise plot of the second peak magnitude with peak information shown, (B) is the tabular peak data displayed in the command window. 46

Figure 3.27: **plotfig** (A) 3D and (B) 2D plots. 47

Figure 3.28: Damage region plots for adjusting *dx* and *dy* inputs, where left is the initial outline alignment and right is the alignment after adjustment. 48

Figure 3.29: Example of a hybrid C-scan (right) and the corresponding front (left) and back (middle) C-scans. 48

Figure 4.1: Front and back side terminology. Reprinted with permission [4]. 49

Figure 4.2: CONT-S-15J-2 Damage Depth Map. (A) Raw Front, (B) Processed Front, (C) UTWin Front, (E) Raw Back, (F) Processed Back, (F) UTWin Back. 50

Figure 4.3: BL-S-10J-2 Damage Depth Map. (A) Raw Front, (B) Processed Front, (C) UTWin Front, (E) Raw Back, (F) Processed Back, (F) UTWin Back. 50

Figure 4.4: BL-S-15J-2 Damage Depth Map. (A) Raw Front, (B) Processed Front, (C) UTWin Front, (E) Raw Back, (F) Processed Back, (F) UTWin Back. 52

Figure 4.5: BL-S-20J-2 Damage Depth Map. (A) Raw Front, (B) Processed Front, (C) UTWin Front, (E) Raw Back, (F) Processed Back, (F) UTWin Back. 52

Figure 4.6: CONT-S-10J-2 Damage Depth Map. (A) Raw Front, (B) Processed Front, (C) UTWin Front, (E) Raw Back, (F) Processed Back, (F) UTWin Back. 53

Figure 4.7: CONT-S-20J-2 Damage Depth Map. (A) Raw Front, (B) Processed Front, (C) UTWin Front, (E) Raw Back, (F) Processed Back, (F) UTWin Back. 53

Figure 4.8: RPR-S-10J-2 Damage Depth Map. (A) Raw Front, (B) Processed Front, (C) UTWin Front, (E) Raw Back, (F) Processed Back, (F) UTWin Back. 54

Figure 4.9: RPR-S-15J-2 Damage Depth Map. (A) Raw Front, (B) Processed Front, (C) UTWin Front, (E) Raw Back, (F) Processed Back, (F) UTWin Back. 54

Figure 4.10: RPR-S-20J-2 Damage Depth Map. (A) Raw Front, (B) Processed Front, (C) UTWin Front, (E) Raw Back, (F) Processed Back, (F) UTWin Back. 55

Figure 4.11: BL-S-10J-2 front, back and hybrid C-scan 3D reconstruction. 56

Figure 4.12: BL-S-15J-2 front, back and hybrid C-scan 3D reconstruction. 56

Figure 4.13: BL-S-20J-2 front, back and hybrid C-scan 3D reconstruction. 57

Figure 4.14: CONT-S-10J-2 front, back and hybrid C-scan 3D reconstruction. 57

Figure 4.15: CONT-S-15J-2 front, back and hybrid C-scan 3D reconstruction. 58

Figure 4.16: CONT-S-20J-2 front, back and hybrid C-scan 3D reconstruction. 58

Figure 4.17: RPR-S-10J-2 front, back and hybrid C-scan 3D reconstruction. 59

Figure 4.18: RPR-S-15J-2 front, back and hybrid C-scan 3D reconstruction. 59

Figure 4.19: RPR-S-20J-2 front, back and hybrid C-scan 3D reconstruction. 60

Figure 4.20: LV-162 TOF (A) back unprocessed, (B) back processed, (C) front unprocessed, (D) front processed. 61

Figure 4.21: LV-162 (a) UT C-scan processed with gating procedure and (b) X-ray CT scan Reprinted with permission [5]. 61

Figure 4.22: LV-162 2 front, back and hybrid C-scan 3D reconstruction. 62

Figure B.1: BL-H-15J-1 Damage Depth. Dimensions (height x width): 51 x 51 mm. 66

Figure B.2: CONT-H-10J-2 Damage Depth. Dimensions (height x width): 39 x 38 mm. 66

Figure B.3: CONT-H-10J-3 Damage Depth. Dimensions (height x width): 35 x 38 mm. 66

Figure B.4: CONT-H-20J-1 Damage Depth. Dimensions (height x width): 63 x 68 mm. 67

Figure B.5: CONT-H-20J-2 Damage Depth. Dimensions (height x width): 55 x 55 mm. 67

Figure B.6: RPR-H-20J-3 Damage Depth. Dimensions (height x width): 51 x 51 mm........... 67

Figure B.7: RPR-H-20J-2 Damage Depth. Dimensions (height x width): 51 x 55 mm........... 68

Figure B.8: RPR-H-20J-3 Damage Depth. Dimensions (height x width): 71 x 76 mm........... 68

Figure C.1: BL-H-15J-1 3D Reconstruction of C-scan. 69

Figure C.2: CONT-H-10J-2 3D Reconstruction of C-scan. 69

Figure C.3: CONT-H-10J-3 3D Reconstruction of C-scan. 70

Figure C.4: CONT-H-20J-1 3D Reconstruction of C-scan. 70

Figure C.5: CONT-H-20J-2 3D Reconstruction of C-scan. 71

Figure C.6: CONT-H-20J-3 3D Reconstruction of C-scan. 71

Figure C.7: RPR-H-20J-2 3D Reconstruction of C-scan. 72

Figure C.8: RPR-H-20J-3 3D Reconstruction of C-scan. 72

Figure D.1: BL-S-10J-2 Morphological TOF processing steps. 73

Figure D.2: BL-S-10J-2 Back Morphological TOF processing steps. 74

Figure D.3: BL-S-15J-2 Morphological TOF processing steps. 75

Figure D.4: BL-S-15J-2 Back Morphological TOF processing steps. 76

Figure D.5: BL-S-20J-2 Morphological TOF processing steps. 77

Figure D.6: BL-S-20J-2 Back Morphological TOF processing steps. 78

Figure D.7: CONT-S-10J-2 Morphological TOF processing steps. 79

Figure D.8: CONT-S-10J-2 Back Morphological TOF processing steps. 80

Figure D.9: CONT-S-15J-2 Morphological TOF processing steps. 81

Figure D.10: CONT-S-15J-2 Back Morphological TOF processing steps. 82

Figure D.11: CONT-S-20J-2 Morphological TOF processing steps. 83

Figure D.12: CONT-S-20J-2 Back Morphological TOF processing steps. 84

Figure D.13: RPR-S-10J-2 Morphological TOF processing steps. 85

Figure D.14: RPR-S-10J-2 Back Morphological TOF processing steps. 86

Figure D.15: RPR-S-15J-2 Morphological TOF processing steps. 87

Figure D.16: RPR-S-15J-2 Back Morphological TOF processing steps. 88

Figure D.17: RPR-S-20J-2 Morphological TOF processing steps. 89

Figure D.18: RPR-S-20J-2 Back Morphological TOF processing steps. 90

Figure D.19: BL-H-15J-1-Morphological TOF processing steps. 91

Figure D.20: CONT-H-10J-2 Morphological TOF processing steps. 92

Figure D.21: CONT-H-10J-3 Morphological TOF processing steps. 93

Figure D.22: CONT-H-20J-1 Morphological TOF processing steps. 94

Figure D.23: CONT-H-20J-2 Morphological TOF processing steps. 95

Figure D.24: CONT-H-20J-3 Morphological TOF processing steps. 96

Figure D.25: RPR-H-20J-2 Morphological TOF processing steps. 97

Figure D.26: RPR-H-20J-3 Morphological TOF processing steps. 98

LIST OF TABLES

Table 2.1: Ply Layups and Stiffnesses of Samples 11

Table 2.2: Sample Conditions and Abbreviations. 11

Table 2.3: Code Testing Dataset Samples. 12

List of Abbreviations

CSAI	Compression-strength-after-impact
FE	Finite element
ROI	Region of interest
TOF	Time-of-flight
UT	Ultrasonic testing, specifically single-sided pulse-echo ultrasonic scanning

ACKNOWLEDGEMENTS

To Professor Hyonny Kim, thank you for being a kind and patient advisor during my master’s thesis program. Thank you for teaching me about composite analysis, design, and repair during my undergraduate degree and for providing me with an opportunity to work in your lab as an undergraduate student. Dr. Kim truly cares for all of his students, and I feel very fortunate to be his student.

To Professor Veronica Eliasson, thank you for your encouragement and guidance throughout my undergraduate, professional, and graduate career. You constantly believed in me even when I was unsure of myself.

To Kyung Brown, thank you for offering support and resources when I was facing challenges and being excited with me when I accomplished something new. Each week, I always look forward to stopping by to say hello and chat.

I would also like to thank members of Professor Kim’s research group for their help understanding the technical aspects of my project. A special thank you to Barrett Romasko for impacting and scanning the C-scan samples that form the dataset for my thesis and explaining details of the C-scan process. Thank you to Andrew Ellison for explaining how his C-scan processing code works, which has formed the basis for my own code. I would also like to thank Janelle Dela-Cueva and Ben Katko for their friendship and support throughout my master’s program.

Thank you to Professor Veronica Eliasson, Professor Michael Todd, and Professor Francesco Lanza Di Scalea for their willingness to be on my defense committee.

Finally, a big thank you to my family and friends for their support throughout my master’s degree and my time at the University of California San Diego.

Figure 1.2, in part, is a reprint of the figure as it appears in Segmentation of X-ray CT and Ultrasonic Scans of Impacted Composite Structures for Damage State Interpretation and Model Generation. Ellison, Andrew, PhD disseration, UC San Diego, 2020. The thesis author received permission to reprint the figure.

Figure 4.21, in part, is a reprint of the figure as it appears in Journal of Composite Materials 2020, Ellison, Andrew; Kim, Hyonny, Sage Journals, 2020. The thesis author received permission to reprint the figure.

ABSTRACT OF THE THESIS

3D Reconstruction and Segmentation of Barely Visible Impact Damage in Composites from Pulse-Echo Ultrasonic C-Scans

by

Jessica Yeu Mao Chan

Master of Science in Structural Engineering

University of California San Diego, 2023

Professor Hyonny Kim, Chair

Despite a high strength to weight ratio, composites are susceptible to impact damage which can be barely visible while still adversely affecting their strength, therefore detecting and characterizing damage is important. Non-destructive evaluation, specifically single-sided pulse-echo ultrasonic C-scans, can be used to detect damage. The goal is to develop a code that aids the quantitative assessment of damage within laminates by segmenting C-scans of damaged regions into 3D reconstructions. The code is open source to create a free alternative to commercial ultrasonic C-scan software and allow users to adapt the code for their own applications. The code processes C-scans and calculates time-of-flight (TOF), then uses unique characteristics in C-scans to create a map of boundaries where the damage changes from one layer to another. Using this map, the code processes the TOF to create a 3D reconstruction of damaged layers. When both front and back side C-scans of a sample are available, the code combines them to produce a hybrid C-scan that merges information from both. The code is developed using a dataset of 25-ply composite panels with varying layups and levels of impact damage. A 24-ply composite sample impacted at 48 Joules was also processed and compared with an X-ray CT dataset. The code's 3D reconstruction output is useful for an enhanced understanding of the impact damage state (e.g., area of damage at specific depth), provides the capability to extract quantitative metrics of the damage, and can be used to create actual-geometry structural analysis model representations of the damage state.

CHAPTER 1: INTRODUCTION

1.1 Scope

The scope of the project is to write a generalized open-source code that has the ability to process pulse-echo ultrasonic C-scans of composite samples with barely visible impact damage from different scanning systems when given raw C-scan data composed of A-scan waveforms at each location on the sample. The goal of the project is to use a dataset of 28 C-scans of impact damaged composite samples to develop a code that can segment the different layers of damage, produce a 3D reconstruction of the layers of damage, and create hybrid C-scans by combining front and back C-scans when both are available. A visual summary of the process is shown in Figure 1.1. The damage depth map produced by the code should be consistent with the output from the commercial Mistras UTWin UT C-scan system and capture detail that is lost in during UTWin software data processing method. The 28 impact damaged composite samples used as the testing dataset have varying levels of impact damage, ply layups, materials, scanning resolution, and sampling frequencies.

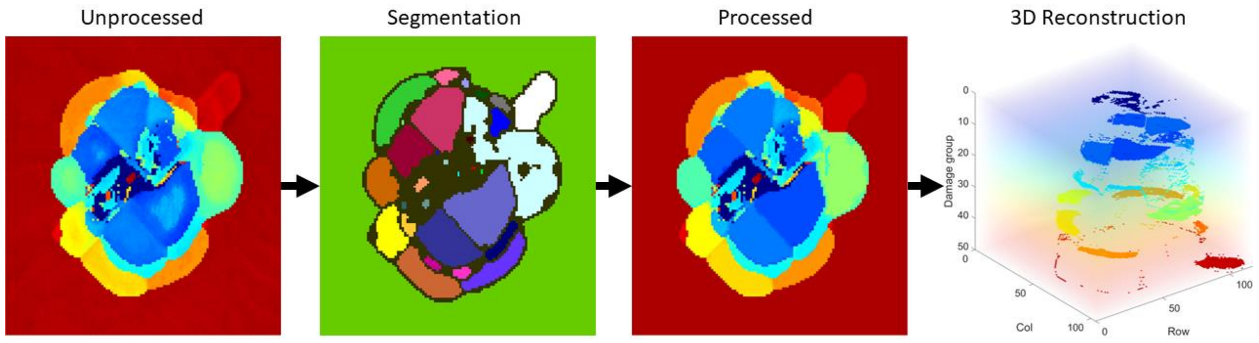


Figure 1.1: Visual summary of C-scan processing code steps.

1.2 Motivation

The project will lay out a method for processing UT C-scans using unique characteristics from the A-scan signals to identify contours of damaged layers and use basic image processing techniques to close gaps in the identified contours. The motivation to create an open-source code is to make available an improved alternative to commercial UT C-scan processing software, with outputs in addition to a damage depth map such as 3D reconstruction, overall damage region identification, and hybrid C-scans. The open-source nature of the code also allows users to look inside the black box, understand how the processing works, and modify input parameters as well as the process to fit the unique requirements for their project.

1.3 Background

Impact damage, specifically barely visible impact damage, is a concern for composites used in aerospace applications, as the Federal Aviation Administration requires that components with this damage be able to withstand the ultimate load anticipated for the part [1]. The main characteristic of barely visible impact damage is that it occurs from impacts that leave little to no visual indication on the front side that damage has occurred, when in fact there is internal damage, namely planar delaminations between the lamina, and there can be visual indication of damage on the back side of the component. Examples of barely visible impact damage include damage to a component impacted by runway debris, hail, or accidentally dropped maintenance tools [1]. The dent depth associated with barely visible impact damage is in the range of 0.25 to 0.50 mm and the velocity associated is between 1 to 10 m/s [2] [3]. An example of barely visible impact damage in shown in Figure 1.2, where Figure 1.2A is a through-thickness slice of the X-ray CT scan and Figure 1.2B is the processed UT C-scan of the same sample. The impact direction is labeled in Figure 1.2A, where the front side of the sample is the face that is impacted and the back side is the opposite face. This convention will be used throughout the rest of the thesis. It can be observed in Figure 1.2A that there is only a small dent on the front side of the sample and a slight curvature on the back side.

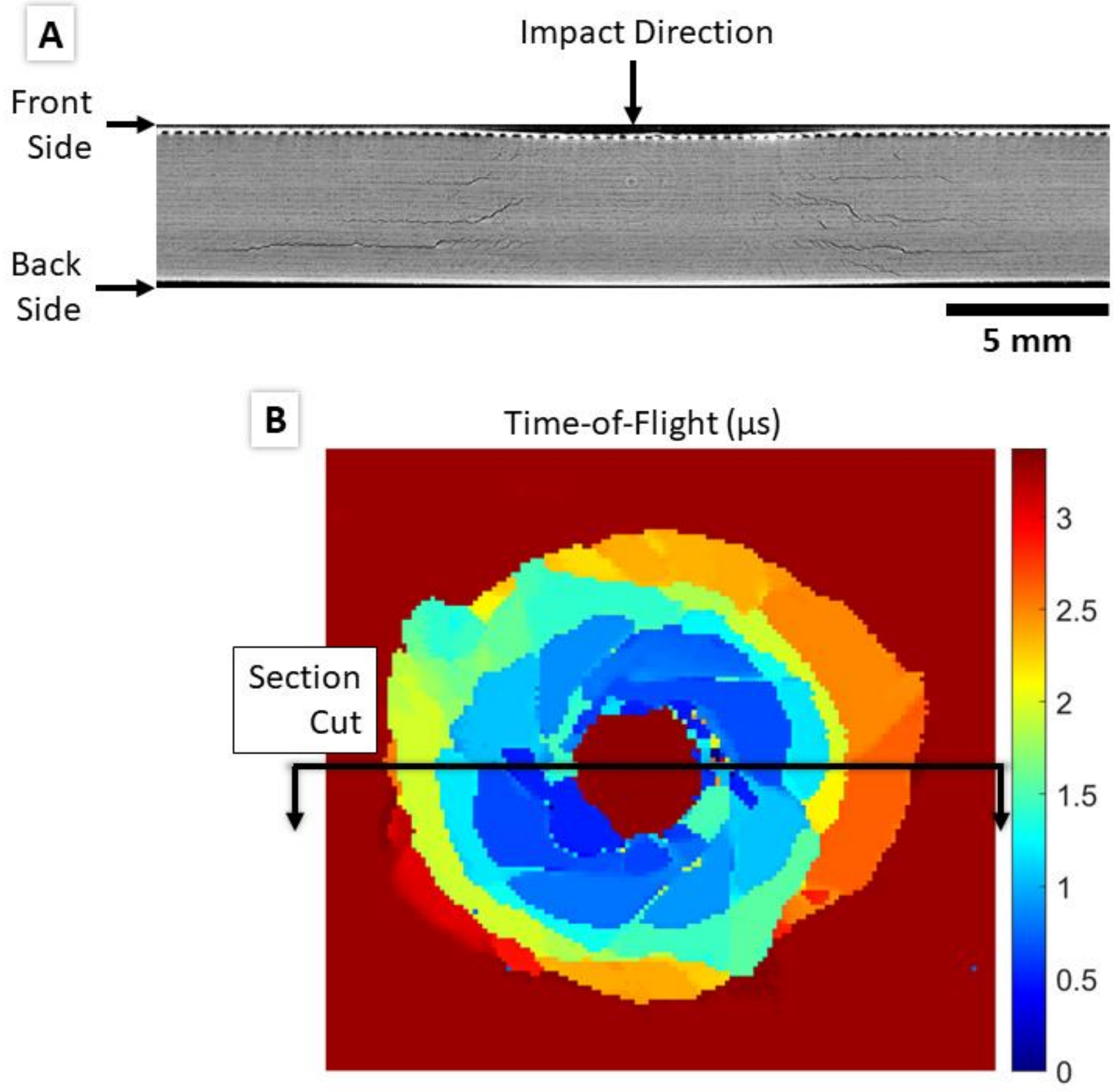


Figure 1.2: (A) Example through-thickness X-ray CT scan slice showing barely visible impact damage labeled with impact direction and front/back side with respect to impact direction. Reprinted with permission [4]. (B) Processed UT C-scan of the same sample as in (A) showing front side damage and representative location for section cut shown in (A).

Pulse-echo ultrasonic C-scans are a commonly used method of finding and characterizing this kind of impact damage in composites and will be the focus of this project. Pulse-echo UT C-scans are made of a 2D array of A-scans taken at points in a uniform grid across a sample. Each A-scan is the measurement of the reflection(s) from the initial ultrasonic signal emitted by the transducer. The first reflected peak is from the top layer of the sample and the second reflected peak is the topmost damaged interface within the sample. An example of the A-scan signal and TOF calculation for an undamaged and damaged A-scan point is shown in Figure 1.3C. A map of the depth of damaged areas can be created by calculating the difference between the first peak and the second peak, the TOF (see Figure 1.3B), which then can be converted to damage depth by using the material's through-thickness wave velocity.

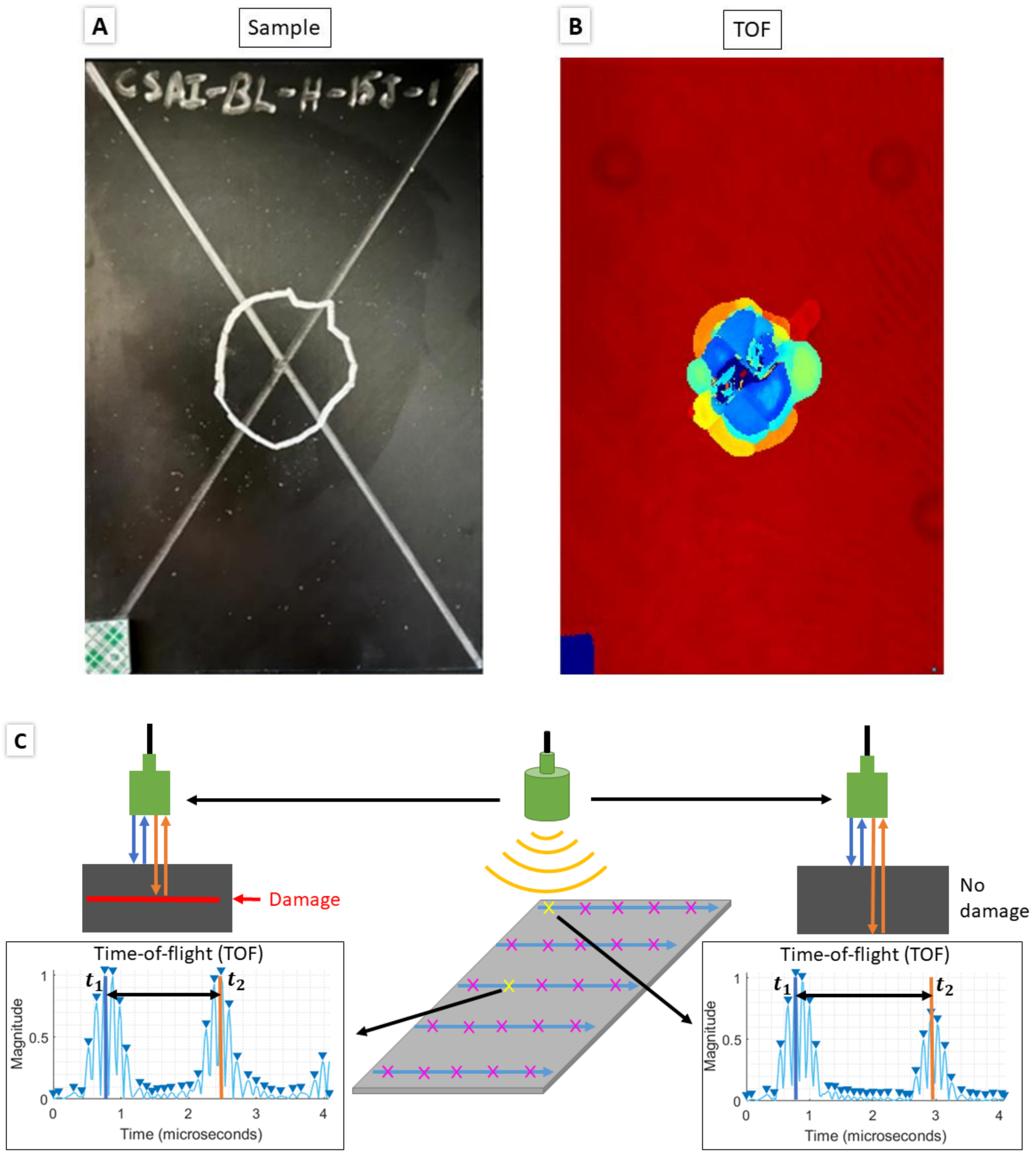


Figure 1.3: (A) Photo of a sample with manually mapped damage region outline, (B) UT C-scan time-of-flight map, (C) C-scan process and calculating TOF for undamaged scan point (right) and damaged scan point (left) using the A-scan signal at each scan point.

1.4 Previous Work

Previous work to process and segment C-scans has been done by another researcher where a gating method was used to calculate the TOF as described by Ellison and Kim [5]. A gate was selected such that the maximum value of the A-scan to the left of the gate is the first peak and the maximum value to the right of the gate is the second peak as shown in Figure 1.4. After the TOF was calculated, the TOF values were segmented by matching each point in the sample to the closest layer. However, when adapting the gating method to process the dataset for this project, the lower sampling frequency resulted in skewed, non-symmetrical signal peaks as shown in Figure 1.5. The skewed peaks caused the resulting TOF to oscillate between values of neighboring data points. In addition, the samples were permanently deformed after impact which caused the first peak location to vary and the TOF value to be calculated incorrectly because of the static gate. Therefore, a different approach was necessary to build a more robust code to account for the different signal characteristics in the test dataset.

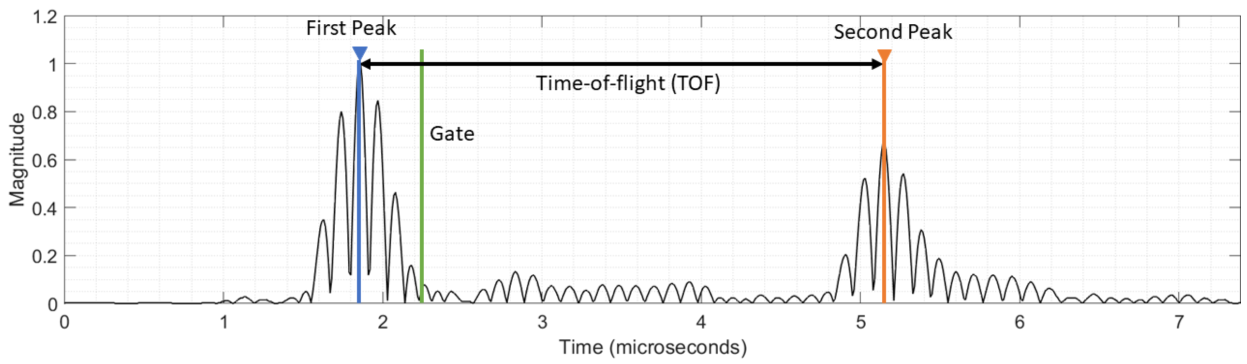


Figure 1.4: A-scan gating procedure.

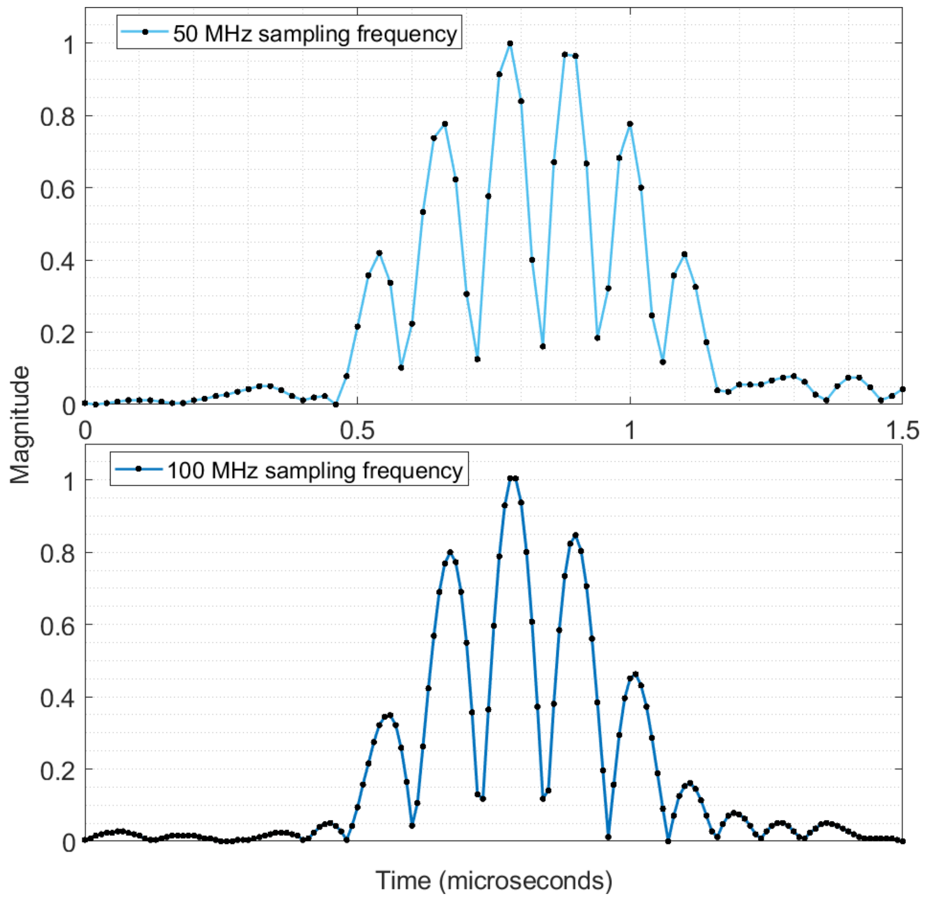


Figure 1.5: A-scan pulses showing low sampling frequency (top) creating asymmetrical peaks compared to high sampling frequency (bottom).

1.5 Literature Review

Approaches to segmentation of C-scans in the existing literature can be categorized in several groups: machine learning, image and morphological processing, clustering, and thresholding as summarized in Wronkowicz et al. [6]. For all of the following methods described, the segmentation was performed on the preprocessed TOF map or image rather than starting from the A-scan data at each point.

In Liu et al. [7], unsupervised machine learning is used to separate the defects from the background. The code was tested on carbon fiber/epoxy samples with artificially generated defects at varying depths using Teflon tape. The shapes of the defects were rectangles, a triangle, a circle, and an irregular blob shape with rough edges, where no shapes shadowed each other through the depth of the sample. The code was able to separate the defects from the background, but the defects were not realistic in shape and did not represent the multi-layered, multi-lobe damage that is typical of actual impact damage in composites.

In Wronkowicz et al. [8] and Wronkowicz and Dragan [11], the authors used the method of image and morphological processing as well as thresholding to separate the background from the damaged region. The background was removed by having the user select a region of interest (ROI), then morphological processing was used to remove noise. The local minima in a histogram of all the color values occurring in the processed TOF image are selected as the segmentation boundaries and the mode value in between the minima is assigned to all the pixel located between the minima. The authors also optionally fill in shadowed regions in lower layers by adding together successive layers, which is an overly simplified assumption for the shape that lower delaminations may take. The assumption does not take into consideration the behavior of impact damaged laminates based on the ply layup. The process is semi-automatic, requiring the user to select a ROI.

In Rodriguez-Hidalgo [9], the authors test five different unsupervised clustering techniques using a composite sample that had been impacted by an unspecified falling object with 3.8 Joules of energy and scanned with a resolution of 1 mm by 1 mm. The results from using the Linkage algorithm (a hierarchical clustering algorithm) and DBSCAN algorithm (Density-Based Spatial Clustering of Application with Noise), showed the best results separating the damage region from the background, but with the very low resolution of the C-scan and the limited sample, it is hard to extrapolate if the clustering methods will extend well to higher resolution C-scans and to composites samples with a variety of energy level impacts.

In the existing body of published work, many of the methods used to remove the undamaged region and segment damaged layers do not utilize information from the A-scan signals and are based on methods that do not always consider the physical meaning of the data being processed. The method developed and demonstrated in this thesis aims to improve upon the methodologies found in the current literature by starting with the A-scan signals at each image point, rather than relying on color-mapped C-scan images produced by other software (e.g., manufacturer's C-scan software).

1.6 Acknowledgements

Figure 1.2, in part, is a reprint of the figure as it appears in Segmentation of X-ray CT and Ultrasonic Scans of Impacted Composite Structures for Damage State Interpretation and Model Generation. Ellison, Andrew, PhD disseration, UC San Diego, 2020. The thesis author received permission to reprint the figure.

CHAPTER 2: EXPERIMENTAL DATASET

2.1 25-Ply UT C-scan Samples

The UT C-scans of the 25-ply samples were collected by a researcher for another project and 26 of the 81 total C-scans from the dataset are used to create and test the code developed in this project. This section gives is a summary of the sample and C-scan information from Romasko [10].

2.1.1 Sample Information

The material used for the samples is unidirectional IM7/977-3 autoclave-cured carbon fiber/epoxy composite. Each sample is 102 mm wide, 152 mm long with the 0° ply aligned in the sample's long dimension direction. The samples are 25 plies thick, 3.3 mm, with three different layups of varying effective modulus as shown in Table 2.1. There are also three sample conditions that were applied to the samples after the C-scans were completed, which are unrelated to this project but are used for giving unique names to each sample, shown in Table 2.2.

The names of the samples used to test the code in this project are listed in Table 2.3. The samples are impacted by a pendulum impactor, shown in Figure 2.1, at 10, 15 and 20 Joules. The samples are fully clamped along all edges and the impact testing is conducted using a modified ASTM D7136 standard.

Table 2.1: Ply Layups and Stiffnesses of Samples

Layup Type	**Ply Layup**	**Effective Modulus (GPa)**
H (hard)	[45/-45/0/0/-45/0/0/45/-45/45/0/0/$\underline{90}$]$_s$	114.0
M (medium)	[45/-45/0/0/45/-45/-/45/-45/0/45/-45/$\underline{90}$]$_s$	79.2
S (soft)	[45-/45/-/45/-45/90/45/-45/0/45/-45/90/$\underline{90}$]$_s$	58.6

Table 2.2: Sample Conditions and Abbreviations.

Sample Condition	**Abbreviation**
Baseline Unrepaired	BL
Surface prepped and repaired	RPR
Contaminated, surface prepped, and repaired	CONT

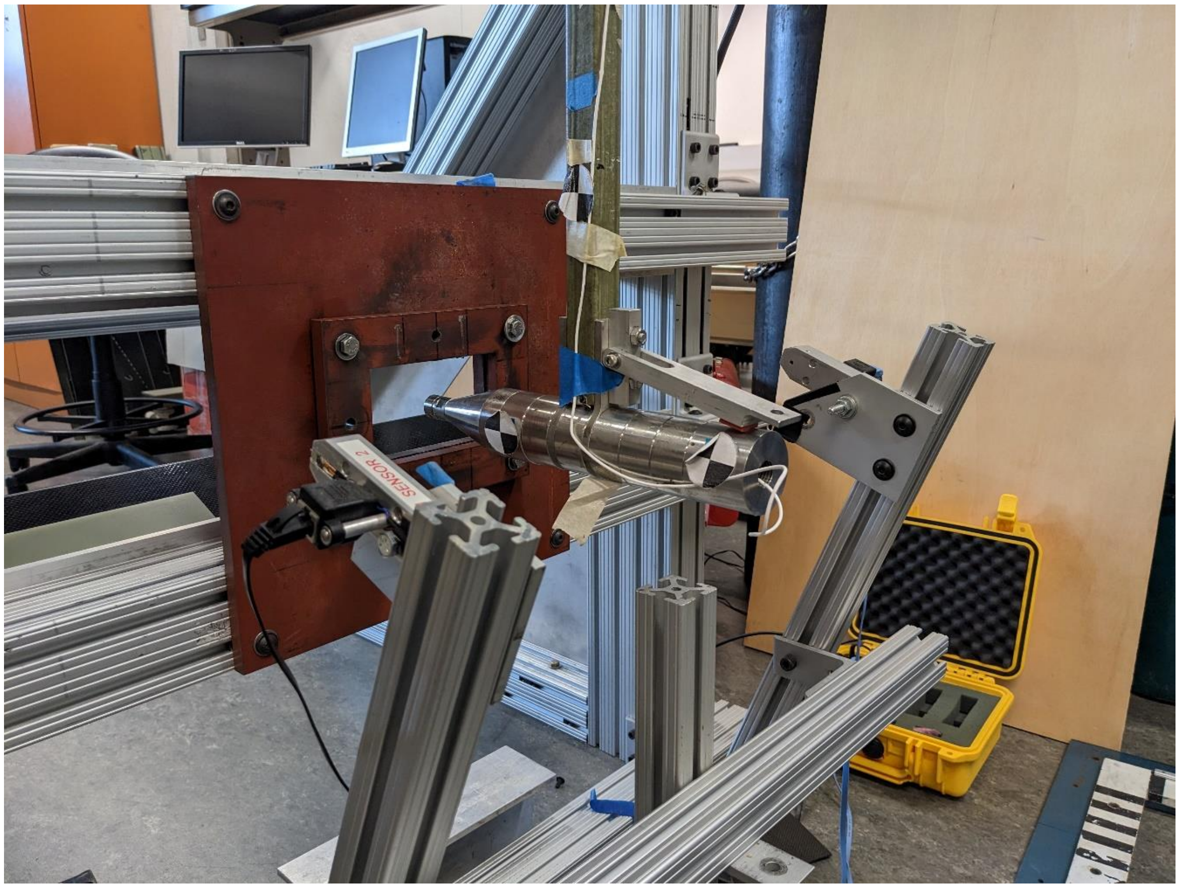


Figure 2.1: Pendulum Impactor.

Table 2.3: Code Testing Dataset Samples.

Sample Name	**Front C-scan**	**Back C-scan**
BL-H-15J-1	X	
CONT-H-10J-2	X	
CONT-H-10J-3	X	
CONT-H-20J-1	X	
CONT-H-20J-2	X	
CONT-H-20J-3	X	
RPR-H-20J-2	X	
RPR-H-20J-3	X	
BL-S-10J-2	X	X
BL-S-15J-2	X	X
BL-S-20J-2	X	X
CONT-S-10J-2	X	X
CONT-S-15J-2	X	X
CONT-S-20J-2	X	X
RPR-S-10J-2	X	X
RPR-S-15J-2	X	X
RPR-S-20J-2	X	X

2.1.2 Pulse-Echo UT C-scan Information

The samples are scanned using a pulse-echo ultrasonic C-scan system inside of an immersion tank filled with de-ionized water. The sample is placed on a set of five standoffs to lift the sample above the bottom of the tank so that the sample is surrounded by water on all sides and remains parallel to the scanning transducer (see Figure 2.2). An NDT System IAHG052 5MHz transducer is used along with a Mistras/NDT Automation 1682 control box and PocketUT system to scan the samples. The samples are scanned with a resolution of 0.4 mm (length) by 0.08 mm (width), a scan speed of 50 mm/s, and sampling frequency of 50 MHz. The setup is shown in Figure 2.2.

[5].

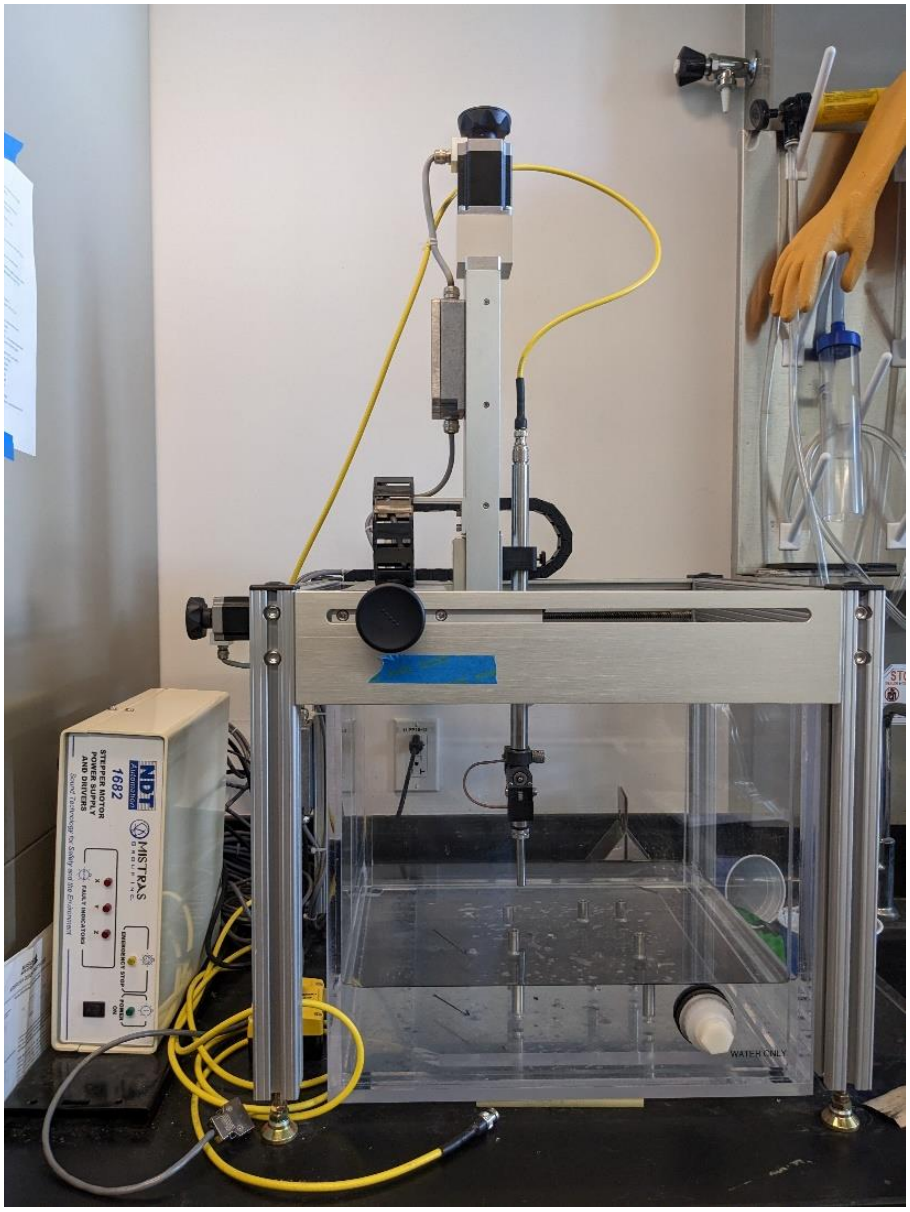


Figure 2.2: Sample in immersion tank and C-scan system.

2.2 24-ply UT C-scan and CT Scanned Sample

The front and back UT C-scans of the 24-ply sample were collected by a researcher for another project and are used to for comparison against the X-ray CT scan of the same sample. This section provides a summary of the sample and C-scan information from Ellison and Kim [5].

2.2.1 Sample Information

The material used for the samples is unidirectional T800S/3900-2B pre-impregnated carbon fiber/epoxy composite. Each sample is 102 mm wide by 154 mm long. The sample is 24 plies thick, 5.0 mm, with the layup $[0/45/90/-45]_{3s}$. The sample was impacted with 48 Joules of energy using the same pendulum impactor setup shown in Figure 2.1.

2.2.2 UT C-scan and CT Scan Information

The UT scan is conducted using the same system as described in Section 2.1.2. The scanning resolution is 0.25 mm by 0.25 mm and is sampled at a frequency of 100 MHz.

The X-ray CT scan is conducted using a NYTEC Inc CT system with a Varian PaxScan 4030E detector array that has a voltage of 70 kV and current of 85μA. The resulting resolution is 13.9 μm/voxel.

CHAPTER 3: CODE DEVELOPMENT

An important goal of the project is to have the code be open source, so the code is published publicly on GitHub and the MathWorks File Exchange for anyone to download, modify as needed for specific applications, and improve functionality. The full code can be found at GitHub (https://github.com/jessica-chan-5/ultrasonic-c-scan-segmentation-for-composites) and on MathWorks File Exchange (https://www.mathworks.com/matlabcentral/fileexchange/125985-ultrasonic-c-scan-segmentation-for-composites). The code is written in MATLAB and requires the MATLAB Parallel Computing, Signal Processing, Curve Fitting, and Image Processing Toolboxes to run. The code was developed in MATLAB 2022b but works for versions 2019b and above.

In the following sections, the functionality of the code will be described in detail. For emphasis, built-in MATLAB functions and custom functions will be indicated by **bold face**. Variable names will be indicated by *italics*.

3.1 Code Overview

The overview of the code structure is shown in Figure 3.1. The code first converts the UT C-scan data input into a 3D matrix in **readcscan**. Then the code processes the C-scan information to find a rectangular box that bounds the damaged region of the sample and calculates the TOF within the damage bounding box in **processcscan**. The code takes the TOF and segments the C-scan into regions of damage using unique characteristics in the A-scan signals to determine when the damage changes from one layer to another in **segcscan**. **plottest** allows the user to plot optional helper figures to adjust input parameters to the previous sections of the code. Once the user has completed adjusting input parameters and rerunning the previous sections, **plotfig** groups the segmented TOF into damage groups that correspond to laminate interfaces where the damage is present in the sample and plots a 3D reconstruction of the damaged region. If there are front and back side C-scans available for sample(s), the user can use **mergecscan** to combine the front and back side damage state information into a single 3D reconstruction of the damage state of the sample. Finally, **plotcustom** plots custom figures that the user may or may not be interested in, specifically a comparison between the unprocessed TOF from **processcscan**, the segmented TOF from **segcscan**, and the UTWin C-scan results.

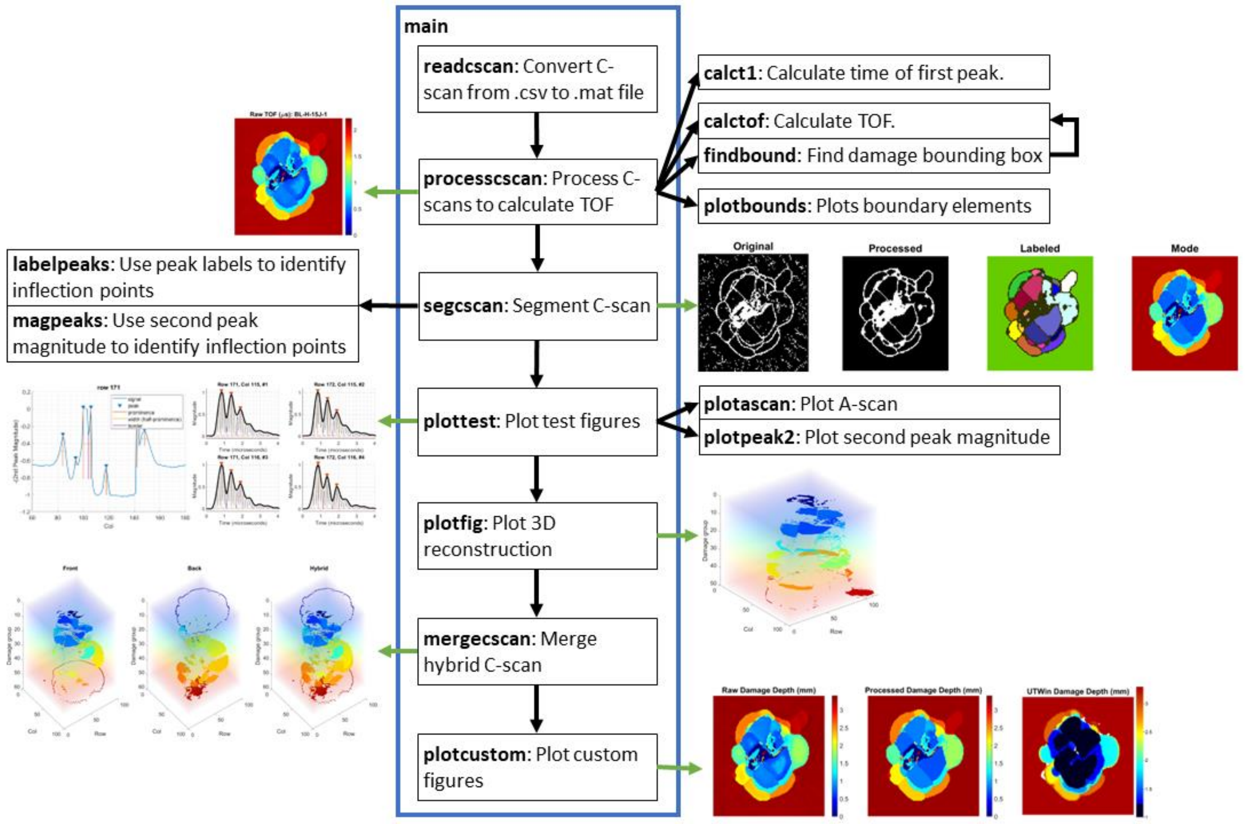
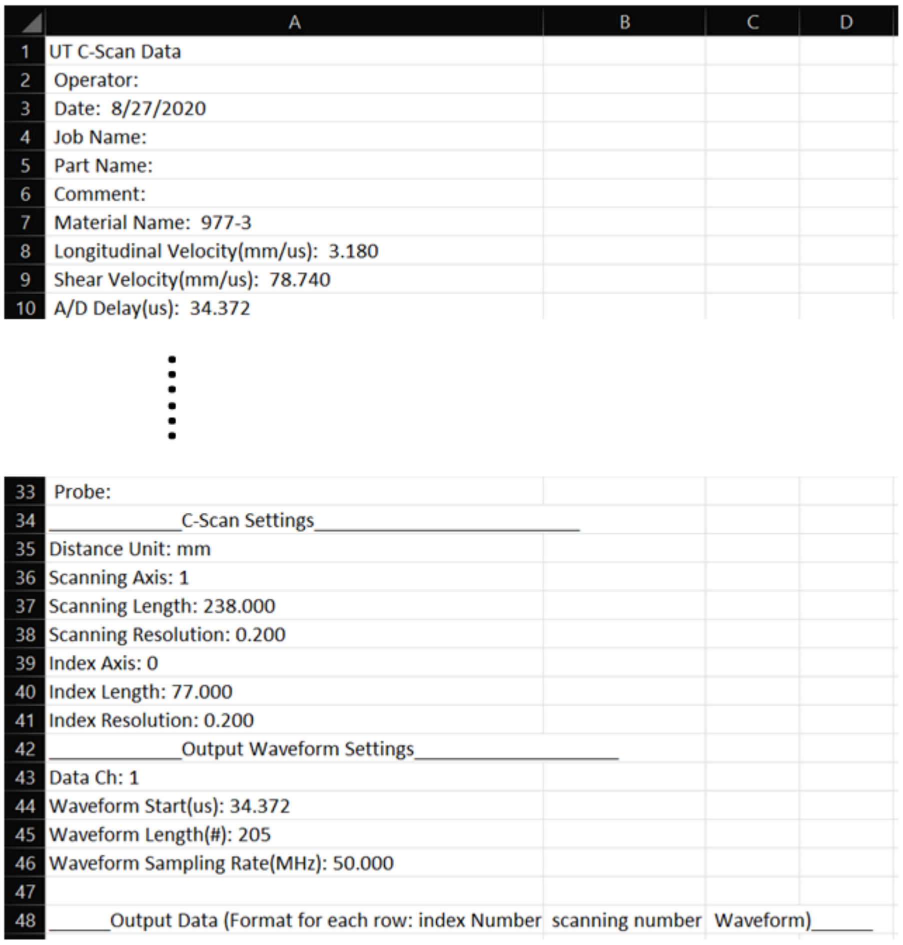

main
readcscan: Convert C-scan from .csv to .mat file
processcscan: Process C-scans to calculate TOF
segcscan: Segment C-scan
plottest: Plot test figures
plotfig: Plot 3D reconstruction
mergecscan: Merge hybrid C-scan
plotcustom: Plot custom figures
calct1: Calculate time of first peak.
calctof: Calculate TOF.
findbound: Find damage bounding box
plotbounds: Plots boundary elements
labelpeaks: Use peak labels to identify inflection points
magpeaks: Use second peak magnitude to identify inflection points
plotascan: Plot A-scan
plotpeak2: Plot second peak magnitude
Original
Processed
Labeled
Mode
Front
Back
Hybrid
Raw Damage Depth (mm)
Processed Damage Depth (mm)
UTWin Damage Depth (mm)

Figure 3.1: Code overview diagram.

To run the code, the user runs the main.m script, which calls in sequence all the functions required to convert the raw output data in character delimited format (.csv, .txt, or .dat) from the user’s ultrasonic C-scan system into TOF information. If front and back side C-scans are available, the code can combine the information from both scans to form a hybrid C-scan. The code also produces a variety of figures to visualize the data and saves data produced in intermediate steps that may be of interest to the user. Intermediate data saved include the time at which the first peak occurs across the sample, an outline of the overall damaged area, and damage layer groups describing damaged areas at varying depths in the sample. The code is designed to batch process large numbers of ultrasonic C-scan samples at a time. The parallel for loop function in MATLAB is implemented in the main.m script and **calctof** function to help speed up processing time for batches of large files. When testing, the user can use the test.m file in place of the main.m file, where the parallel for loops wrapping functions that plot helper figures are changed to normal for loops to allow figures to be visible, as figures plotted inside of parallel for loops do not show even when figure visibility is set to on.

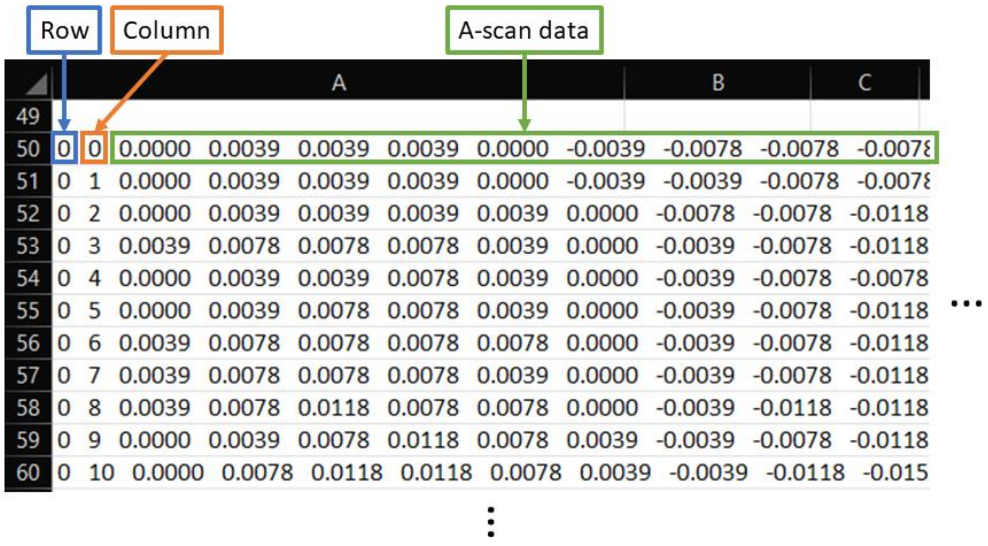
3.2 File Input Requirements

The input file type from the ultrasonic C-scan system must be a character delimited file such as .csv, .txt or .dat. Header information at the beginning of the file, such as shown in Figure 3.2, is acceptable and will be ignored. A few example rows from a .csv file are shown in Figure 3.3 where the input file must be formatted with each line containing in order: (1) row number, (2) column number, and (3) A-scan data for the corresponding point on the sample.



	A	B	C	D
1	UT C-Scan Data			
2	Operator:			
3	Date: 8/27/2020			
4	Job Name:			
5	Part Name:			
6	Comment:			
7	Material Name: 977-3			
8	Longitudinal Velocity(mm/us): 3.180			
9	Shear Velocity(mm/us): 78.740			
10	A/D Delay(us): 34.372			
⋮				
33	Probe:			
34	____________C-Scan Settings________________________			
35	Distance Unit: mm			
36	Scanning Axis: 1			
37	Scanning Length: 238.000			
38	Scanning Resolution: 0.200			
39	Index Axis: 0			
40	Index Length: 77.000			
41	Index Resolution: 0.200			
42	____________Output Waveform Settings__________________			
43	Data Ch: 1			
44	Waveform Start(us): 34.372			
45	Waveform Length(#): 205			
46	Waveform Sampling Rate(MHz): 50.000			
47				
48	_____Output Data (Format for each row: index Number	scanning number	Waveform)_____	

Figure 3.2: File header information.



	A							B	C
49									
50	0 0 0.0000	0.0039	0.0039	0.0039	0.0000	-0.0039	-0.0078	-0.0078	-0.007
51	0 1 0.0000	0.0039	0.0039	0.0039	0.0000	-0.0039	-0.0039	-0.0078	-0.007
52	0 2 0.0000	0.0039	0.0039	0.0039	0.0039	0.0000	-0.0078	-0.0078	-0.0118
53	0 3 0.0039	0.0078	0.0078	0.0078	0.0039	0.0000	-0.0039	-0.0078	-0.0118
54	0 4 0.0000	0.0039	0.0039	0.0078	0.0039	0.0000	-0.0039	-0.0078	-0.0078
55	0 5 0.0000	0.0039	0.0078	0.0078	0.0039	0.0000	-0.0039	-0.0078	-0.0118
56	0 6 0.0039	0.0078	0.0078	0.0078	0.0078	0.0000	-0.0039	-0.0078	-0.0118
57	0 7 0.0039	0.0078	0.0078	0.0078	0.0039	0.0000	-0.0039	-0.0078	-0.0118
58	0 8 0.0039	0.0078	0.0118	0.0078	0.0078	0.0000	-0.0039	-0.0118	-0.0118
59	0 9 0.0000	0.0039	0.0078	0.0118	0.0078	0.0039	-0.0039	-0.0078	-0.0118
60	0 10 0.0000	0.0078	0.0118	0.0118	0.0078	0.0039	-0.0039	-0.0118	-0.015

Figure 3.3: Input file format.

3.3 Primary Script – main.m

Editing and running the main.m script is the primary way the user interacts with the code. The script gathers all the function inputs in one location and allows the user to control which section and which files the code runs and analyzes. The script is broken up into sections for readability and usability. The structure of the sections is shown in Figure 3.4 and is broken up into three different section groups: (1) Processing Setup, Sections i through iii, (2) Function Inputs, Sections A through G, and (3) Run Functions, Sections 1 through 7.

In the Processing Setup sections group, Sections i through iii, the code first clears the workspace in Section i by closing all figures and clearing variables. The user enters the file names of samples to be processed as a string array in Section ii. Section iii controls which Sections 1 through 7 are selected to run and which files are processed in each section. Additionally, for **processcscan**, **segcscan**, **mergecscan**, and **plotcustom**, there is an additional option to show or hide helper figures that assist with adjusting function inputs. The flexibility from the options in Section iii is useful when adjusting function inputs or testing a sample before batch processing the full list of samples. For example, when adjusting function inputs, the user can set one function to run on a single sample with the helper figures set to visible. Later, when the user finishes adjusting function inputs, the user can change the function options so that all sections run and all files are processed during batch processing, as shown in Figure 3.5. It should be noted that **mergecscan** has a differing method of designating indices of files. If a front side scan is named "sample-name," the back side scan should be named "sample-name-*back*," so only the indices corresponding to the file names for the front side C-scans with corresponding back side C-scans should be included.

In the Function Input sections group (see Figure 3.4), Sections A through G, inputs are organized by function in the order they appear in Sections 1 through 7 for easy access and quick adjustments by the user. The details of inputs for each function will be described in the respective section.

In each of the Run Functions sections group (see Figure 3.4), Sections 1 through 7, the section's function is called using the inputs from above, for the requested file indices. Progress messages are displayed in the Command Window letting the user know which functions are being run, the name and index of the file(s) currently being processed, and when complete, the total amount of time the function took to process the requested files. An example of a section, Section 1, is shown in Figure 3.6 and the corresponding progress message is shown in Figure 3.7.

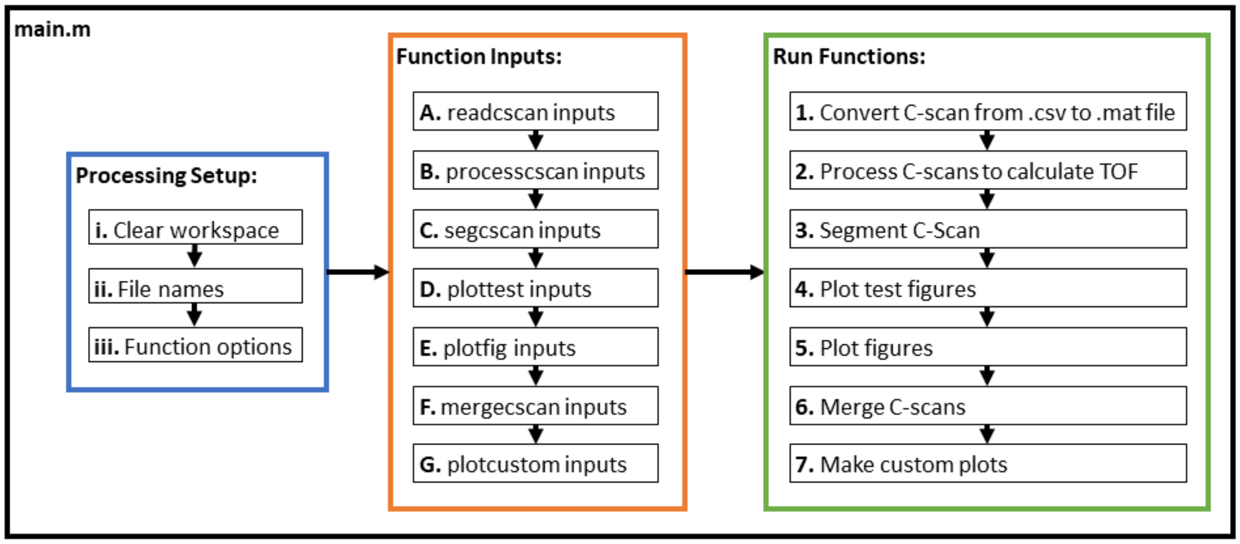
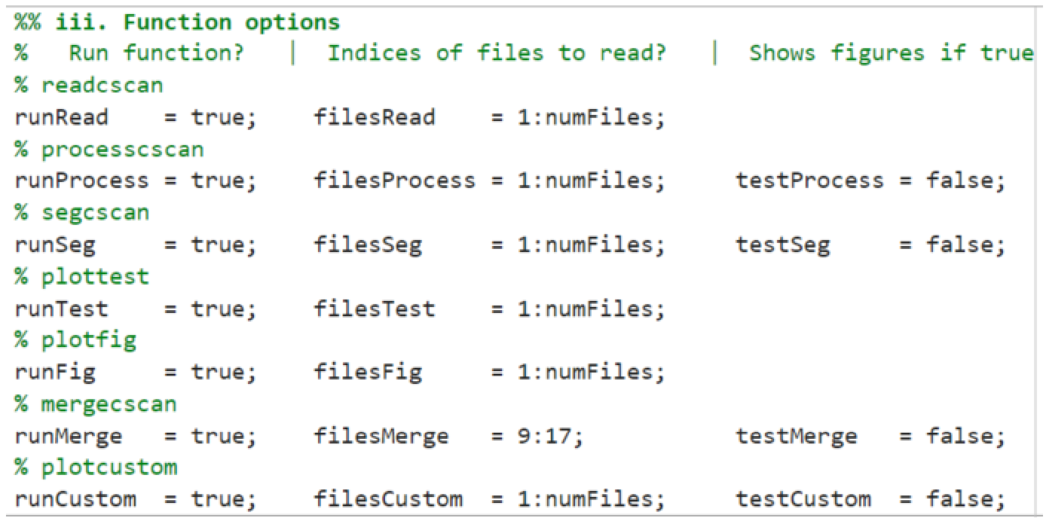
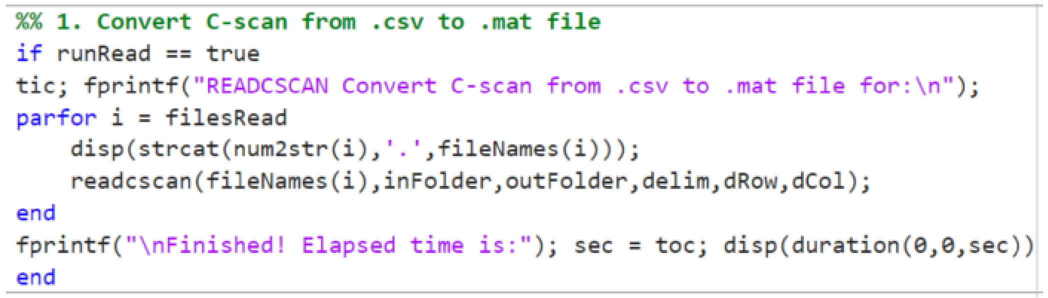


Figure 3.4: main.m script section structure.



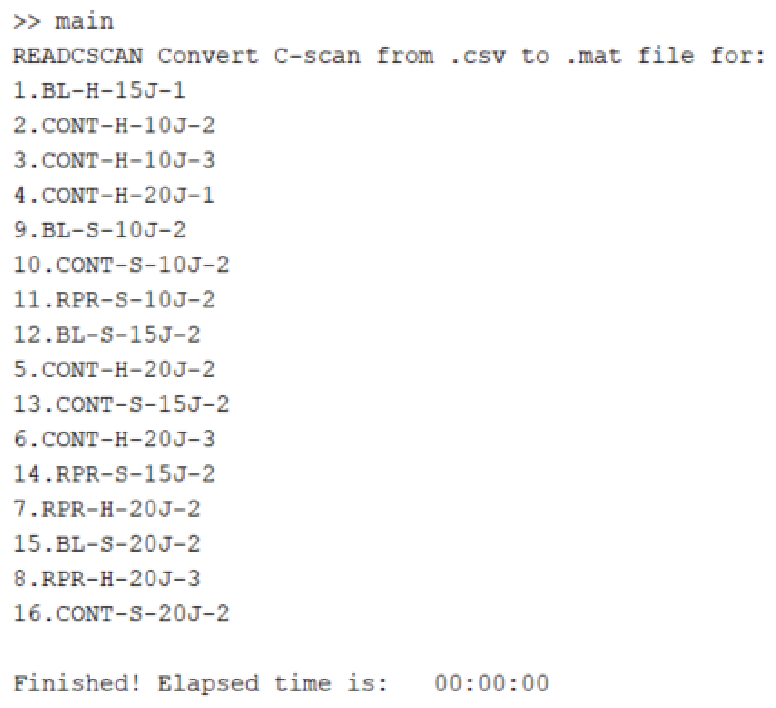
```
%% iii. Function options
%   Run function?    |  Indices of files to read?    |  Shows figures if true
% readcscan
runRead     = true;    filesRead    = 1:numFiles;
% processcscan
runProcess = true;    filesProcess = 1:numFiles;     testProcess = false;
% segcscan
runSeg     = true;    filesSeg     = 1:numFiles;     testSeg     = false;
% plottest
runTest    = true;    filesTest    = 1:numFiles;
% plotfig
runFig     = true;    filesFig     = 1:numFiles;
% mergecscan
runMerge   = true;    filesMerge   = 9:17;           testMerge   = false;
% plotcustom
runCustom  = true;    filesCustom  = 1:numFiles;     testCustom  = false;
```

Figure 3.5: main.m Section iii, function options.



```
%% 1. Convert C-scan from .csv to .mat file
if runRead == true
tic; fprintf("READCSCAN Convert C-scan from .csv to .mat file for:\n");
parfor i = filesRead
    disp(strcat(num2str(i),'.',fileNames(i)));
    readcscan(fileNames(i),inFolder,outFolder,delim,dRow,dCol);
end
fprintf("\nFinished! Elapsed time is:"); sec = toc; disp(duration(0,0,sec))
end
```

Figure 3.6: Run Function section example, Section 1.



```
>> main
READCSCAN Convert C-scan from .csv to .mat file for:
1.BL-H-15J-1
2.CONT-H-10J-2
3.CONT-H-10J-3
4.CONT-H-20J-1
9.BL-S-10J-2
10.CONT-S-10J-2
11.RPR-S-10J-2
12.BL-S-15J-2
5.CONT-H-20J-2
13.CONT-S-15J-2
6.CONT-H-20J-3
14.RPR-S-15J-2
7.RPR-H-20J-2
15.BL-S-20J-2
8.RPR-H-20J-3
16.CONT-S-20J-2

Finished! Elapsed time is:   00:00:00
```

Figure 3.7: Example progress message from Section 1.

3.4 Read C-scan – readcscan

The inputs for **readcscan** include *inFolder* and *outFolder*, which are also used in other functions. In general, any inputs that are shared between different functions will appear in the first function where it is used and not repeated in the Function Input section for the rest of the functions. *inFolder* is the name of the folder containing the character delimited C-scan files. *outFolder* is the name of the folder to save intermediate and final variables, most often matrices, produced by functions in .mat format. *inFolder* and *outFolder* should both be created by the user prior to running the code in the same folder as the main.m script. Input specific to **readcscan** are *delim*, *dRow,* and *dCol*. *delim* is the field delimiter character used in the C-scan files. *dRow* and *dCol* indicate which ith row should be saved after the C-scan file is read. For example, if *dRow* is equal to 1, then every 1st row will be saved, or in other words every row will be saved. If *dRow* is instead equal to 5, then every 5th row will be saved. Down sampling is necessary at this point if the C-scan resolution along both directions is not equal. The rest of the code assumes an equal C-scan resolution along the row and column direction.

readcscan uses the built-in MATLAB function, **readmatrix**, to read the character delimited C-scan file and discards any non-numeric data such as headers. The function takes the absolute value of the A-scan signal at each point, as shown in Figure 3.8, removes the row and column information at the beginning of each row in the original file, and reshapes the C-scan into a 3D matrix with the dimensions: (1) number of C-scan rows, (2) number of C-scan columns, and (3) number of points in each A-scan. **readcscan** saves the C-scan 3D matrix as a .mat file with the file path, “*outFolder*\cscan\sampleName-cscan.mat.” All other variables saved in the code follow the same format, where the variable name matches the subfolder in *outFolder* and is appended to the sample name.

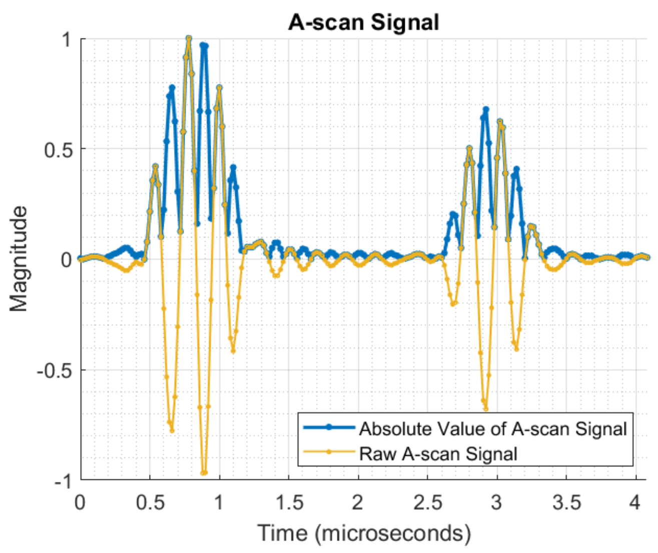


Figure 3.8: Raw and absolute value of A-scan signal

3.5 Process C-scan – processcscan

The inputs for **processcscan** include *figFolder* which is also used for the rest of the functions. *figFolder* is the name of the folder where figures created in each function will be saved in .fig and/or .png file formats. *dt* is the sampling period in microseconds of the C-scan receiver. *bounds*, *incr*, *baseRow*, *baseCol*, *cropThresh*, and *pad* are inputs that define where and how the function searches for a box bounding the damaged area in each sample. *minProm1*, *noiseThresh*, and *maxWidth* are inputs that control how time-of-flight (TOF) values are calculated at each point. If *calcT1* is true, the function saves the time at which the first peak appears. Finally, *res* is an input used for figures created by all functions that sets image resolution in dots per inch (dpi) for .png figures saved to *figFolder*.

3.5.1 Damage Bounding Box

processcscan first loads the C-scan 3D matrix saved by **readcscan** in *outFolder*. The function then uses the *bounds*, *incr*, *baseRow*, *baseCol*, *cropThresh*, and *pad* inputs to systematically search for the sample's damage bounding box. *bounds* can be used to exclude features that are not of interest, but that may be erroneously detected as damage, such as standoffs used to hold the sample in the C-scan tank or foam tape used to label the orientation of samples, as shown in Figure 3.9A and B. *bounds* defines the x and y coordinates of the search area in the format, [startX endX startY endY]. The function searches for the damage bounding box within the given coordinates and ignores the region outside. An example of a bounding box is shown in red in Figure 3.9C.

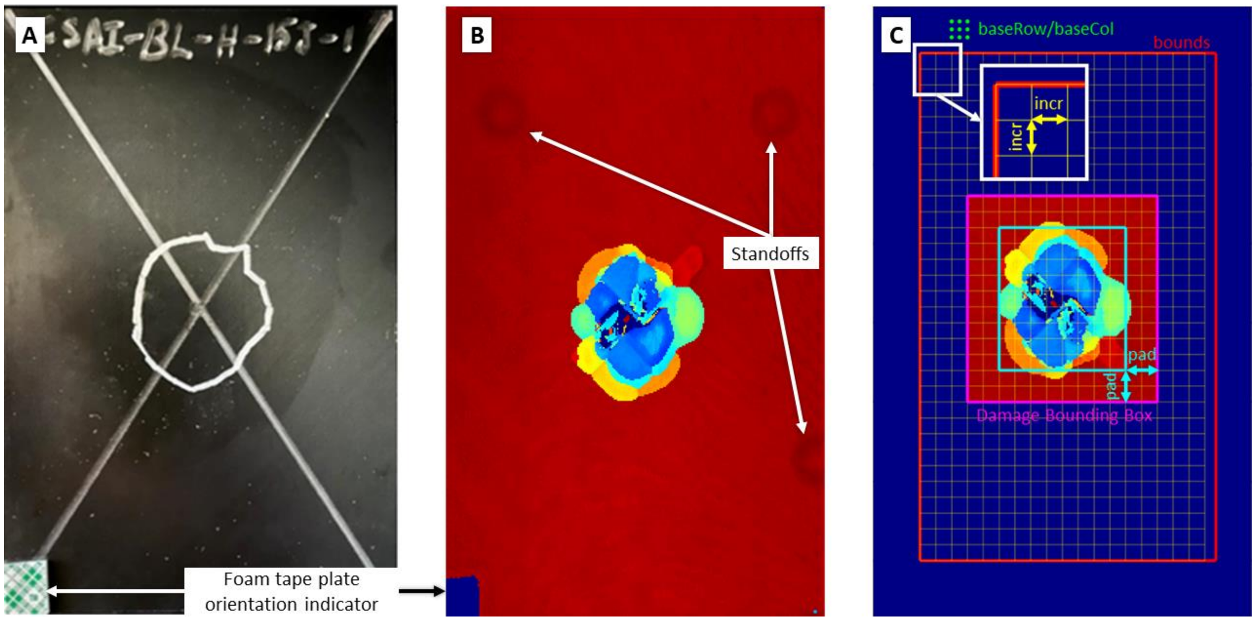


Figure 3.9: Searching for damage bounding box example. (A) Photo of sample showing manually mapped damage, (B) C-scan of damage, (C) Damage bounding box search input parameters.

The function then calculates a square grid of search points within the search area defined by *bounds*, using the spacing indicated by *incr*. If the length and/or width of the search box is not wholly divisible by *incr*, then the search grid (yellow) will stop short of the search box (red) as is the case for the sample in Figure 3.9C. A baseline TOF value is calculated using a grid of points defined by *baseRow* and *baseCol* for comparison later. *baseRow* and *baseCol* should be picked in an area that is expected to be undamaged as shown in Figure 3.9C as a grid of green dots.

The damage bounding box search process begins by defining the horizontal centerline of the sample and calculating TOF values at each grid search point vertically from the top and the bottom towards the centerline, as shown in Figure 3.10A. The search along each vertical stops when the TOF value at the grid search point differs from the calculated baseline TOF by more than the *cropThresh*, and is saved as a start or end row candidate, as shown in Figure 3.10B. The row of outermost start and end row candidates are set as the *startRow* and *endRow* for the sample. The damage bounding box search continues by defining the vertical centerline for the sample and calculating TOF values at each grid search point horizontally from the left and right towards the centerline, but only along rows between the *startRow* and *endRow*, as shown in Figure 3.10C. Similar to the vertical direction, the search stops when the TOF value at a grid point differs from the baseline TOF more than the *cropThresh*, and is saved as a start or end column candidate, as shown in Figure 3.10D. The outermost start and end column candidates are saved as the *startCol* and *endCol* for the sample.

The function moves the *startRow*, *endRow*, *startCol*, and *endCol* locations outwards by a (1+*pad*) factor of *incr*. In the example shown in Figure 3.9C, *pad* is equal to one, so the damage box boundaries are moved outwards by (2×*incr*). If *pad* is equal to zero, the boundaries would be moved out by a factor of (1×*incr*). *pad* can be adjusting according to the sample's characteristics and user preference. A lower value will create a tighter damage bounding area and vice versa.

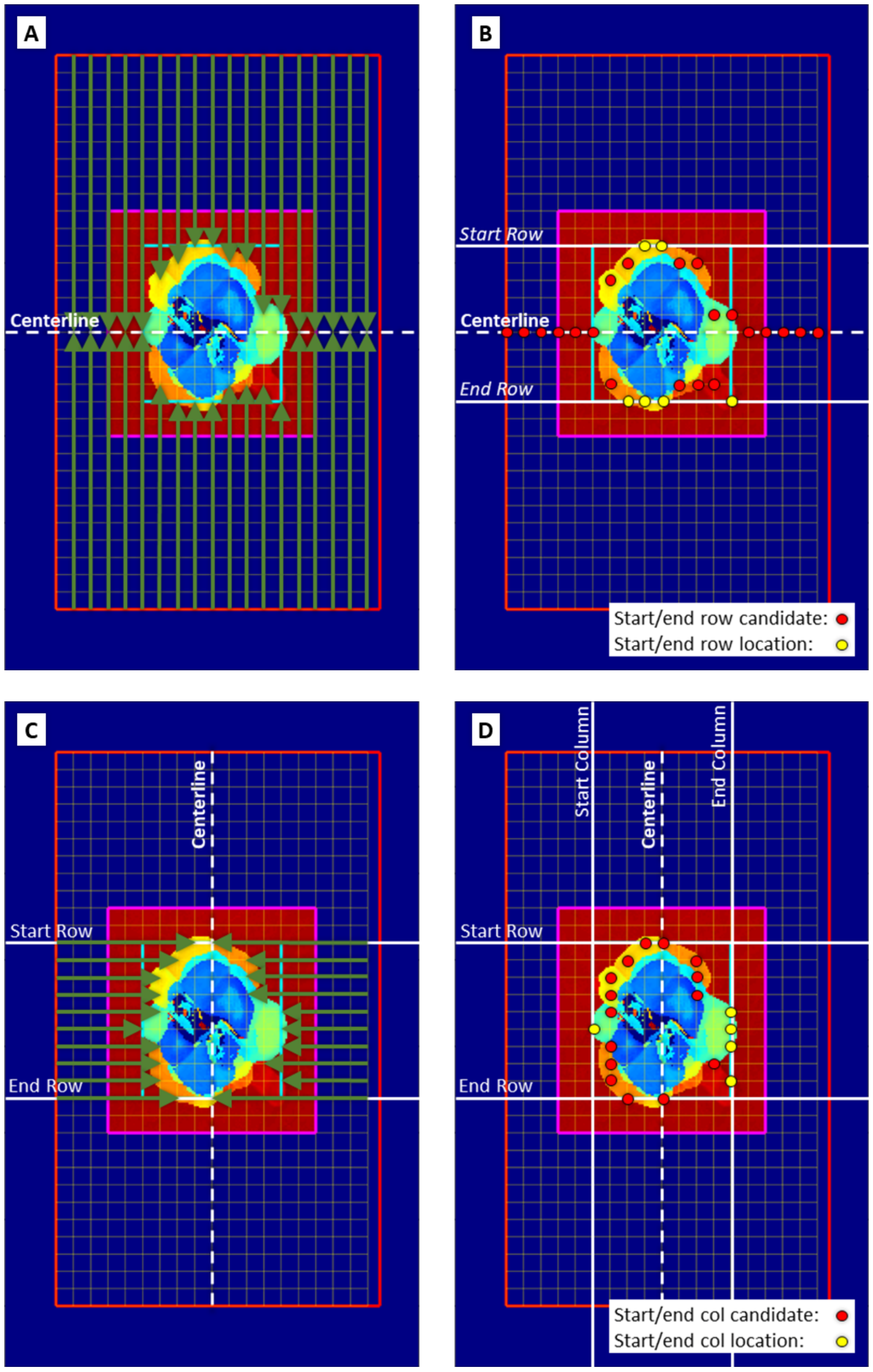


Figure 3.10: Damage bounding box search process. (A) Search along columns, (B) picking start row and end row indices, (C) search along rows, (D) picking start and end column indices.

3.5.2 Calculate TOF – calctof

The helper function **calctof** calculates TOF only for points within the damage bounding box to decrease code run time. For each point in the bounding box, the first step is to use the built in MATLAB function, **findpeaks** (part of the MATLAB Signal Processing Toolbox) to find all peaks in the A-scan signal, which is shown for an example A-scan in Figure 3.11A. The next step is to fit a smoothing spline to the peaks using the built in MATLAB function, **fit** (part of the MATLAB Curve Fitting Toolbox) to interpolate points between the found peaks, which is shown in Figure 3.11B. Of the fitting options available in the **fit** function, smoothing splines created the best fit. The final step is to use the **findpeaks** function again, this time to find peaks in the spline fit and applying the *minProm1* input to ignore any peaks below a prominence level that occur because of a small fluctuation in data. Prominence is a measure by **findpeaks** of how much a peak stands out because of its height and location relative to neighboring peaks. Two examples of peaks that would not be detected when an appropriate *minProm1* is used are t_A and t_B in Figure 3.11C. The two peaks that would be detected would be are t_1 and t_2, the first and second peak respectively. *minProm1* may be different from dataset to dataset because of the characteristics of the A-scan signals. In Section 4 (see Figure 3.4), **plottest** plots figures to assist in adjusting *minProm1* as well as other inputs. Finally, the TOF of the point is calculated by taking the difference between t_2 and t_1, where t_2 is the time of occurrence of the peak with the highest magnitude, excluding the first peak.

Once finished looping through all points in the bounding box, **calctof** returns the raw TOF, the magnitude and time of occurrence of all peaks found at each point, the number of peaks at each point, and whether the width of the first peak is greater than the input *maxWidth*. If only one peak is found at an A-scan point, the TOF is considered too small to resolve and is set to zero. All of this information will be used in the next section to process the raw TOF.

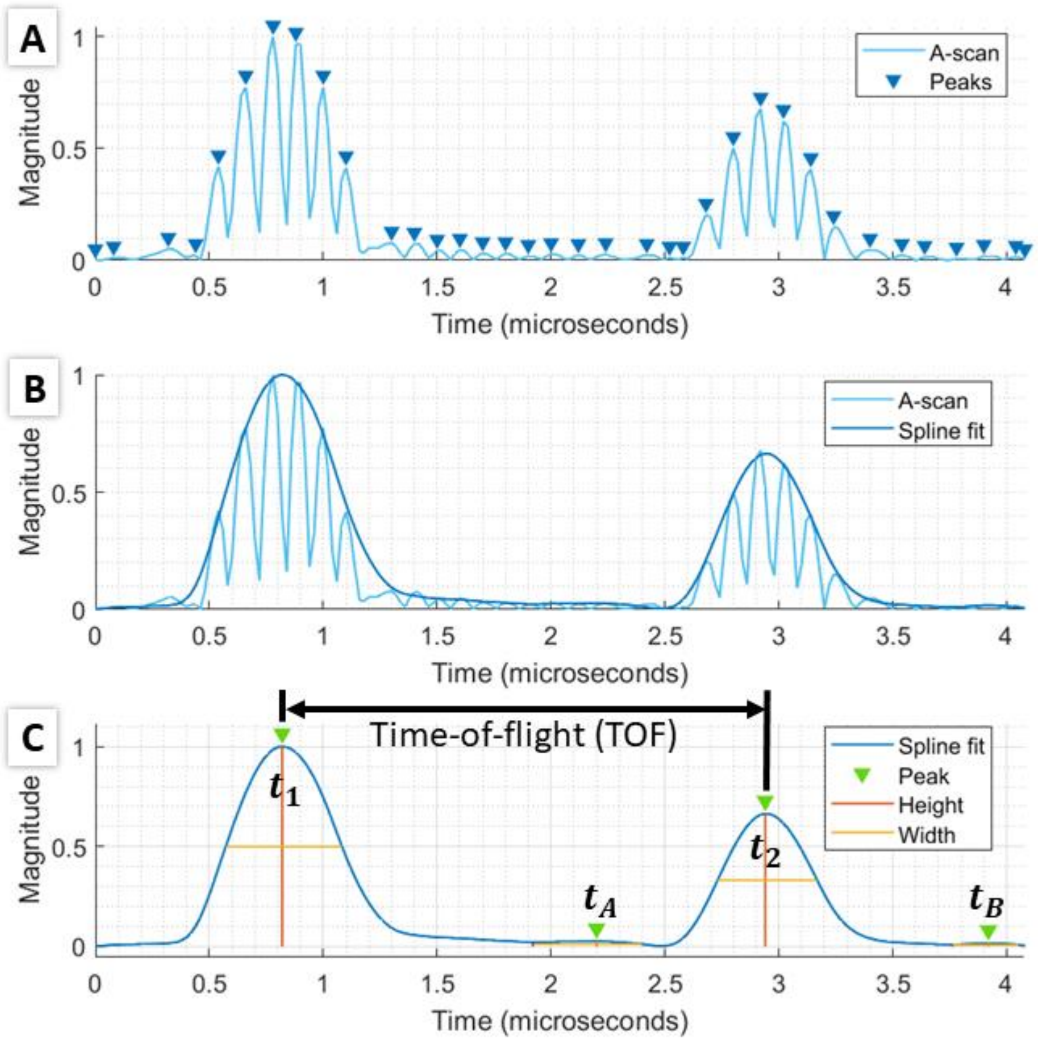


Figure 3.11: TOF calculating process. (A) Finding peaks, (B) Fitting curve to peaks, (C) Calculating TOF.

As an optional output, if the user is interested in creating a dent depth map in addition to calculating TOF, the user can set *calcT1* equal to true. **processcscan** will call helper function **calct1**, which in turn calls **calctof**, but instead of just processing the damage bounding box area, it will process the whole sample. **calct1** extracts the time of occurrence of the first peak from each A-scan point and then plots the raw TOF and time of the first peak for the whole sample, side-by-side, as shown in Figure 3.12. The first peak information, *t1*, is saved by **calct1** in *outFolder* in the subfolder t1 and can be processed to create a dent depth map of the sample.

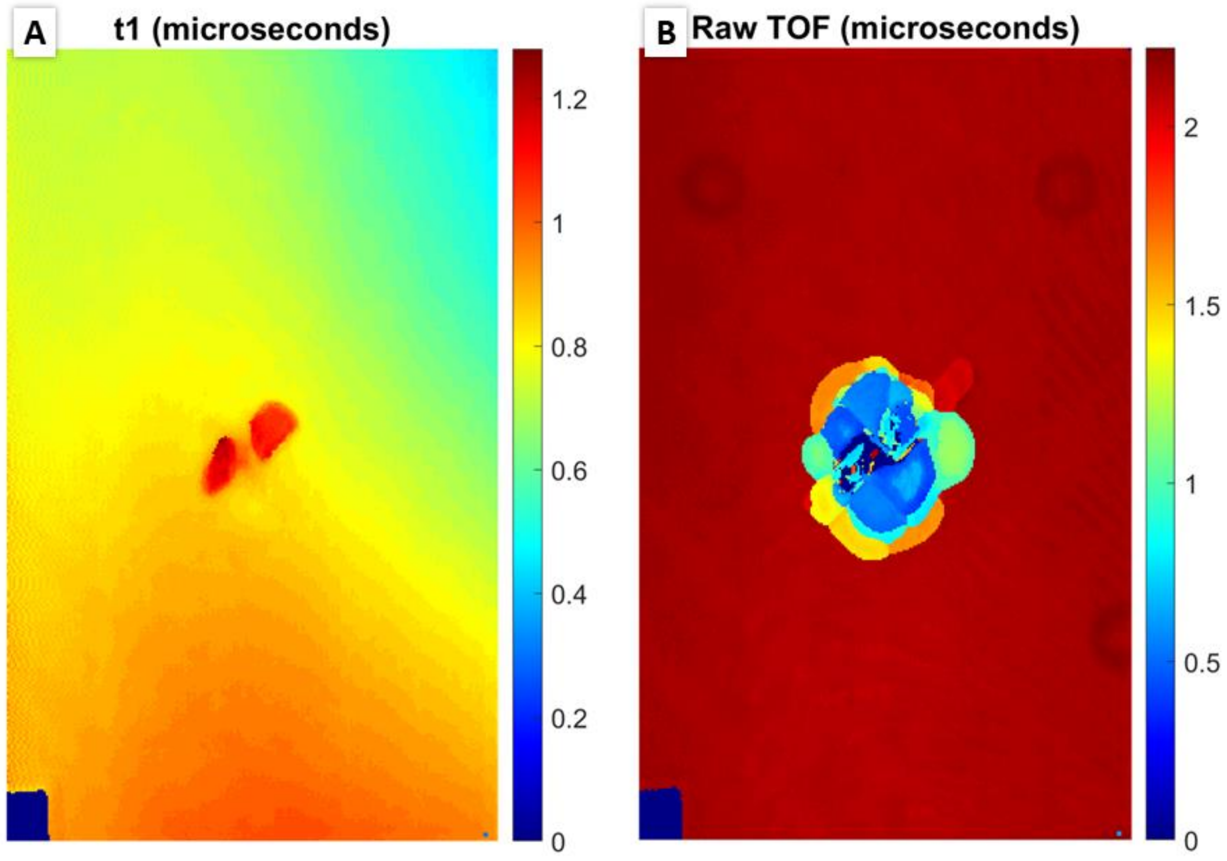


Figure 3.12: (A) Time of first peak, (B) raw TOF.

3.5.3 Figures and Output Variables

processcscan can produce up to four figures:

(1) side-by-side time of first peak and raw TOF, Figure 3.12,

(2) damage bounding box inputs visualization, Figure 3.9, without the text annotations,

(3) a queryable plot of raw TOF, Figure 3.12A, and

(4) a non-queryable plot of raw TOF, Figure 3.12B.

All figures are saved in *figFolder* and are in subfolders called t1, damBoundBox, rawTOFquery, and rawTOF respectively.

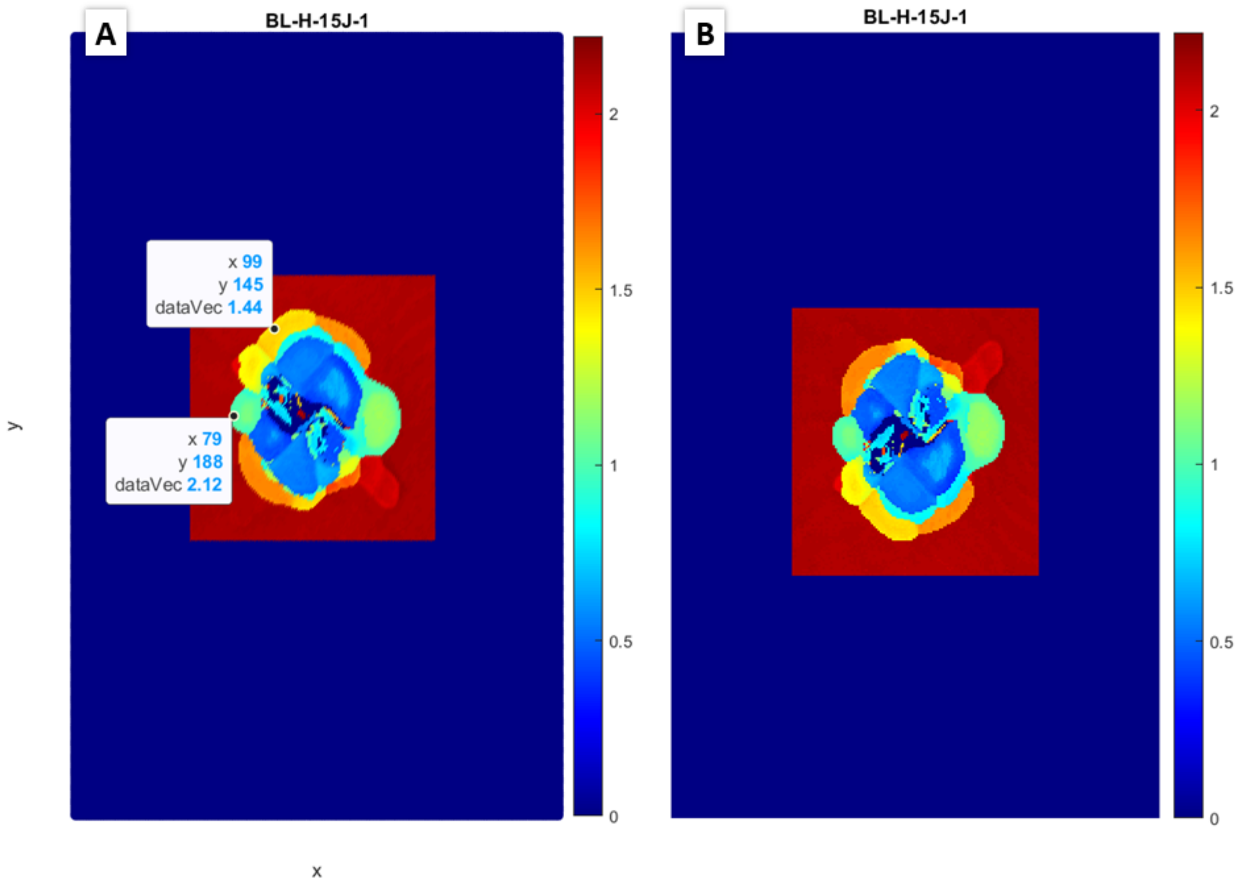


Figure 3.13: (A) Queryable raw TOF plot, (B) Non-queryable raw TOF plot.

The queryable plot of raw TOF is a **scatter** plot superimposed on raw TOF displayed as an image using **imshow** as shown in Figure 3.13A. The figure is saved as a reference for following sections of the code where it is useful to be able to query the coordinates and raw TOF values when adjusting inputs. The non-queryable raw TOF figure displays the raw TOF using **imshow** and is saved as a separate file from the queryable version because the **scatter** function causes the image to be mirrored across x-axis and the scatter plot markers at each A-scan point are visible, which distorts the image. The coordinates in the queryable figure are accurate despite the mirroring effect. It should be noted that points outside of the damage bounding box that are not processed by **calctof** are set to zero and shown in dark blue.

processcscan saves six variables, in subfolders in *outFolder* with names matching the variable names:

(1) raw TOF, *rawTOF*,

(2) magnitude of all peaks found at each A-scan point, *peak*,

(3) time of occurrence of all peaks found at each A-scan point, *locs*,

(4) A-scan point locations where the first peak was wider than *maxWidth*, *wide*,

(5) the number of peaks found at each A-scan point, *nPeaks*, and

(6) the damage bounding box coordinates, *cropCoord*.

These variables will be used in the following sections to process the raw TOF.

3.6 Segment C-scan – segcscan

The inputs for **segcscan** include *minProm2*, *peakThresh*, *modeThresh*, and *seEl*. *minProm2* and *peakThresh* are used in the processes of finding locations at which there is a layer change, where a layer change is defined as when the damage area depth location changes from one layer to another. *modeThresh* and *seEl* are used in the processes of segmenting the damage areas into separate layers. A diagram summarizing **segcscan** is shown in Figure 3.14.

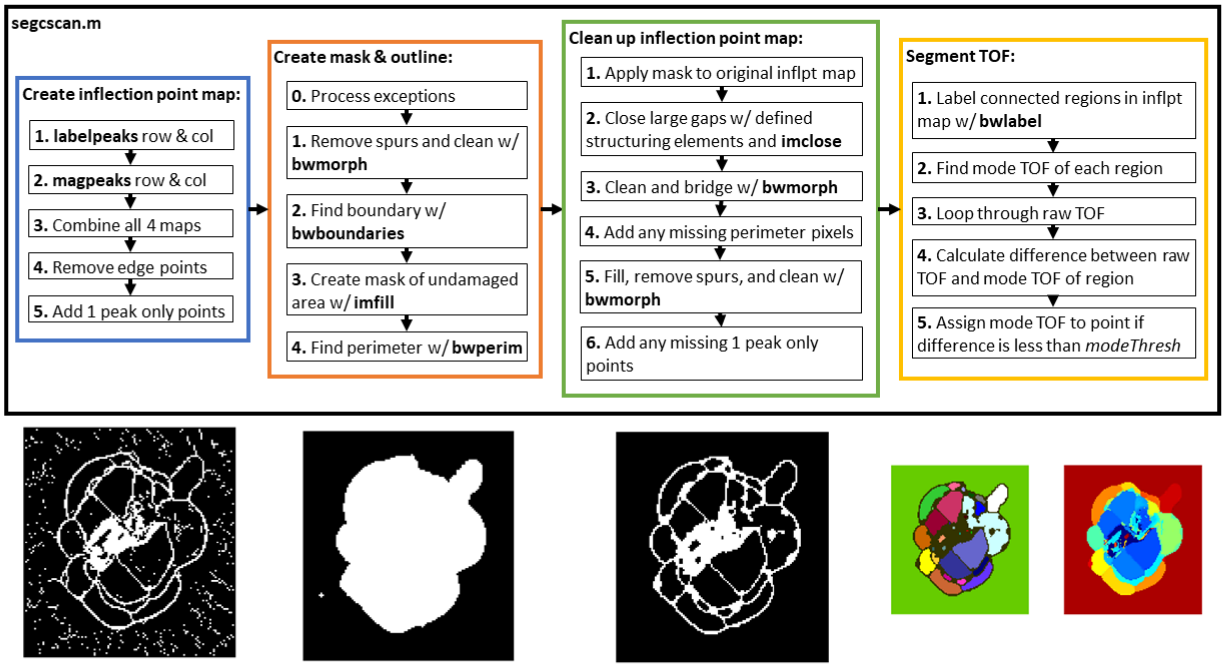

segcscan.m
Create inflection point map:
1. labelpeaks row & col
2. magpeaks row & col
3. Combine all 4 maps
4. Remove edge points
5. Add 1 peak only points
Create mask & outline:
0. Process exceptions
1. Remove spurs and clean w/ bwmorph
2. Find boundary w/ bwboundaries
3. Create mask of undamaged area w/ imfill
4. Find perimeter w/ bwperim
Clean up inflection point map:
1. Apply mask to original inflpt map
2. Close large gaps w/ defined structuring elements and imclose
3. Clean and bridge w/ bwmorph
4. Add any missing perimeter pixels
5. Fill, remove spurs, and clean w/ bwmorph
6. Add any missing 1 peak only points
Segment TOF:
1. Label connected regions in inflpt map w/ bwlabel
2. Find mode TOF of each region
3. Loop through raw TOF
4. Calculate difference between raw TOF and mode TOF of region
5. Assign mode TOF to point if difference is less than modeThresh

Figure 3.14: **segcscan** code diagram.

segcscan starts by loading variables saved at the end of **processcscan**: *rawTOF*, *peak*, *locs*, *wide*, *nPeaks*, and *cropCoord* from their respective subfolders in *outFolder*. The function then calls the helper functions **labelpeaks** and **magpeaks** to find locations of layer changes. The point at which a layer change occurs will be referred to as an inflection point in this thesis.

3.6.1 Change in Peak Labels – labelpeaks

First, **labelpeaks** gives labels to all peaks found at each A-scan point and checks for layer changes. An example is shown in Figure 3.15A, where the point locations in the sample are highlighted. Figure 3.15B shows the spline fit of the A-scan at each point along with the peaks and their labels from **labelpeaks**. From Column 95 to 96, there is no change in the peak labels because the change in peak location between the two points for all the peaks is less than *peakThresh*, which is 0.04 microseconds in this example. From Column 96 to 97, peaks 1, 2, and 4 do not change, but peaks 3 and 5 disappear because their prominence decreased below the *minProm1* in **processcscan** and are no longer detected. At this point, the peak labels 3 and 5 are recycled and used to label future peaks that appear. There is a layer change detected between Column 96 and 97 because the peak label of the second peak changes from 3 (Col 96) to 2 (Col 97). From Column 97 to 98, the second peak retains the same peak label, 2, because the change in location is less than the *peakThresh*. From Column 98 to 99, a new peak is labeled, 3, and peak 4 disappears because its prominence has decreased to below *minProm1*. There is a layer change detected between Column 98 and 99 because the peak label of the second peak changes from 2 to 3. From Column 99 to 100, there is no change in peak labels and no layer change detected.

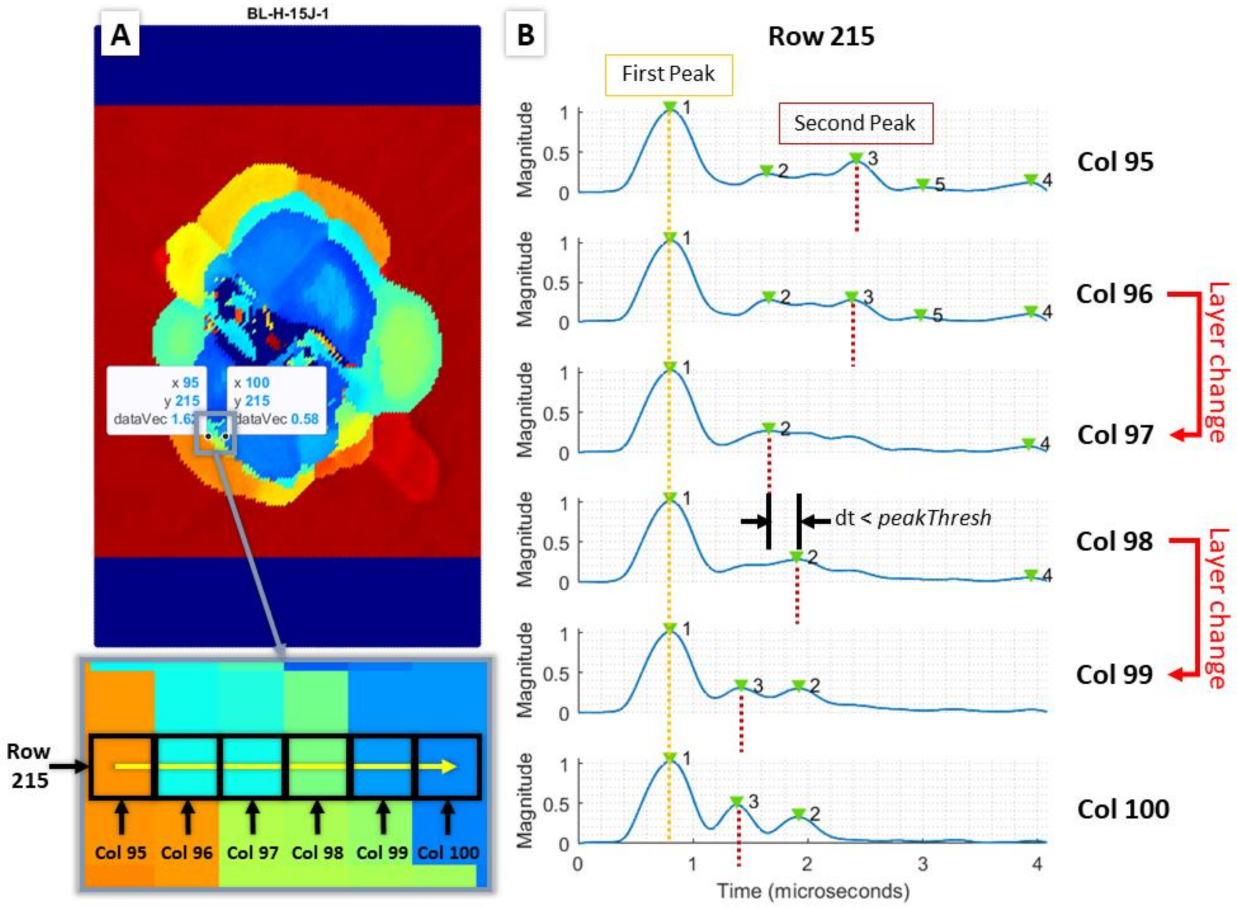


Figure 3.15: Peak label process example. (A) Example path direction along Row 215 from Column 95 to 100, (B) A-scan signals along path direction showing changes in peak labels and detected peak changes.

labelpeaks is called twice, once looping through raw TOF across the rows and once across the columns. The layer change locations detected in each direction are shown in Figure 3.16A (row) and Figure 3.16B (column). Both directions are required because some inflection points are only able to be detected when looping across rows and vice versa for columns. However, **labelpeaks** is only able to detect some of the inflection points, so the next step is to use **magpeaks** to detect the rest of the inflection points.

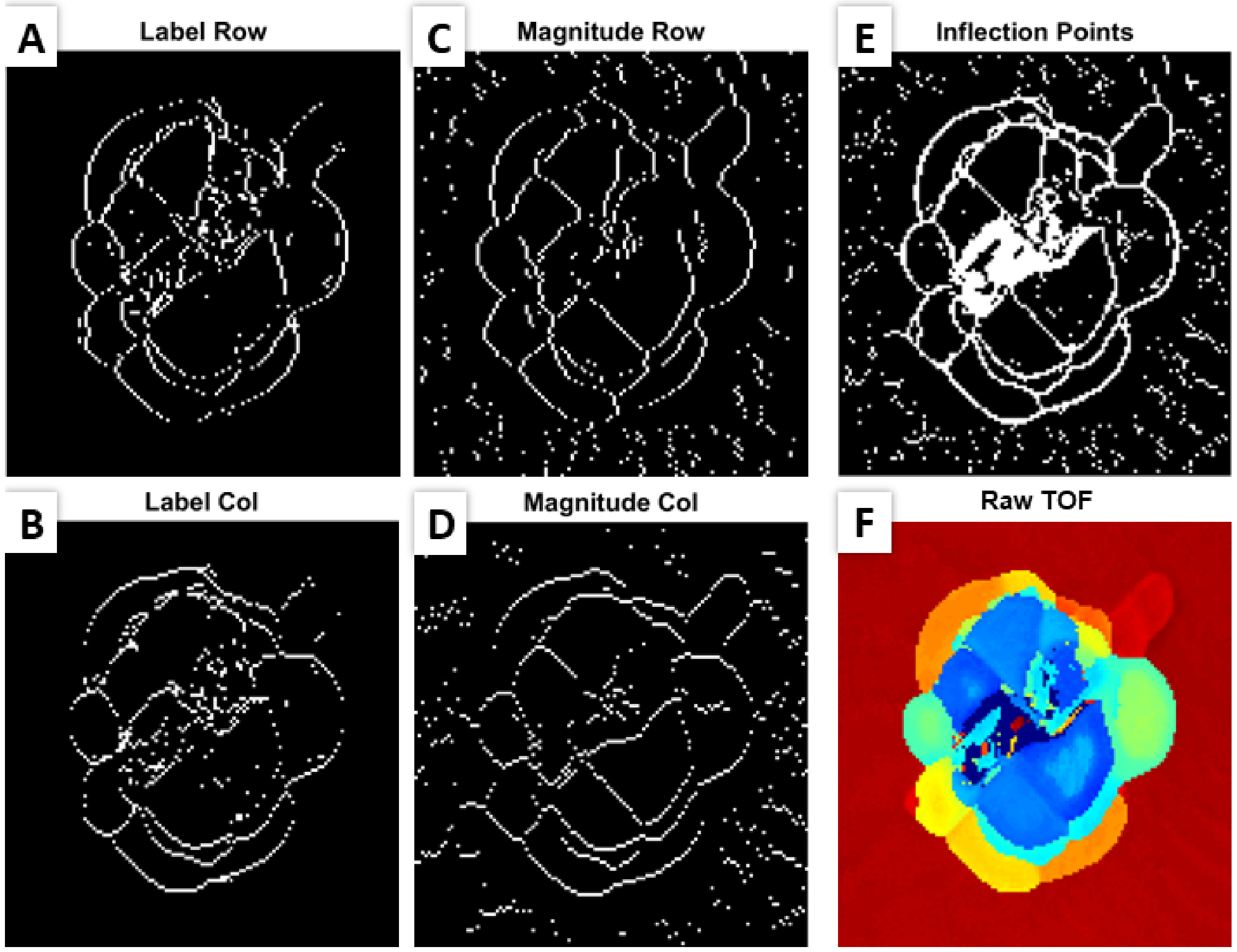


Figure 3.16: Inflection points found via (A) peak labels changes along rows, (B) peak label changes along columns, (C) peaks in second peak magnitude along rows, (D) peaks in second peak magnitude along columns. (E) Inflection points from all methods (A)-(D). (F) Unprocessed TOF for reference.

3.6.2 Peaks in Second Peak Magnitude – magpeaks

magpeaks plots the negative value of magnitude of the second peak along each row or each column of the sample, then uses **findpeaks** and *minProm2* to find peaks. The column (plot along rows) or row (plot along columns) corresponding to the peak location is the location of a layer change and marked as an inflection point. *minProm2* helps the function ignore any peaks that are a result of noise in the second peak magnitude. The negative value of the second peak magnitude is taken to turn the valleys into peaks that can be detected by the **findpeaks** function. An example is shown in Figure 3.17 where seven inflection points are located along Row 171 using this method. In Section 4 (see Figure 3.4), **plottest** plots figures similar to Figure 3.17 to help the user adjust the input *minProm2*.

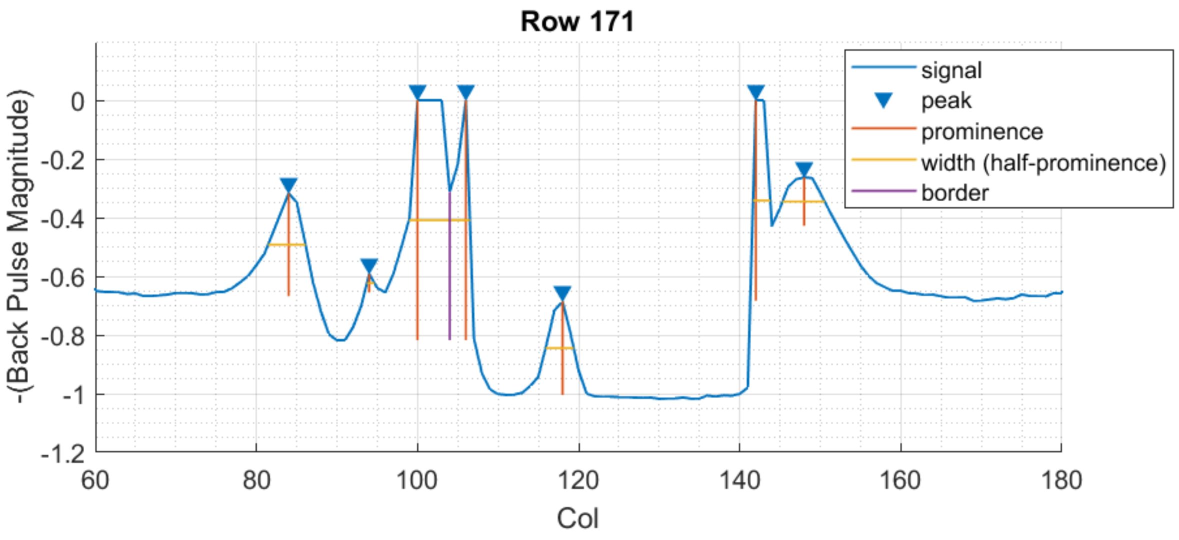


Figure 3.17: Second peak magnitude plot example.

magpeaks is able to detect layer changes where distinct second peak changes are not present. An example is shown in Figure 3.18A, along Row 171 and from Column 115 to 121. As shown in Figure 3.18B, from Column 115 to 118, the magnitude of the second peak monotonically decreases, and then from Column 118 to 121, the magnitude of the second peak monotonically increases. Figure 3.18C shows the physical meaning of the monotonic increase and decrease. The monotonic decrease corresponds to the signal from the transducer reflecting off just the red layer, shown in i, then reflecting off a combination of a decreasing area of the red layer and an increasing area of the blue layer, shown in ii. The red layer is closer to the transducer than the blue layer. The layer change corresponds to when the signal reflects off an equal area of the red and blue layers, shown in iii. The monotonic decrease corresponds to the signal reflecting off a decreasing area of the red layer and an increasing amount of the blue layer, shown in iv, and then reflecting off just the blue layer, shown in v. Like **labelpeaks**, **magpeaks** is called twice, once taking slices of the raw TOF along the rows and once across the columns.

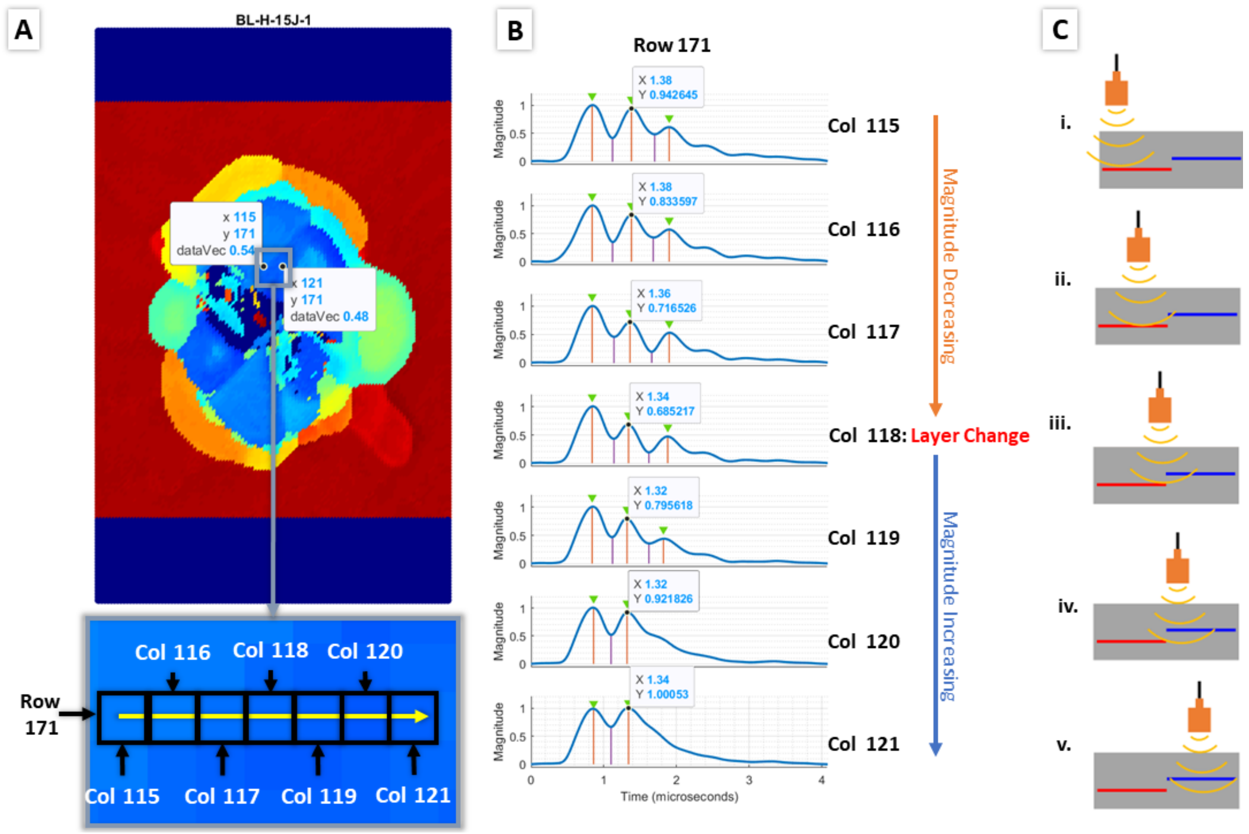


Figure 3.18: Second peak magnitude layer change detection method example. (A) Example path direction along Row 171 from Column 115 to 121, (B) A-scan signals along path direction showing changes in peak magnitude and detected layer change, (C) simplified example of damage layer and UT C-scan transducer progression before, during, and after layer change.

The layer change locations detected by **magpeaks** in each direction are shown in Figure 3.16C (row) and Figure 3.16D (column). Similar to **labelpeaks**, both directions are required because some inflection points are only able to be detected when taking slices across rows and vice versa for columns. The combination inflection point map of all four methods – **labelpeaks** and **magpeaks** applied across rows and columns – are shown in Figure 3.16E next to the raw TOF plot in Figure 3.16F for reference. To complete the inflection point map, any pixels along the edge of the map are removed (first row, last row, first column, last column) and points with only one peak detected are set to a value of 1. Having only one peak detected indicates that the damage is too close to the surface to resolve a TOF value with the current transducer frequency.

The next steps are to obtain a mask of the overall damage area, then clean up and close gaps in the inflection point map using morphological operations.

3.6.3 Mask of Undamaged Region

To create a mask of the undamaged region, the inflection point map is first cleaned up using the MATLAB built in function, **bwmorph** (from the Image Processing Toolbox). First the **bwmorph** function is used with the ‘clean’ option to remove isolated pixels that are surrounded by zeros in all directions – diagonal, vertical, and horizontal, an example of which is shown in Figure 3.19A. Next, **bwmorph** is applied again with the ‘spur’ option to remove spur pixels, which are pixels that have exactly one 8-connected neighbor, or in other words, a pixel that is only connected to one other pixel where the connection can be diagonal, horizontal, or vertical, an example of which is shown in Figure 3.19B. If the pixel is at the edge of the image, such as the bottom left pixels in Figure 3.19B and Figure 3.19C, then it counts as connected to the edge(s) that is it adjacent to. The morphological operation spur is used with the option ‘inf,’ which means that the operation will be applied repeatedly until the image does not change, an example of which is shown in Figure 3.19C. In general, the ‘clean’ operation is used to remove noisy pixels and the ‘spur’ operation is used to remove stray lines that are not part of an enclosed contour. A few examples of areas where ‘clean’ and ‘spur’ will remove unwanted features are shown in Figure 3.19D. The ‘clean’ operation will also help remove stray pixels in the background (undamaged region). An example from before and after processing is shown in Figure 3.20A and B.

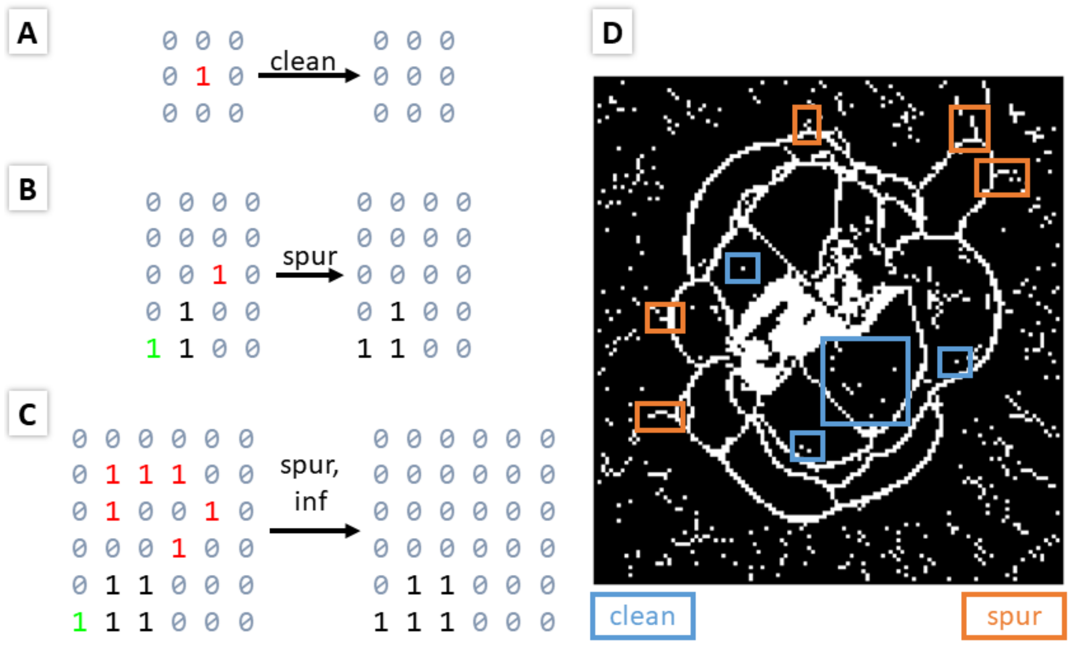


Figure 3.19: Examples of morphological operations (A) clean, (B) spur, operated once, (C) spur, operating with option ‘inf’ (until image no longer changes). (D) Examples of regions in inflection point map that will be cleaned from clean (blue) and spur operations (orange).

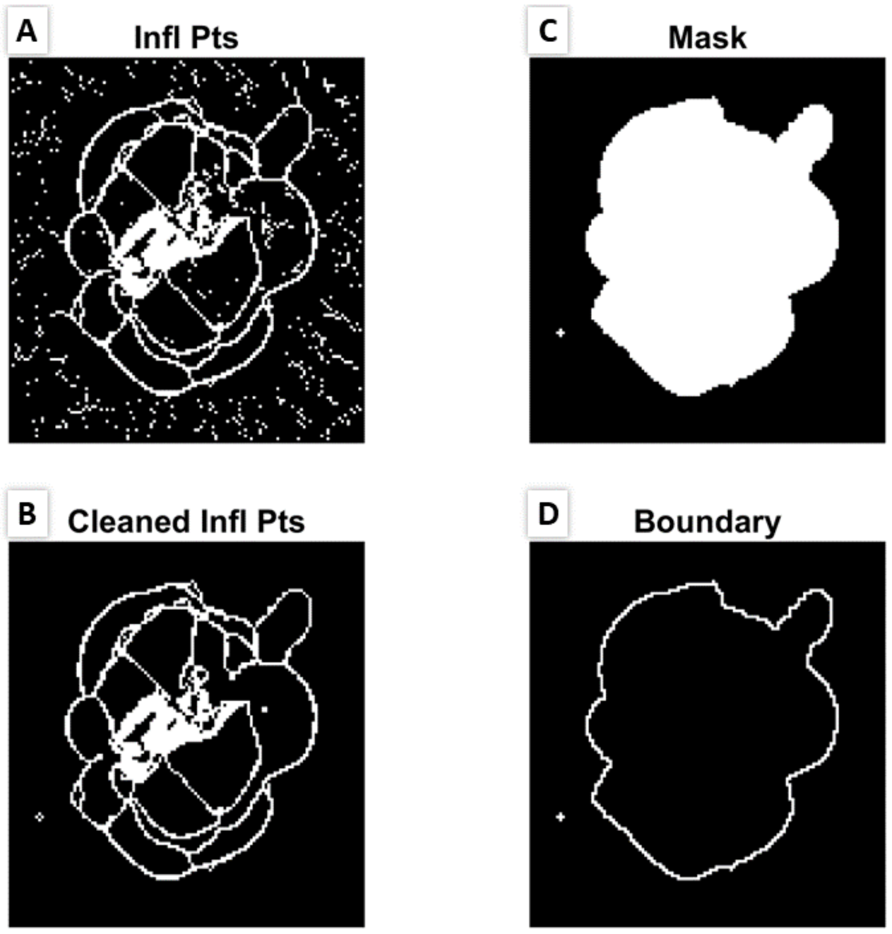


Figure 3.20: (A) Inflection points map before morphological operation cleanup, (B) inflection point map after morphological operation cleanup for creating damage area mask, (C) damage area mask, (D) damage region boundary.

After the clean-up process, the **bwboundaries** (ignoring any interior holes) and **bwperim** functions (also from the Image Processing Toolbox) are used to create a mask and perimeter of the undamaged region, which are shown in Figure 3.20C and Figure 3.20D respectively.

3.6.4 Inflection Points Cleanup

The mask is applied to the original inflection points map to remove all points outside of the damaged region. The input *seEl* is used at this point to create line structuring element(s) with the given length and angle, which is input as [length45deg, length-45deg, length0deg, length90deg] for each sample. If the length is designated as zero, then the structuring element is not created or used. Each of the line structuring elements is one pixel wide and the origin of the element is located at the center of the line. An example of a line structuring element with length 3 and angle 45 degrees is shown in Figure 3.21A. The function **imclose** (from the Image Processing Toolbox) is used to morphologically close the image using each of the designated structuring element(s).

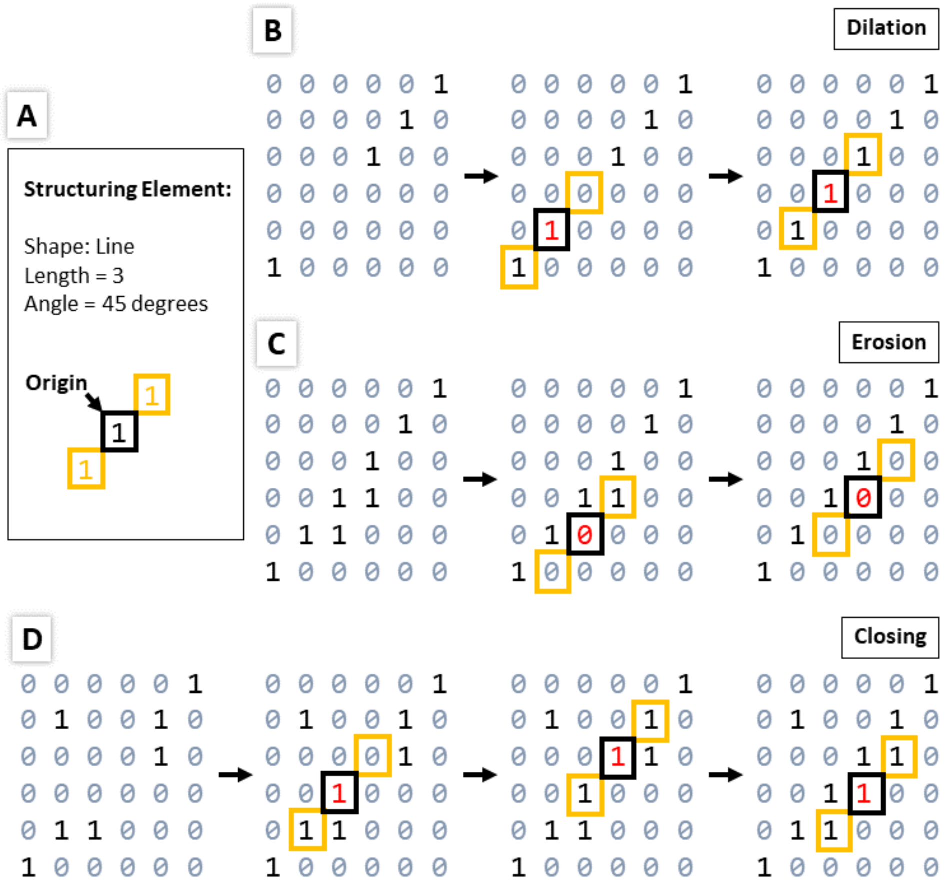


Figure 3.21: (A) Example of a structuring element. Examples of morphological operations (B) dilation, (C) erosion, and (D) closing.

Closing is a dilation operation followed an erosion operation using the same structuring element for both operations. Dilation is an operation in which the origin of the structuring element is placed at each background pixel (0), and if any of the pixels under the structuring element are a foreground pixel (1), then the pixel at the origin is changed to 1. An example is shown in Figure 3.21B, where the operation connects the previously unconnected center diagonal. Erosion is an operation in which the origin of the structuring element is placed at each foreground pixel (1), and if any of the pixels under the structuring element are a background pixel (0), then the pixel at the origin is changed to 0. An example is shown in Figure 3.21C, where the operation thins the center diagonal. Morphological closing will tend to close areas that are similar to the shape of the structuring element. An example is shown in Figure 3.21D, where the operation connects the center diagonal and does not affect the isolated pixel in the top left corner.

The option to use structuring elements to close gaps in the inflection points map is available for specific samples that have regions that would otherwise not be fully enclosed by the original inflections points or still retain gaps after the default morphological operations are performed. The user should use the least number of structuring elements and the shortest element necessary because the **imclose** operation will connect neighboring lines that may not need to be connected. The element length should be equal to the size of the largest gap to be closed with an extra pixel added. If multiple structuring elements are used for one sample (e.g., 90 degrees and 45 degrees), the **imclose** operations are independent and the resulting inflection points maps are combined with an OR operation.

After the **imclose** operations, a **bwmorph** ‘clean’ operation is performed. Then a **bwmorph** ‘bridge’ operation with the ‘inf’ option is applied. The bridge operation changes any zeros into ones if it has two neighboring pixels that are ones, for horizontal, vertical, and diagonal neighbors. An example is shown in Figure 3.22.

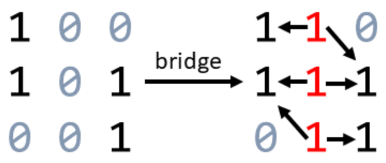


Figure 3.22: **bwmorph** bridge operation.

The boundary shown in Figure 3.20D is added back to the inflection points map in case any pixels were accidentally removed in the previous operations. Then three **bwmorph** operations are applied: (1) fill, which fills in any lone background pixels and is the opposite of the clean operation, (2) spur, which is applied twice, and (3) clean. Finally, any inflection points located at a point where only one peak was detected is set to a value of 1 in case it was removed during the previous operations. The original map is shown next to the final result after all the previous processing of the inflection point map for comparison in Figure 3.23A and B.

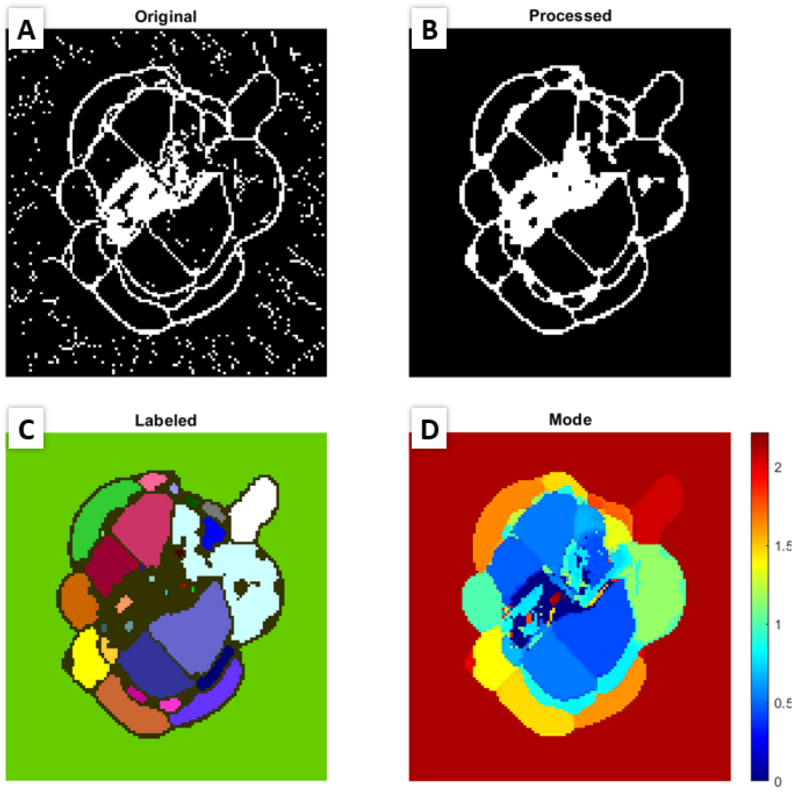


Figure 3.23: (A) Inflection point map before morphological operation cleanup, (B) inflection point map after morphological operation cleanup for segmentation, (C) labeled connected regions, (D) processed TOF using mode value of respective connected regions.

Once all gaps in the inflection point map are closed, the function **bwlabel** (from the Image Processing Toolbox) is used to find 8-connected objects in the inflection point map and labeling each connected object as a region, as shown in Figure 3.23C. The mode TOF value is found within each area, and if the difference between the TOF value at a point and the mode TOF value of the region the point belongs to is less than *modeThresh*, then the TOF value at the point is set to the mode TOF value. Any points not part of a labeled region, i.e., points located at a foreground pixel in the processed inflection point map, is left as the raw TOF value. The result is shown for a sample in Figure 3.23D. A plot of the raw TOF is shown next to the processed TOF for comparison in Figure 3.24A and B. The processing removes "donut" artifacts, labeled 'A' in Figure 3.24A, and removes noise from the undamaged region. The *modeThresh* allows for regions that are not picked up by the **labelpeaks** and **magpeaks** functions or otherwise have gap(s) that are unable to be closed by the morphological operations to still appear in the processed TOF. Example regions are labeled 'B' in Figure 3.24A.

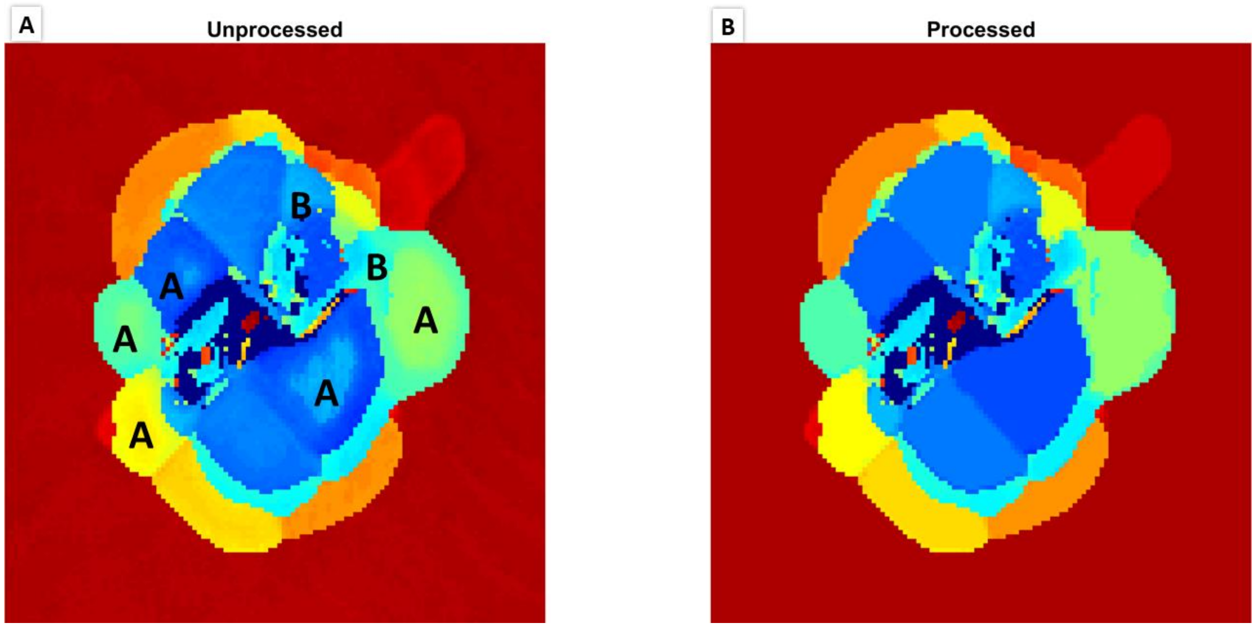


Figure 3.24: (A) Raw TOF compared to (B) processed TOF.

3.6.5 Figures and Output Variables

segcscan can produce eight figures:

(1) queryable plot of raw TOF cropped to the damage bounding box, not saved,

(2) inflection points map breakdown between **labelpeaks** and **magpeaks** along row and column directions, Figure 3.16,

(3) a queryable plot of inflection points map,

(4) a non-queryable plot of inflection points map,

(5) mask and boundary of undamaged region, Figure 3.20,

(6) processed inflection points map, labeled regions, and processed TOF, Figure 3.23,

(7) comparison of raw and processed TOF, Figure 3.24, and

(8) processed TOF.

All figures are saved in *figFolder* and are in subfolders called comboInflpt, inflptsQuery, inflpt, masks, process, compare, and tof respectively.

segcscan saves five variables, in subfolders in *outFolder* with names matching the variable names:

(1) processed TOF, *tof*,

(2) unprocessed inflection points map, *inflpt*,

(3) mask of undamaged region, *mask*,

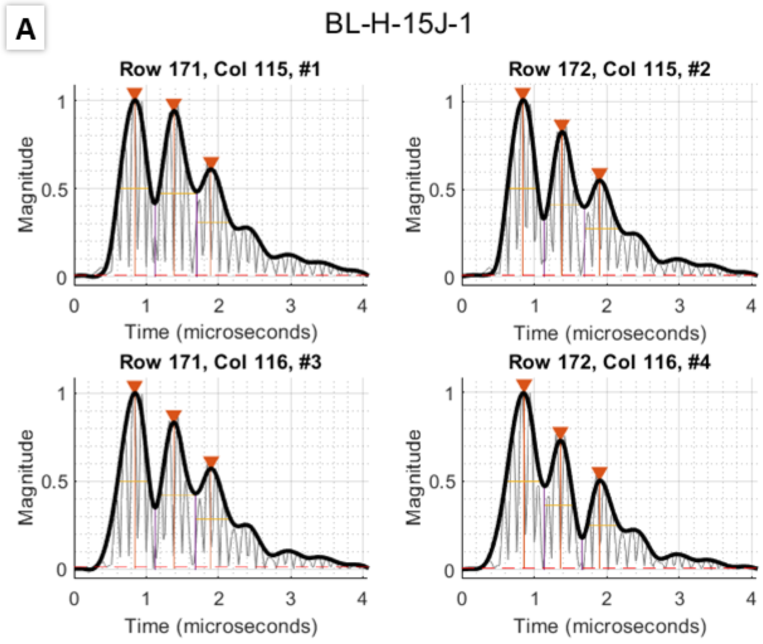
(4) perimeter of damaged region, *bound*, and

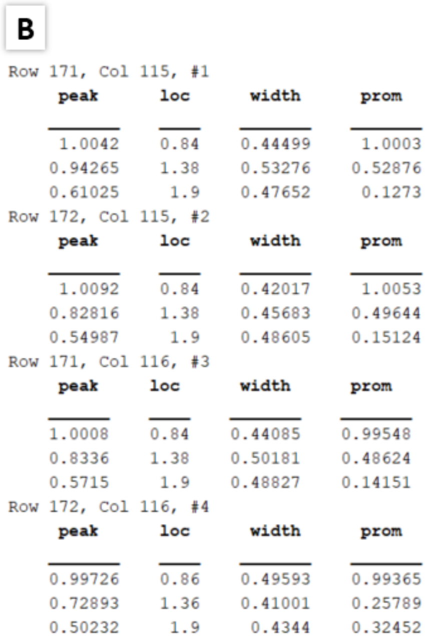
(6) the magnitude of the second peak at each point, *peak2*.

These variables will be used in the following sections to perform additional processing steps on the TOF and for plotting figures.

3.7 Plot Test Figures – plottest

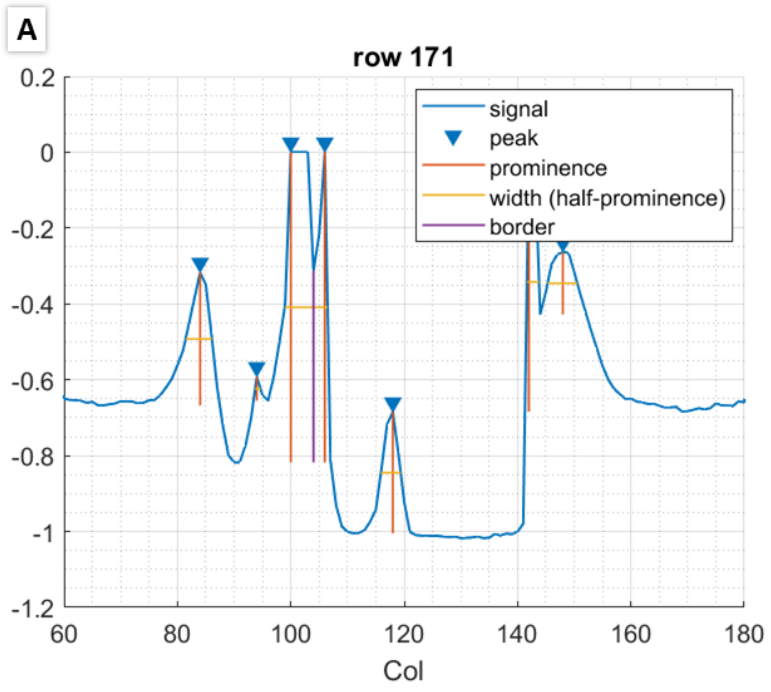
plottest has inputs *rowRange*, *colRange*, *dir*, and *num*. *rowRange* and *colRange* are used for helper function **plotascans**, which loads *cscan* from **readcscan** and plots A-scans from the corresponding rows and columns as shown in Figure 3.25A. In this case the requested rows are Row 171 and 172 and the columns are Column 115 and 116. *dir* and *num* are used for helper function **plotpeak2** which reads in *peak2* from **segcscan** and plots the magnitude of the second peak, along a specific row or column, which is input to *dir* as ‘row’ or ‘col’ and *num* specifies which row or column, as shown in Figure 3.26A, where Row 171 is plotted. The user can use the queryable raw TOF and inflection points map figures to help find locations of interest for adjusting input parameters from previous sections. The peak information for both graphs are also output in the command window for reference as shown in Figure 3.25B and Figure 3.26B.





```
Row 171, Col 115, #1
     peak      loc      width      prom
    ______    ____    _______    _______

     1.0042    0.84    0.44499     1.0003
    0.94265    1.38    0.53276    0.52876
    0.61025     1.9    0.47652     0.1273
Row 172, Col 115, #2
     peak      loc      width      prom
    ______    ____    _______    _______

     1.0092    0.84    0.42017     1.0053
    0.82816    1.38    0.45683    0.49644
    0.54987     1.9    0.48605    0.15124
Row 171, Col 116, #3
     peak     loc     width      prom
    ______    ____    _______    _______

    1.0008    0.84    0.44085    0.99548
    0.8336    1.38    0.50181    0.48624
    0.5715     1.9    0.48827    0.14151
Row 172, Col 116, #4
     peak      loc      width      prom
    _______    ____    _______    _______

    0.99726    0.86    0.49593    0.99365
    0.72893    1.36    0.41001    0.25789
    0.50232     1.9     0.4344    0.32452
```

Figure 3.25: Example of **plotascan** output, where (A) are the A-scan plots with smoothing spline fit, peaks, peak width/prominence, and *noiseThresh* shown, (B) tabular data displayed in command window about the peak magnitude, location, width, and prominence for each A-scan.



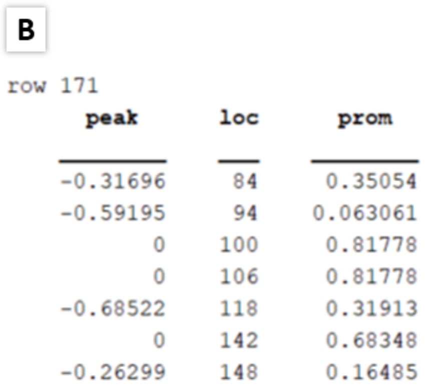

```
B
row 171
      peak        loc       prom
    ________    ___    ________

    -0.31696     84     0.35054
    -0.59195     94    0.063061
           0    100     0.81778
           0    106     0.81778
    -0.68522    118     0.31913
           0    142     0.68348
    -0.26299    148     0.16485
```

Figure 3.26: Example of **plotpeak2** output, where (A) is the row-wise plot of the second peak magnitude with peak information shown, (B) is the tabular peak data displayed in the command window.

3.8 Plot Figures – plotfig

plotfig has two inputs: (1) *plateThick*, the thickness of the sample in millimeters, and (2), *nLayers*, number of layers in the composite sample. The function reads in the processed TOF, *tof*, from **segcscan**, and uses the plate thickness and number of layers to calculate the TOF associated with each layer, then discretizes the processed TOF, *tof*, from **segcscan** into (2×*nLayers* + 1) groups using the MATLAB built in function **discretize**. The mask of undamaged areas is used to remove the undamaged plate areas by setting the point equal to NaN.

Two figures are produced by **plotfig**: (1) damage groups 3D plot, Figure 3.27A, and (2) damage groups 2D plot, Figure 3.27B. The figures are saved in the folders damLayers and 3Dplot respectively. The damage layer groups are saved as *damLayers* in *outFolder*.

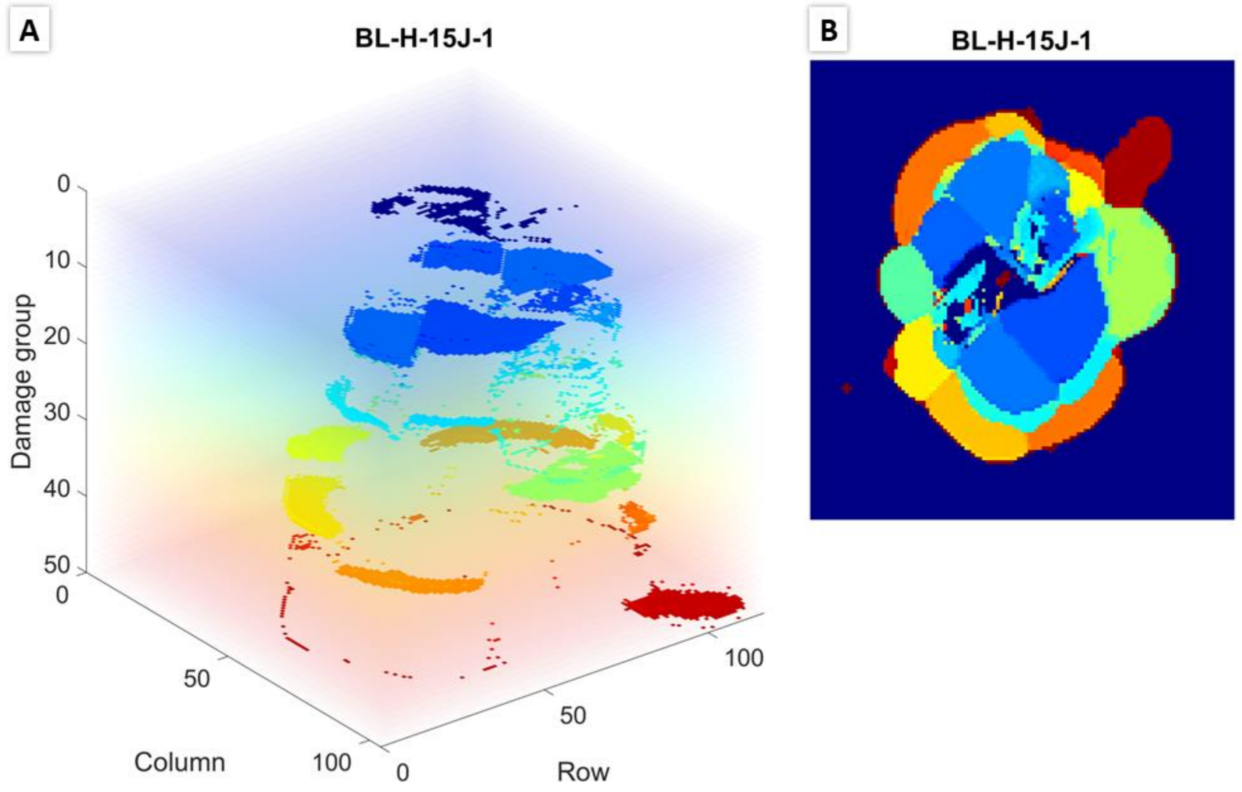


Figure 3.27: **plotfig** (A) 3D and (B) 2D plots.

3.9 Merge C-scan – mergecscan

mergecscan has two inputs, *dx* and *dy*, where positive corresponds to right/up and negative corresponds to left/down. *dx* and *dy* are the x and y offsets of the front scan relative to the back scan and can be adjusted with helper figures plotted by **mergecscan**. **mergecscan** reads in the damage layer groups, *damLayers*, from **plotfig** and the outline of the damaged region, *bound*, from **segcscan**. The function plots the outline of the damaged region from the front and back scans on top of each other before and after the horizontal and vertical adjustments are applied, shown in Figure 3.28. The back scan is flipped left to right and top to bottom to align with the top scan before the two are merged. The 3D plot of the front, back, and hybrid scan are shown in Figure 3.29. The merged hybrid C-scan is saved as *hybridCscan*. A second figure is saved with just the hybrid C-scan in the folder hybridCscan.

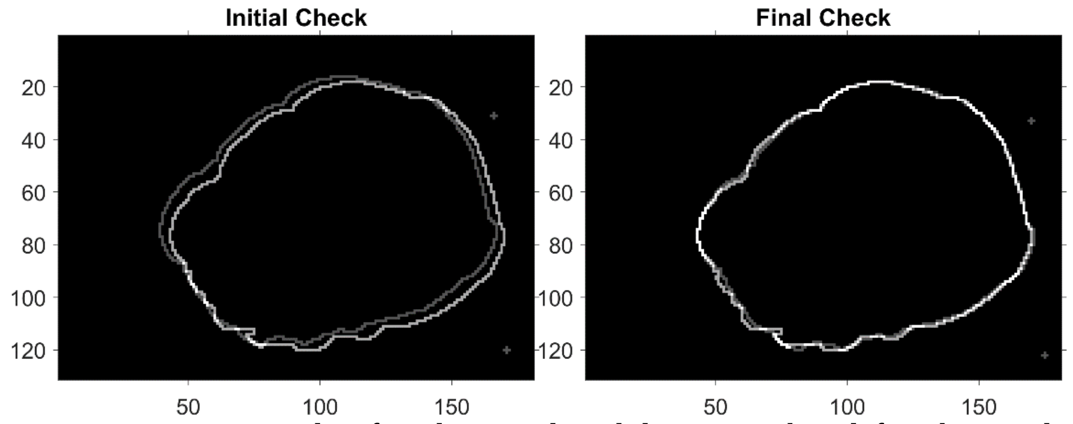


Figure 3.28: Damage region plots for adjusting *dx* and *dy* inputs, where left is the initial outline alignment and right is the alignment after adjustment.

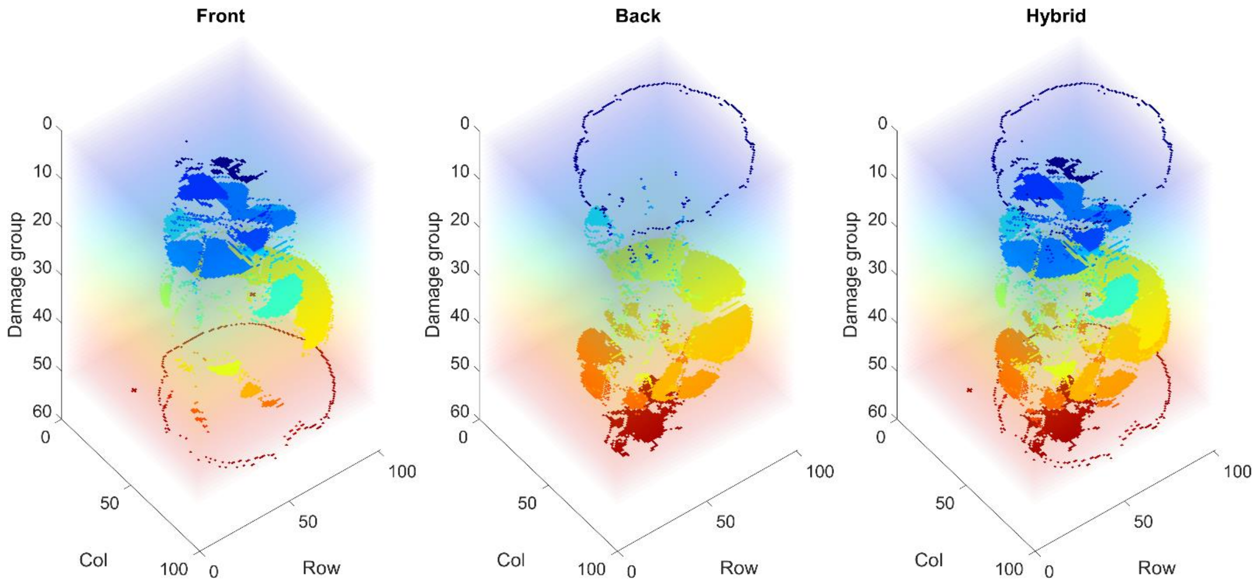


Figure 3.29: Example of a hybrid C-scan (right) and the corresponding front (left) and back (middle) C-scans.

3.10 Plot Custom Figures – plotcustom

plotcustom plots custom figures specific to this project and that not all users may be interested in generating. There are two inputs for **plotcustom**, *utwinCrop* and *dy*. *utwinCrop* are the coordinates at which the UTWin figure should be cropped in [startRow endRow startCol endCol] format. *dy* is the amount of vertical adjustment the UTWin figure should be adjusted up (positive) or down (negative). The resulting figures plotted are the raw, processed, and UTWin damage depth map plotted together for comparison, which are shown in the following chapter.

CHAPTER 4: RESULTS AND DISCUSSION

The chapter will present the processed damage depth map, 3D reconstruction, and hybrid 3D reconstruction results from the code using: (1) the 25-ply samples, specifically nine front and back C-scan samples and (2) the 24-ply sample with X-ray CT scan for comparison. All results and data used from the 25-ply sample dataset are posted on UC San Diego's Research Data Curation as a collection ([Link to be added here when dataset is published on UCSD RDCP site, before defense date.]).

4.1 25-ply Damage Depth Maps

The TOF results converted to damage depth after processing the samples are shown in Figure 4.3 through Figure 4.10 with the raw, processed, and UTWin damage depth plots side-by-side for comparison. Instead of showing all 26 C-scans, the nine samples with front and back C-scans are shown, where the front scan results are along the top row and the back scan results are along the bottom row. The remaining eight C-scans are shown in Appendix B. It should be noted that the back scan results are mirrored horizontally to allow for easier side-by-side comparison with the front scan. The depth represented by the colormap is measured relative to the face of the sample closest to the transducer closest to the transducer, where dark blue or black (value of 0) indicates the damage is closest to the transducer. For front scans, the closest face is the front face and for back scans, the closest face is the back face. Figure 1.2A is copied here in Figure 4.1 as visual for the front and back face terminology.

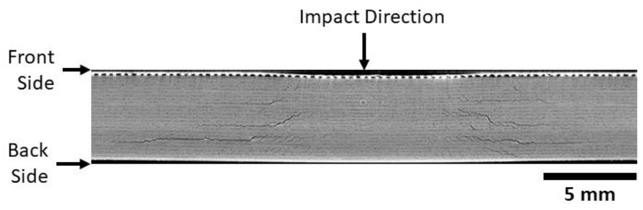


Figure 4.1: Front and back side terminology. Reprinted with permission [4].

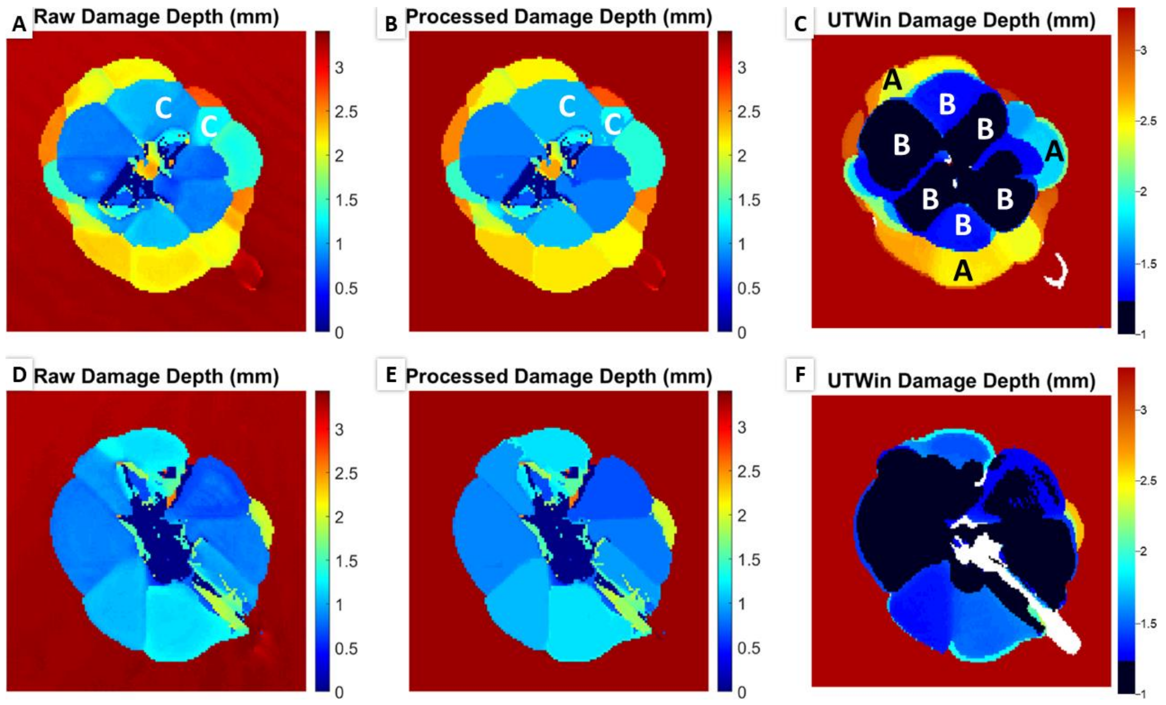


Figure 4.2: CONT-S-15J-2 Damage Depth Map. (A) Raw Front, (B) Processed Front, (C) UTWin Front, (E) Raw Back, (F) Processed Back, (F) UTWin Back.
Dimensions (height x width): front – 51 x 55 mm, back – 51 x 55 mm.

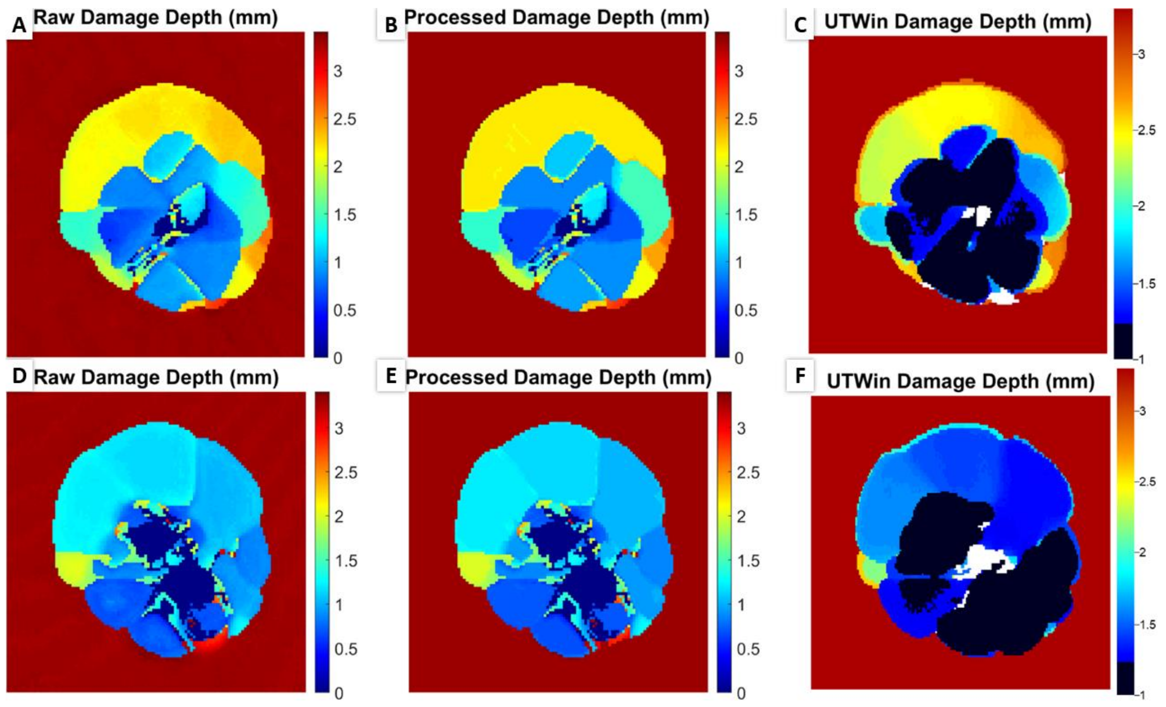


Figure 4.3: BL-S-10J-2 Damage Depth Map. (A) Raw Front, (B) Processed Front, (C) UTWin Front, (E) Raw Back, (F) Processed Back, (F) UTWin Back.
Dimensions (height x width): front – 47 x 46 mm, back – 43 x 46 mm.

As seen in the front scan in Figure 4.2, the code developed is able to capture both details included in the UTWin image, such as the lobes at the edge of the damaged region, labeled A, as well as details that are lost in the UTWin image, specifically the central black and blue lobes, labeled B. However, the code still encounters difficulty with some of the central light blue lobes that are close to the top surface, labeled C, which is an area of improvement for future editions of the code.

As seen in the back side scan in Figure 4.2, the results are a large improvement compared to UTWin, as several lobes of damage near the top surface are well separated in comparison to UTWin which was not able to resolve the region into separate lobes.

Future versions of the code could improve on separating regions that gradually transition between layers such as shown in the front scan in Figure 4.3, which the current code is unable to resolve into separate regions of damage. The topmost area transitions from yellow to light orange to orange in the unprocessed image, but merges into a single orange region in the processed image. It should be noted that the *modeThresh* input described in Section 3.6.4 can be adjusted to a lower value to allow for the transition to show up in the processed image. However, for gradual transitions between adjacent layers, the lower value will also mean that other areas will not be adjusted to the local mode value. As a result, the processed image will look very similar to the unprocessed image.

In general, the code is a significant improvement over the UTWin images with more lobes defined on front and back side scans, especially for lobes close to the top surface. The code was able to capture all of the lobes shown in the UTWin images and distinguish lobes in areas that UTWin showed as dark blue or black.

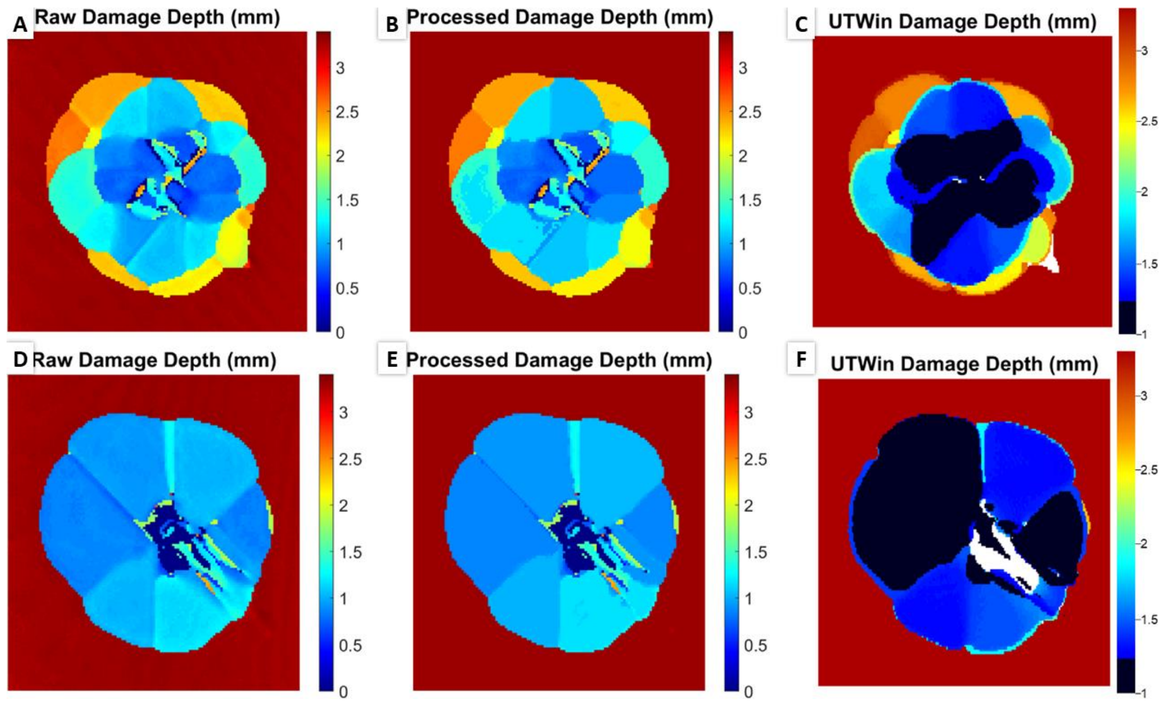


Figure 4.4: BL-S-15J-2 Damage Depth Map. (A) Raw Front, (B) Processed Front, (C) UTWin Front, (E) Raw Back, (F) Processed Back, (F) UTWin Back.
Dimensions (height x width): front – 51 x 55 mm, back – 51 x 51 mm.

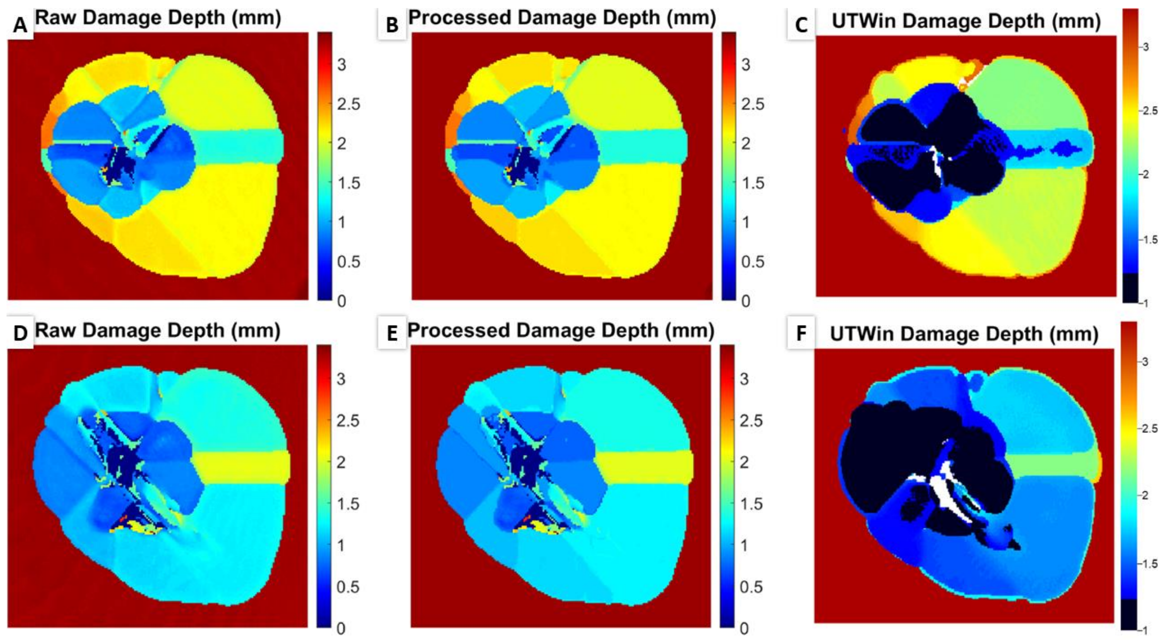


Figure 4.5: BL-S-20J-2 Damage Depth Map. (A) Raw Front, (B) Processed Front, (C) UTWin Front, (E) Raw Back, (F) Processed Back, (F) UTWin Back.
Dimensions (height x width): front – 63 x 76 mm, back – 63 x 72 mm.

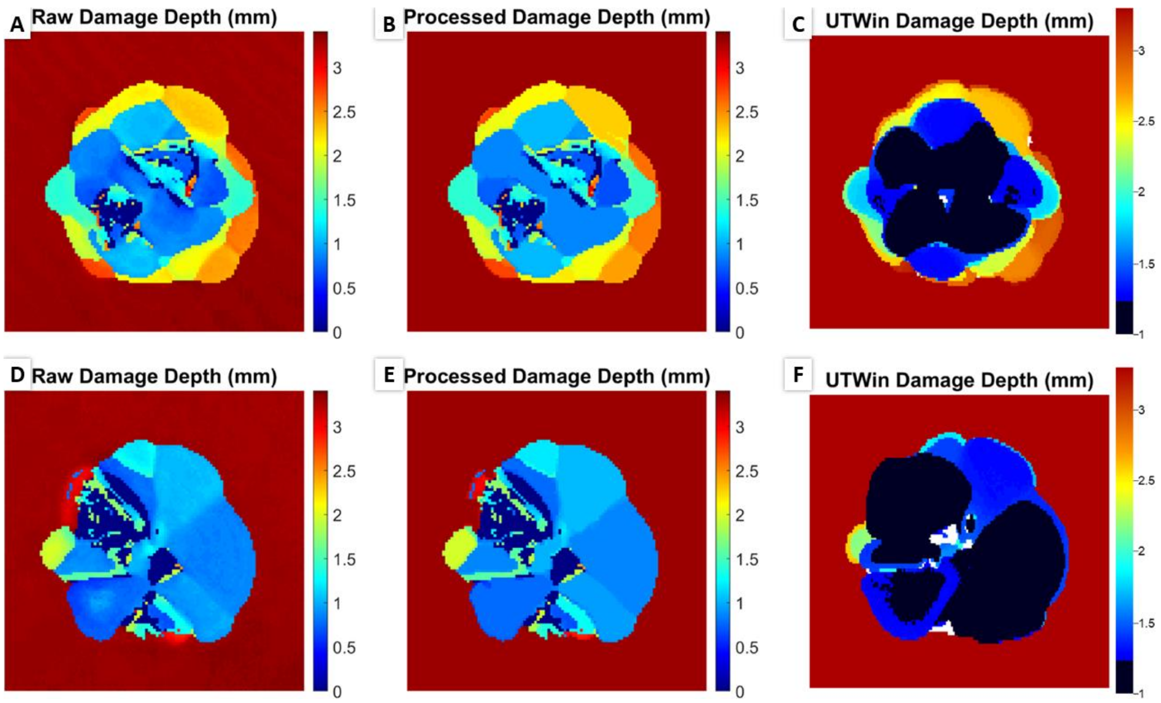


Figure 4.6: CONT-S-10J-2 Damage Depth Map. (A) Raw Front, (B) Processed Front, (C) UTWin Front, (E) Raw Back, (F) Processed Back, (F) UTWin Back.
Dimensions (height x width): front – 43 x 46 mm, back – 43 x 46 mm.

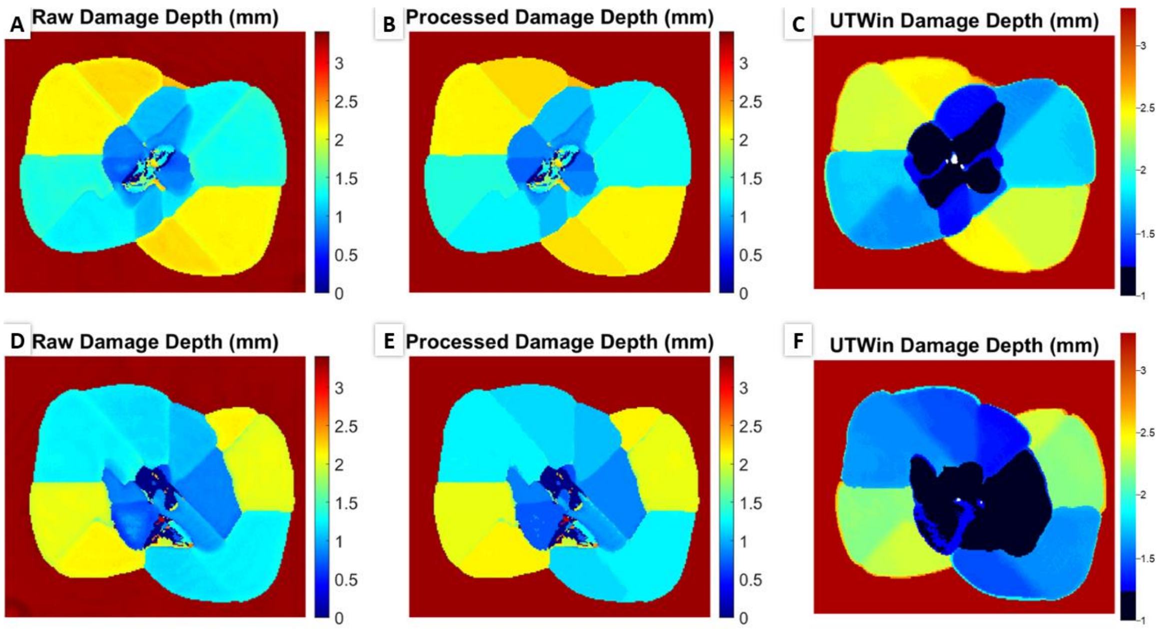


Figure 4.7: CONT-S-20J-2 Damage Depth Map. (A) Raw Front, (B) Processed Front, (C) UTWin Front, (E) Raw Back, (F) Processed Back, (F) UTWin Back.
Dimensions (height x width): front – 75 x 93 mm, back – 75 x 93 mm.

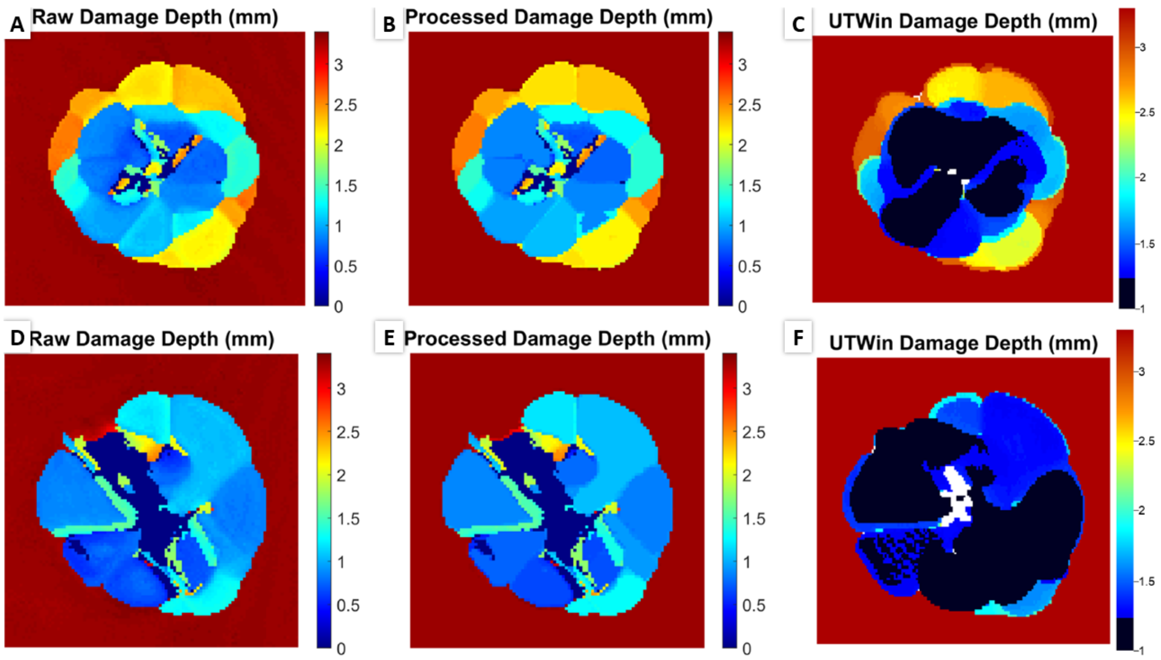


Figure 4.8: RPR-S-10J-2 Damage Depth Map. (A) Raw Front, (B) Processed Front, (C) UTWin Front, (E) Raw Back, (F) Processed Back, (F) UTWin Back.
Dimensions (height x width): front – 39 x 46 mm, back – 39 x 42 mm.

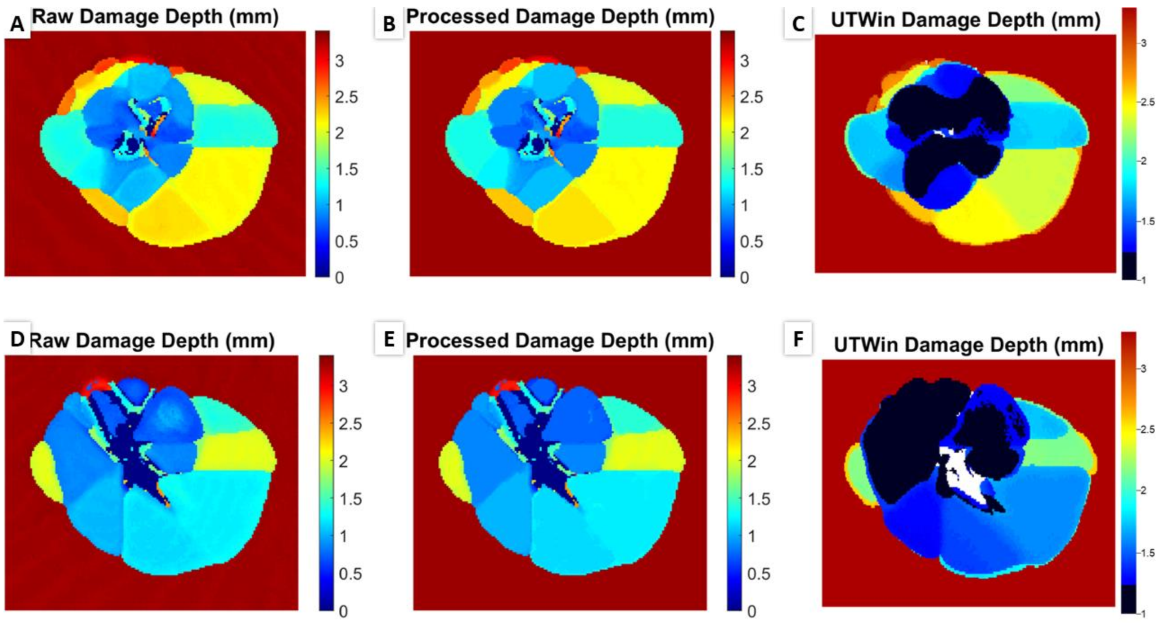


Figure 4.9: RPR-S-15J-2 Damage Depth Map. (A) Raw Front, (B) Processed Front, (C) UTWin Front, (E) Raw Back, (F) Processed Back, (F) UTWin Back.
Dimensions (height x width): front – 51 x 68 mm, back – 51 x 64 mm.

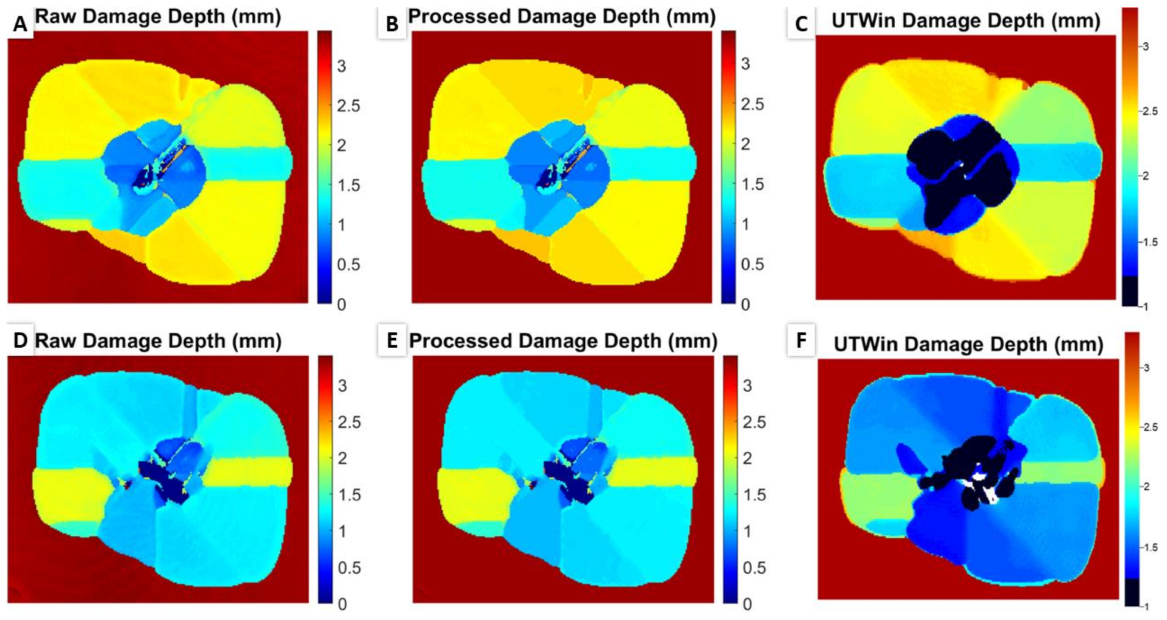


Figure 4.10: RPR-S-20J-2 Damage Depth Map. (A) Raw Front, (B) Processed Front, (C) UTWin Front, (E) Raw Back, (F) Processed Back, (F) UTWin Back.
Dimensions (height x width): front – 75 x 89 mm, back – 71 x 93 mm.

4.2 25-ply Hybrid C-scans and 3D C-scan Reconstruction

The nine hybrid C-scans are shown in Figure 4.11 to Figure 4.19 with plots of the 3D reconstruction of the front and back scans side-by-side for comparison. 3D reconstructions of the remaining eight samples without back scans are included in Appendix C. The resolution along row and column directions is 0.4 mm.

The hybrid C-scans were able to combine and merge regions detected by both the front and back scans and create a more complete picture of the damaged areas of the sample. When merged, the samples have consistent matching depths between the front and back side C-scans. The merged hybrid dataset provides an enhanced picture of the damage state, which is not as easy to ascertain when looking at the individual front or back C-scan. However, in future versions of the code a different method of associating TOF with the respective damage interface could be developed. The current method of dividing the range of TOF into (2 × number of layers

+ 1) was implemented to prevent adjacent damage groups from merging to a single group but requires the user to manually interpret which interface each damage group in the hybrid C-scan and 3D C-scan reconstruction should be associated to.

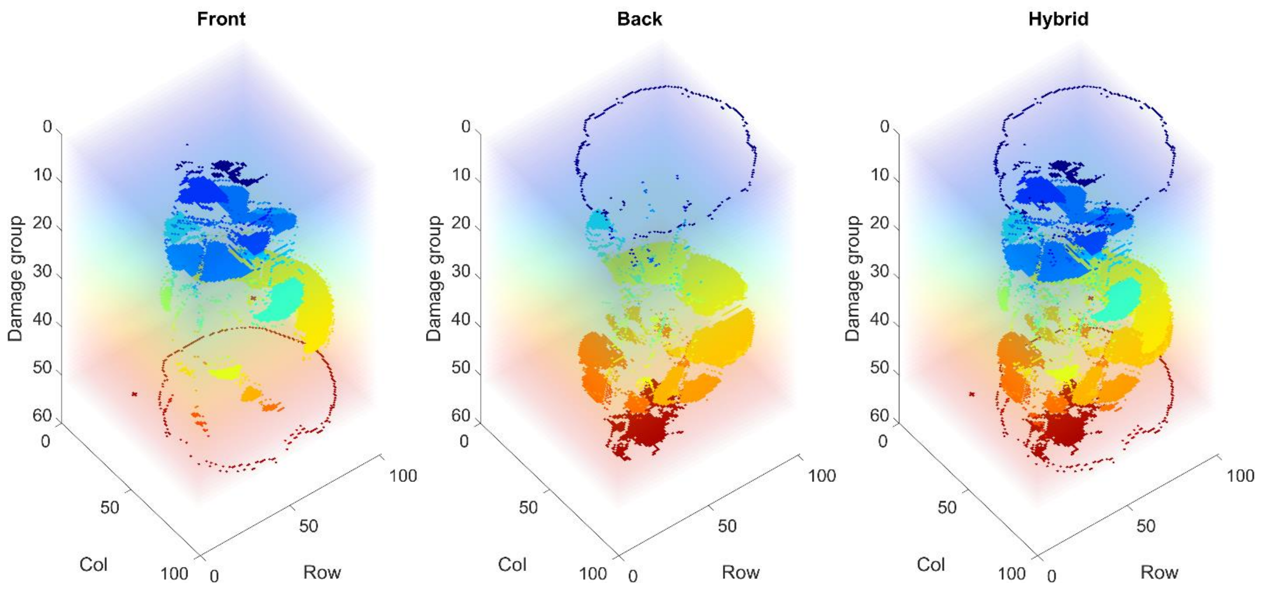


Figure 4.11: BL-S-10J-2 front, back and hybrid C-scan 3D reconstruction.

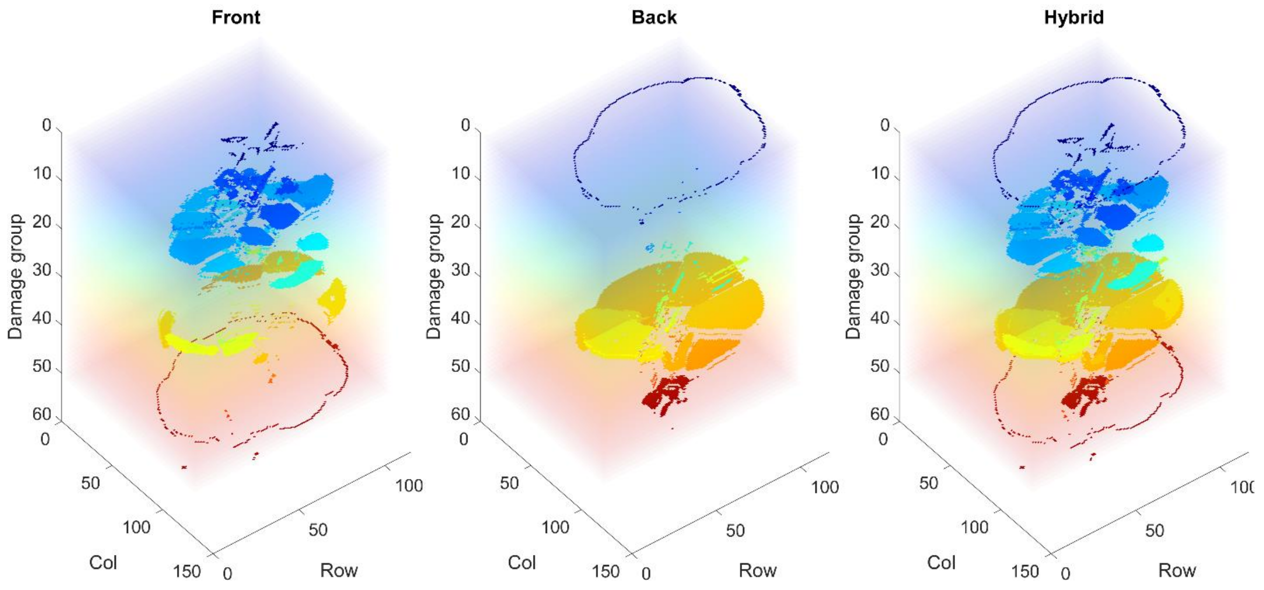


Figure 4.12: BL-S-15J-2 front, back and hybrid C-scan 3D reconstruction.

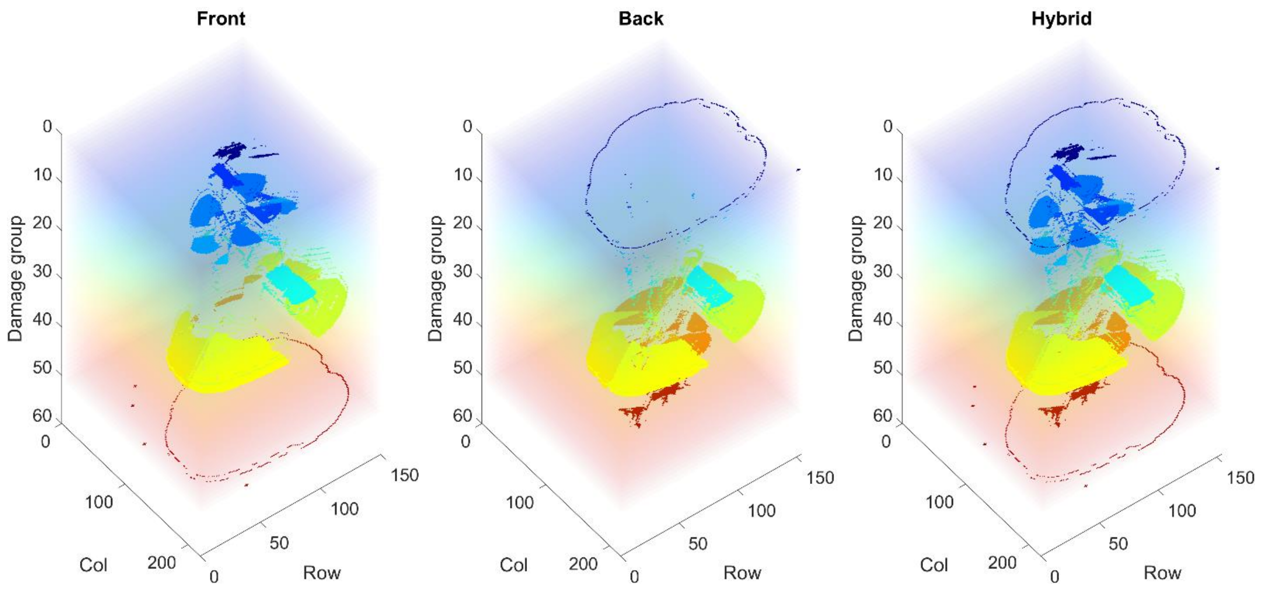


Figure 4.13: BL-S-20J-2 front, back and hybrid C-scan 3D reconstruction.

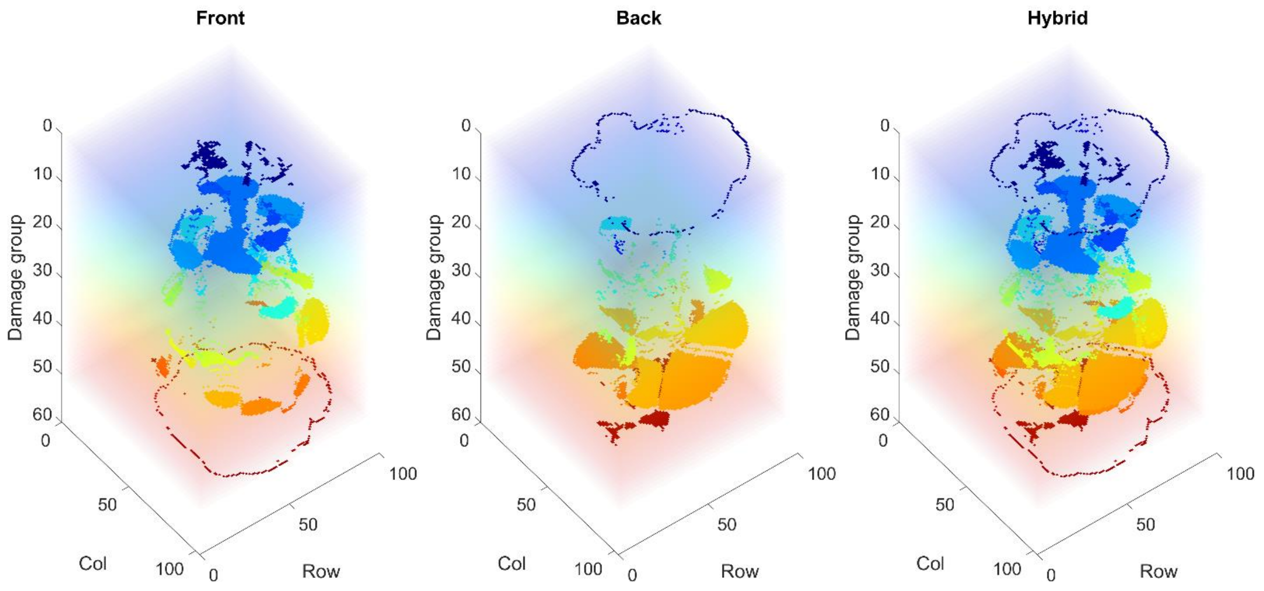


Figure 4.14: CONT-S-10J-2 front, back and hybrid C-scan 3D reconstruction.

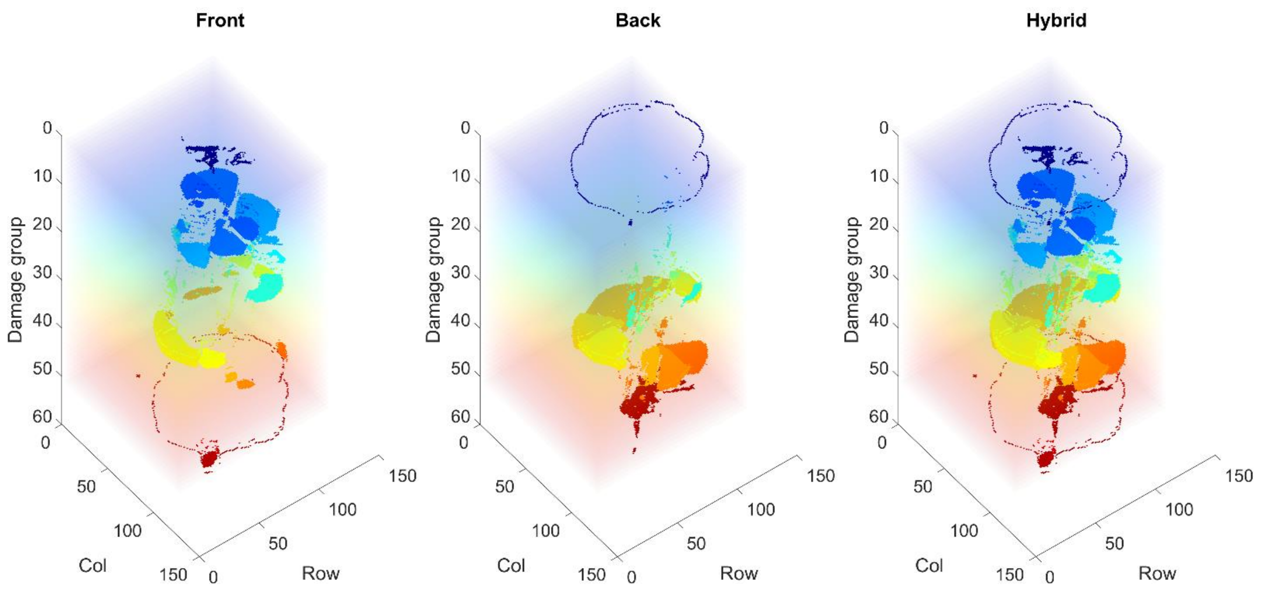


Figure 4.15: CONT-S-15J-2 front, back and hybrid C-scan 3D reconstruction.

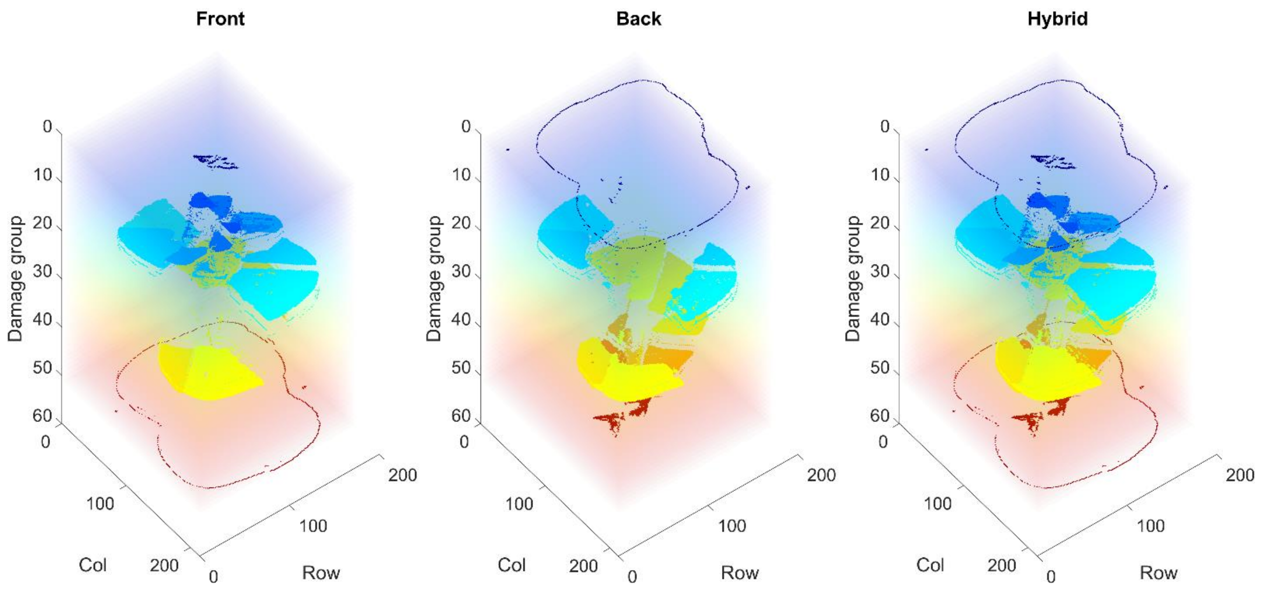


Figure 4.16: CONT-S-20J-2 front, back and hybrid C-scan 3D reconstruction.

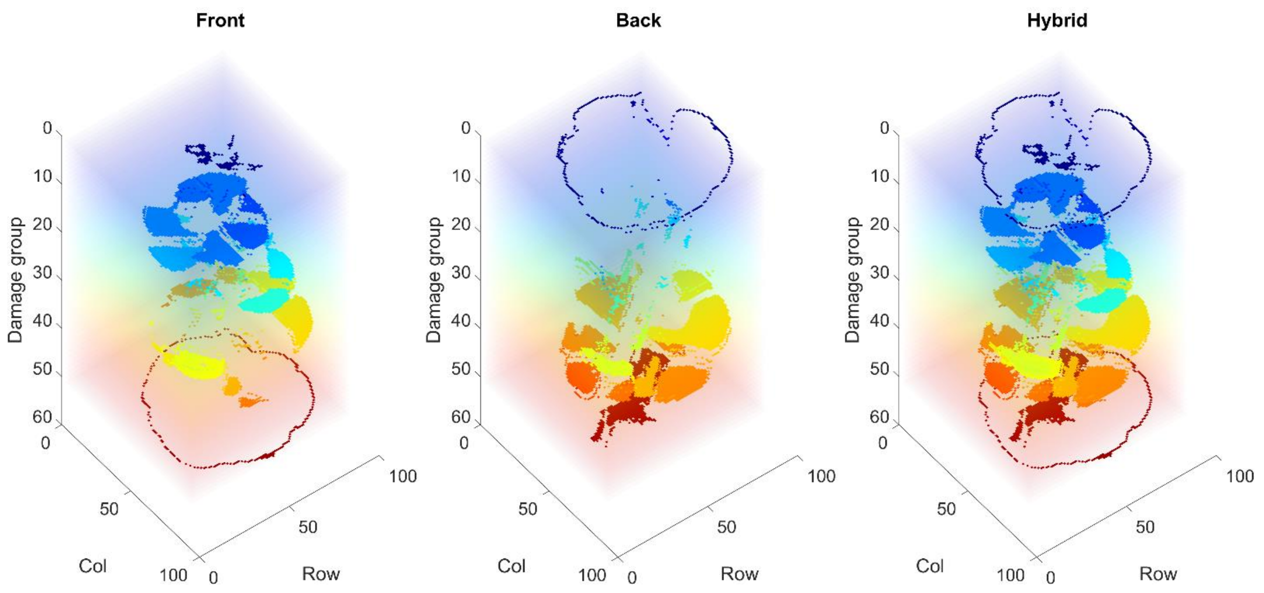


Figure 4.17: RPR-S-10J-2 front, back and hybrid C-scan 3D reconstruction.

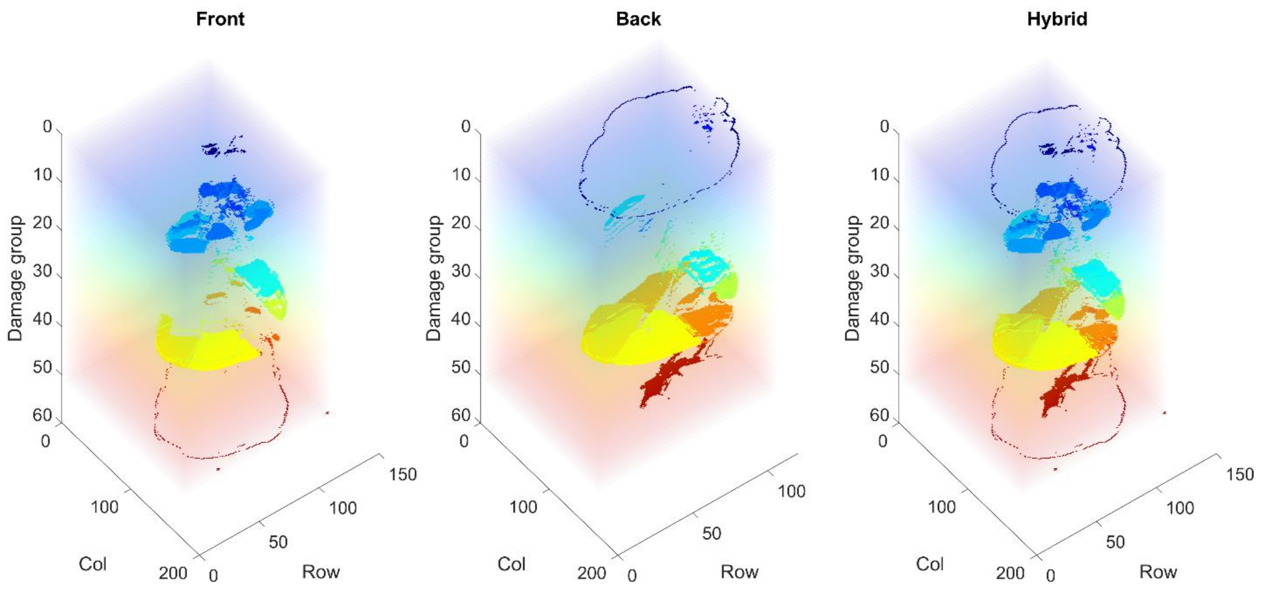


Figure 4.18: RPR-S-15J-2 front, back and hybrid C-scan 3D reconstruction.

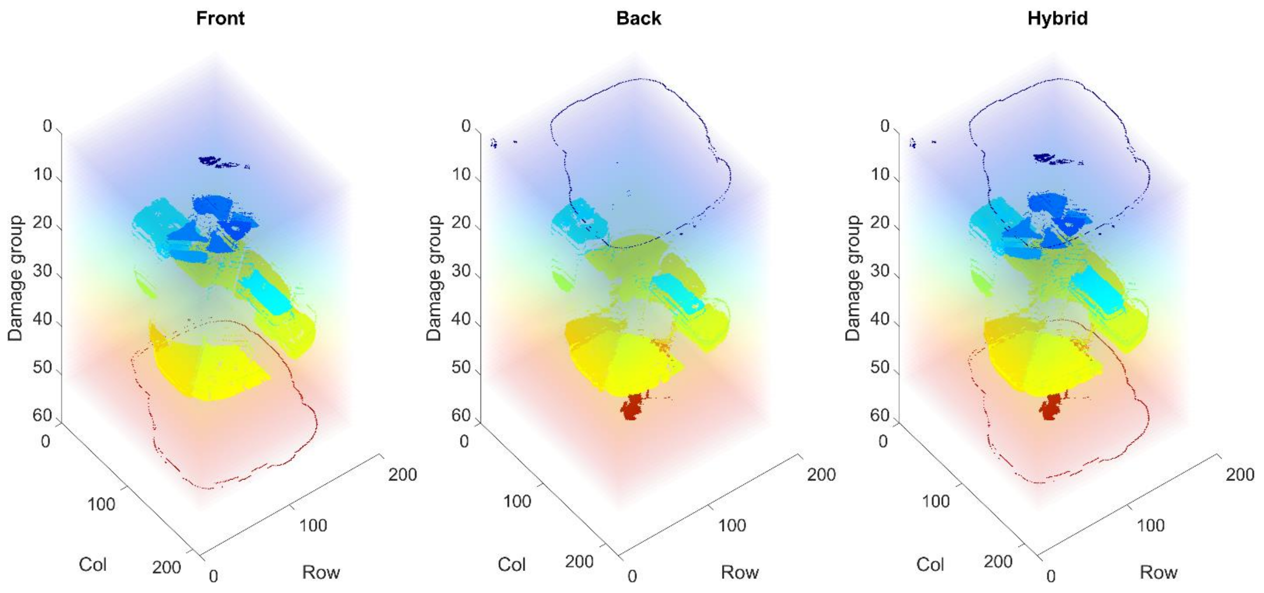


Figure 4.19: RPR-S-20J-2 front, back and hybrid C-scan 3D reconstruction.

4.3 24-ply Damage Depth Maps

The damage depth maps for the back and front scans of the 24-ply sample are shown in Figure 4.20. The same UT C-scan segmented using the gating procedure described in Previous Work section, and in [5], is shown along with the CT scan in Figure 4.21.

The processed front scan from the code in Figure 4.20D closely duplicates the result from using the gating procedure from [5] in Figure 4.21a. This match shows that the process developed in this project functions correctly, while being more robust in handling datasets with lower sampling frequencies and variable deformation such as the dataset used to test the code. The X-ray CT scan in Figure 4.21b also matches up with the processed front scan from the code with some regions not detected by the X-ray CT scan due to resolution limitations.

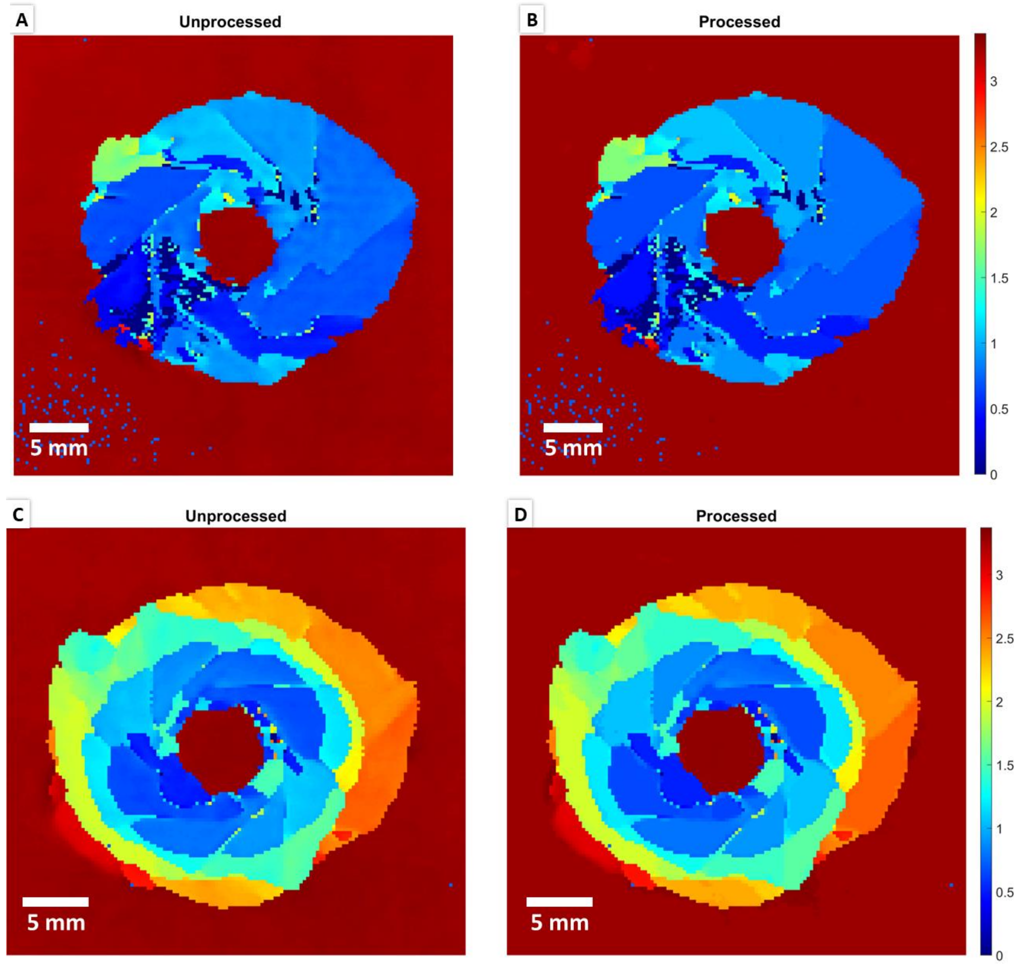


Figure 4.20: LV-162 TOF (A) back unprocessed, (B) back processed, (C) front unprocessed, (D) front processed.

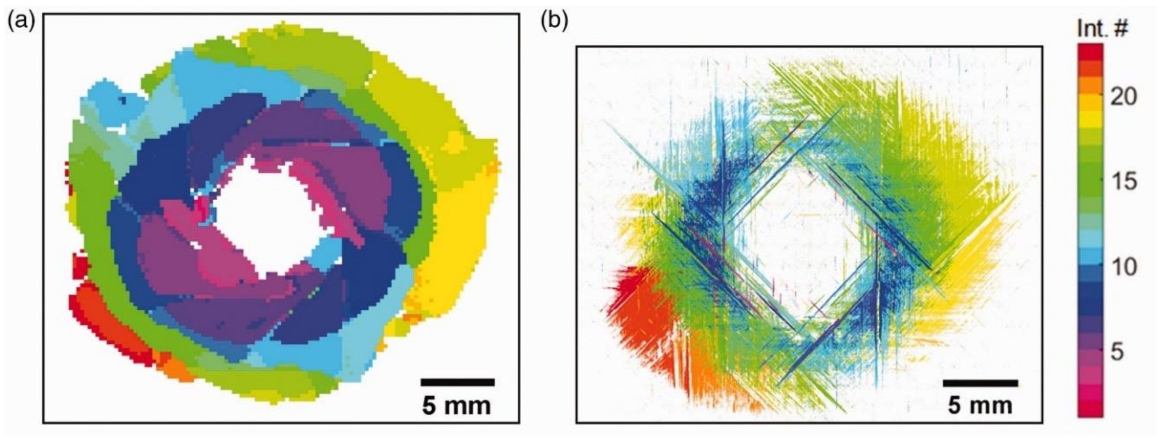


Figure 4.21: LV-162 (a) UT C-scan processed with gating procedure and (b) X-ray CT scan Reprinted with permission [5].

4.4 24-ply Hybrid C-scans and 3D C-scan Reconstruction

The hybrid C-scan and 3D C-scan reconstruction are shown in Figure 4.22. Resolution along row and column directions is 0.25 mm.

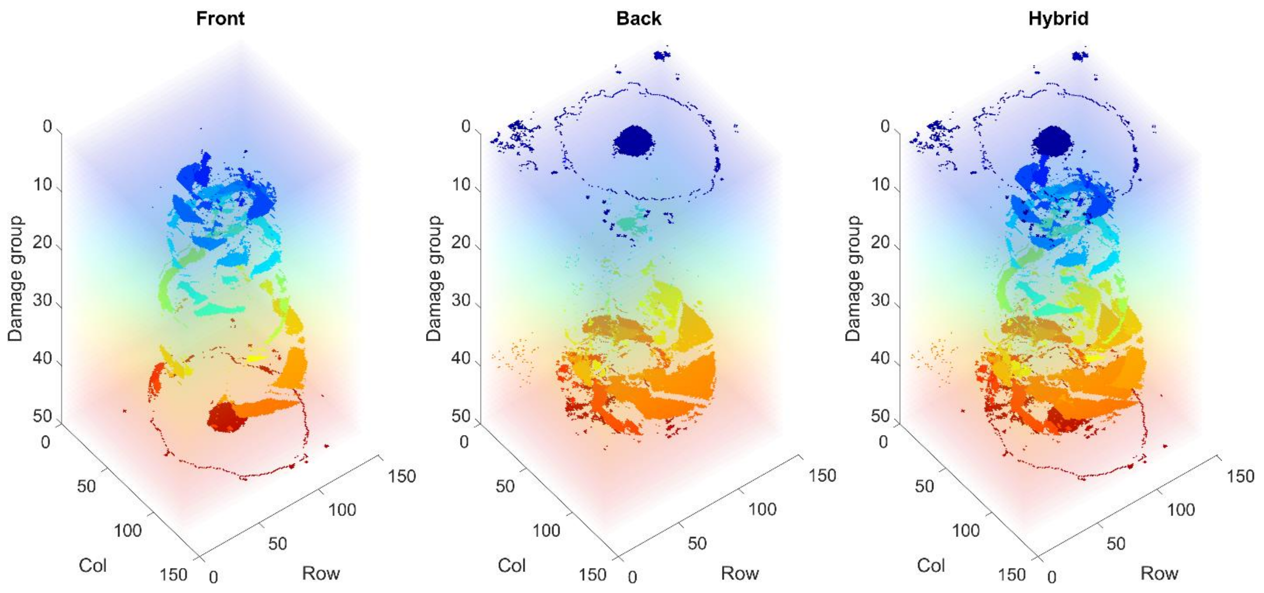


Figure 4.22: LV-162 2 front, back and hybrid C-scan 3D reconstruction.

4.5 Acknowledgements

Figure 4.1, in part, is a reprint of the figure as it appears in Segmentation of X-ray CT and Ultrasonic Scans of Impacted Composite Structures for Damage State Interpretation and Model Generation. Ellison, Andrew, PhD disseration, UC San Diego, 2020. The thesis author received permission to reprint the figure.

Figure 4.21, in part, is a reprint of the figure as it appears in Journal of Composite Materials 2020, Ellison, Andrew; Kim, Hyonny, Sage Journals, 2020. The thesis author received permission to reprint the figure.

CHAPTER 5: CONCLUSION AND FUTURE WORK

5.1 Conclusion

The results from testing the code on the dataset of 28 C-scans demonstrated successful segmentation and 3D reconstruction of the pulse-echo ultrasonic C-scans of impact damaged composite samples. Additionally, with the ten samples with front and back C-scans, the code was able to create hybrid C-scans by combining front and back side C-scans of samples and merge damage information from both scans to form a more complete picture of the damage. After cleaning up the inflection points map with image processing functions, the code is able to remove artifacts and noise from the TOF and damage depth maps. The code has tunable parameters that allow for processing of datasets with different characteristics which was demonstrated by using two different datasets, the 24-ply and the 25-ply samples, that were taken with different sampling frequencies and sampling resolution. The code is published as open source on GitHub and MathWorks File Exchange which will allow for users to improve and adapt the code for their unique application. The code is a marked improvement on the UTWin C-scan processing software with much increased details of the damage. The code also produces additional data useful to the user, not provided by the UTWin software, such as the layer segmentation, 3D reconstruction, damage area outline, inflection points map, and hybrid C-scans. These 3D datasets of the damage state are useful for validation of computational modeling of impact damage.

5.2 Future Work

The code has a couple different applications for work by future researchers including building finite element (FE) models of impact damaged composites and filling in shadowed regions of damage.

The 3D reconstruction of the damage can be used to create detailed and realistic FE models of impact damaged as well as impact damaged then repaired composite samples, where the model can be updated and tuned to match testing results from compression-strength-after-impact (CSAI) tests. In turn, parameters from the updated and tuned models can be used to accurately predict CSAI strength of composites with impact damage and impact damage that has been repaired. The CSAI tests of the 25-ply samples used as part of the dataset in this project started in August 2021 and are currently ongoing.

The code written in this project can be combined with a shadow delamination extension code, as described in Ellison [5], that was written previously by another researcher to predict the damage regions shadowed by layers above to create a more complete 3D reconstruction of the damaged region. The shadow extension code uses information about the ply layup direction to predict the shape and size of the shadowed region. Combining the two methods together, shadow extension and the hybrid C-scan from the current project could form a complete picture of the damage region that could be used for the FE model work described above.

Appendix A: LINKS TO CODE AND DATA

Data:

UC San Diego Research Data Collections (RDCP):
https://doi.org/10.6075/J0G16118

Code:

GitHub:
https://github.com/jessica-chan-5/ultrasonic-c-scan-segmentation-for-composites

MathWorks File Exchange:
https://www.mathworks.com/matlabcentral/fileexchange/125985-ultrasonic-c-scan-segmentation-for-composites

Appendix B: DAMAGE DEPTH MAPS

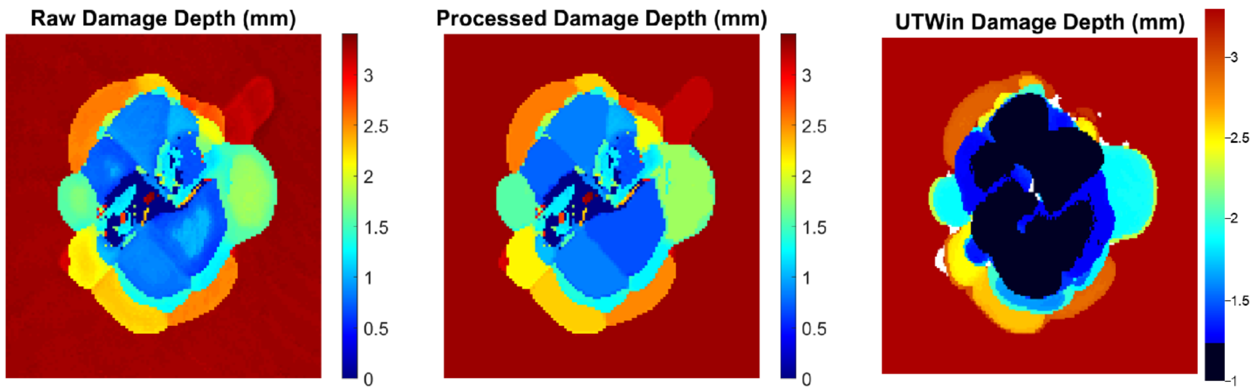


Figure B.1: BL-H-15J-1 Damage Depth. Dimensions (height x width): 51 x 51 mm.

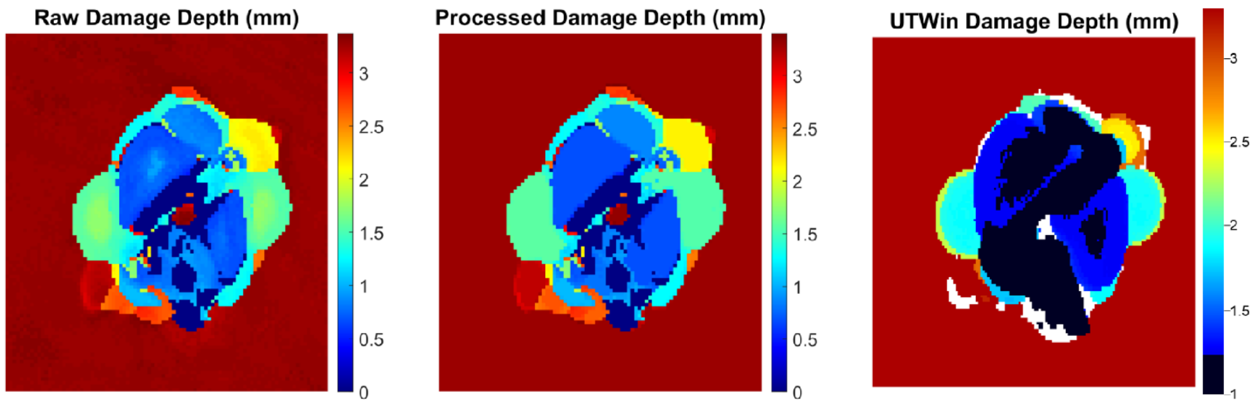


Figure B.2: CONT-H-10J-2 Damage Depth. Dimensions (height x width): 39 x 38 mm.

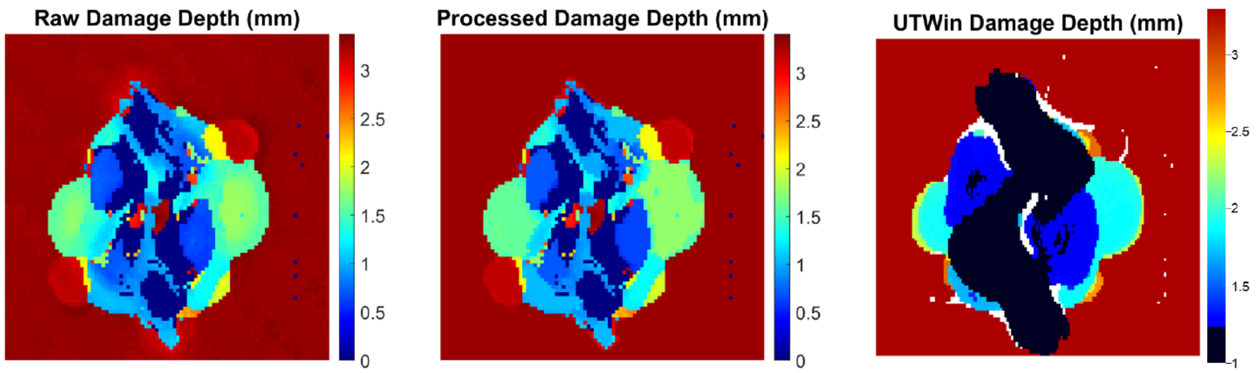


Figure B.3: CONT-H-10J-3 Damage Depth. Dimensions (height x width): 35 x 38 mm.

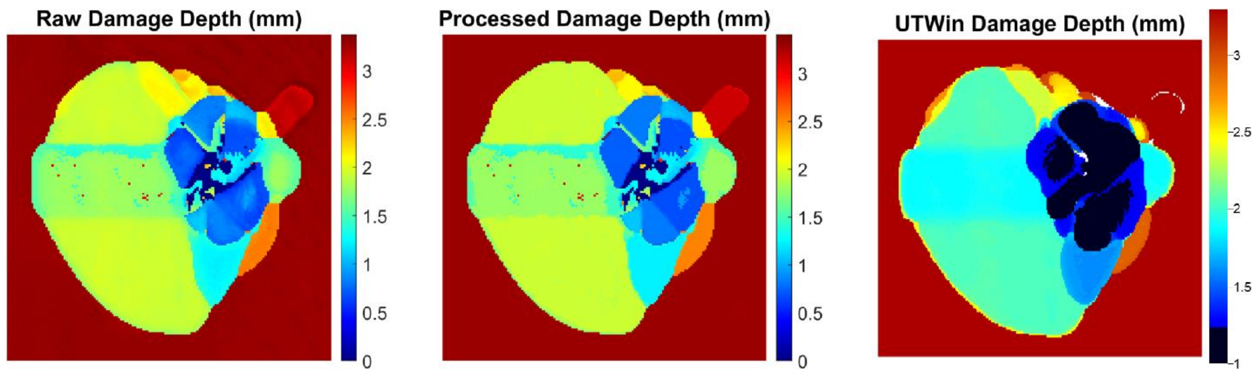


Figure B.4: CONT-H-20J-1 Damage Depth. Dimensions (height x width): 63 x 68 mm.

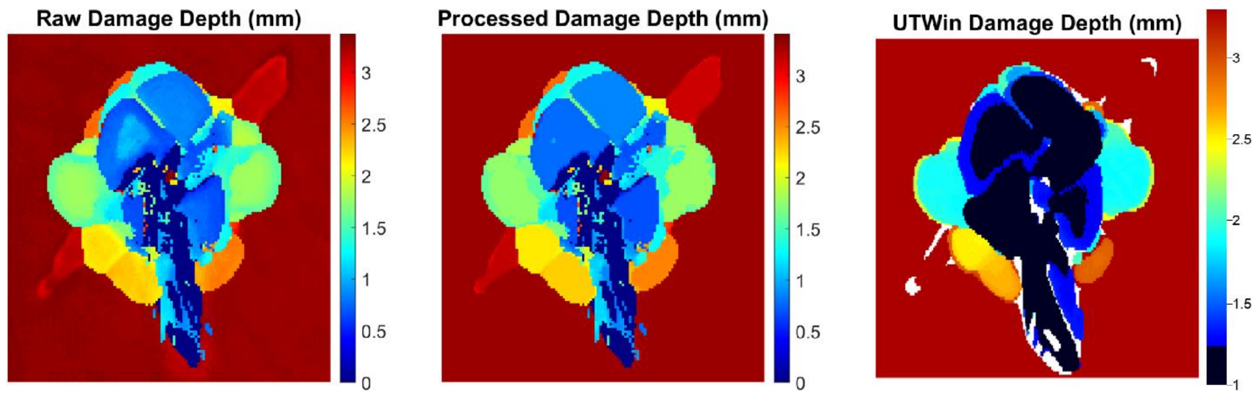


Figure B.5: CONT-H-20J-2 Damage Depth. Dimensions (height x width): 55 x 55 mm.

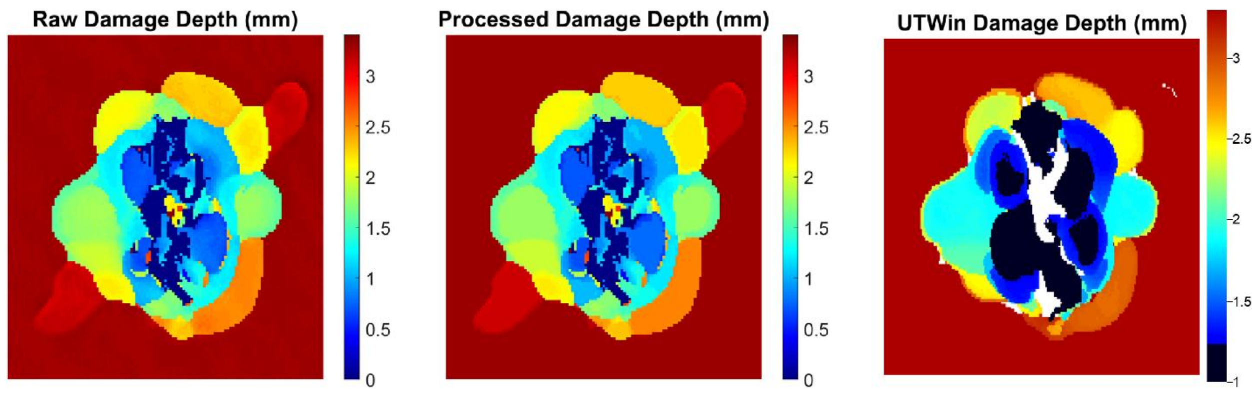


Figure B.6: RPR-H-20J-3 Damage Depth. Dimensions (height x width): 51 x 51 mm.

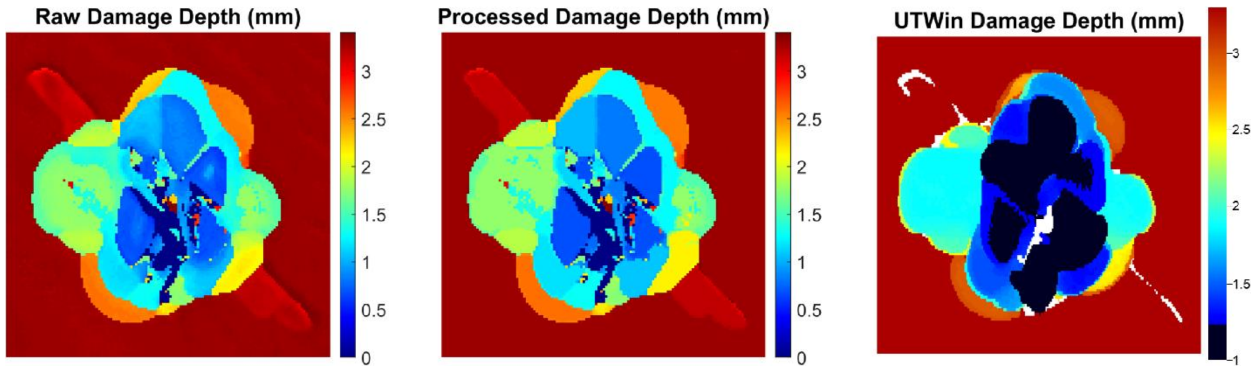


Figure B.7: RPR-H-20J-2 Damage Depth. Dimensions (height x width): 51 x 55 mm.

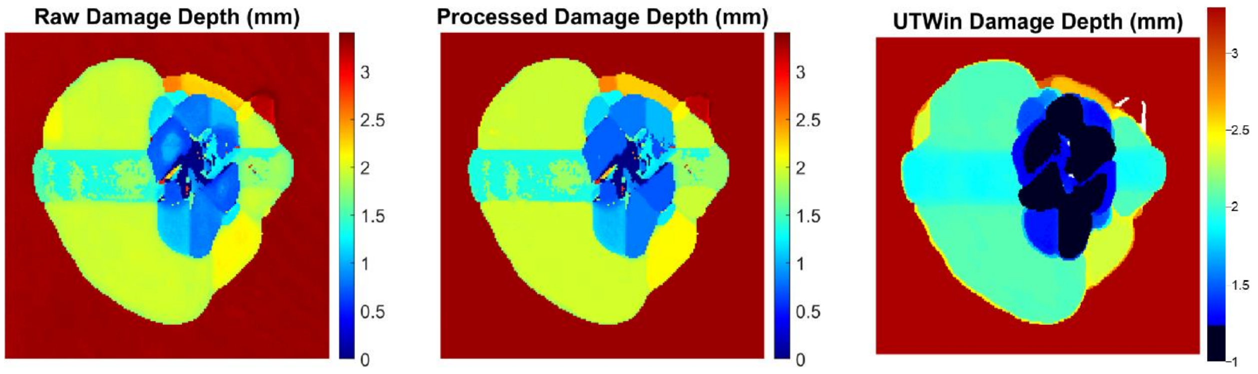


Figure B.8: RPR-H-20J-3 Damage Depth. Dimensions (height x width): 71 x 76 mm.

Appendix C: 3D RECONSTRUCTIONS OF C-SCANS

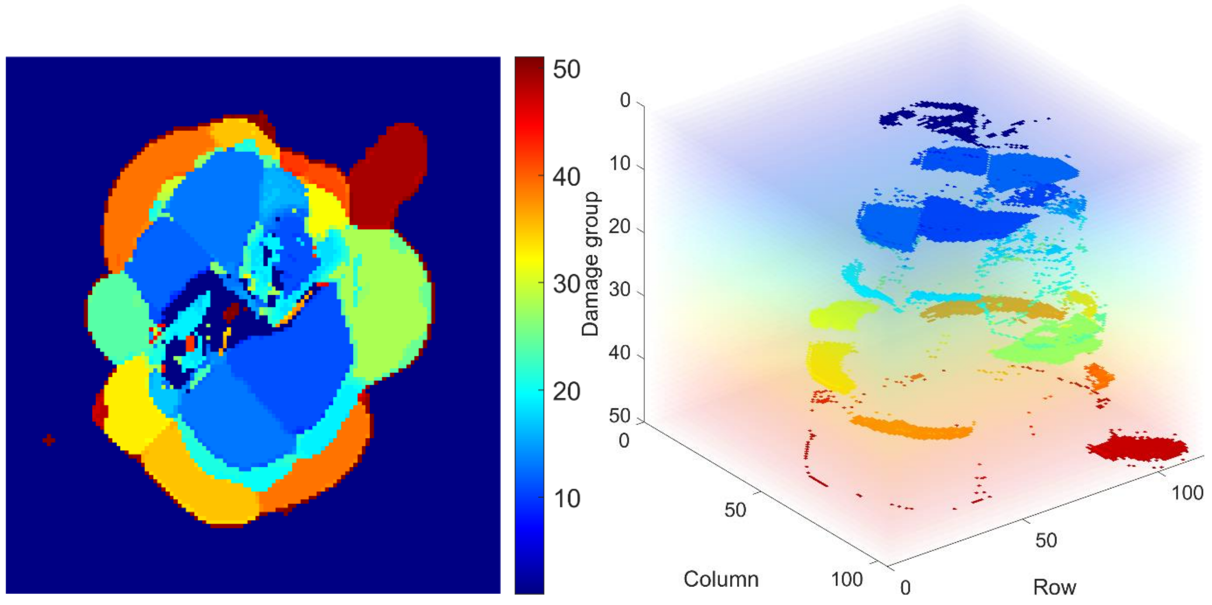


Figure C.1: BL-H-15J-1 3D Reconstruction of C-scan.

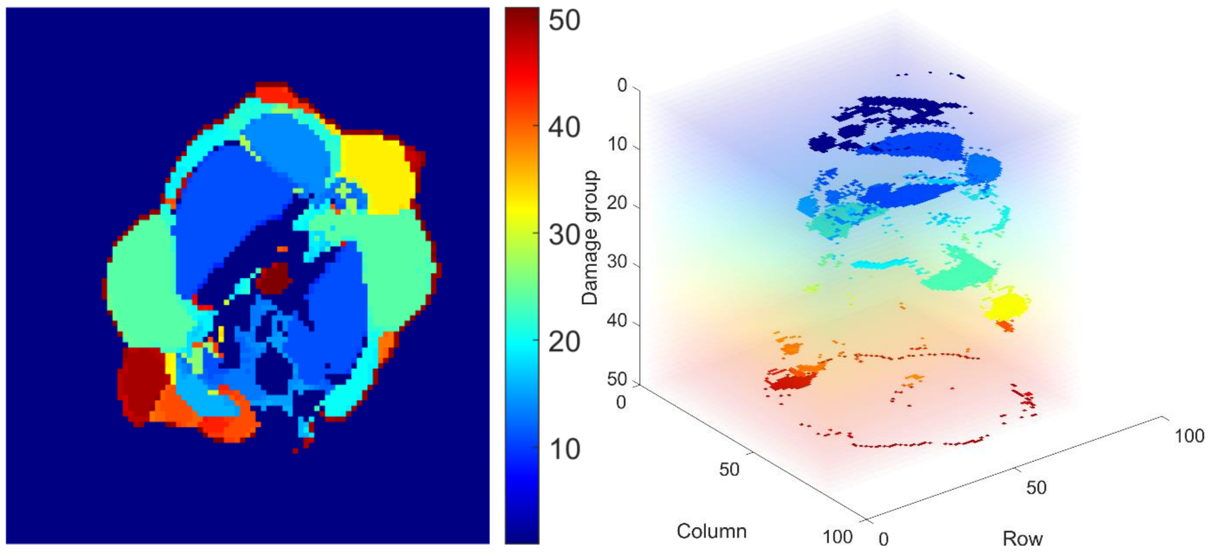


Figure C.2: CONT-H-10J-2 3D Reconstruction of C-scan.

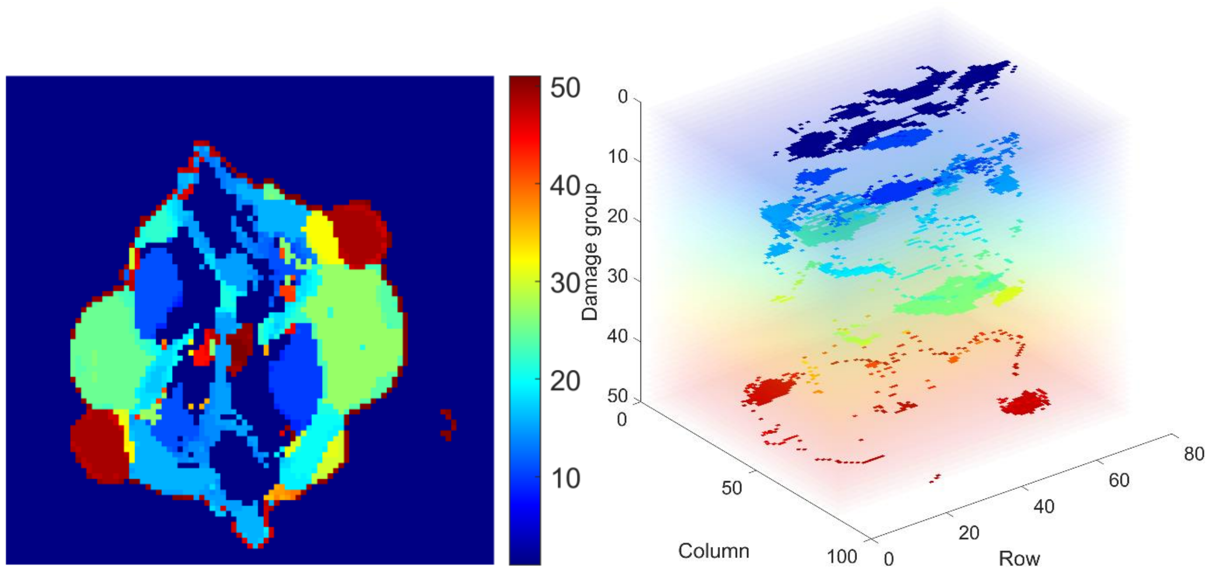


Figure C.3: CONT-H-10J-3 3D Reconstruction of C-scan.

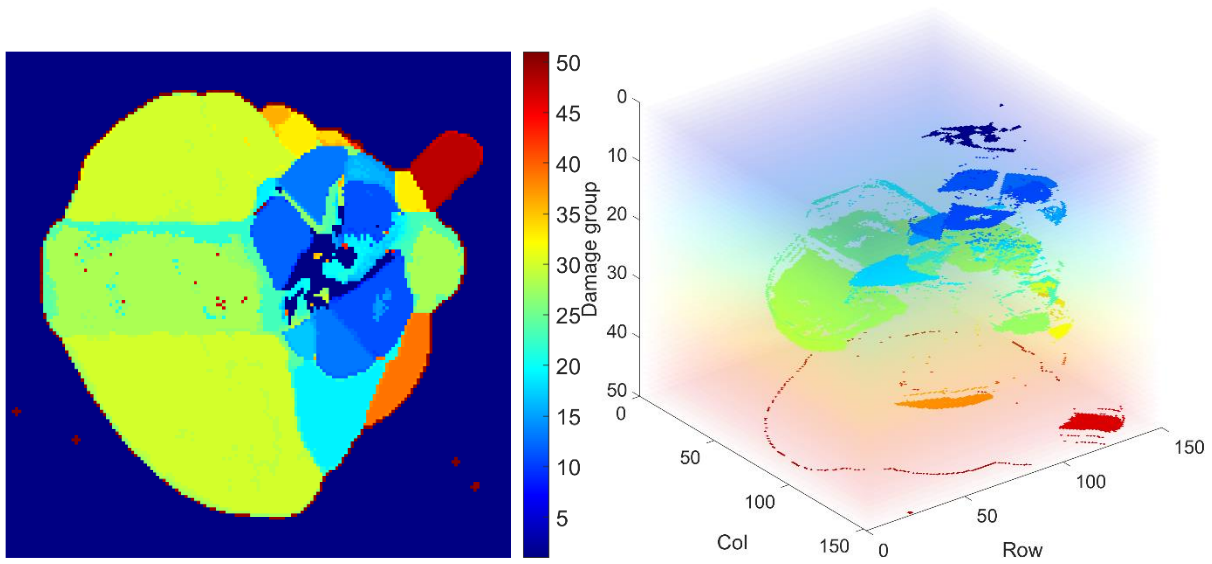


Figure C.4: CONT-H-20J-1 3D Reconstruction of C-scan.

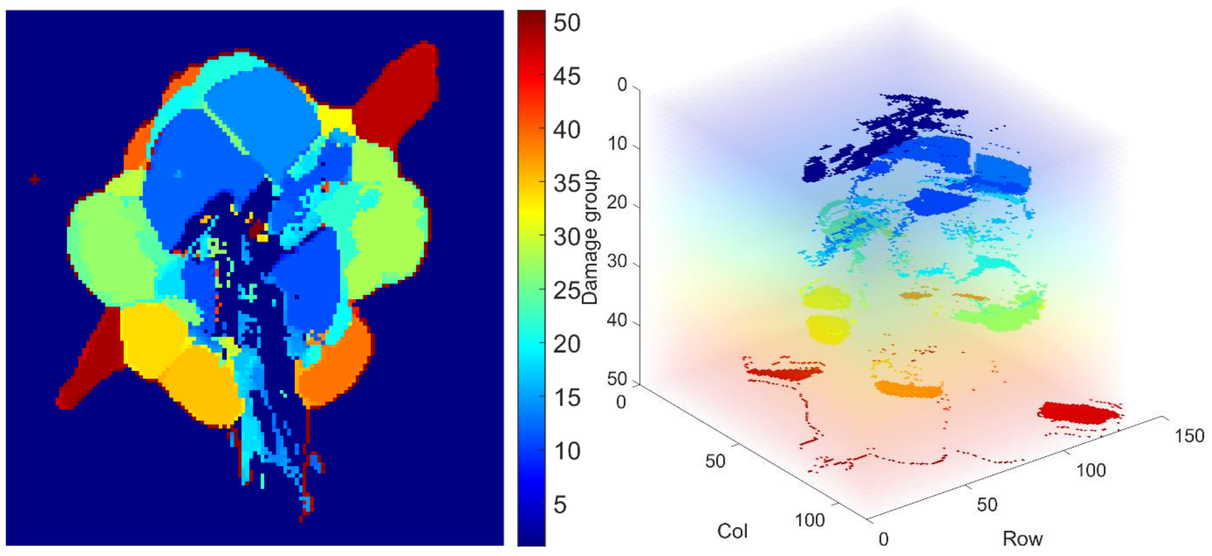


Figure C.5: CONT-H-20J-2 3D Reconstruction of C-scan.

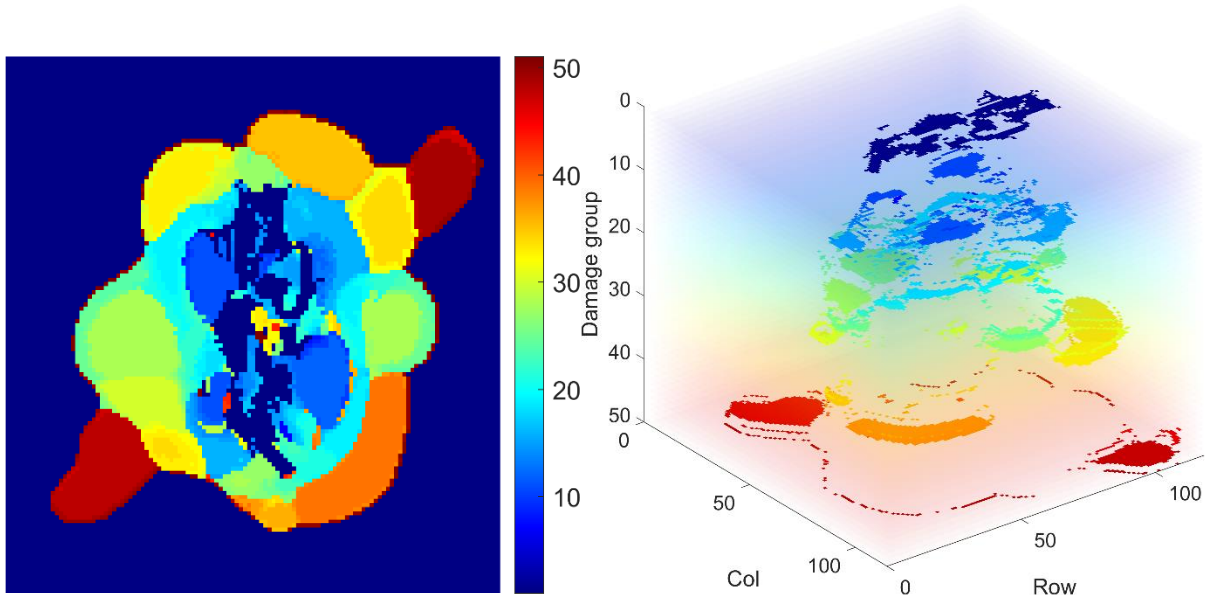


Figure C.6: CONT-H-20J-3 3D Reconstruction of C-scan.

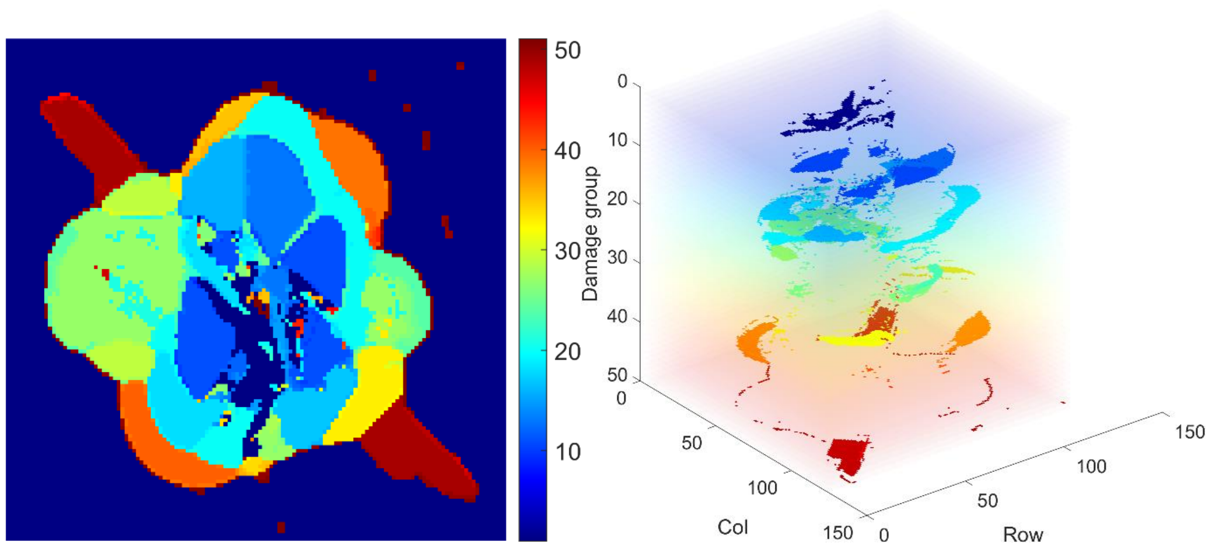


Figure C.7: RPR-H-20J-2 3D Reconstruction of C-scan.

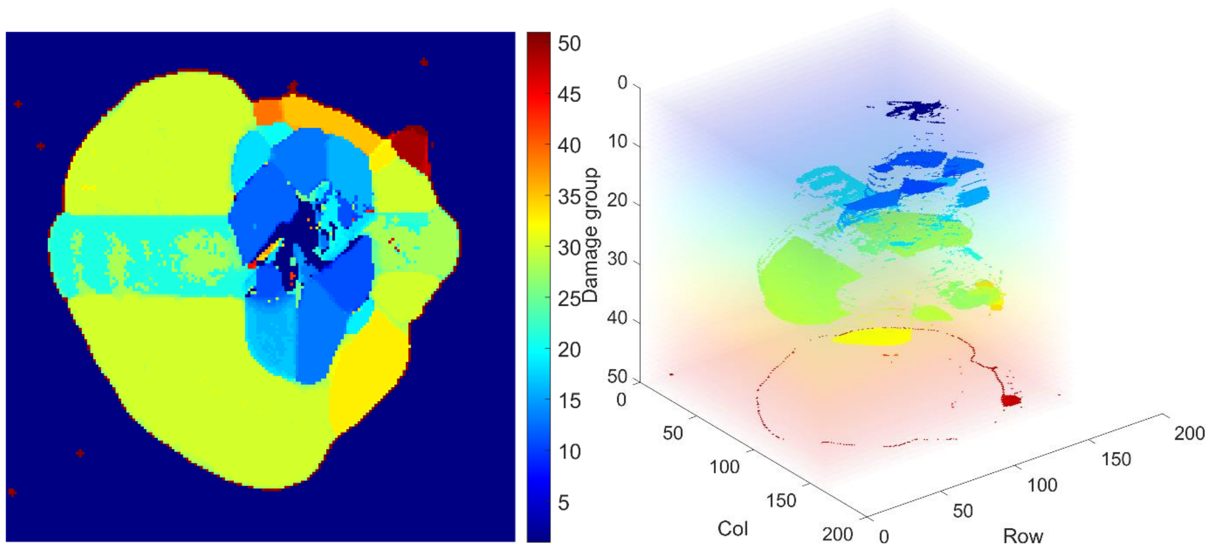


Figure C.8: RPR-H-20J-3 3D Reconstruction of C-scan.

Appendix D: Morphological TOF Processing Steps

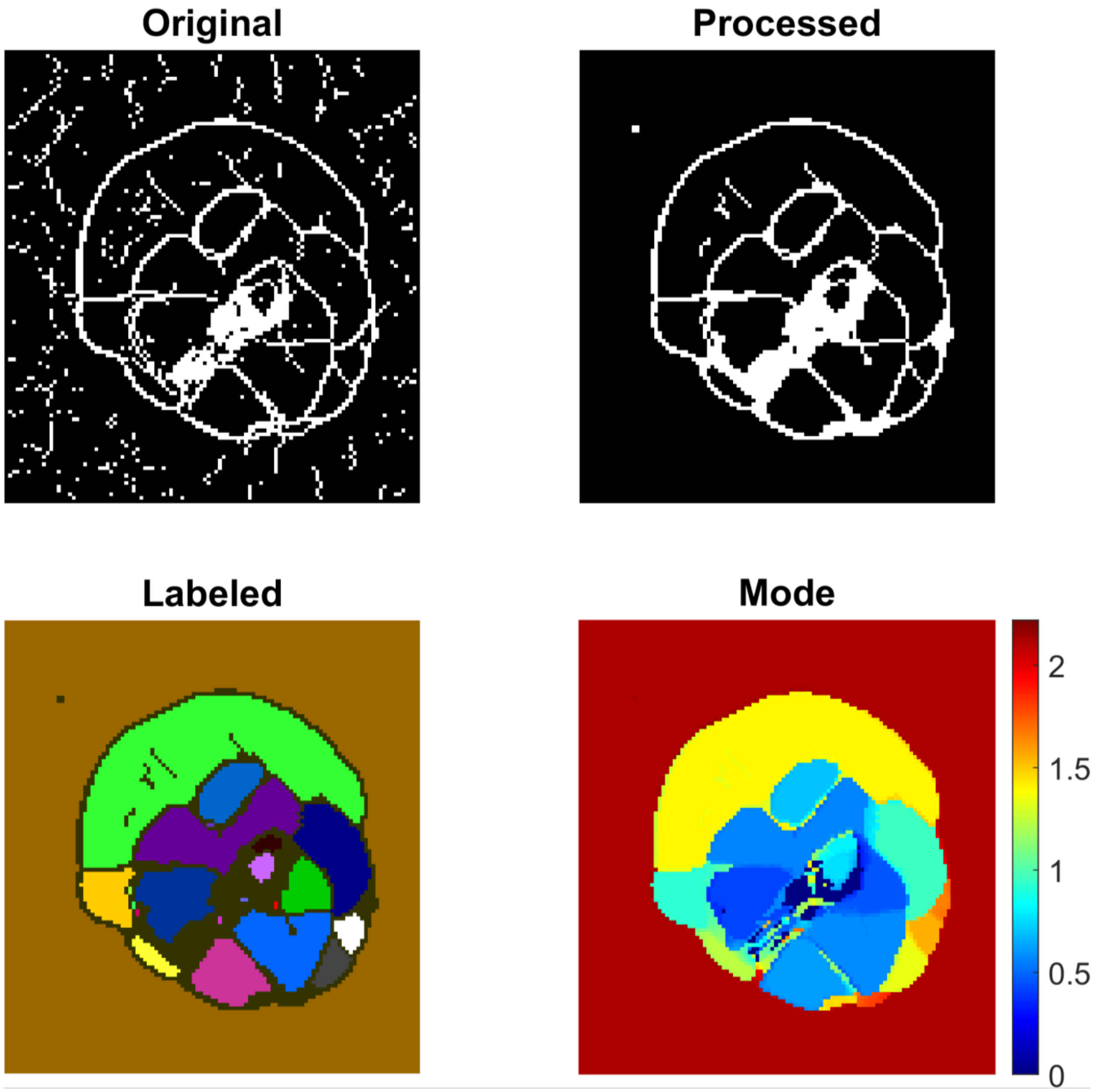


Figure D.1: BL-S-10J-2 Morphological TOF processing steps.

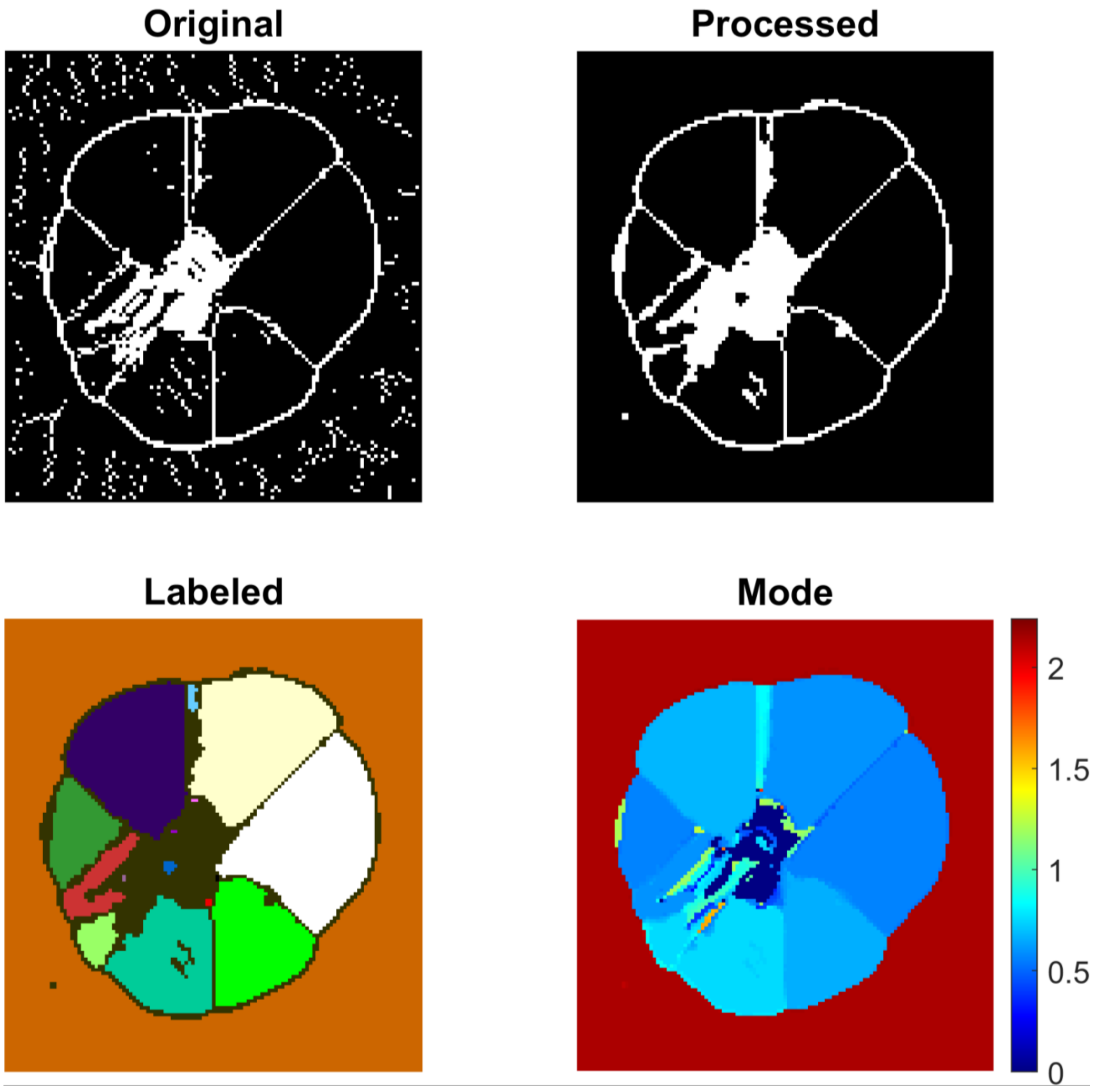


Figure D.2: BL-S-10J-2 Back Morphological TOF processing steps.

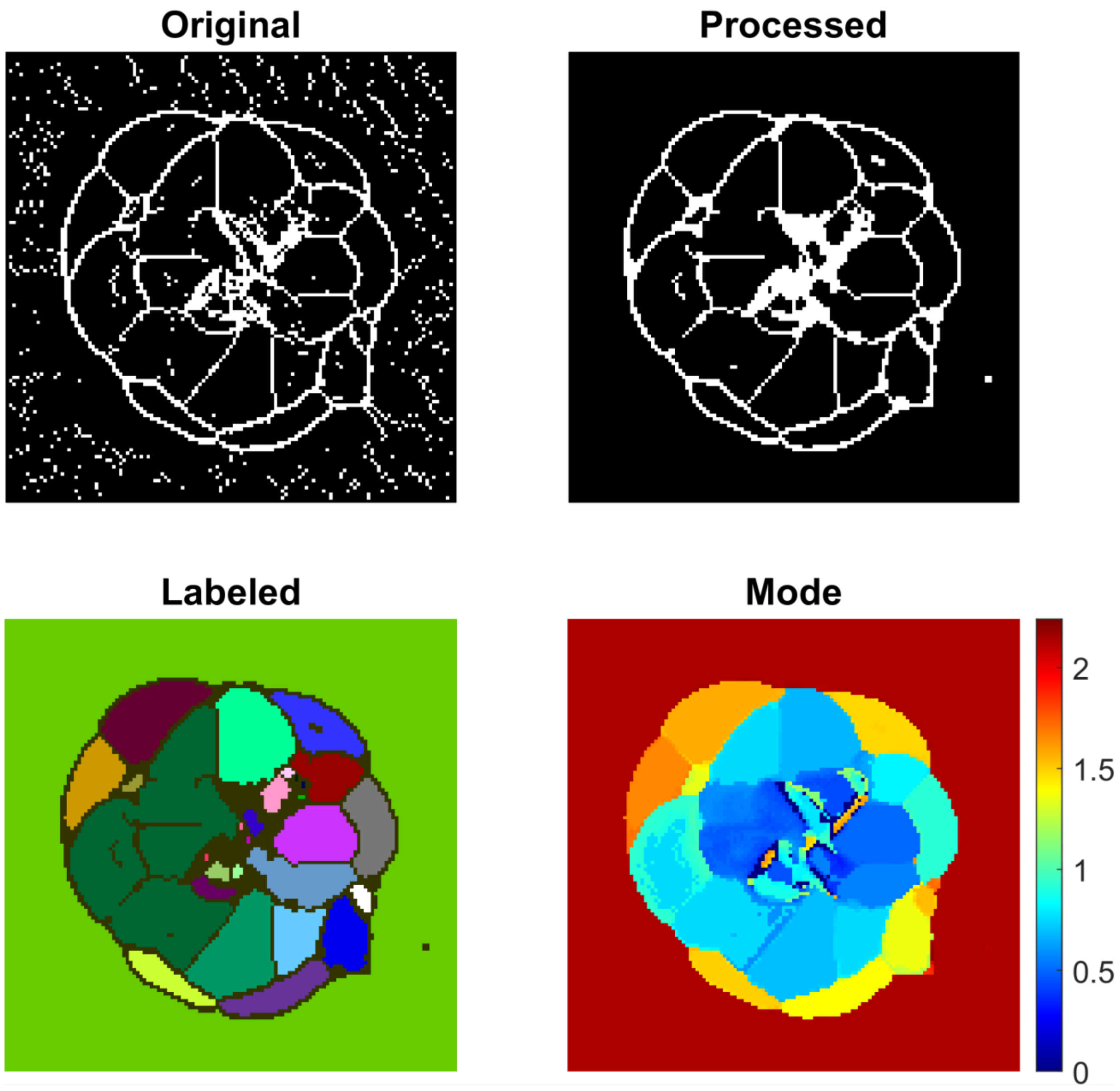


Figure D.3: BL-S-15J-2 Morphological TOF processing steps.

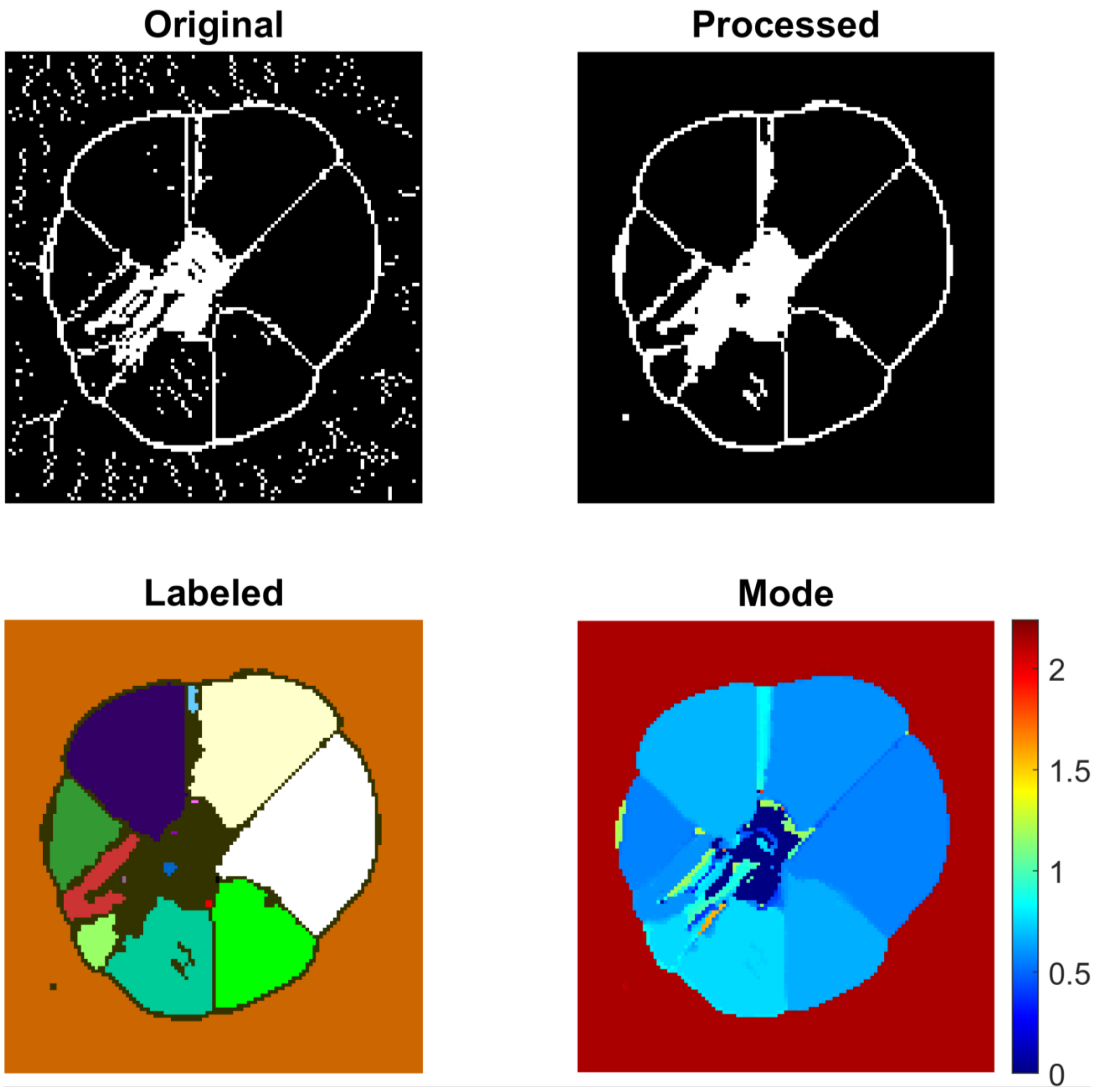


Figure D.4: BL-S-15J-2 Back Morphological TOF processing steps.

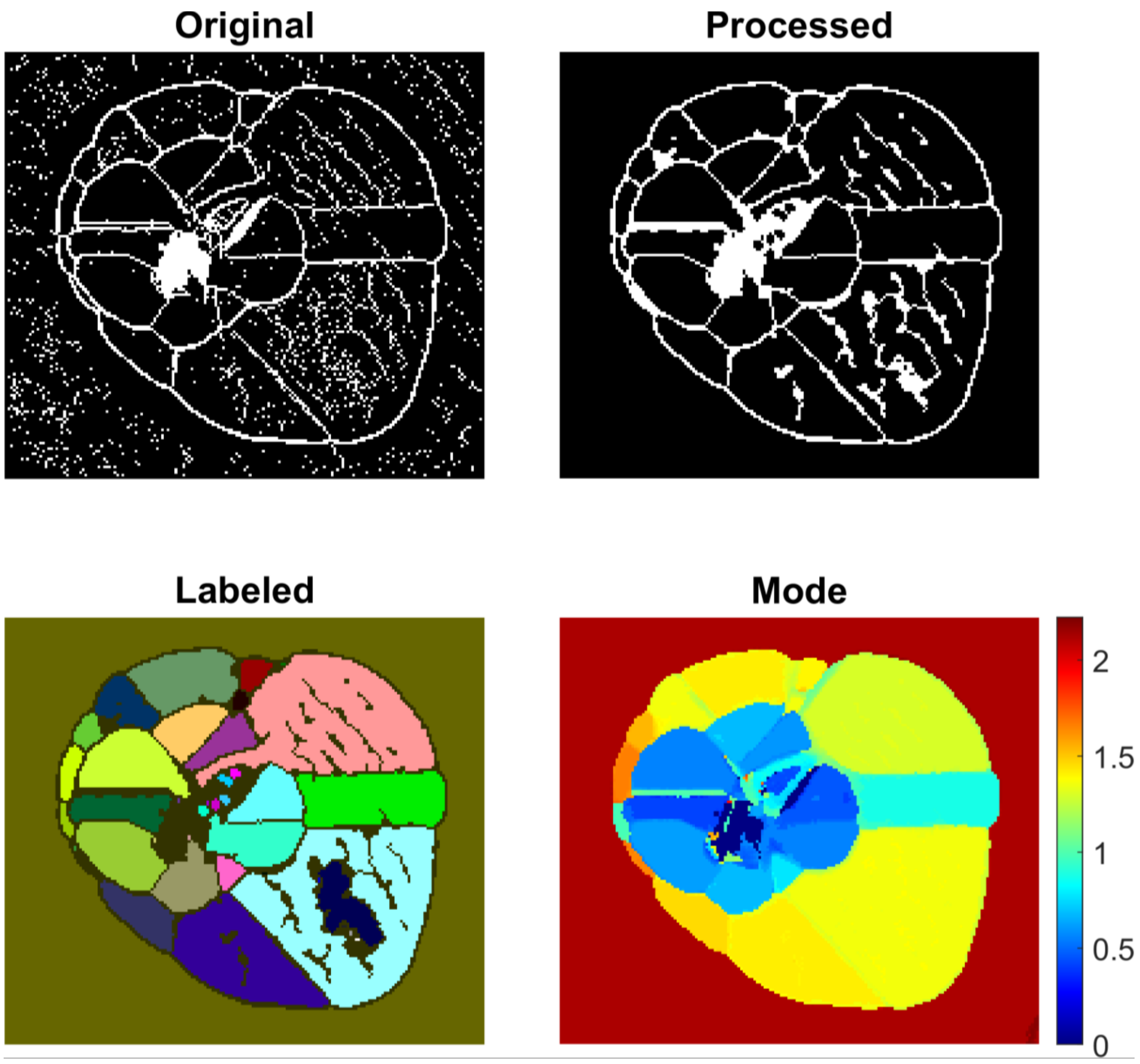


Figure D.5: BL-S-20J-2 Morphological TOF processing steps.

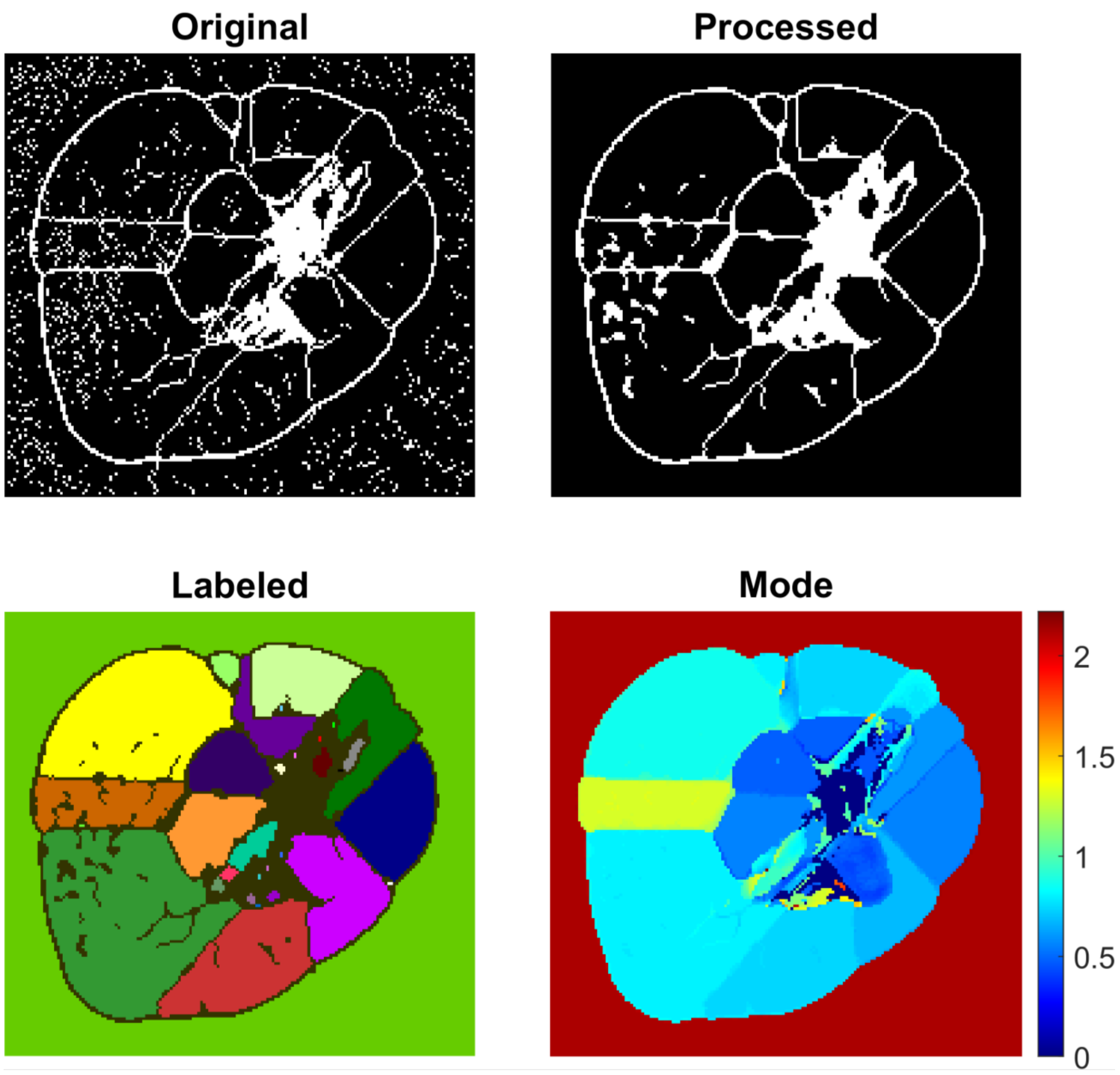


Figure D.6: BL-S-20J-2 Back Morphological TOF processing steps.

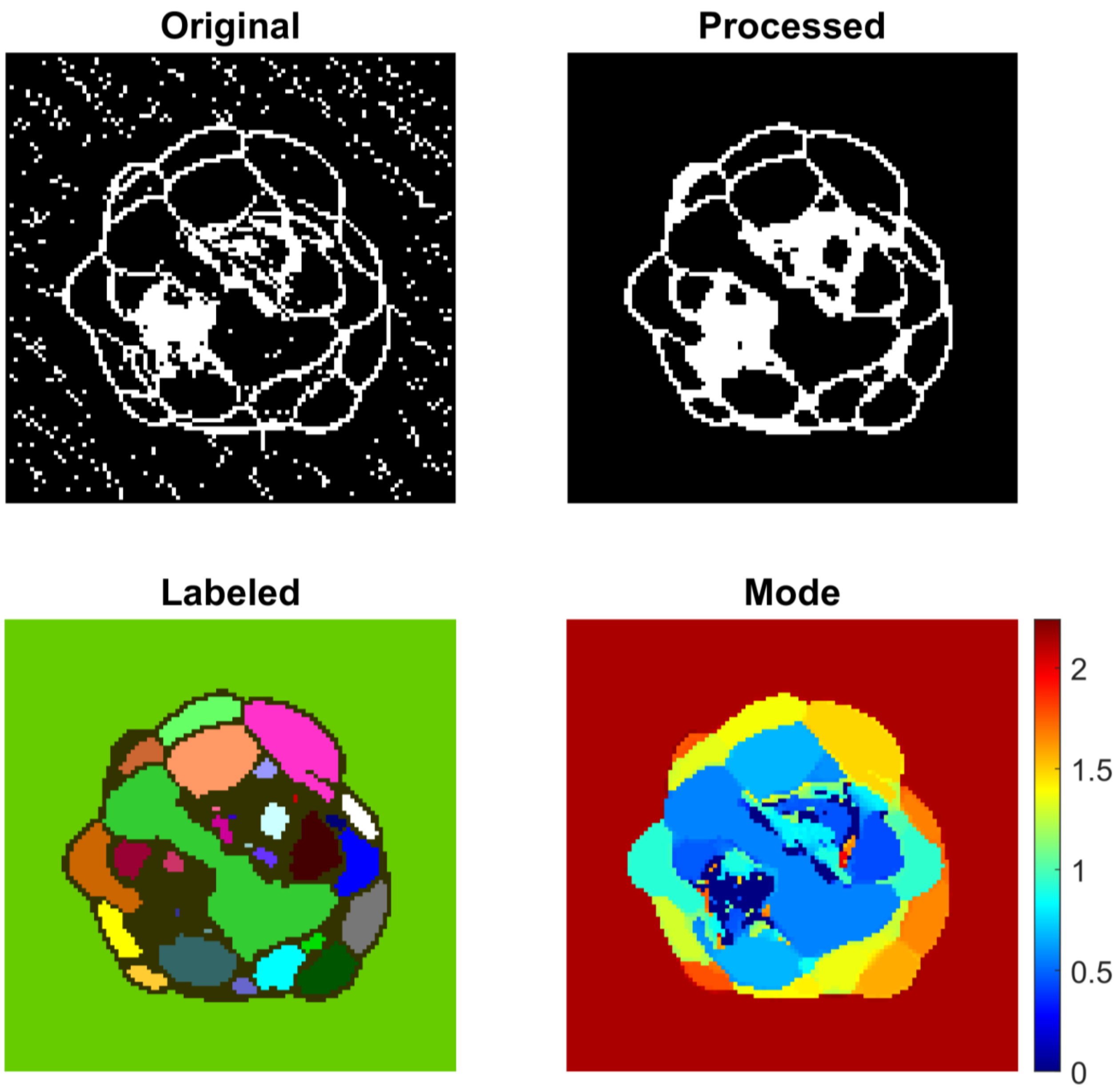


Figure D.7: CONT-S-10J-2 Morphological TOF processing steps.

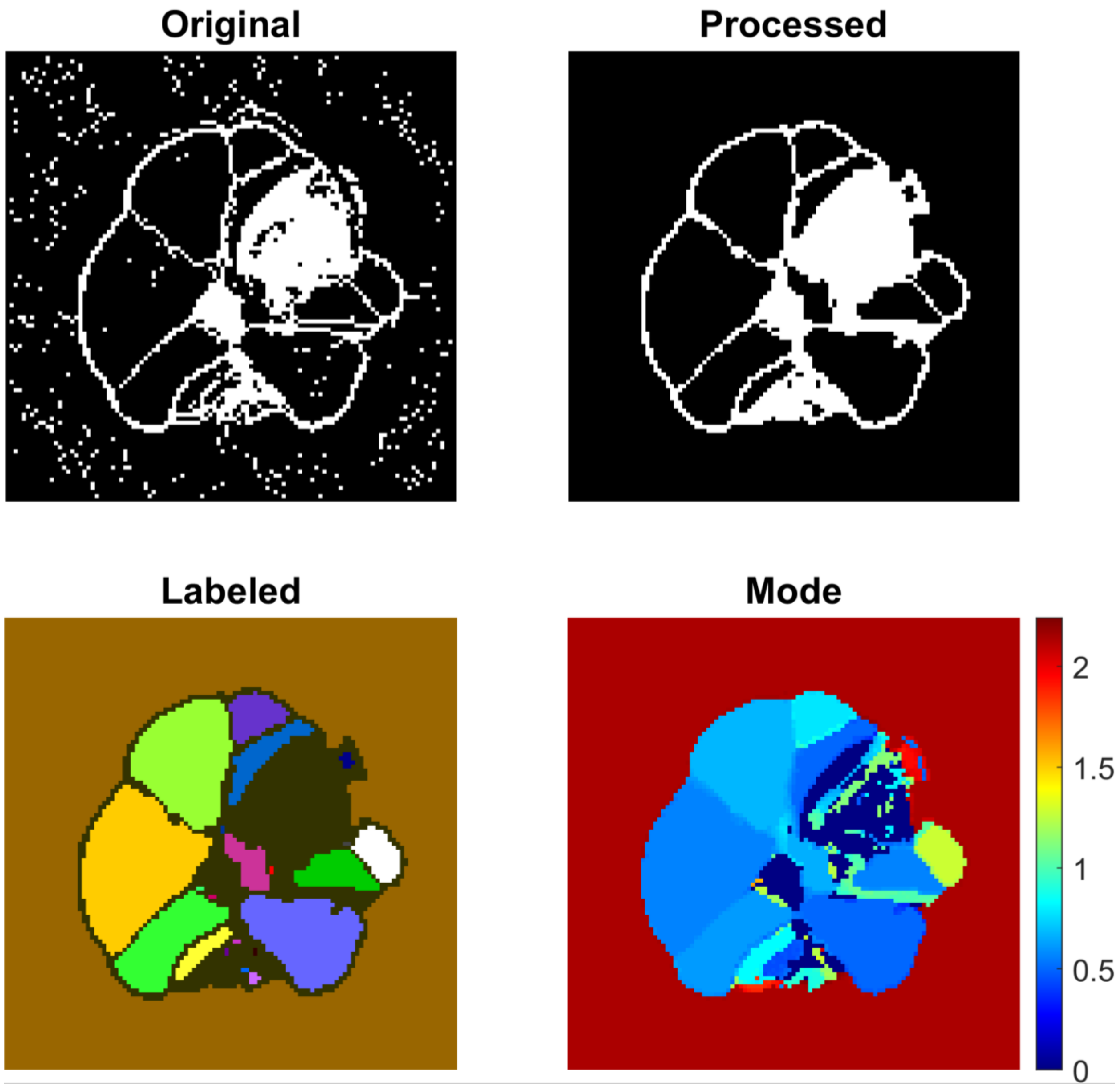


Figure D.8: CONT-S-10J-2 Back Morphological TOF processing steps.

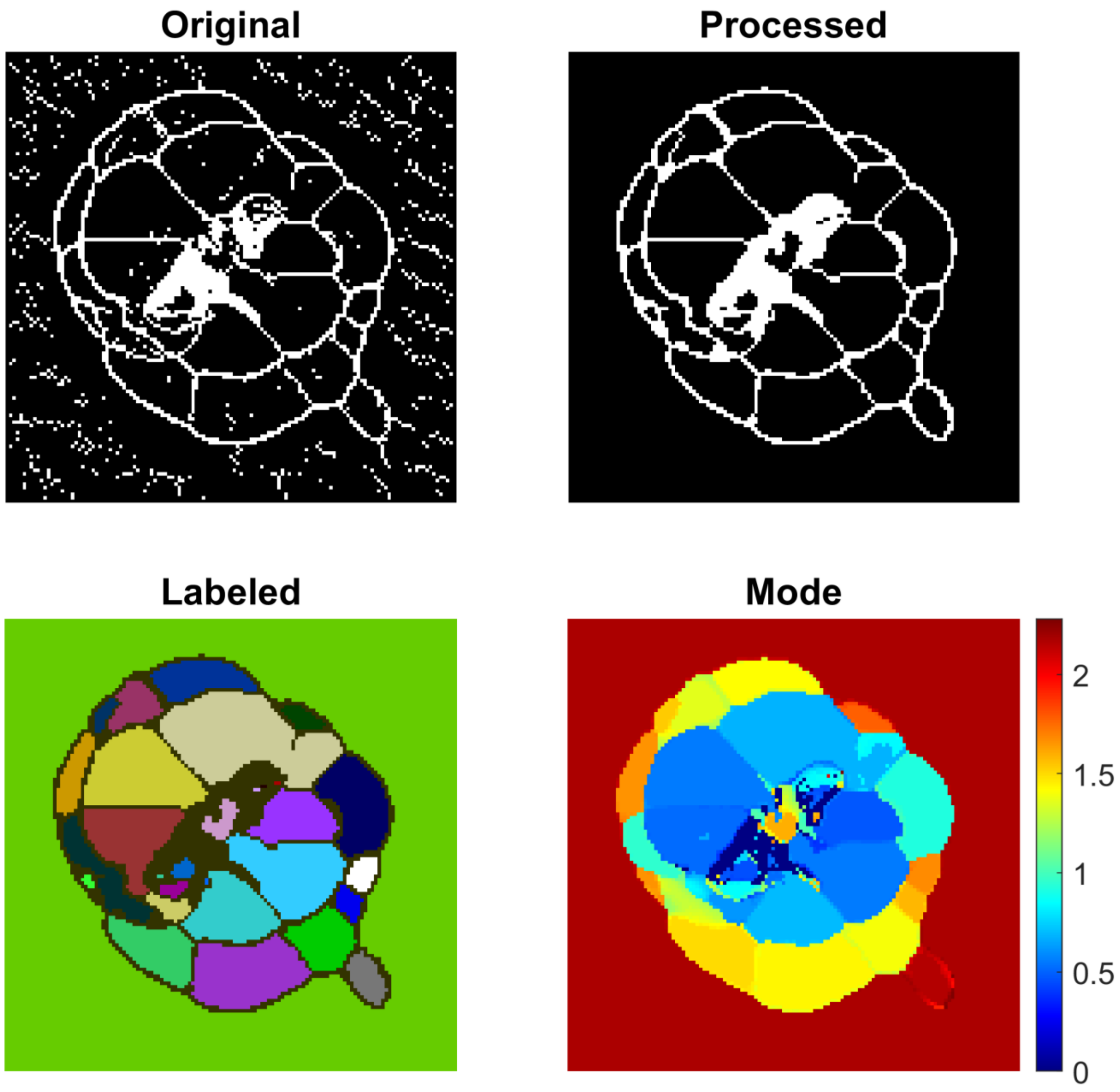


Figure D.9: CONT-S-15J-2 Morphological TOF processing steps.

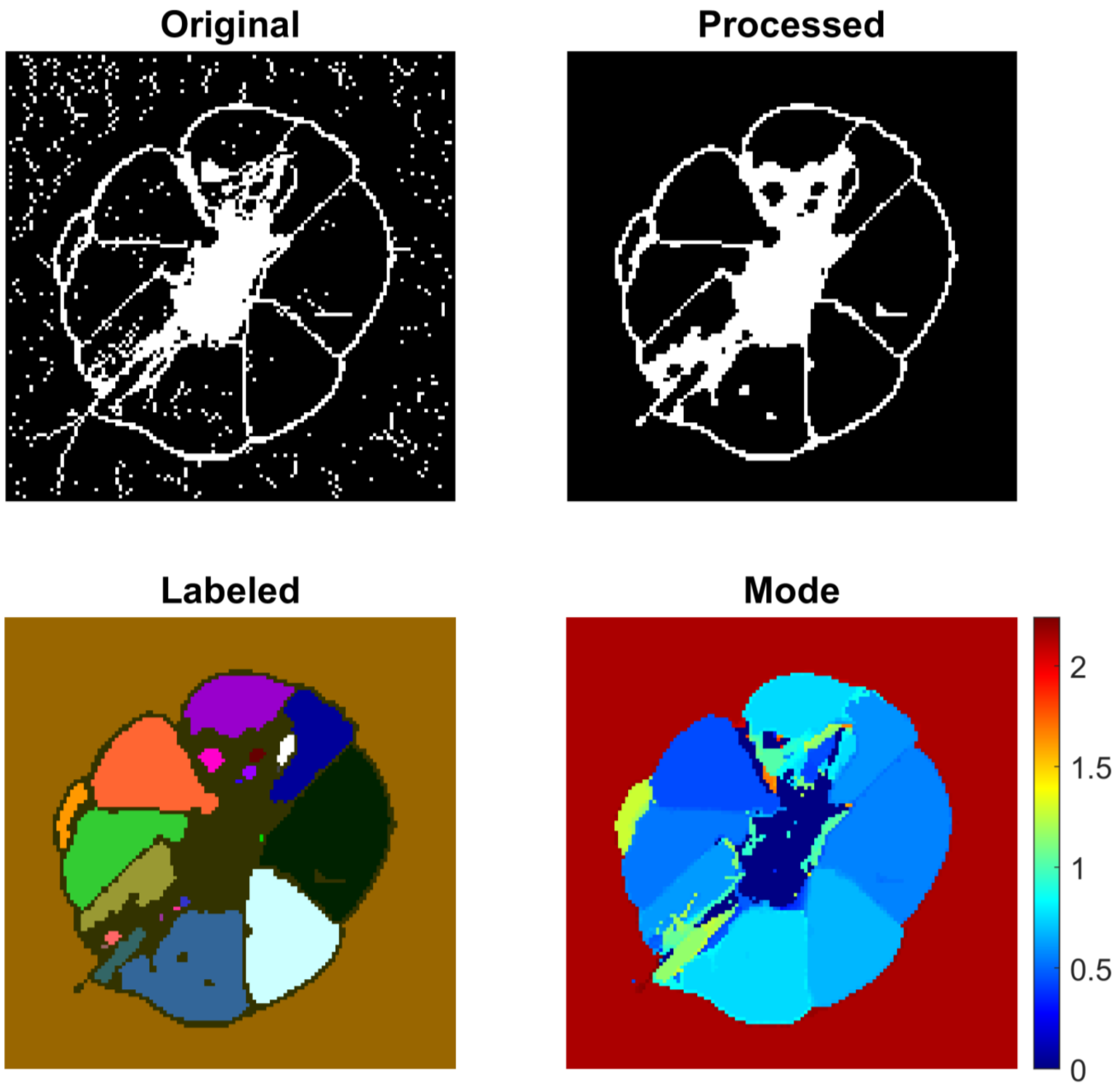


Figure D.10: CONT-S-15J-2 Back Morphological TOF processing steps.

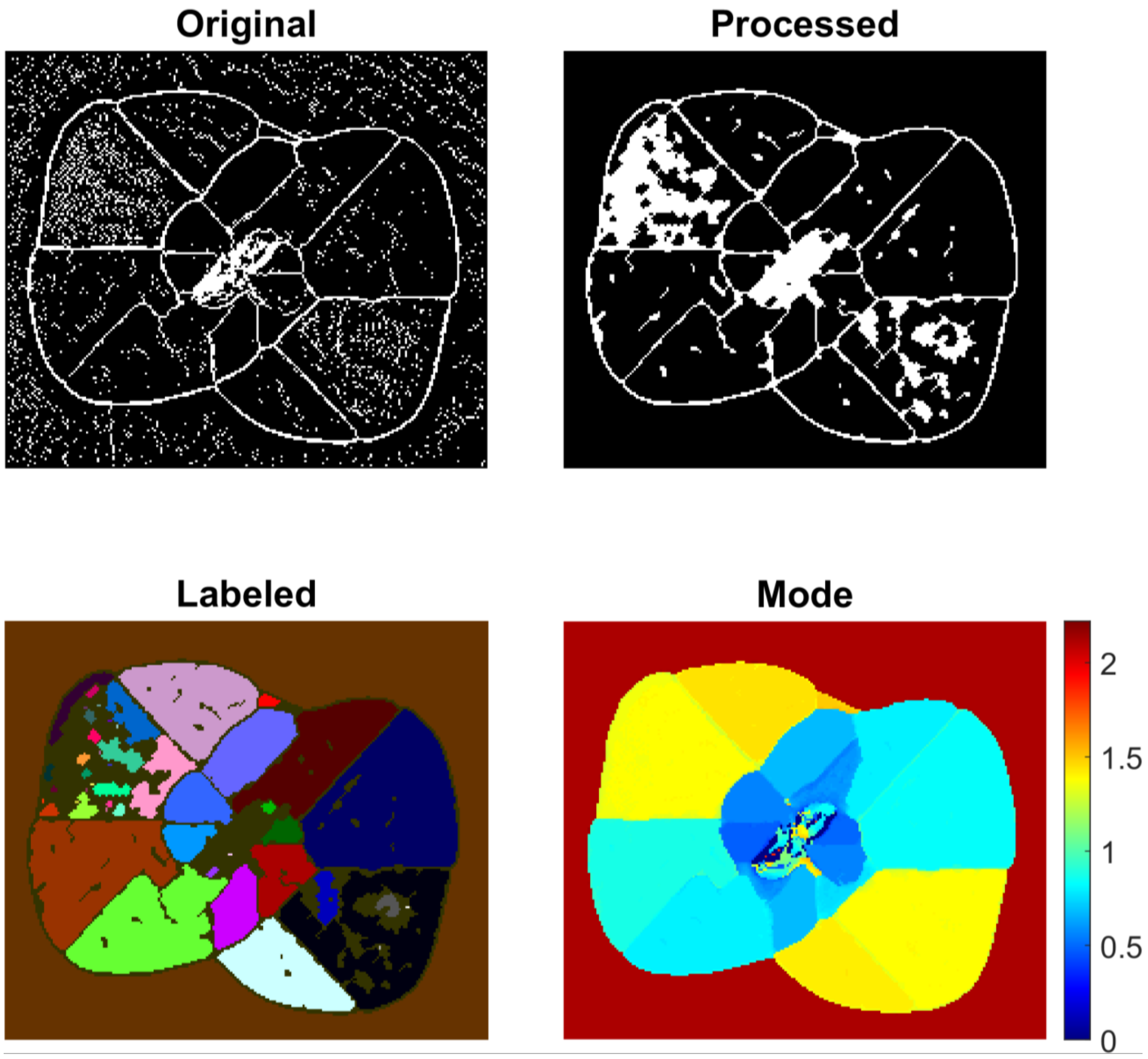


Figure D.11: CONT-S-20J-2 Morphological TOF processing steps.

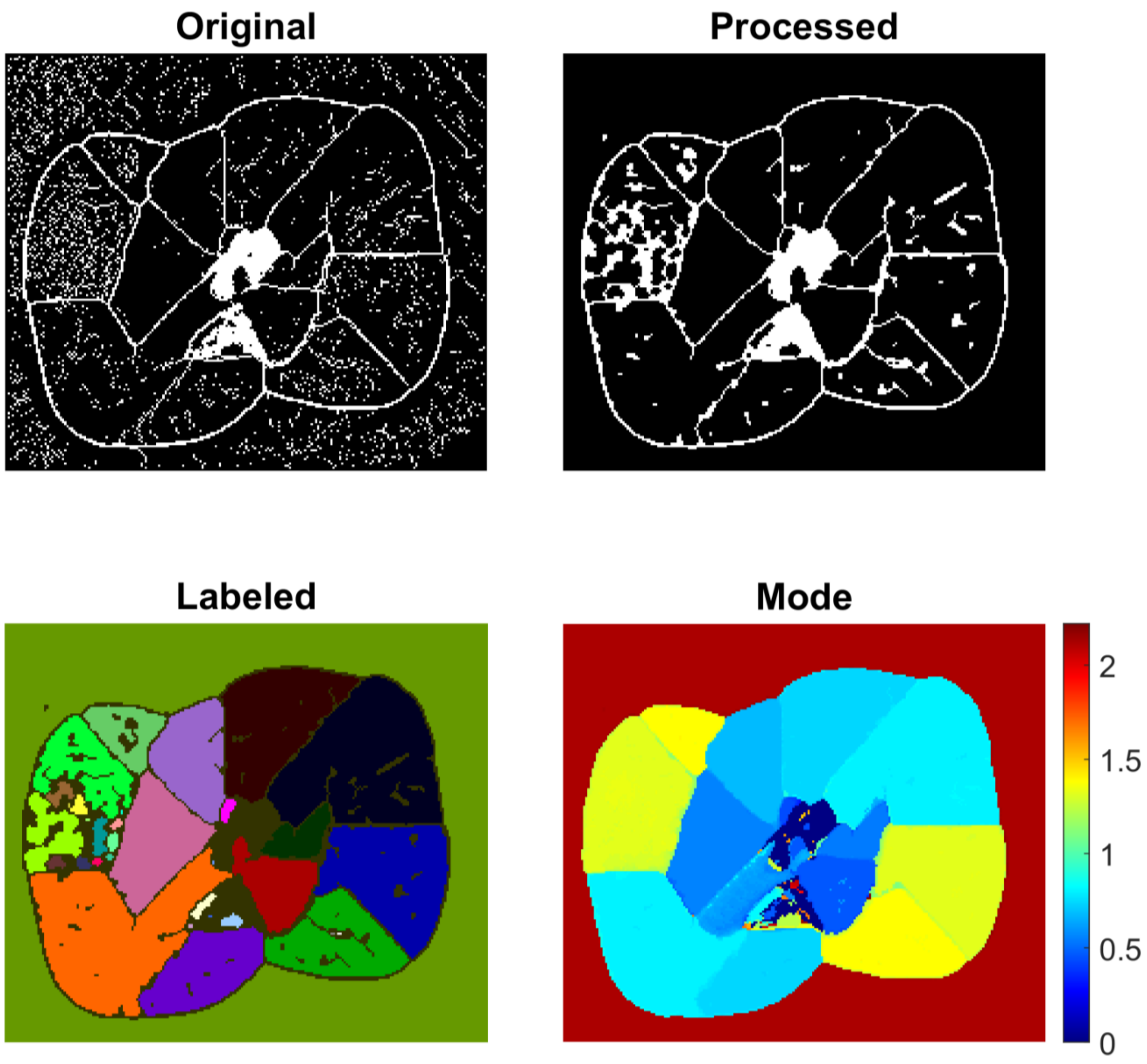


Figure D.12: CONT-S-20J-2 Back Morphological TOF processing steps.

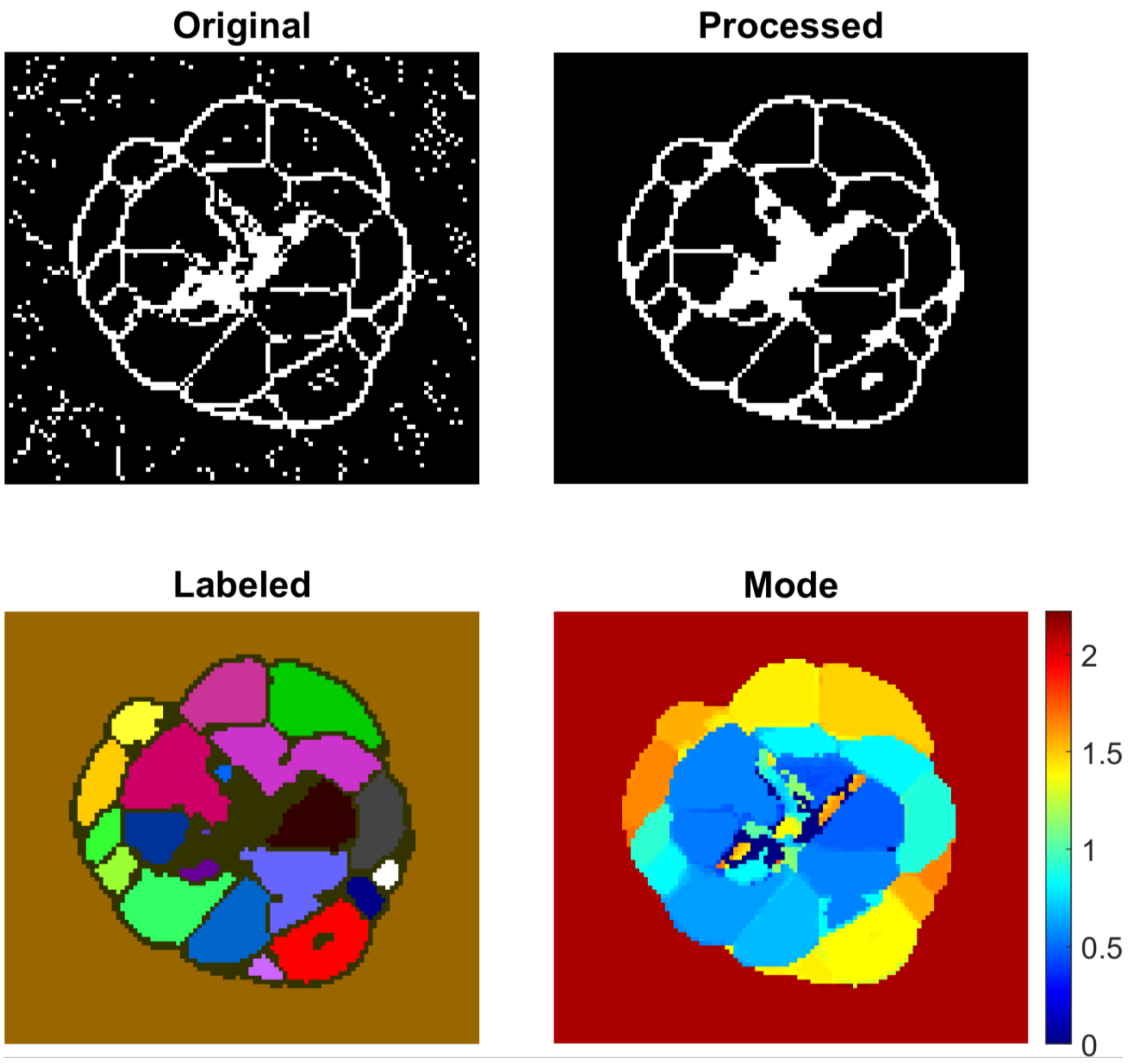


Figure D.13: RPR-S-10J-2 Morphological TOF processing steps.

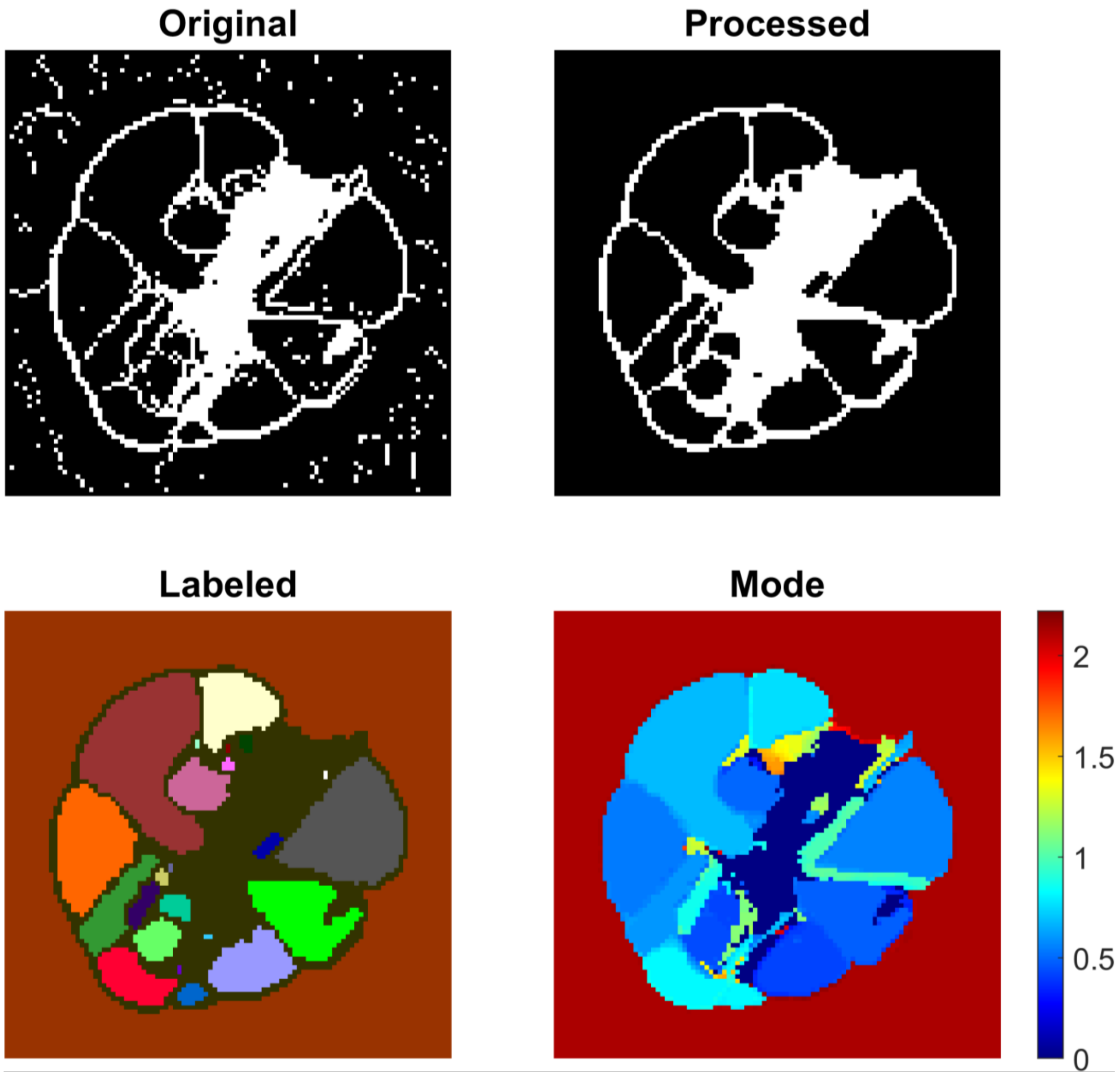


Figure D.14: RPR-S-10J-2 Back Morphological TOF processing steps.

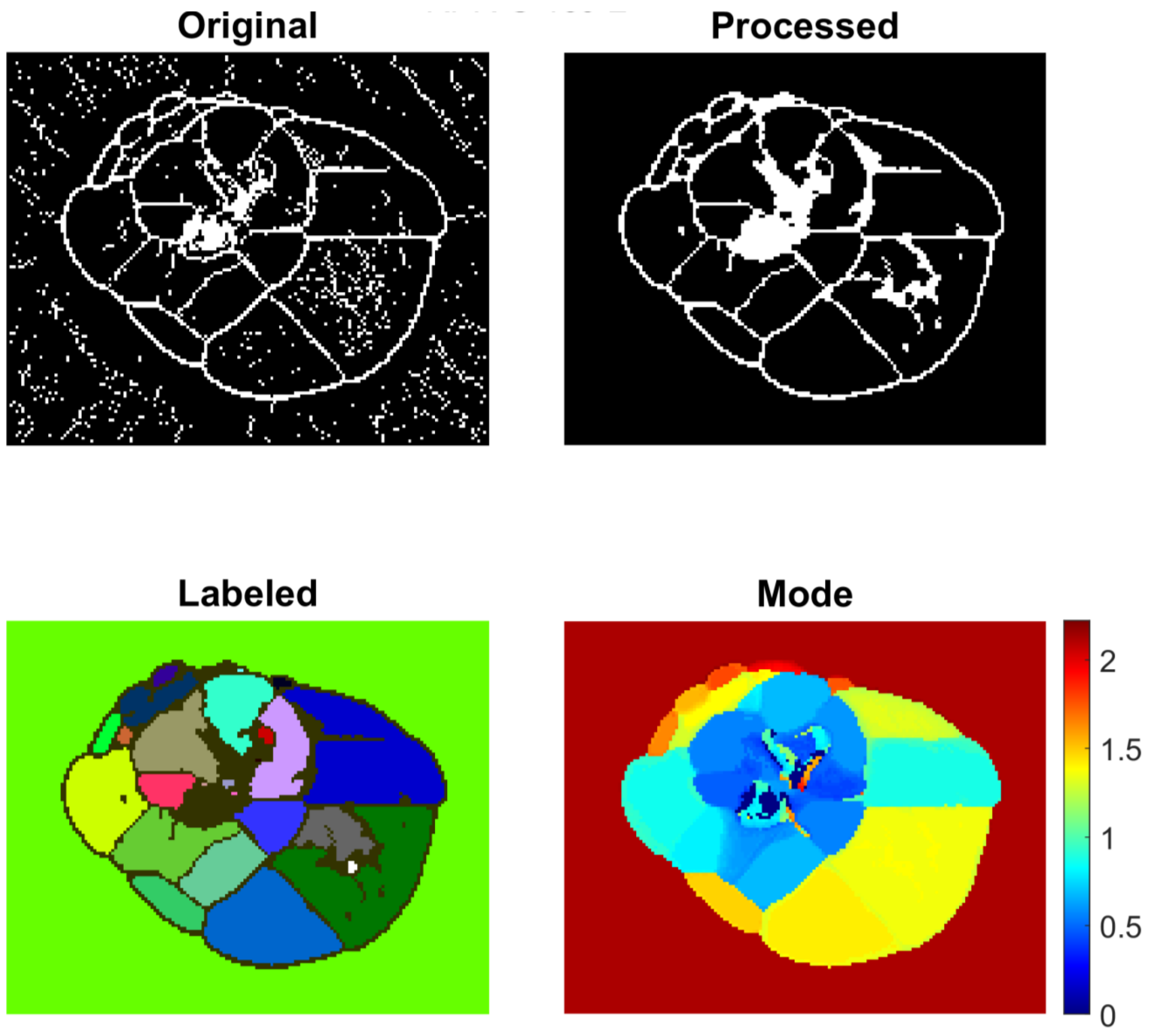


Figure D.15: RPR-S-15J-2 Morphological TOF processing steps.

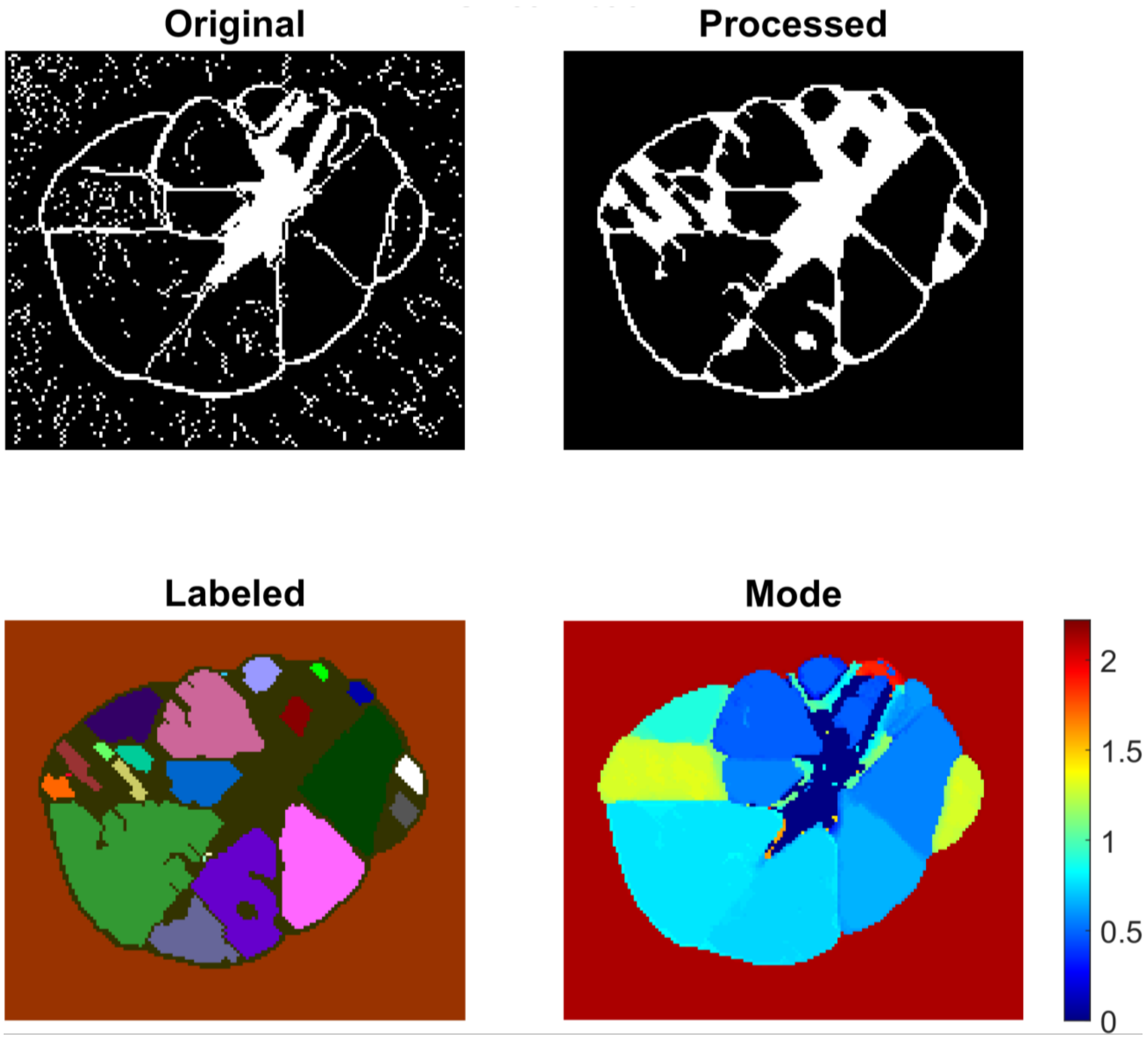


Figure D.16: RPR-S-15J-2 Back Morphological TOF processing steps.

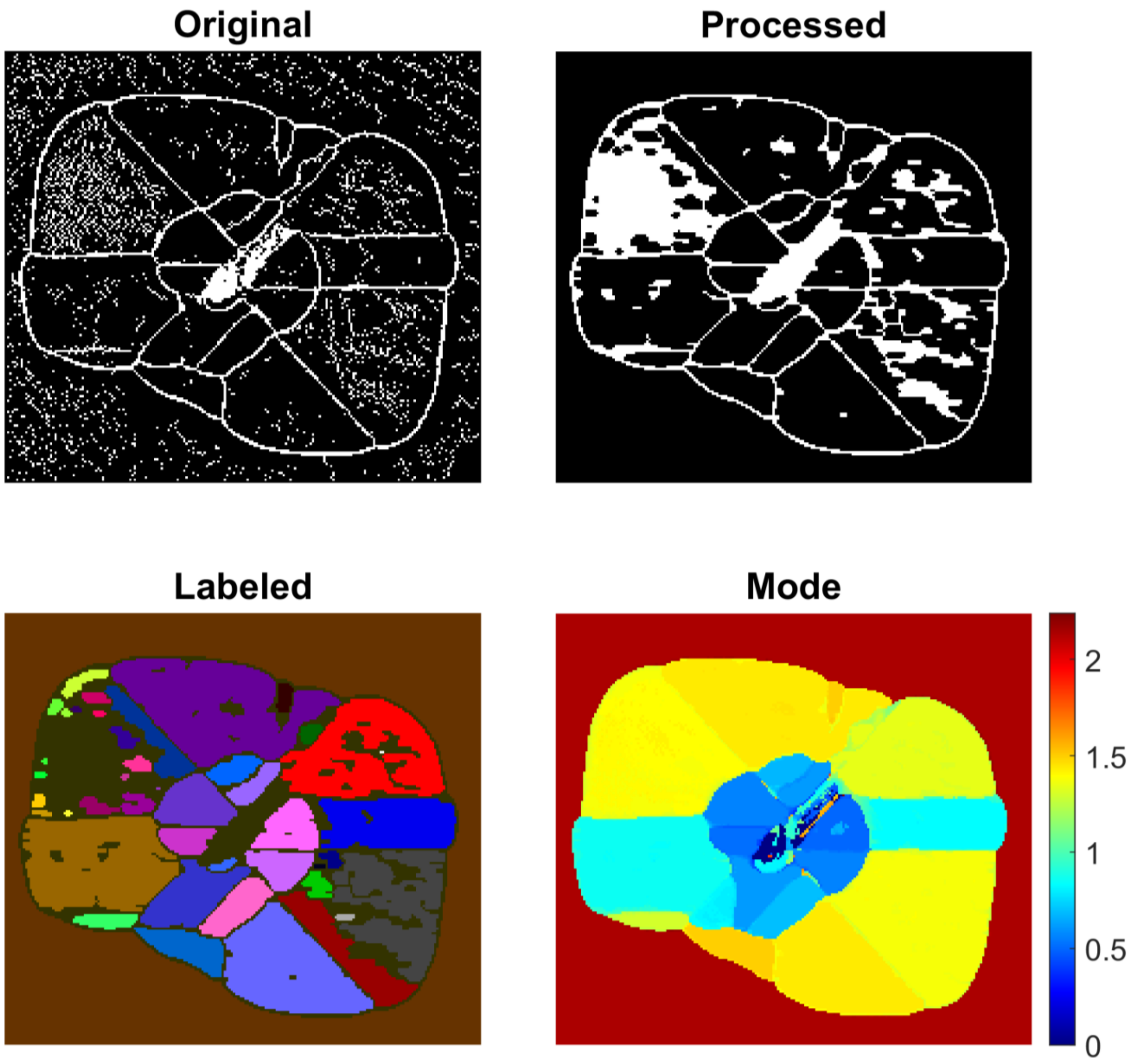


Figure D.17: RPR-S-20J-2 Morphological TOF processing steps.

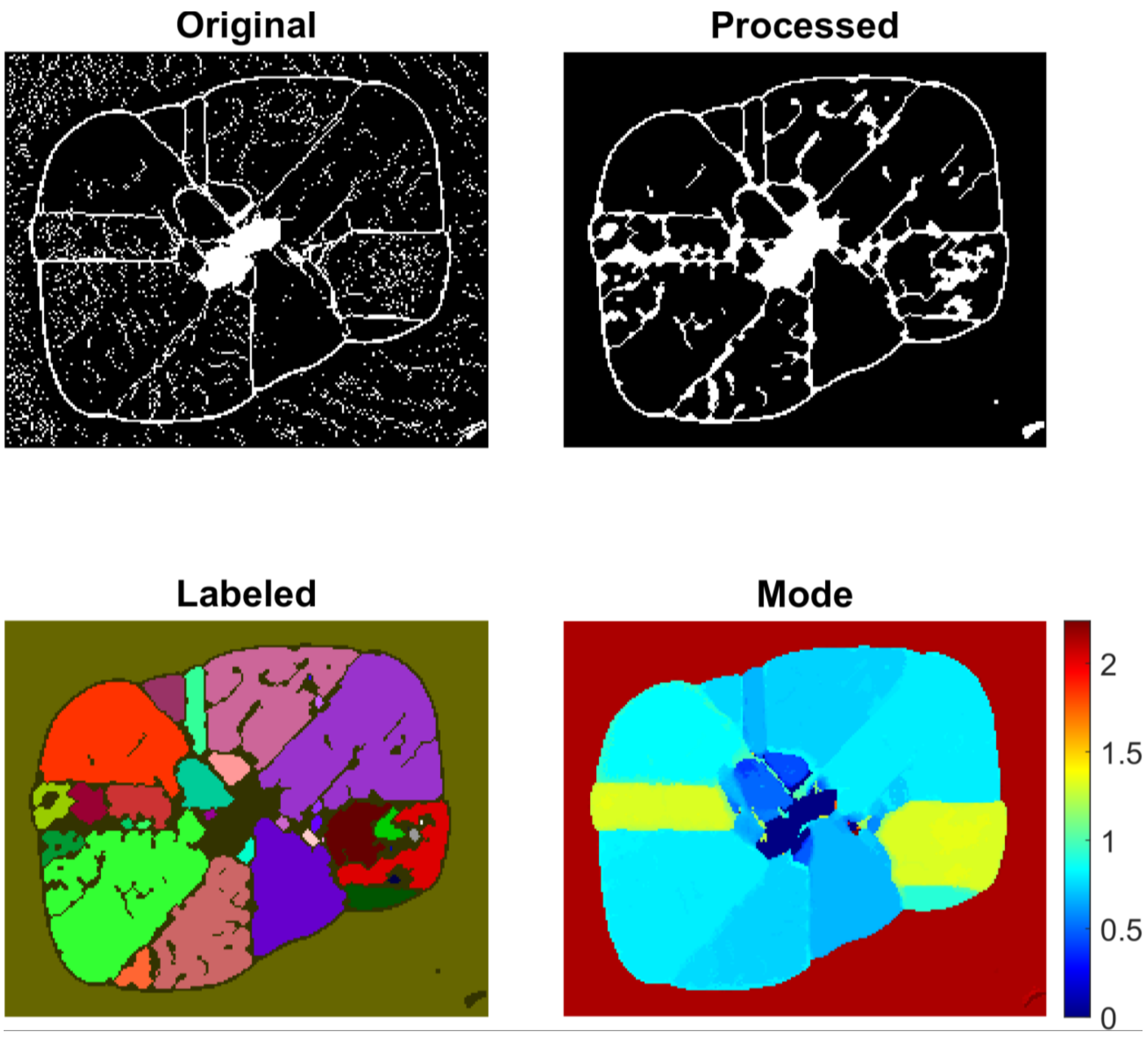


Figure D.18: RPR-S-20J-2 Back Morphological TOF processing steps.

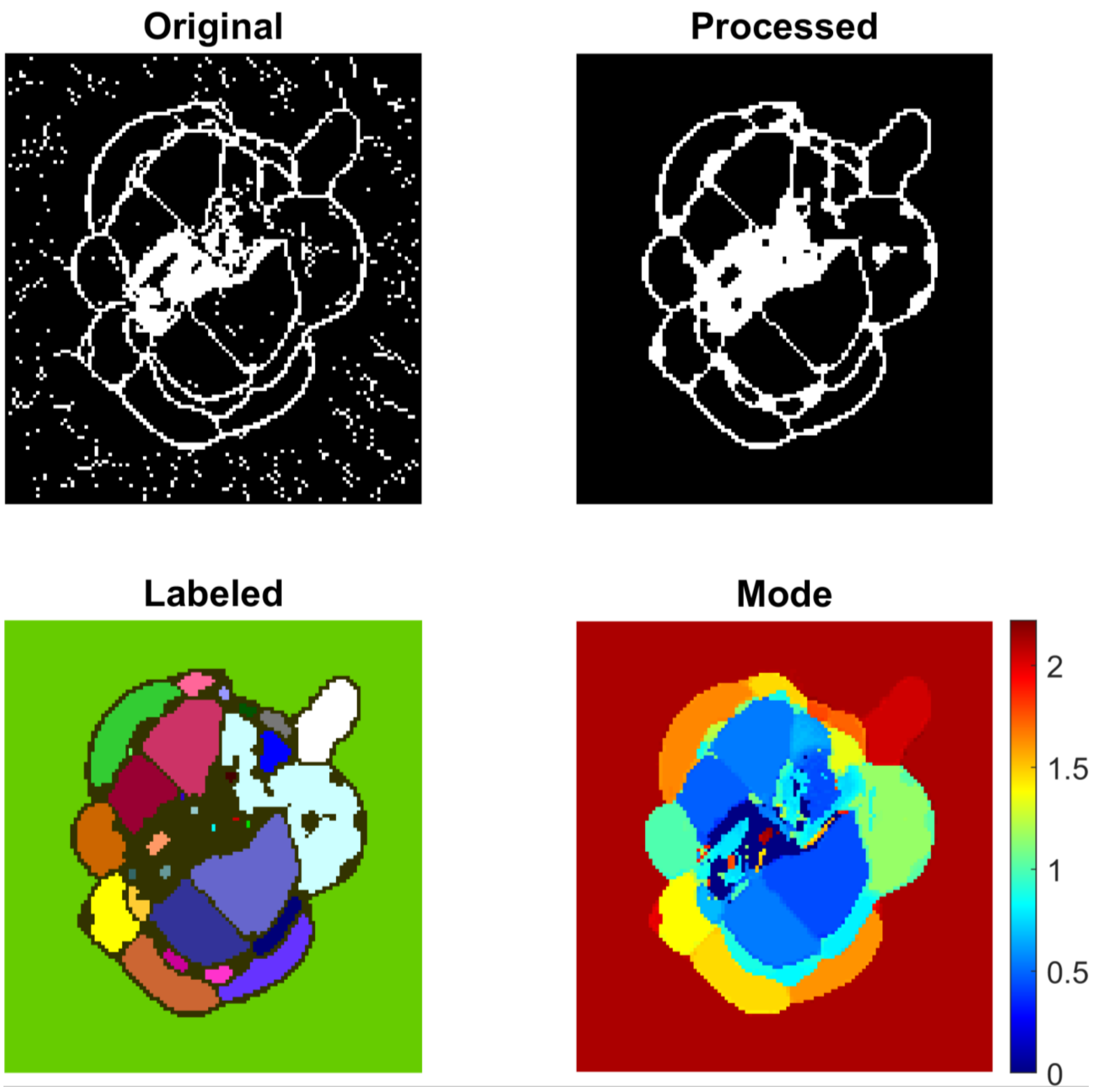


Figure D.19: BL-H-15J-1-Morphological TOF processing steps.

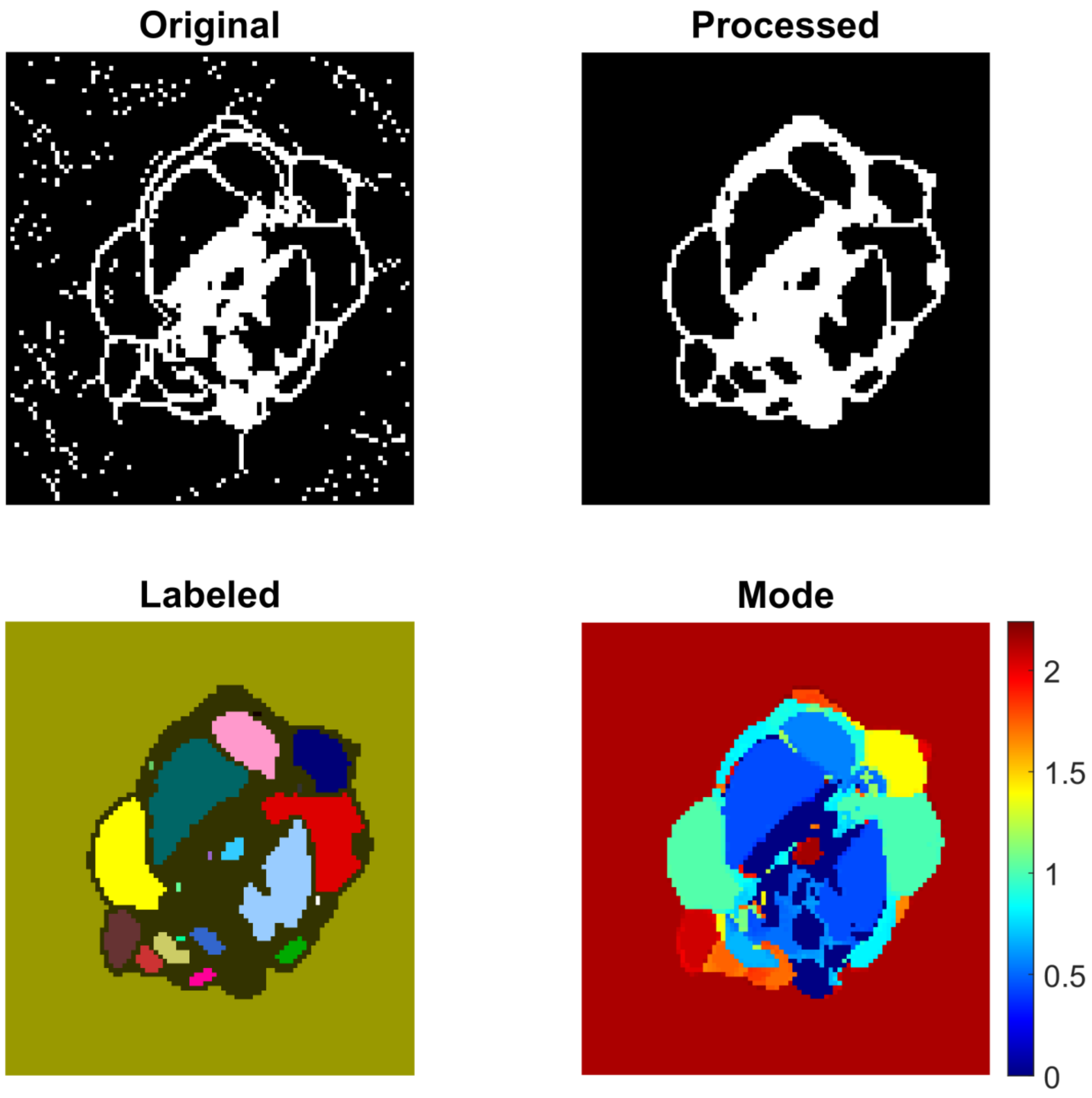


Figure D.20: CONT-H-10J-2 Morphological TOF processing steps.

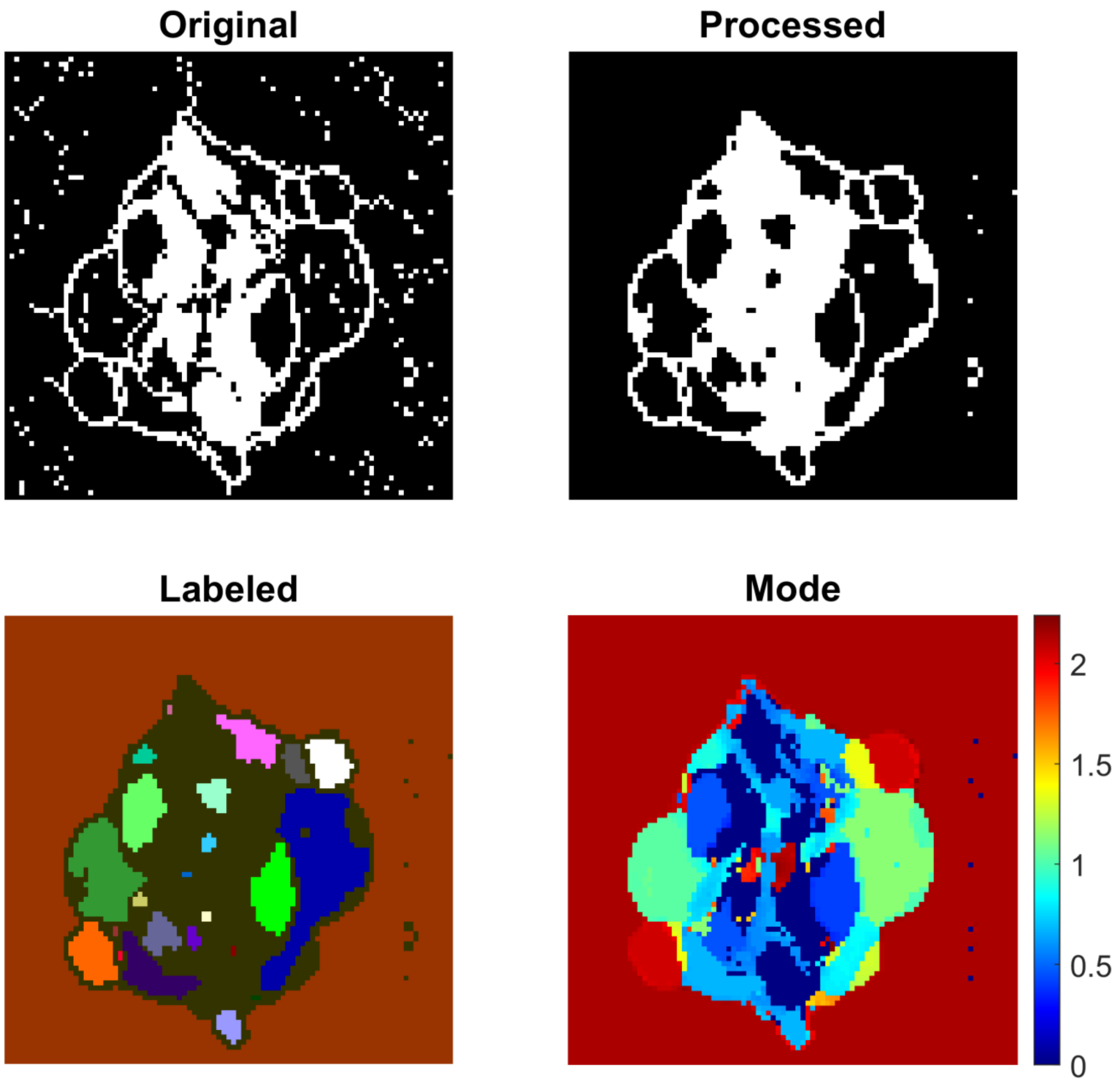


Figure D.21: CONT-H-10J-3 Morphological TOF processing steps.

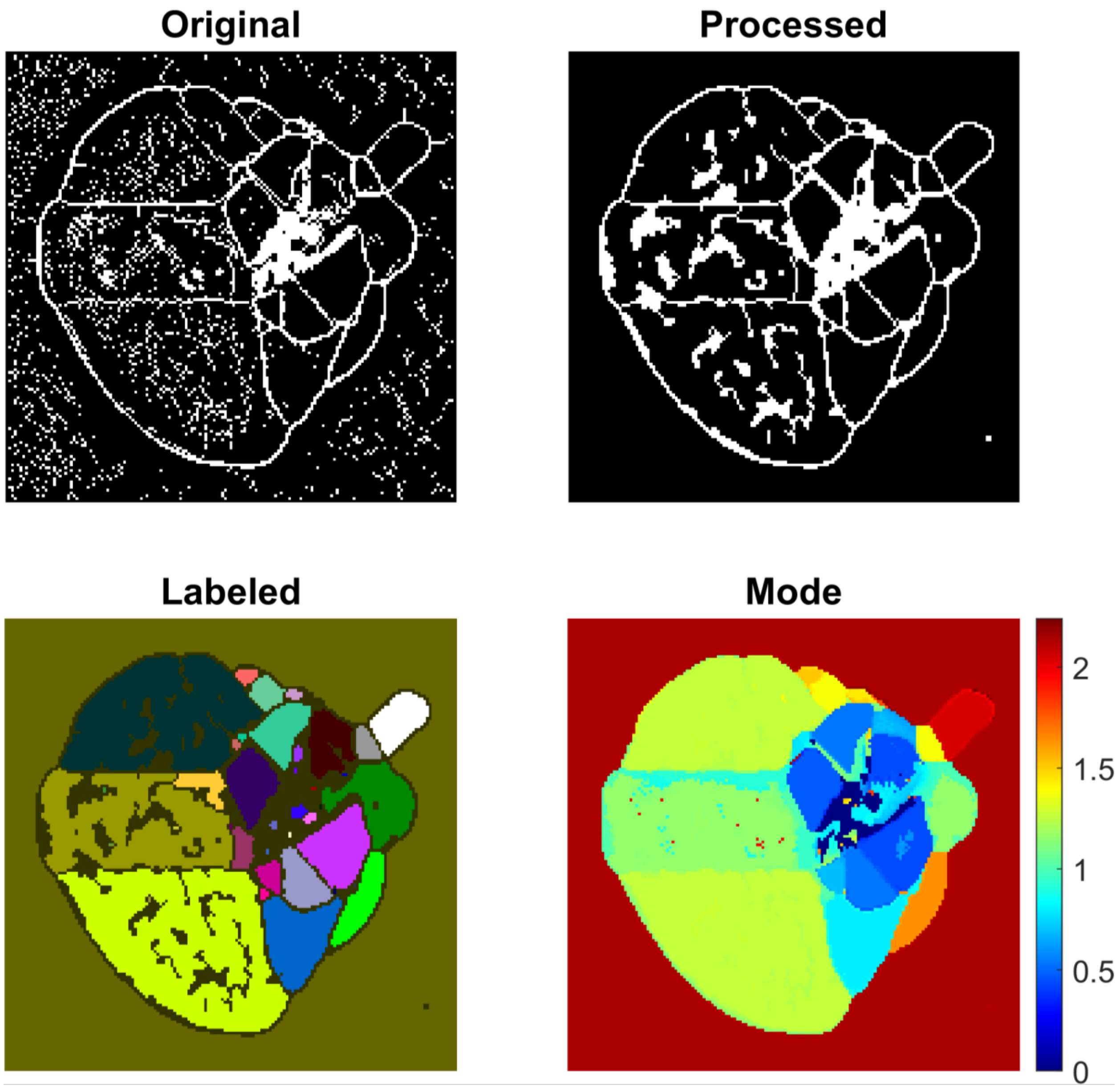


Figure D.22: CONT-H-20J-1 Morphological TOF processing steps.

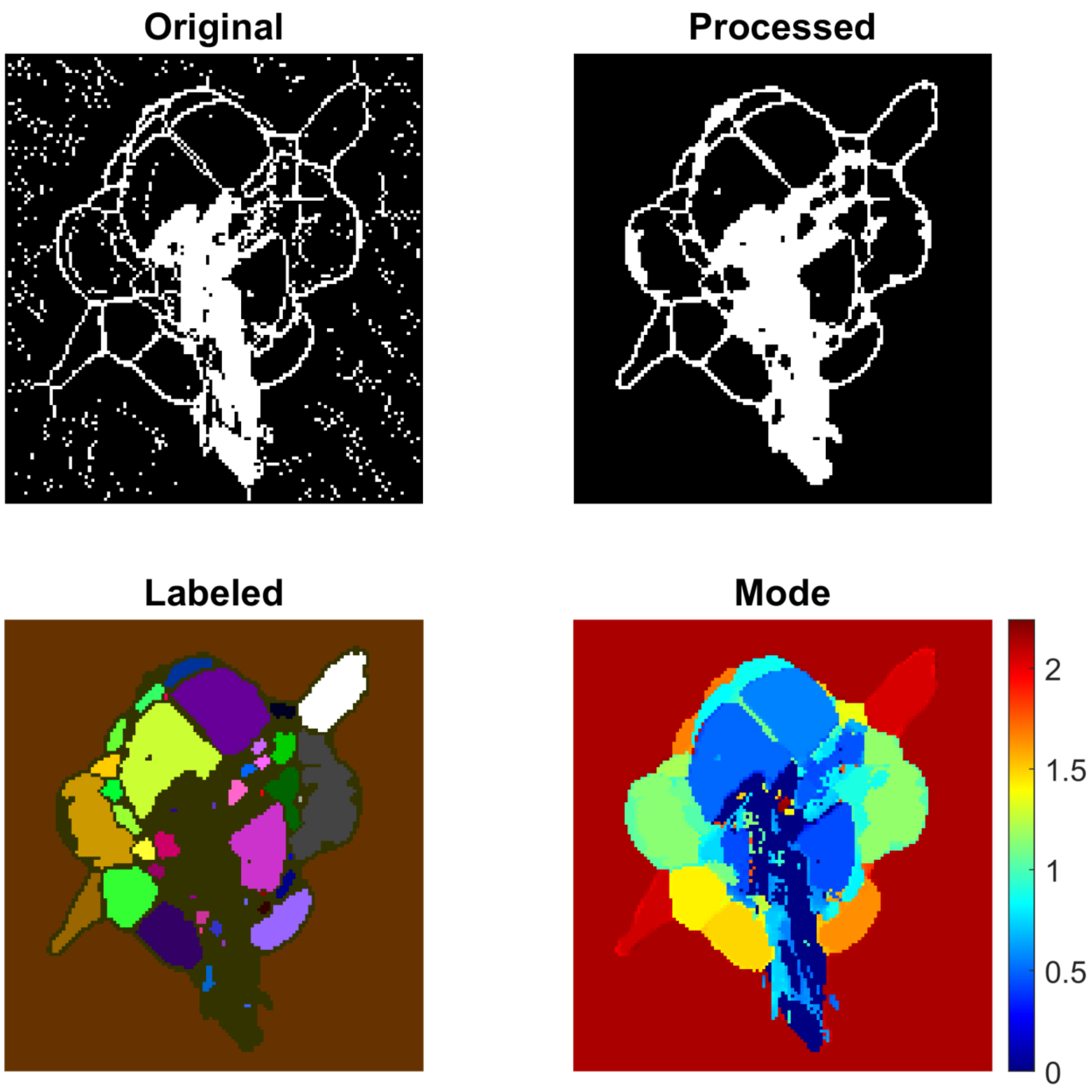


Figure D.23: CONT-H-20J-2 Morphological TOF processing steps.

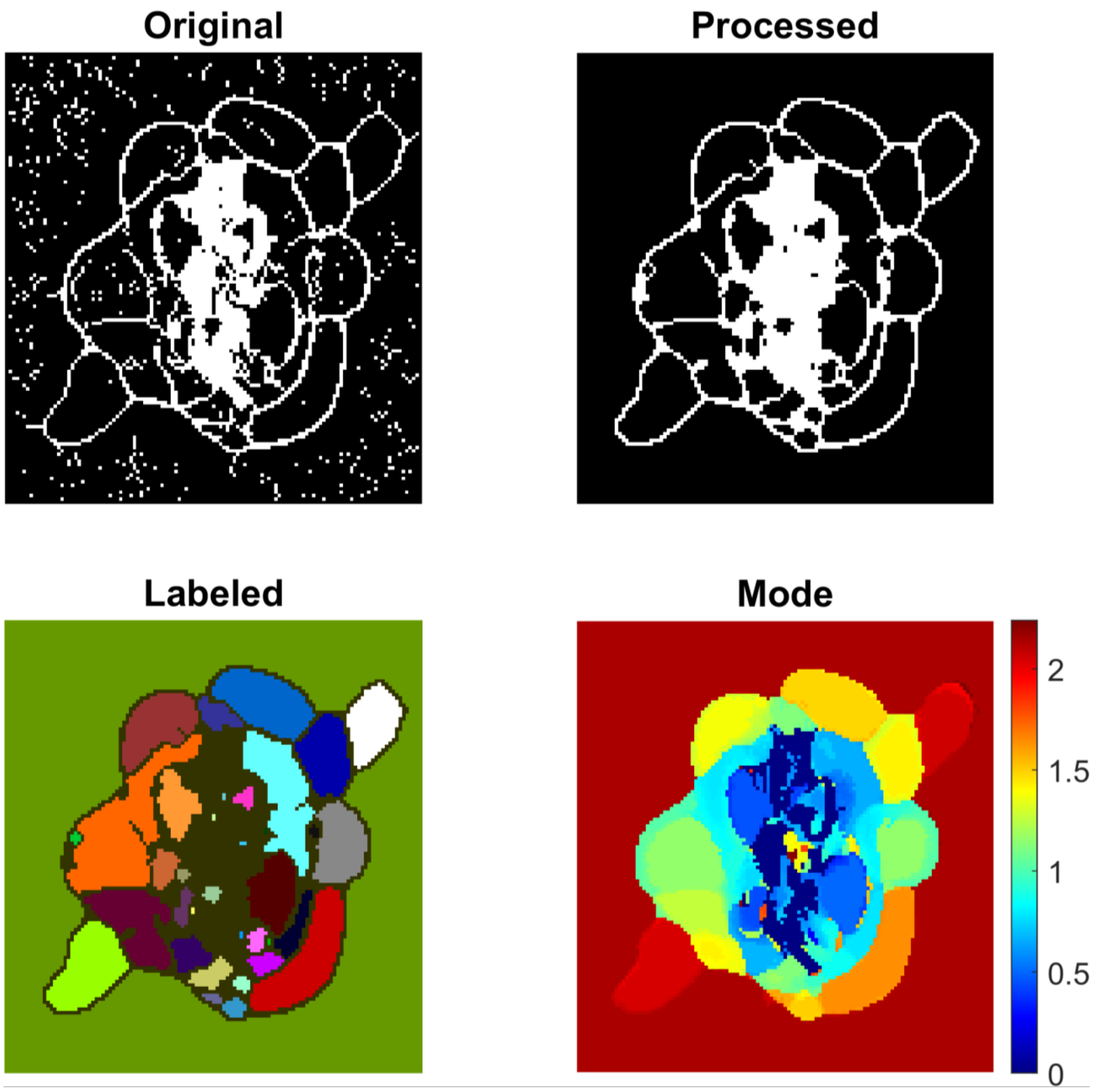


Figure D.24: CONT-H-20J-3 Morphological TOF processing steps.

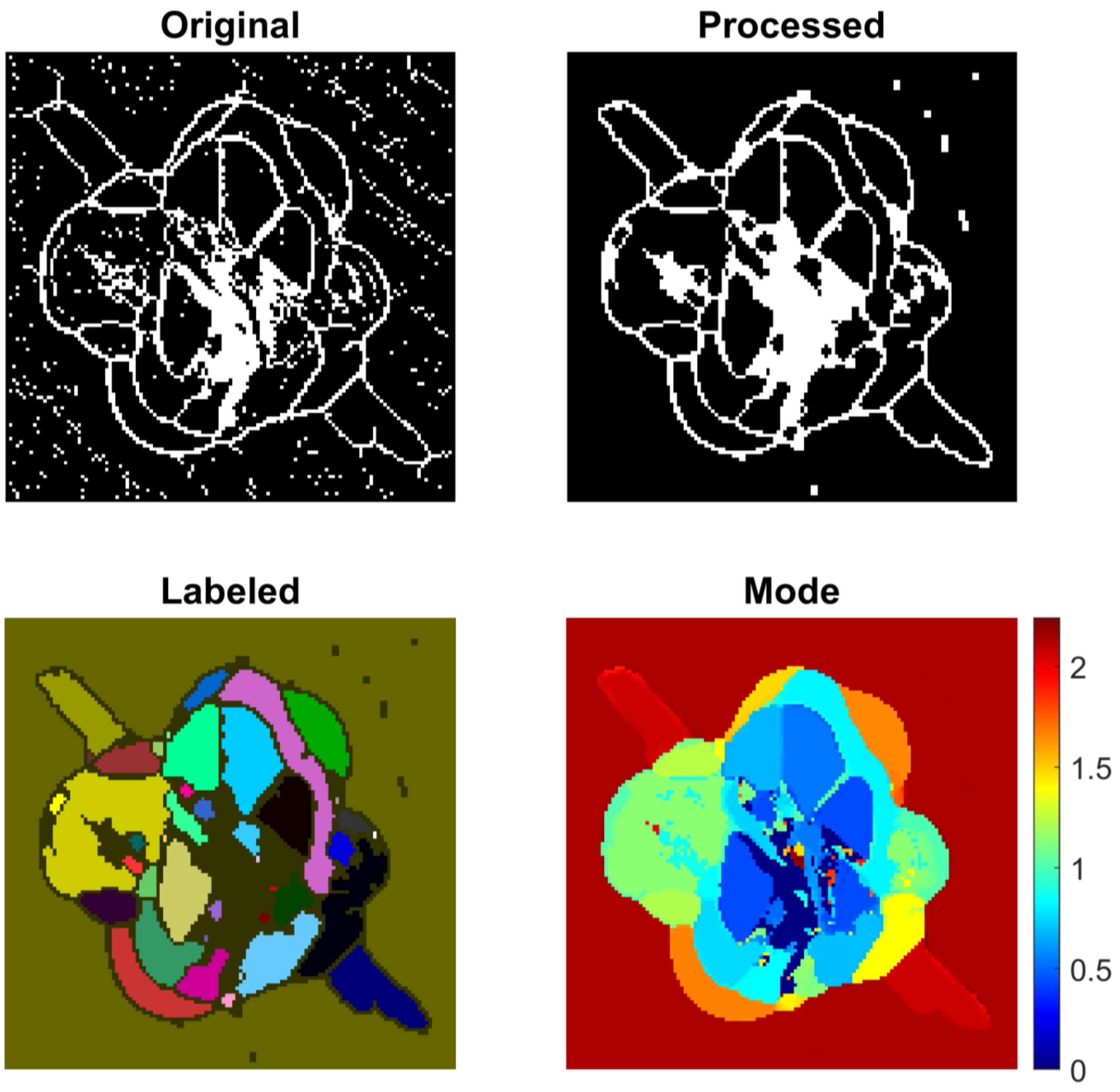


Figure D.25: RPR-H-20J-2 Morphological TOF processing steps.

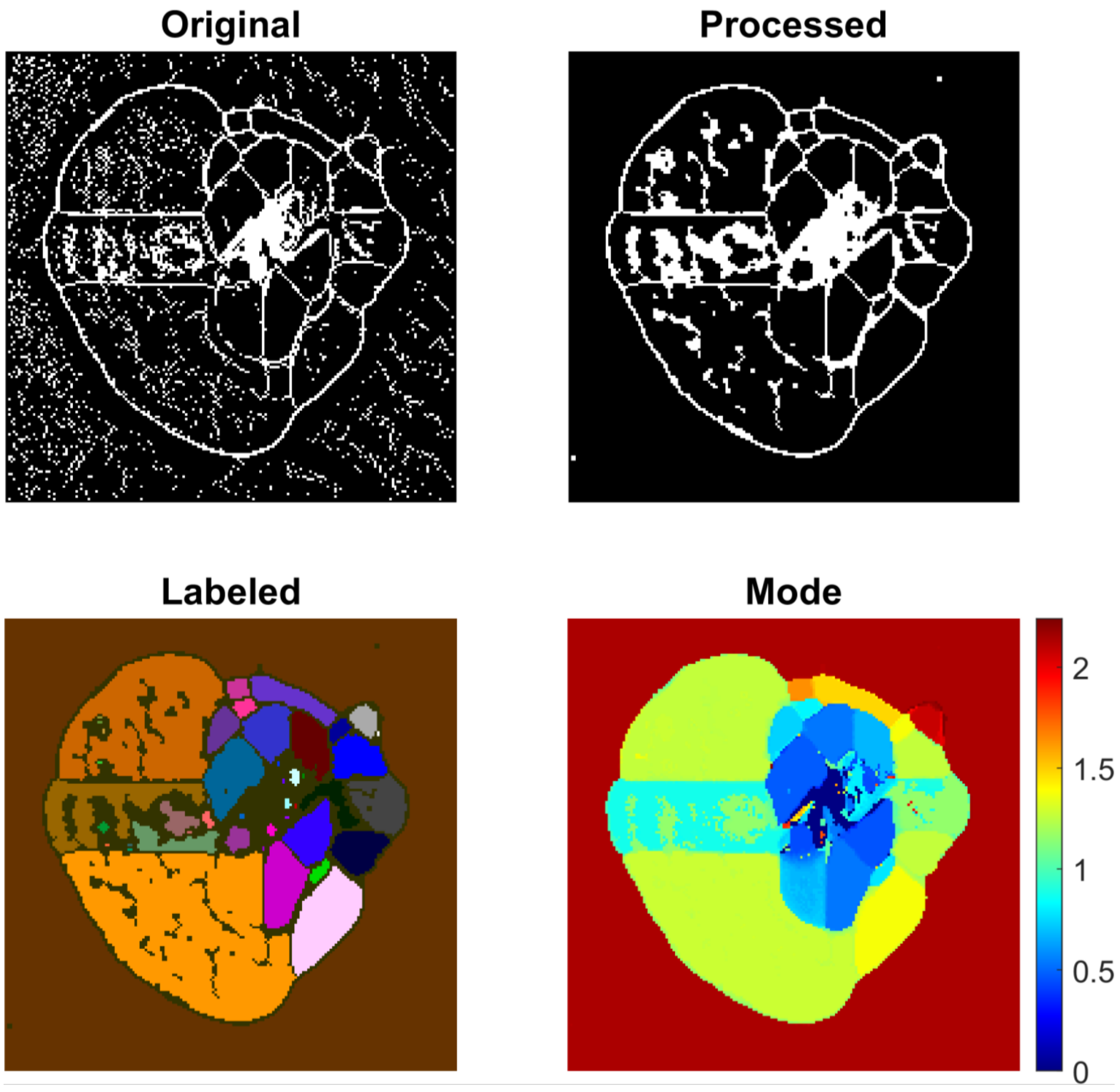


Figure D.26: RPR-H-20J-3 Morphological TOF processing steps.

References

[1] Federal Aviation Administration, "Advisory Circular: Composite Aircraft Structure," *U.S. Department of Transportation*, 2009.

[2] R. Talreja and N. Phan, "Assessment of damage tolerance approaches for composite aircraft with focus on barely visible impact damage," *Composite Structures*, vol. 219, pp. 1-7, 2019.

[3] M. Richardson and M. Wisheart, "Review of low-velocity impact properties of composite materials," *Composites Part A: Applied Science and Manufacturing*, vol. 27, no. 12, pp. 1123-1131, 1996.

[4] A. Ellison, "Segmentation of X-ray CT and Ultrasonic Scans of Impacted Composite Structures for Damage State Interpretation and Model Generation," PhD Dissertation, *UC San Diego*, 2020.

[5] A. Ellison and H. Kim, "Shadowed delamination area estimation in ultrasonic C-scans of impacted composites validated by X-ray CT," *Journal of Composite Materials*, vol. 54, no. 4, pp. 549-561, 2020.

[6] A. Wronkowicz-Katunin, G. Mihaylov, K. Dragan and A. Timofiejczuk, "Uncertainty Estimation for Ultrasonic Inspection of Composite Aerial Structures," *Journal of Nondestructive Evaluation*, vol. 38, 2019.

[7] K. Liu, Q. Yu, W. Lou, S. Sfarra, Y. Liu, J. Yang and Y. Yao, "Manifold learning and segmentation for ultrasonic inspection of defects in polymer composites," *Journal of Applied Physics*, vol. 132, 2022.

[8] A. Wronkowicz-Katunin, A. Katunin and K. Dragan, "Reconstruction of Barely Visible Impact Damage in Composite Structures Based on Non-Destructive Evaluation Results," *Sensors*, vol. 19, no. 21, 2019.

[9] A. Rodriguez-Hidalgo, A. M. Gomez, N. Bochud, J. M. Soto and A. M. Peinado, "A clustering-based damage segmentation for ultrasonic C-Scans of CFRP plates," in *IEEE International Ultrasonics Symposium (IUS)*, Taipei, Taiwan, 2015.

[10] B. W. Romasko, "Flat Composite Panel Impact Testing and Characterization by Ultrasonic Non-Destructive Evaluation," MS Thesis, *UC San Diego*, 2022.

[11] A. Wronkowicz and K. Dragan, "Damage size monitoring of composite aircraft structures based on ultrasonic testing and image processing," *Composites Theory and Practice*, vol. 16, no. 3, pp. 154-160, 2016.