The Pennsylvania State University

The Graduate School

HIGH-FREQUENCY ULTRASOUND IMAGING WITH THIN-FILM TRANSDUCERS AND EMBEDDED SYSTEM

A Thesis in

Computer Science and Engineering

by

Geesun Jang

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

August 2020

The thesis of Geesun Jang was reviewed and approved by the following:

Kyusun Choi

Associate Teaching Professor of Computer Science and Engineering

Thesis Advisor

Thomas Jackson

Robert E. Kirby Chair Professor of Electrical Engineering

Chita R. Das

Distinguished Professor of Computer Science and Engineering

Head of the Department of Computer Science and Engineering

ABSTRACT

Ultrasound imaging is a methodology that utilizes ultrasound waves to create digital images of biomedical structures such as organs and skeletal structures inside the body. As ultrasound imaging does not have harmful side-effects like radiation exposure, the application of ultrasound imaging varies from disease diagnosis to fingerprint detection. Nowadays, people are using smartphones more extensively than ever and storing their personal information on them. The most popular biometric authentication methods for mobile devices are face recognition, iris recognition, and fingerprint recognition. But there are security limitations in each technique due to the possibility of forgery. One critical reason is the rapid improvement in AI technology, which makes it possible to synthesize a realistic fake biometric signature of a target person using data on the web. With this possibility, smartphones become a critical target for adversaries trying to hack other people's private data. Given the importance of enhancing the authentication system's security level in our mobile environment, we propose initial imaging system prototypes as the proof-of-concept of a finger vein scanning system, which has the potential to be the most secure authentication method. We conducted two types of ultrasonic scanning experiments: (1) 1-Dimensional ultrasonic scanning and (2) 2-Dimensional ultrasonic scanning. For the experiments, we developed an ultrasonic scanning embedded system that consists of an ultrasound sensor, an analog-to-digital converter, a CNC-milling machine, and a Raspberry Pi 4 computer system. Our ultrasonic scanning embedded system enables the automatic ultrasonic scanning experiment that is an essential process in ultrasonic imaging. With the data collected from ultrasonic scanning experiments, we went through the process of ultrasound imaging. We displayed the ultrasound images that show the structure and shape of a target object. We believe that the system we built can be applied to further ultrasonic scanning experiments targeting complex objects like finger-vein structure.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First of all, I would like to express my sincere gratitude to my advisors, Dr. Kyusun Choi, and Dr. Thomas N. Jackson. They not only introduced me to this field of study but also showed me how a true researcher should approach to solve a problem. I greatly enjoyed my discussions with them, which often provided me with motivation for my research.

Also, I would like to thank my project members. They have always been helpful whenever I encountered obstacles during my project. They are brilliant project mates and good friends to me. It has been an honor to be part of the project team.

Finally, I want to give all the credit to my beloved ones. Without the love and support you have given me, I could not have become the person I am today. I appreciate your trust given to me along my journey. This thesis is dedicated to my father Yunsang Jang, my mother Sunhee Jung, and my brother Bosun Jang. Thank you.

Geesun Jang

July 2020

Chapter 1

Introduction

With the recent advancement in the performance of processor chips, smartphones have evolved into a general-purpose device from a communication device [1]. Nowadays, people are using smartphones more extensively than ever. Many mobile apps provide customized services utilizing the user's personal information created by user activities ranging from GPS log and browser history to financial data. Unfortunately, as the usage rate of mobile devices continuously increases, as shown in Figure 1.1, a large variety of private information is being stored and processed on this platform. As such, the private information in the mobile system is increasingly under risk of being a target of malicious adversaries [1]. For this reason, it is a pressing issue to enhance the security level of the user authentication system in our mobile environment.

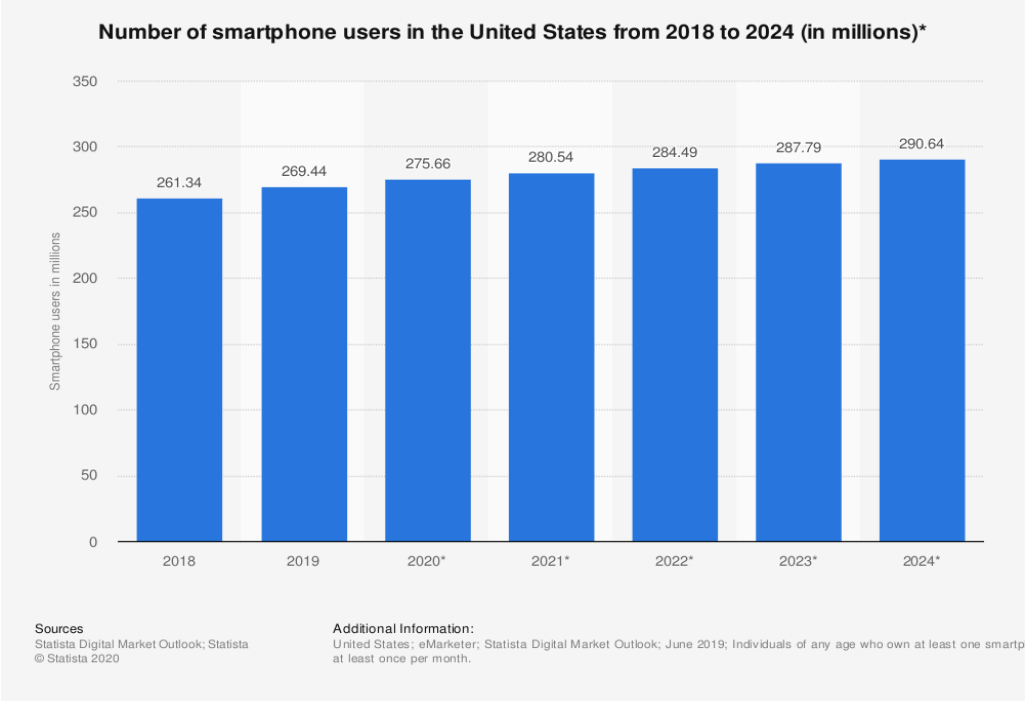


Figure 1.1 : Number of smartphone users in the United States 2018-2024 [4]

1.1 Authentication in a mobile system

In cybersecurity, the term 'authentication' is defined as the process of verifying whether the user who is trying to access a system is valid [2]. The early model of smartphones had a simple authentication process that was usually based on passwords with a cryptographic hash function. But as various techniques were introduced in password cracking, many researchers suggested more secure methods that use personal characteristics to prevent the vulnerability of mobile devices [1]. Table 1 shows various biometric authentication methods. In a survey on biometric authentication conducted by Rui et al., they categorized biometric authentication into two categories: physical-based biometric authentication and behavioral-based authentication. Physical biometric authentication contains methods that utilize static physical features such as the

Biometrics	**Accuracy**	**Size of data**	**Cost**	**Security Level**	**Long-term stability**
Fingerprint	Medium	Small	Low	Low	Low
Facial Recognition	Low	Large	High	Low	Low
Iris Scan	High	Large	High	Medium	Medium
Voice Recognition	Low	Small	Medium	Low	Low
Hand Geometry	Low	Large	High	Low	Low

Table 1: Comparision of biometric authentication methods [12]

fingerprint, face, and iris that verifies the identity of the user. On the other hand, behavioral-based authentication contains methods that utilize dynamic behavioral features like voice and keystroke [3]. In 2017, Apple released the face recognition system called FaceID with the iPhoneX model.

Compared to other face recognition method, it utilizes a deep learning classifier on the phone to improve accuracy [5]. In 2020, Samsung released Galaxy S20, the latest smartphone providing multiple authentication options: pin code, face recognition, iris recognition, and fingerprint recognition. Face recognition is the most popular option among users because it is convenient and easy to use [6]. But there are security limitations with facial recognition. During the Black Hat USA 2019 session, titled "Biometric Authentication Under Threat: Liveness Detection Hacking.", researchers show how to bypass a victim's FaceID and hack their phone by putting a modified glasses on their face as shown in Figure 1.2 [7]. Also, the improvements in AI technology make it possible to synthesize a realistic fake-image of a target person using images on social network services. Likewise, other methods such as fingerprint and iris recognition also have vulnerabilities in its system due to the possibility of forgery.

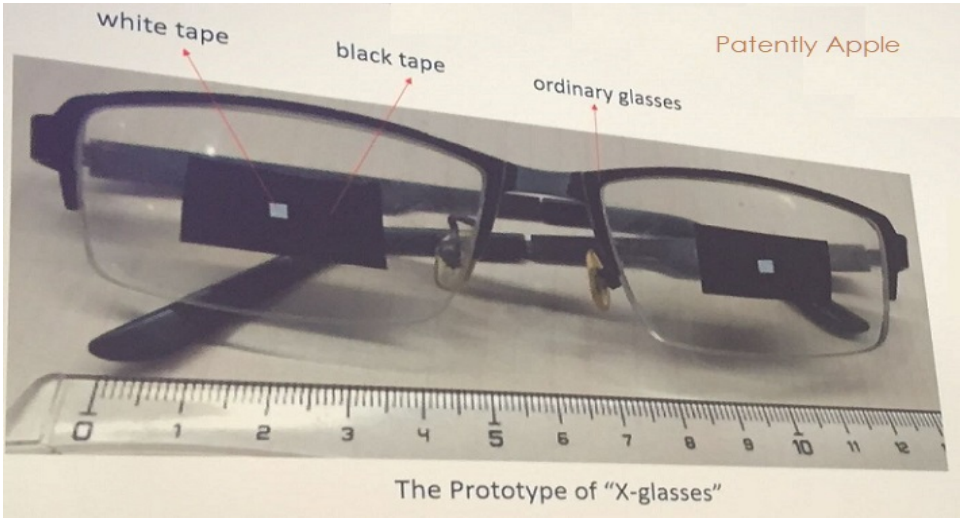


Figure 1.2: Modified glasses used to bypass FaceID in 2019 Blackhat conf [8]

1.2 Finger-vein structure as a biometric source

Researchers proposed a finger-vein structure as a new source of biometric [10] to enforce the security level of biometric authentication. Yanagawa et al. reviewed the effectiveness of a

finger-vein structure for personal identification with 500 samples. As a result, they suggested that the performance and accuracy of finger-vein pattern identification have as much validity as iris pattern recognition [11]. Also, unlike other biometric sources shown in Table 1, the finger vein pattern has the advantage of being difficult to forge by adversaries because it is inside the human body. Miura et al. proposed a biometric system using patterns of a finger-vein structure. In this system, a finger is placed between a LED light source and a Charge-Coupled Device (CCD) camera. When a light source is transmitted towards the backside of the finger, a camera captures the finger's shape, which reveals the pattern of veins. As shown in Figure 1.3 (b,c), the structure of veins in the palm side of the hand appears as a shadow figure because Hemoglobin inside the blood absorbs the light [9].

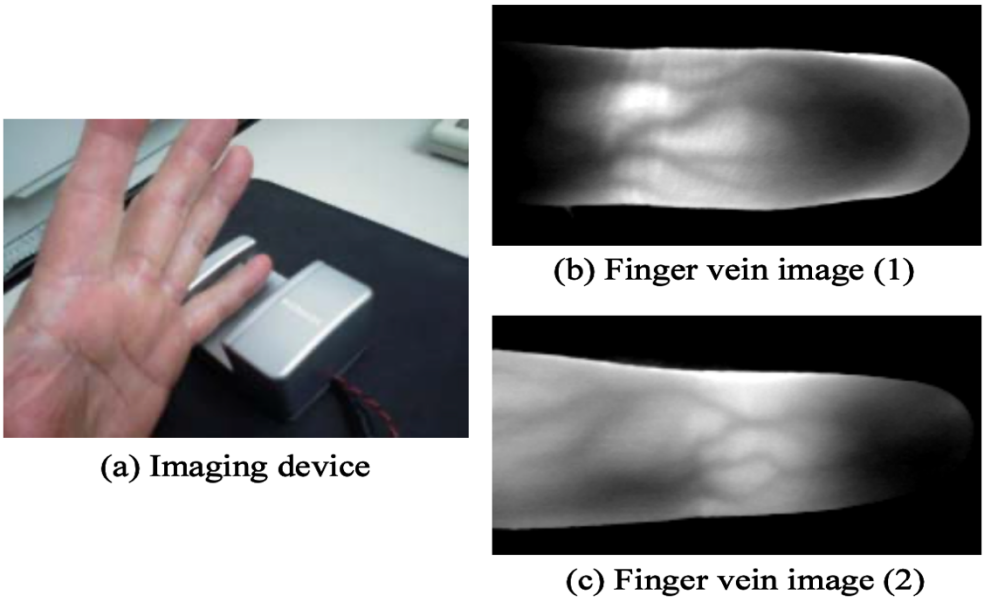


Figure 1.3: Prototype of a finger vein imaging device (a) and examples of infrared images (b,c) [9]

1.3 Ultrasonic sensor for Finger-vein structure scanning in mobile devices

Although the finger-vein imaging system in Figure 1.3 shows the finger-vein structure with high resolution and high contrast, the optical method has its drawbacks.

(a) The optical scanning is limited to 2-dimensional fingerprint imaging, that is prone to biometric forgery [13]

(b) As modern smartphone design pursues thinner and bezel-less design, the optical method can not comply with design specifications [13].

Samsung released the Samsung Galaxy S10 smartphone on March 8th, 2019. Galaxy S10 was the first device that adopted the 3D dimensional ultrasonic in-display fingerprint sensor developed by Qualcomm [14]. As shown in Figure 1.4, the ultrasound sensor is an ultra-thin (0.2mm) sensor placed under the screen without extra space.The Qualcomm 3D Sonic Sensor uses ultrasound waves to read the valleys and ridges of a finger. By transmitting an ultrasonic pulse towards the finger's surface, it creates a detailed reproduction of the scanned fingerprint [15]. Likewise, if a high-frequency ultrasound sensor can be applied to finger-vein structure scanning, it will provide in-depth biometric information that identifies users accurately and securely. Also, the size of the ultrasound sensor will meet the standards of the design trend.

Figure 1.4: Qualcomm ultrasound in-display fingerprint sensor [15]

1.4 Terminology in Ultrasonic Scanning

In ultrasonic scanning, there are special terms used to classify system setups. Firstly, the *pulse-echo* testing refers to a single transducer configuration for transmitting and receiving signals for scanning. As in Figure 1.5 (a), a single transducer diagnoses an inner flaw of an object by monitoring echo signals. Next, the *pitch-catch* testing refers to a system configuration that consists of two separate transducers to transmit and receive signals. When a transmitter bursts ultrasound signals, a receiver catches the signals reflected or scattered from an inner crack of an object, as shown in Figure 1.5 (b). Lastly, the *through-transmission* testing refers to a system configuration with two separate transducers to transmit and receive signals while facing each other, as in Figure 1.5 (c).

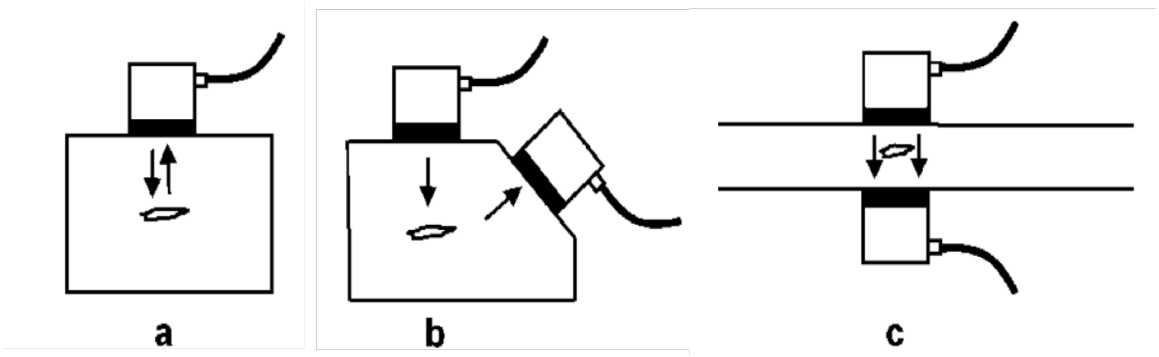


Figure 1.5: (a) A pulse-echo setup (b) a pitch-catch setup (c) a through-transmission setup [23]

Also, there are three types of displays that contain information about an object in scanning experiments. Firstly, the *A-scan* is the standard view on an oscilloscope showing the voltage versus time display. Next, the *B-scan* is one of the ultrasonic displays that shows the transducer's distance information along a linear scanning path. In Figure 1.6, a transducer scans the surface of an object to diagnose the inner flaw. The *B-scan* display reveals the side view of flaw signals by collecting the distance data from a transducer. Lastly, the *C-scan* display shows two-dimensional information of scanning

objects. In Figure 1.7, a transducer scans the surface of an object moving vertically and horizontally. With the collected data, the *C-scan* display shows the shape and formation of inner flaws inside the target object in the scanning diagnosis.

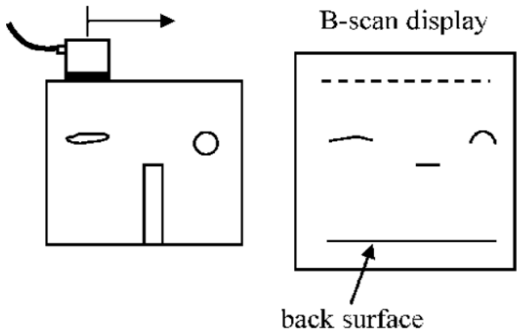


Figure 1.6: Ultrasonic B-scan display showing a side view of flaw signals [23]

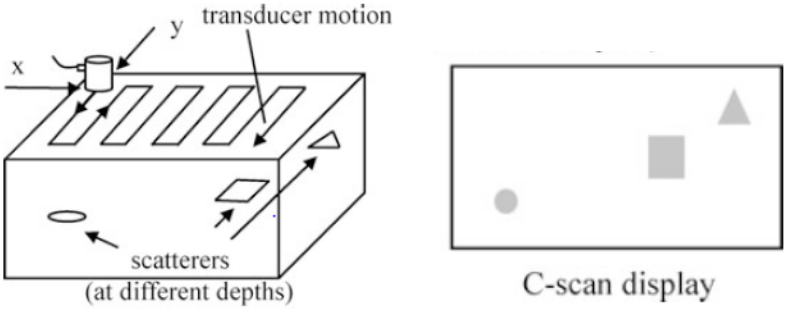


Figure 1.7: Ultrasonic C-scan display showing an upper surface view of flaw signals [23]

In this dissertation, we conducted a series of experiments using a pair of transducers, a transmitter, and a receiver. We will show the *B-scan* experiment in Chapter 2 and the *C-scan* experiment in Chapter 3.

1.5 Objectives

As the prior research above shows, the ultrasound scanning system of finger vein structure is needed to achieve higher security of the authentication process in our mobile devices. This dissertation aims to propose an initial imaging system prototype for a finger vein scanning system's proof-of-concept. The implementation details and imaging process of ultrasound signals will be covered in Chapter 2 and Chapter 3. The processed imaging result will be presented in Chapter 4. Finally, we will discuss the experiment result and future works in Chapter 5.

Chapter 2

1-Dimensional Scanning with Vehicle Machine

In this chapter, we propose a 1-dimensional ultrasonic object scanning system as a prototype model for the finger vein imaging system. In this system, we used a 40kHz ultrasound transducer to examine the basic characteristics of an ultrasound signal and get the intuition of the overall system configuration. We conducted a 1-dimensional B-scan experiment of sample objects using a vehicle machine that is created for this experiment.

2.1 Experiment Settings

HC-SR04, the transducer we used in this experiment, provides 2cm - 400cm non-contact measurement function. As shown in Figure 2.1 (a), HC-SR04 includes two transducer module, a transmitter for sending ultrasound signal and receiver for listening bounced ultrasound signal [16]. To control each module directly, we disassembled transducer modules. We used a Raspberry pi 4 board, a single computer board, for the main control system. The Raspberry pi 4 provides 3.3V 40kHz square wave pulse signals to the transducer and receives the receiver's return signal. When receiving the signal, we used an analog-to-digital converter called Analog Discovery 2, developed by Digilent. To conduct 1-dimensional scanning of sample objects, we built a vehicle machine using the Raspberry pi car starter kit. Figure 2.2 (b) shows the overall design of the ultrasound scanning system used for the B-Scan experiment.

(a) (b)

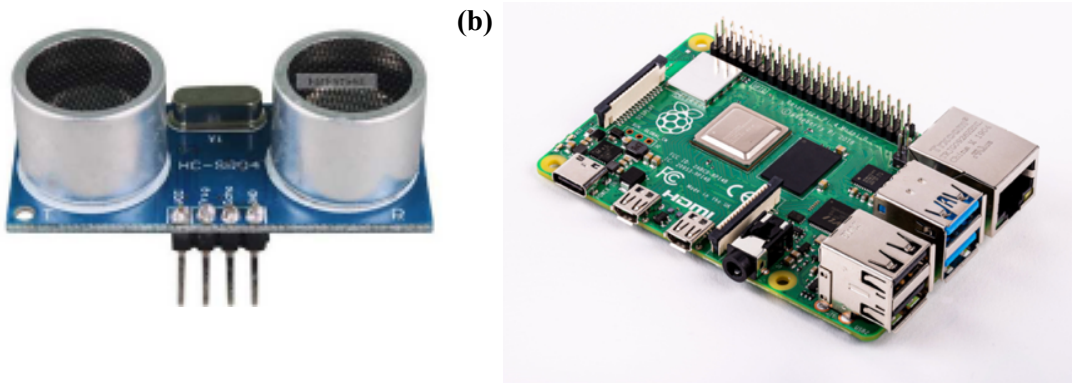
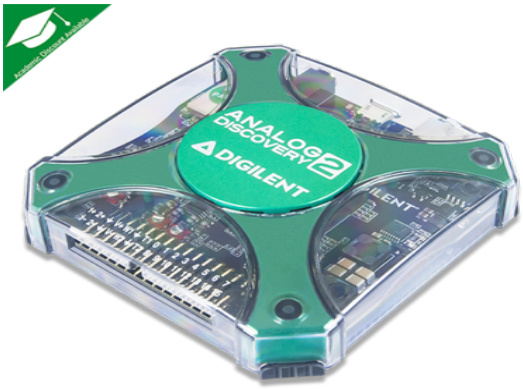


Figure 2.1: Experiment components (a) 40kHz Ultrasound Transducer [16] (b) Raspberry Pi 4

(a)



(b)

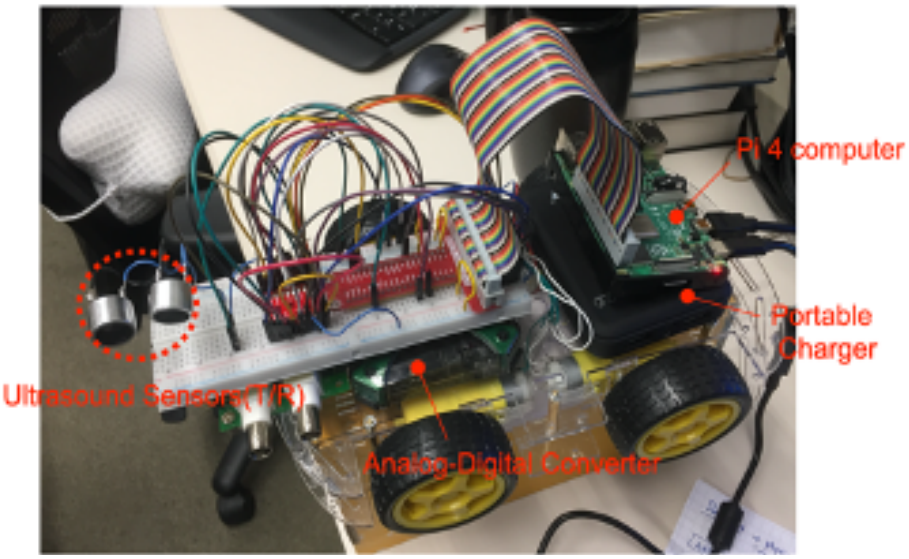


Figure 2.2: Experiment components (a) Analog Discovery 2 (b) Vehicle Machine for scanning

2.2 Experiment Process

With the ultrasonic scanning system we built, we conducted multiple experiments with a different arrangement of objects: Single object scan (Figure 2.4 (a)), Multiple objects type 1(Figure 2.5(a)), Multiple objects type2 (Figure 2.5(b)). As Figure 2.4 (b) shows, when the object scanning program is executed, the vehicle motor moves for unit distance and triggers the transmitter transducer. When the reflected ultrasound signal reaches the receiver transducer, the Analog Discovery 2 converts the analog signal to digital signal and saves it into storage. The software is written in C language and included in Appendix A.

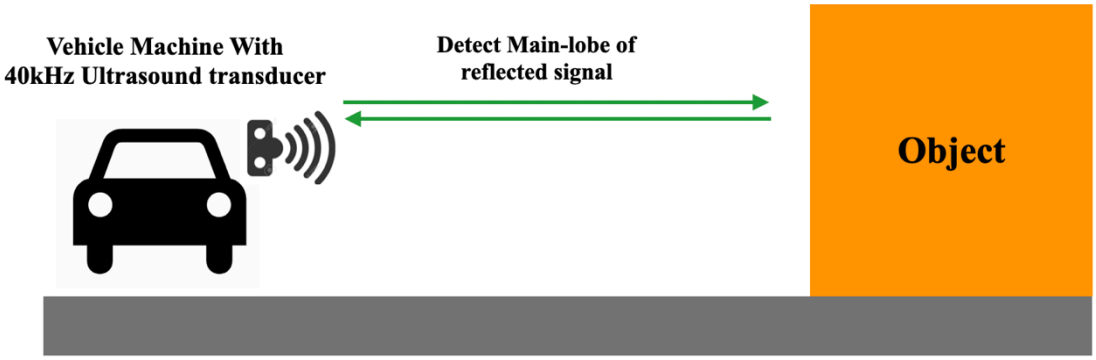
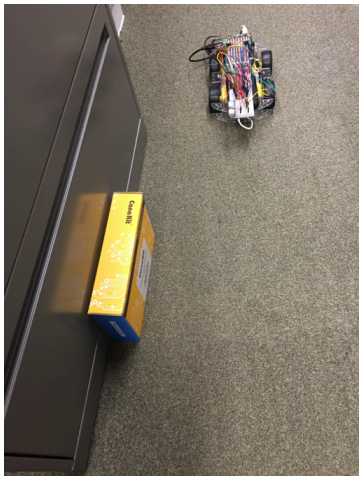


Figure 2.3: Design of object detection experiment using Vehicle Machine

(a)



(b)

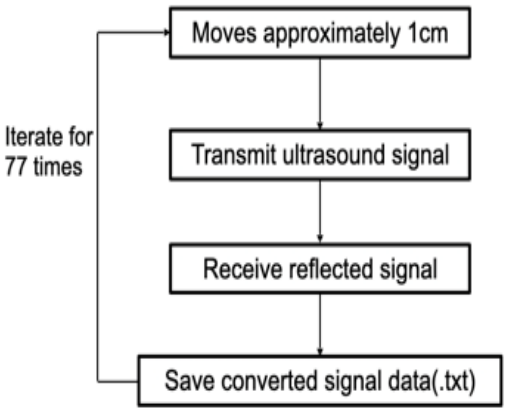


Figure 2.4: (a) Single object scanning (b) Process diagram of Scanning

Figure 2.5: Multiple object scanning: type 1 (a) and type 2 (b)

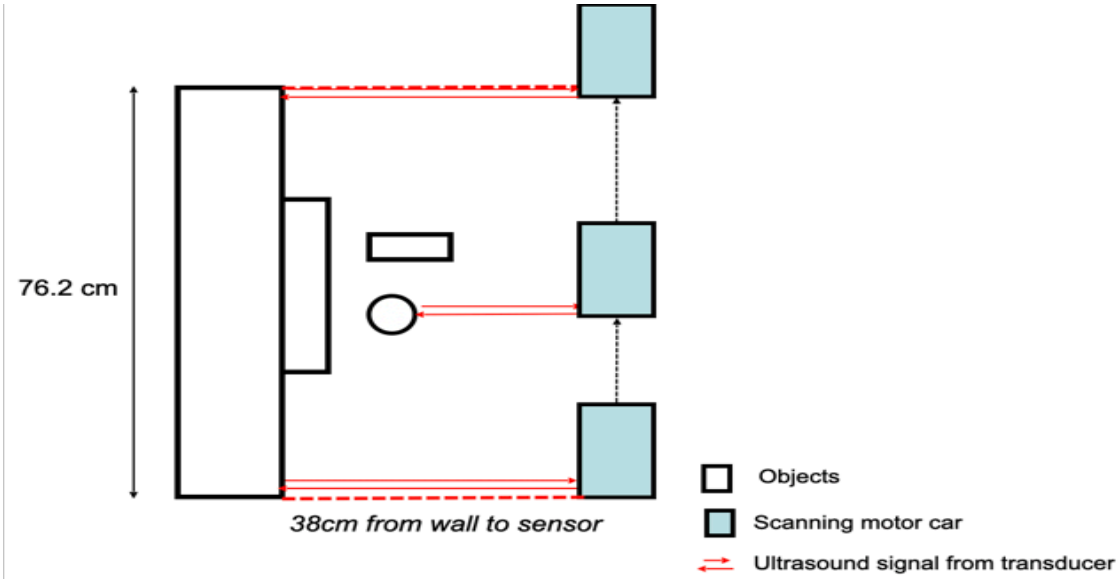


Figure 2.6: Bird's eye view of multiple object scanning

Chapter 3

2-Dimensional Scanning with CNC-Milling Machine

In this chapter, we propose a 2-dimensional ultrasonic object scanning system as a prototype model for the finger vein imaging system. In this system, we used 2.7 MHz ultrasound transducers to scan an object placed underwater. We conducted a 2-dimensional B-Scan experiment using a CNC-milling machine that provides x-axis, y-axis, z-axis stepping functionality.

3.1 Experiment Settings

For the underwater object scan experiment, we created a mounter attached with tilted transducers, as shown in Figure 3.1. For 2-dimensional scanning, we used the Titoe 3018 Pro CNC-milling machine (Figure 3.2) to provide x-axis and y-axis stepping functionality. The Raspberry Pi 4 board and the Analog Discovery 2 were used for analog-to-digital conversion, trigger signal, and system control. We connected the control interface of the CNC-milling machine with our Raspberry Pi 4 board, as shown in Figure 3.3, and wrote a C language software that tracks the position of scanning point and executes commands from keyboard input. The software code is included in Appendix B.

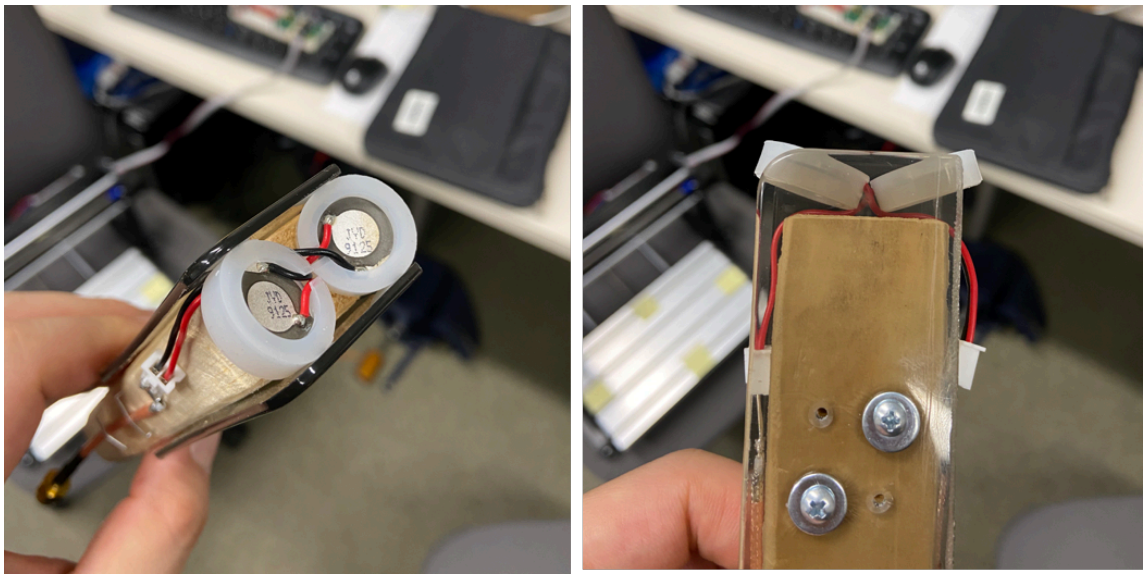


Figure 3.1: 2.7MHz transducers used for 2-dimensional scanning

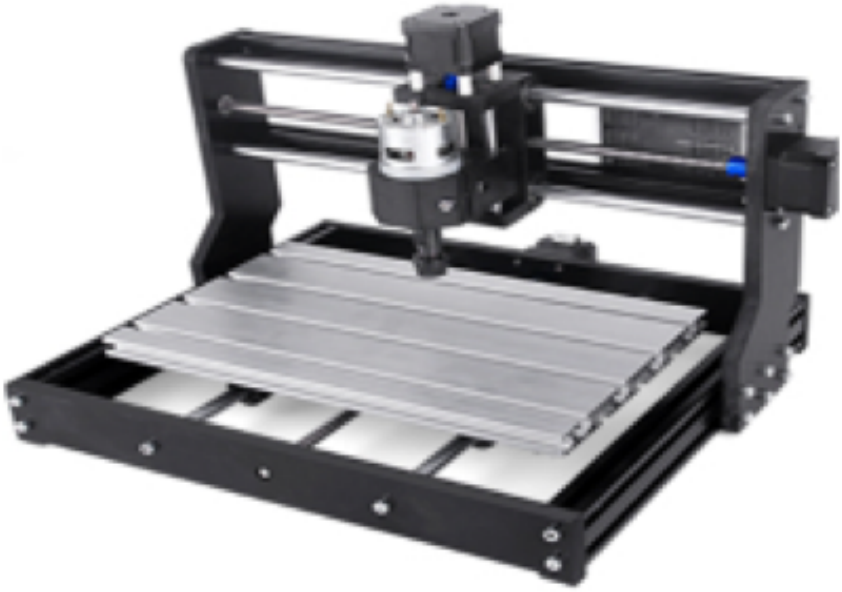


Figure 3.2: The Titoe 3018 Pro CNC-milling machine for x-axis, y-axis stepping

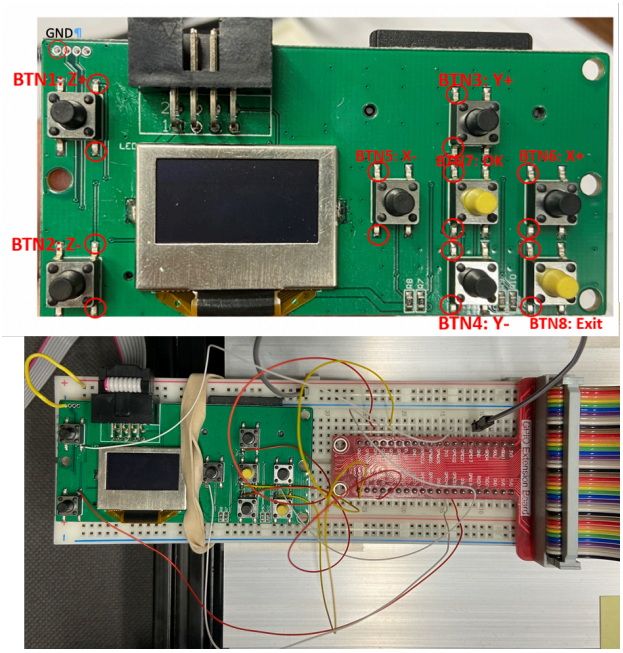


Figure 3.3: Controllable interface with a CNC-milling machine and Raspberry Pi 4 board

3.2 Experiment Process

We wrote software that automates the process of underwater object scanning. When the software starts, the CNC-milling machine moves the scan point, the focal point of transducers, to the middle of an object, coordinate (0,0). The user types in the scanning area range, and the scanning starts from the upper left point to the lower right point. As shown in Figure 3.4, we scanned a quarter-dollar coin as a sample object underwater. Since the diameter of a quarter-dollar is 24.26mm, we scanned 40mm by 40mm square area (Figure 3.5 (a)). Discovery 2 triggers two pulses of 2.7 MHz sine wave (Figure 3.5 (b)) and converts the received signal to digital data. With the collected data, we generated ultrasound images of the scanned object.

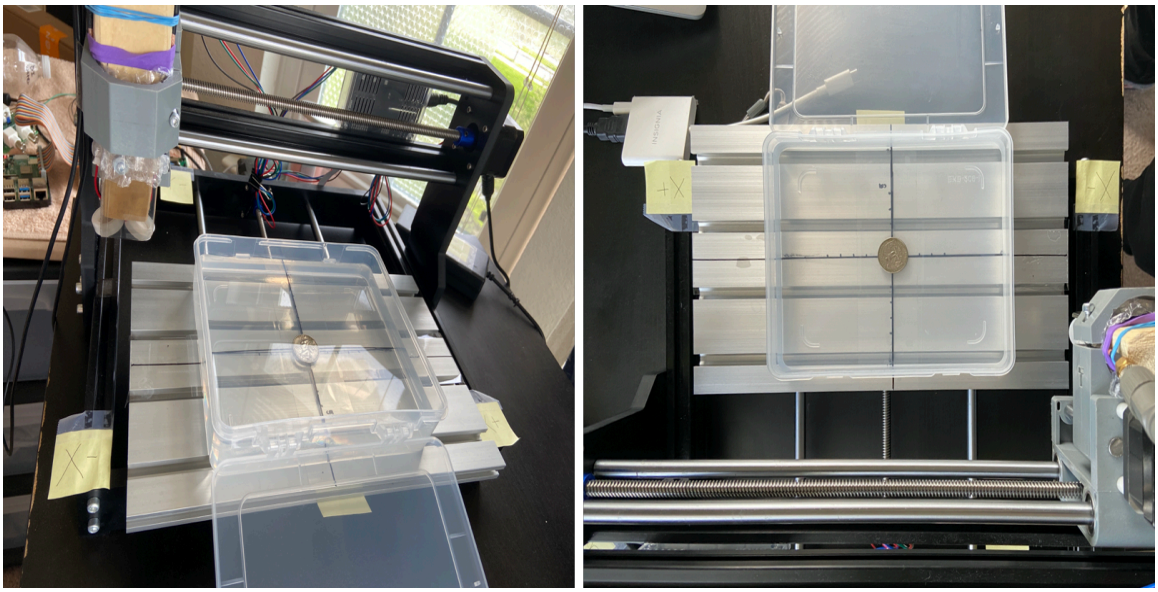


Figure 3.4: Ultrasonic scanning experiment with a quarter-dollar coin underwater

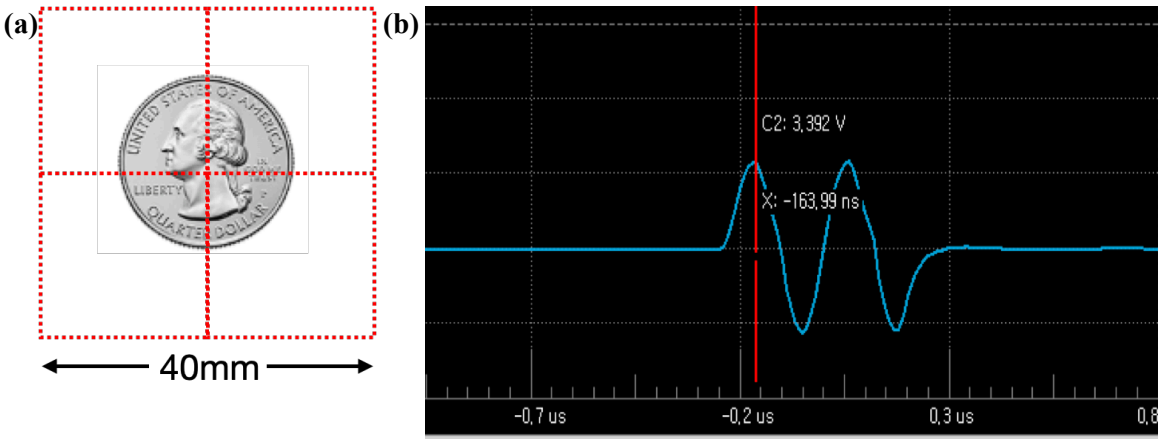


Figure 3.5: (a) Scanning area of a coin (b) 2.7 MHz Trigger signal

Chapter 4

Ultrasound Imaging

In this chapter, we proposed how to process the digitized ultrasound data and generate an ultrasound image. Based on an example of the collected ultrasound signal from the scanning experiments in .jpg and .xls format (Figure 4.1 and Figure 4.2), we showed how each process affects the initial data step by step. The code for ultrasound imaging is included in Appendix C.

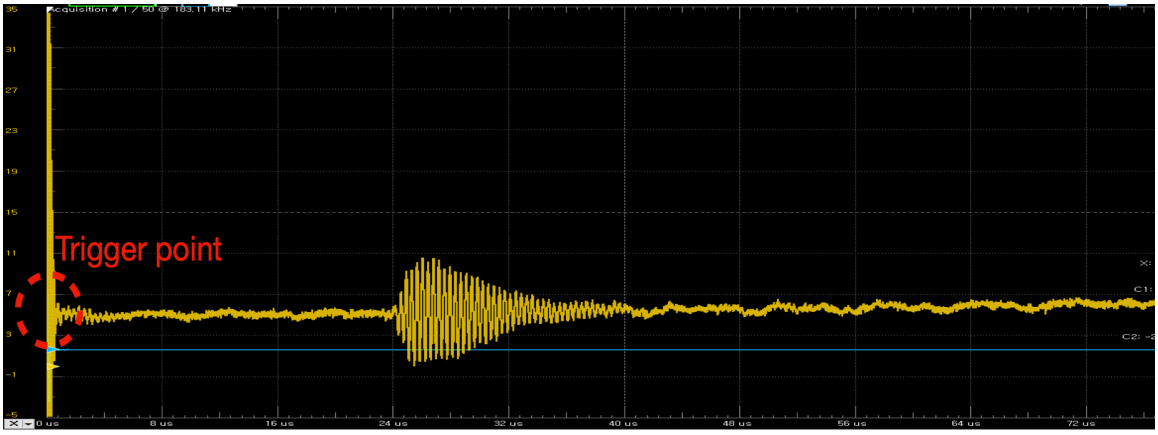


Figure 4.1: Example of collected ultrasound signal data in underwater object scanning (.jpg)

data_r1c1

#Digilent WaveForms Oscilloscope Acquisition		
#Device Name: Discovery2		
#Serial Number: SN:210321ABEE79		
#Date Time: 2020-04-14 15:11:15.274		
#Sample rate: 1e+08Hz		
#Samples: 8192		
#Trigger: Source: Channel 2 Type: Edge Condition: Rising Level: 0 V Hyst.: Auto HoldOff: 0 s		
#Average: 50		
#Channel 1: Range: 4 mV/div Offset: -15 mV Attenuation: 1 X		
#Channel 2: Range: 3 V/div Offset: -10 V Attenuation: 1 X		
#Wavegen Channel 1: Running		
#Mode: Basic		
#Type: Sine		
#Frequency: 4 MHz		
#Period: 250 ns		
#Amplitude: 4.555 V		
#Offset: 0 V		
#Symmetry: 50 %		
#Phase: 0		
Time (s)	Channel 1 (V)	Channel 2 (V)
-9.599999999999998E-07	0.004885239609138900	-0.020439051777641100
-9.499999999999996E-07	0.004885642773953540	-0.020664539978893000
-9.399999999999994E-07	0.004880819361531150	-0.022523105532567700
-9.3E-07	0.004835692732502110	-0.022041563495047600
-9.199999999999998E-07	0.005010051962499980	-0.021211854431745700
-9.099999999999996E-07	0.004897934071472580	-0.019520396472013100

Figure 4.2: Example of collected ultrasound signal data in underwater object scanning (.xls)

4.1 Process of Ultrasound Imaging

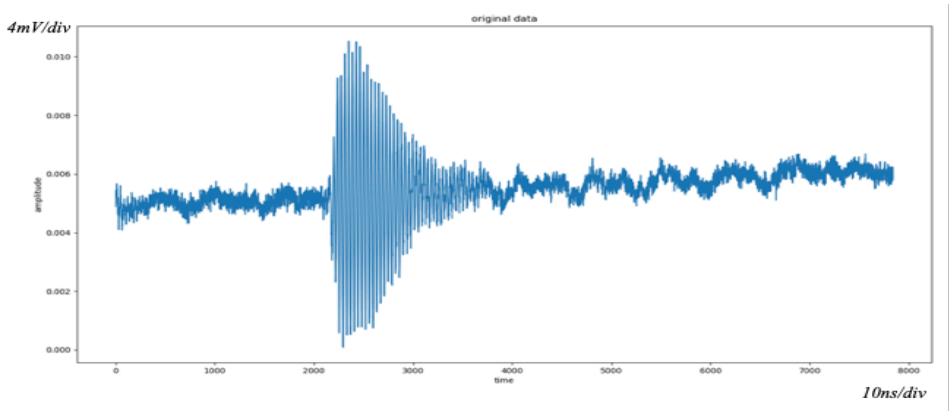


Figure 4.3: Plot of collected ultrasound signal data with noise

Noise Reduction by Averaging

In the original data collected from the experiments in Figure 4.3, there are noise signals that change the base amplitude value. To reduce the effect of noise and set the amplitude of data to be zero, we took averaging steps for the values within each period of the transducer overlappingly as a first step. Averaging is a technique that uncovers small amplitude signals in the noisy data, as shown in the upper plot of Figure 4.4. Finally, we subtracted the averaged values from the original data resulting in the lower plot of Figure 4.4.

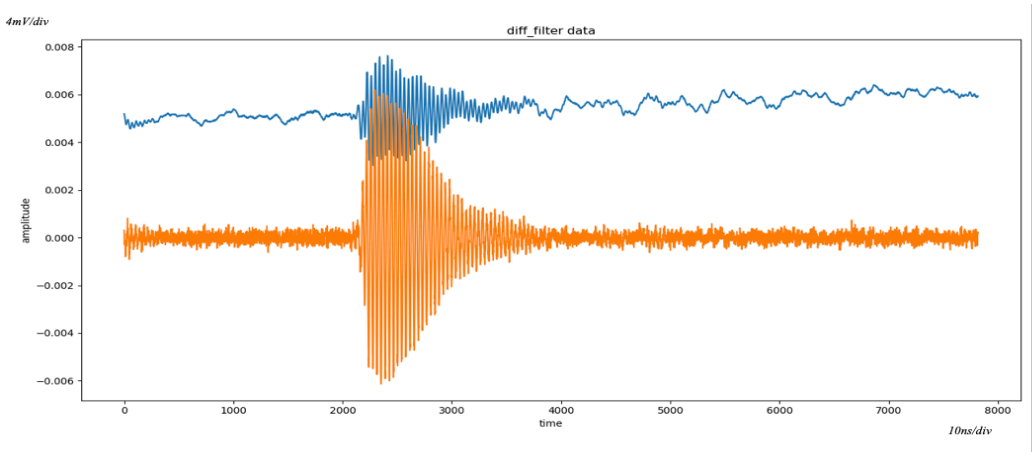


Figure 4.4: Plot of data after averaging and noise reduction

Hilbert Transform

When an ultrasound transducer generates a sound beam, the main lobe, which contains the highest power, reaches the surface of a target and then travels back to the receiver. The Hilbert transform is a specific linear operator that derives the analytic representation of a real-valued signal [19]. The Hilbert transform of the function $x(t)$ is defined by an integral transform [20]:

$$H(x(t)) = \tilde{x(t)} = \pi^{-1} \int_{-\infty}^{\infty} \frac{x(\tau)}{t-\tau} d\tau \qquad (1)$$

For our experiment, we applied the Hilbert Transform to get the upper envelope of our signal data to identify the strongest signal, which is the main lobe effect. The upper envelope of Figure 4.5 represents the result of the Hilbert transform of the lower graph.

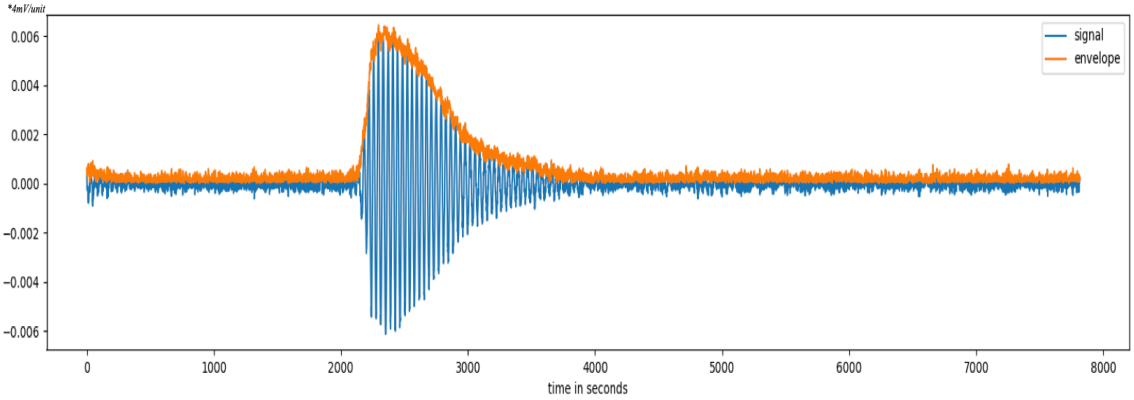


Figure 4.5: Result of the Hilbert Transform, presenting the upper envelope of the data

Signal conversion to a pixel value

Figure 4.6 shows how the ultrasound waves are reflected from the front surface. Since the distance from the scanning point to the bottom differs, the time delay of the reflected ultrasound wave also differs. With this characteristic, we calculated the ratio of the time delay of reflection

over the time delay, which is the time difference between each trigger of the transmitter. Let d as the element of data we processed through, then the equation of the pixel data is as below:

$$Pixel(d) = \left[\frac{Argmax(d)}{length\ of\ scan\ point\ data}\right] * 255 \qquad (2)$$

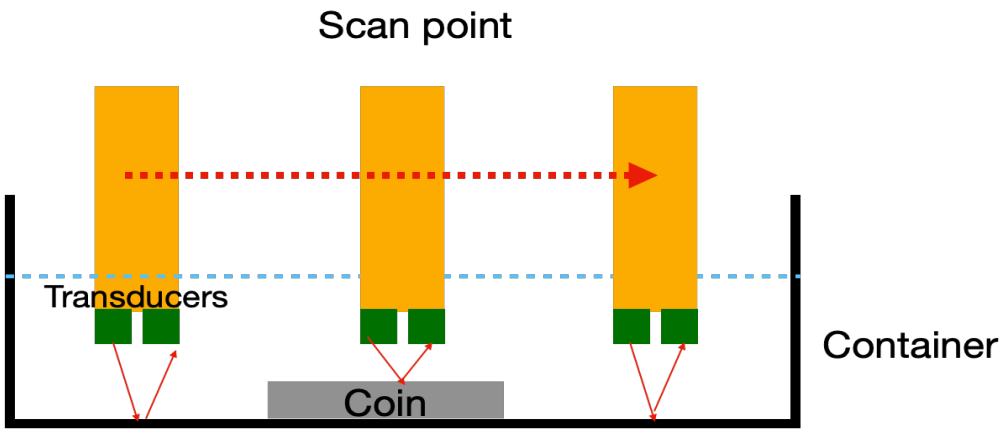


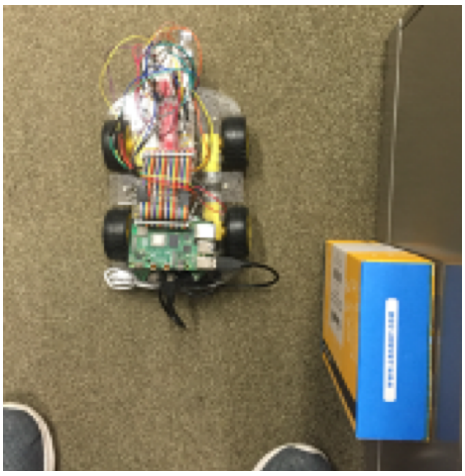
Figure 4.6: Schematic of the 2-dimensional ultrasound scanning experiment

4.2 Result of Ultrasound Imaging

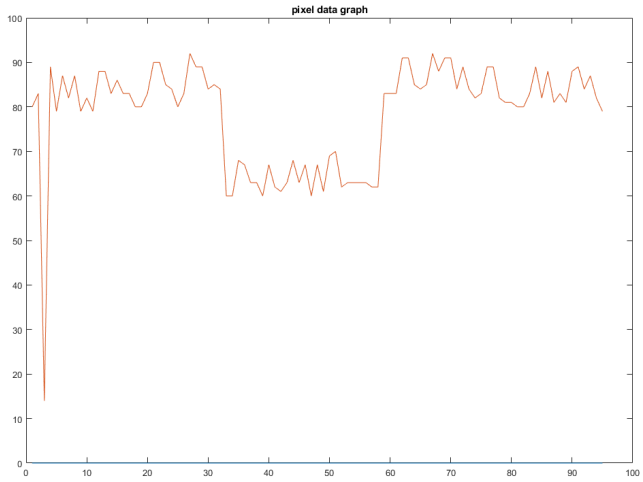
1-Dimensional Scanning experiment

For the 1-dimensional scanning experiment, we tested with two different settings, scanning a single flat object and scanning multiple objects of different shapes. Since this is a 1-dimensional scanning, the processed result represents the distance from the scan point along the scanning route, as shown in Figure 4.7 (b), 4.8 (b), 4.9 (b). Also, we copied the processed array multiple times and arranged it vertically to visualize the scanned surface as an image in Figure 4.7 (c), 4.8 (c), 4.9 (c).

(a)



(b)



(c)

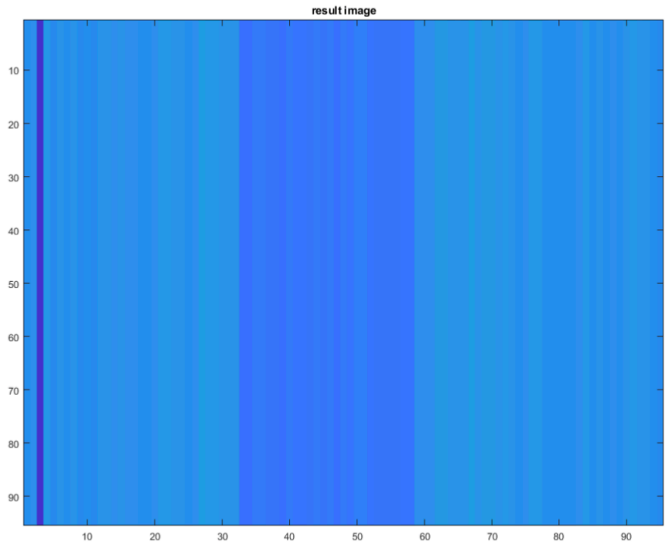
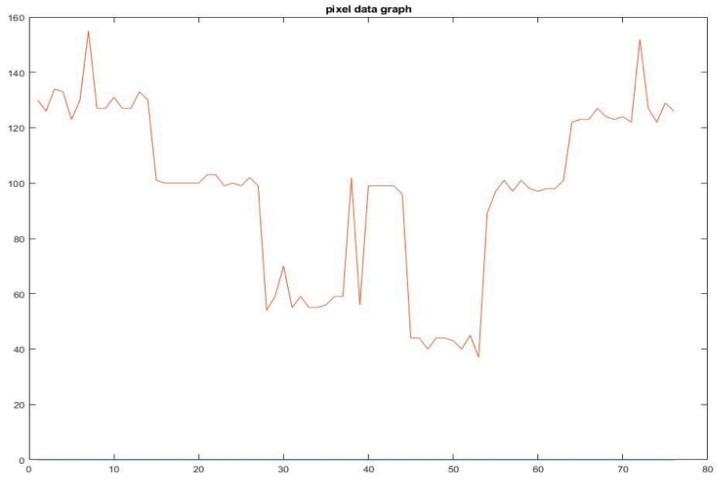


Figure 4.7: Scanning a single flat object (a) Original setting (b) Plot of the received signal (c) Ultrasound Image

(a)

(b)



(c)

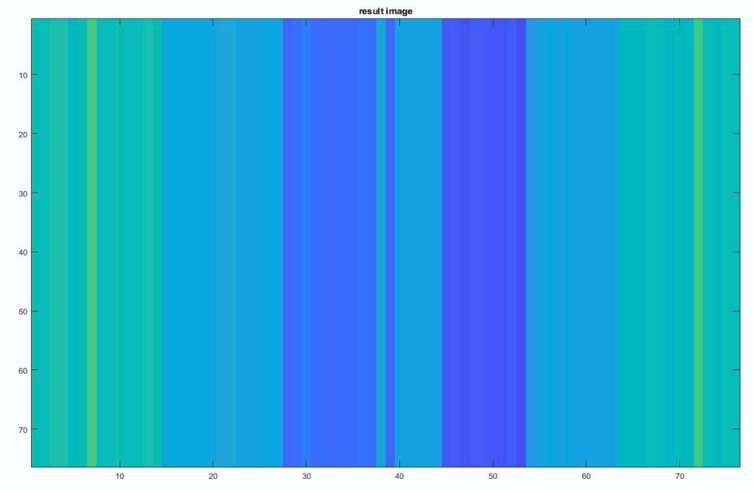
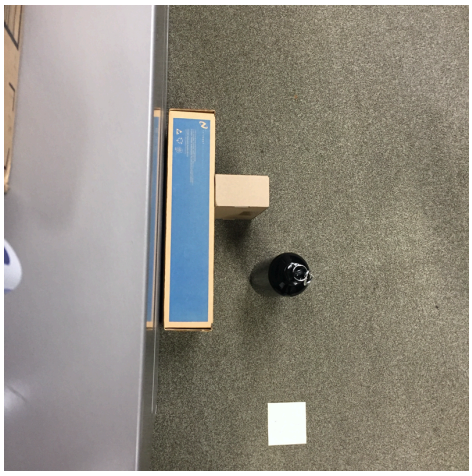
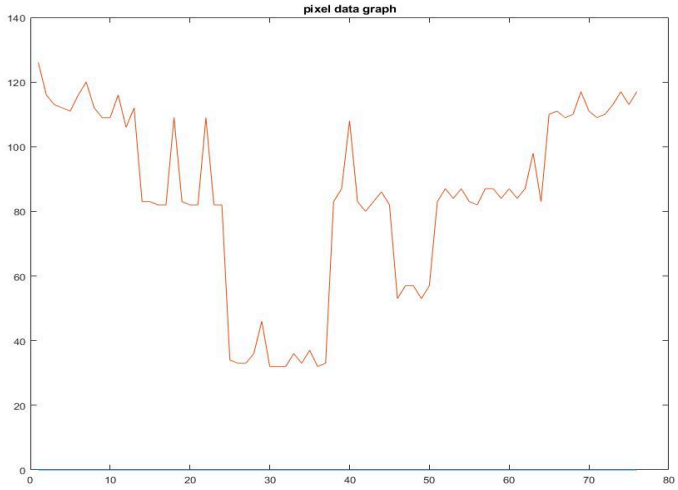


Figure 4.8: Scanning multiple objects with different shape experiment 1 (a) Original setting (b) Plot of the received signal (c) Ultrasound Image

(a)



(b)



(c)

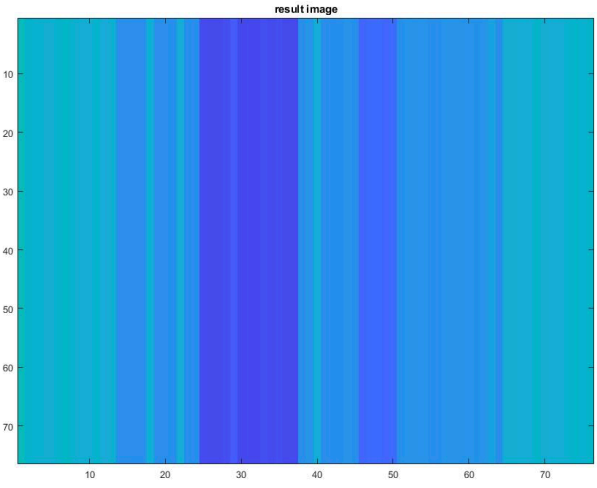


Figure 4.9: Scanning multiple objects with different shape experiment 2 : (a) Original setting (b) Plot of the received signal (c) Ultrasound Image

2-Dimensional Scanning experiment

For the 2-dimensional scanning experiment, we tested with a quarter dollar coin inside a container filled with distilled water, as shown in Figure 4.10. Figure 4.11 shows the initial ultrasound imaging result in grayscale. We also conducted additional noise reduction processes to improve the quality of the image presented in the Jet color scale in Figure 4.12.

Figure 4.10: A quarter-dollar coin used for scanning experiment

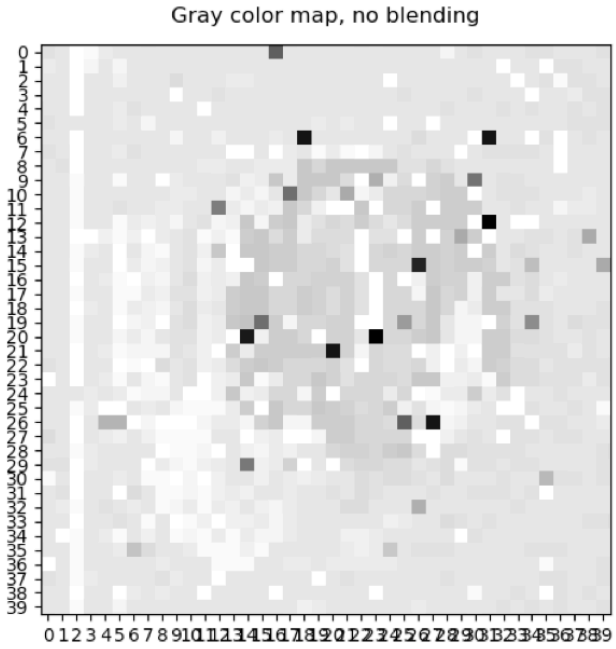


Figure 4.11: Ultrasound imaging result presented in grayscale

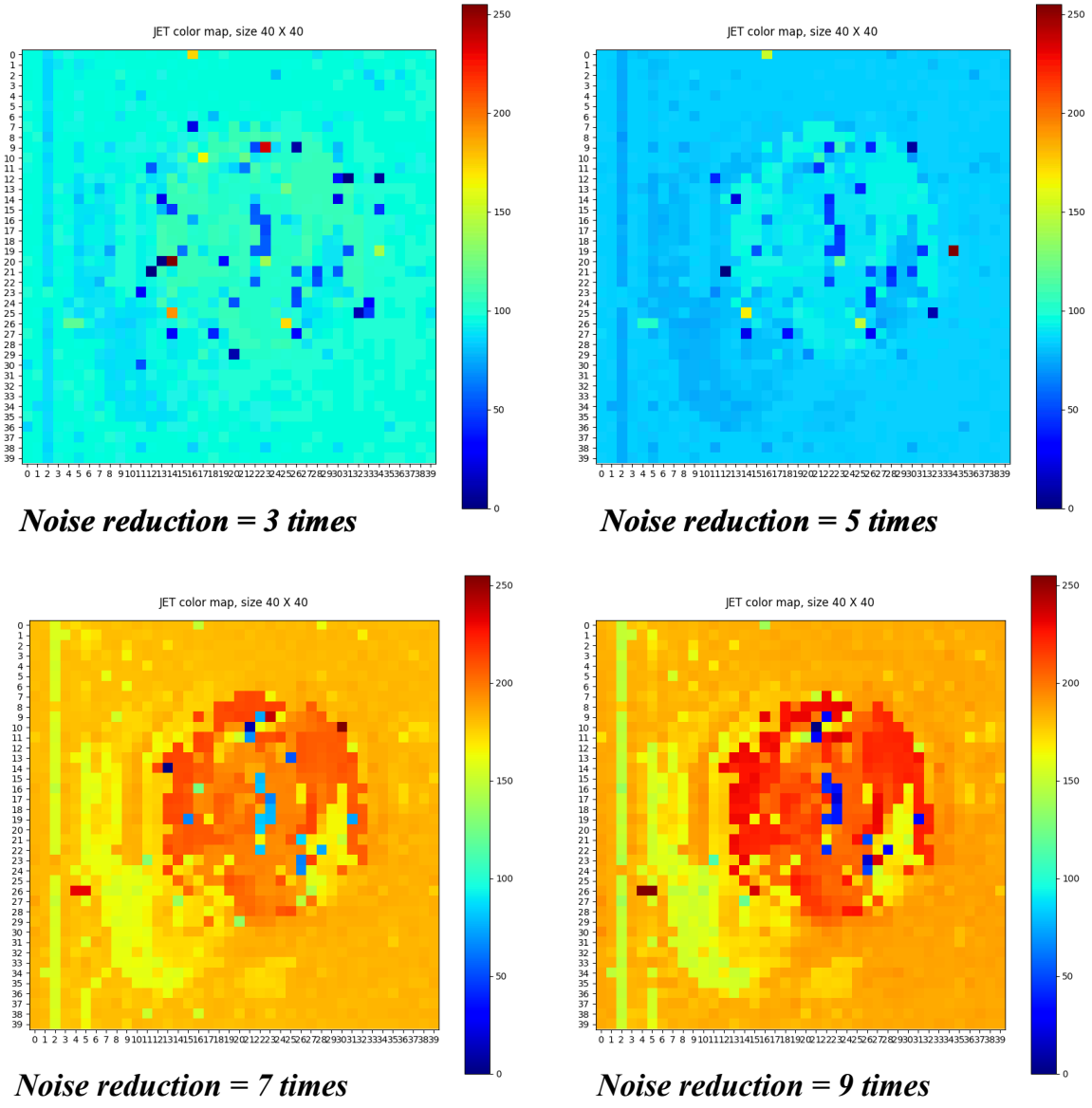


Figure 4.12: Ultrasound imaging result with additional noise reduction presented in the Jet color scale

Chapter 5

Conclusion

5.1 Limitation

In this dissertation, we proposed an initial imaging system prototypes as a proof-of-concept of the finger vein scanning system and showed the results of ultrasonic scanning experiments of multiple objects in air and water. In Chapter 4.2, we distinguish the relative distance and shape of scanned objects. However, as shown in Figure 4.12, a limitation that exists is that we could not identify the engraved pattern of a quarter-dollar coin that we used for the experiment. The reason for the low resolution of our result is that the transducer's resonance frequency was not capable of detecting the detailed patterns of the coin. Lateral resolution is a term used to specify the performance of an ultrasound transducer. It refers to the minimum distance between two reflectors placed horizontally to the direction of the ultrasound beam, that can be distinguished. If the lateral resolution is high, a transducer can better detect a target object's precise patterns. Lateral resolution is proportional to a parameter called *Near-zone length*. The *Near-zone length* is determined by the factors in the equation below:

$$Lateral\ Resolution\ \propto Near\ zone\ length = \frac{diameter^2}{4 \times wavelength} \qquad (3)$$

In equation (3), the *diameter* means the width of a transducer, and the *wavelength* means the transducer's resonant wavelength [24]. The lateral resolution becomes higher when we use a transducer with a wider width and shorter resonant wavelength. Figure 5.1 shows two transducers with different near-zone length. Additionally, there is another method to improve the lateral resolution called *focusing*. *Focusing* is a process of operating Piezoelectric elements in a

transducer with different delay times to create a narrow pulse beam. As shown in Figure 5.2, the narrow pulse beam also shortens the near-zone length to a smaller value called focal length, increasing the lateral resolution at the same time. In conclusion, using a focused-transducer with a higher resonance frequency, which would result in a high lateral resolution, is expected to improve the ultrasound image quality.

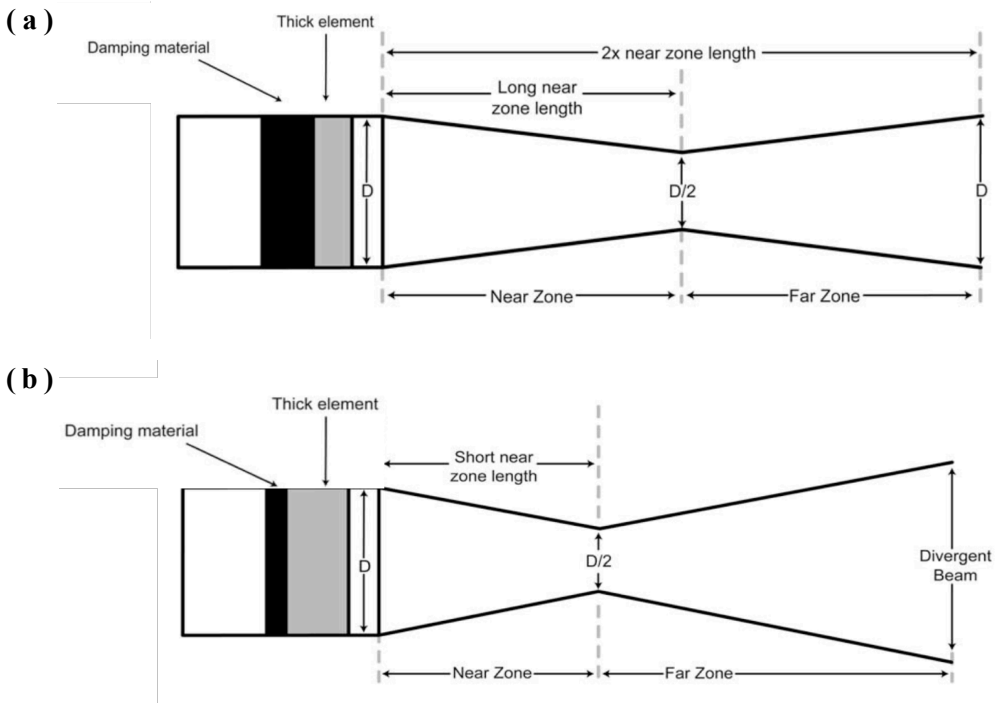


Figure 5.1: (a) High-frequency transducer with narrow width and long near-zone length (b) Low-frequency transducer with wide width and short near-zone length

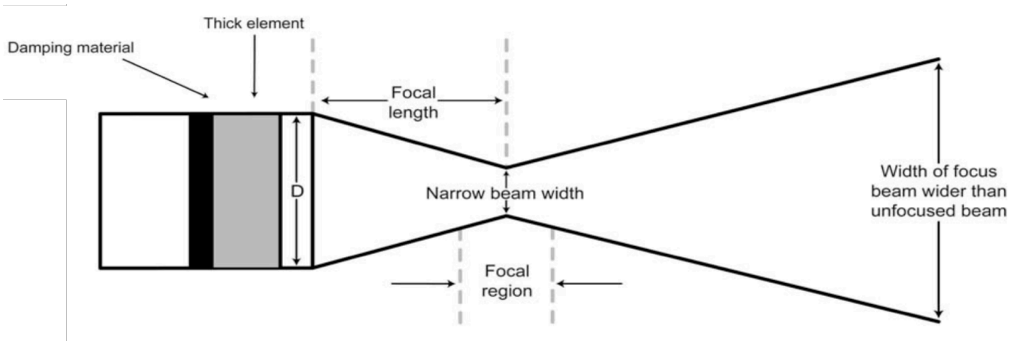


Figure 5.2: Effect of Focusing process [24]

5.2 Future works

We implemented an ultrasonic scanning embedded system that consists of an ultrasound sensor, an analog-to-digital converter, a CNC-milling machine, and a Raspberry Pi 4 computer system. Our ultrasonic scanning embedded system enables the automatic ultrasonic scanning experiment that is an essential process in ultrasonic imaging. With the data collected from ultrasonic scanning experiments, we went through the process of ultrasound imaging. We displayed the ultrasound images that show the structure and shape of a target object. We believe that the system we built can be applied to further ultrasonic scanning experiments targeting complex objects like finger-vein structure.

For the next-step, we expect that additional ultrasonic scanning experiments of more complex structured objects are required to achieve our ultimate goal, an ultrasonic finger-vein biometric authentication system. The human finger has a multi-layer structure with various components inside, as shown in Figure 5.3. For this reason, we expect that identifying the echo signals from each layer and sub-structures will be the major challenge in finger-vein scanning.

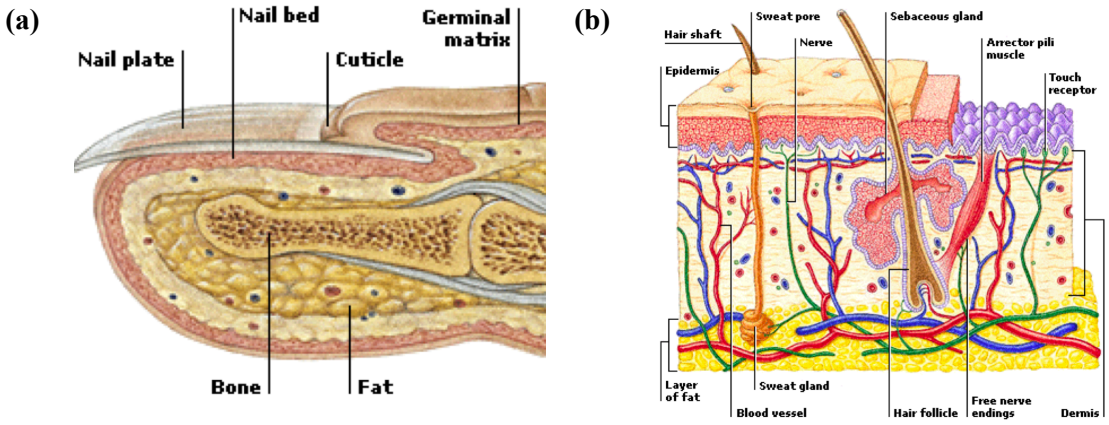


Figure 5.3: Human finger structure: (a) cross-section and (b) skin-diagram [22]

References

[1] Davrondzhon G., Kirsi H., and Torkjel S., "Biometric Gait Authentication Using Accelerometer Sensor", *JOURNAL OF COMPUTERS, VOL. 1, NO. 7, OCTOBER/NOVEMBER* (2006)

[2] Mark Stamp, *Information Security Principles and Practice*, pp. 242-251, John Wiley & Sons Inc. (2011)

[3] Zhang Rui and Zheng Yan, "A Survey on Biometric Authentication: Toward Secure and Privacy-Preserving Identification", *IEEE Access Volume.7* (2018)

[4] S. O'Dea, "Number of smartphone users in the United States from 2018 to 2024, https://www.statista.com/statistics/201182/forecast-of-smartphone-users-in-the-us/, Statistica (Apr 21, 2020)

[5] Samsung Electronics Inc., "Which biometric authentication method is the most secure?", https://insights.samsung.com/2020/02/12/which-biometric-authentication-method-is-the-most-secure-2/, (Feb 12, 2020)

[6] Andrew Bud, "Facing the future: the impact of Apple FaceID", *Biometric Technology Today* (2018)

[7] Lindsey O'Donnell, "Researchers Bypass Apple FaceID Using Biometrics 'Achilles Heel' ", https://threatpost.com/researchers-bypass-apple-faceid-using-biometrics-achilles-heel/147109/ , (August 8, 2019)

[8] Article, https://www.patentlyapple.com/patently-apple/2019/08/while-face-id-was-hacked-at-the-black-hat-conference-the-plausibility-of-it-occurring-could-only-be-found-in-a-bad-b-movie.html

[9] Naoto M., Akio N., and Takafumi M., "Feature extraction of finger-vein patterns based on repeated line tracking and its application to personal identification", *Machine Vision and Applications* (2004)

[10] Wonseok S., Taejeong K., Hee C. K., Joon H. C., Hyoun-Joong K., and Seung-Rae L., "A finger-vein verification system using mean curvature", *Pattern Recognition Letters 32* (2011)

[11] Yanagawa T., Aoki S., Ohyama T., "Human finger vein images are diverse and its patterns are useful for personal identification", *MHF 2007-12* (2007)

[12] Mofiria Corporation, "Comparison of biometric technologies", http://a-mofiria.tokyocms.com/en/about

[13] Kim, Jeong Nyeon, "CLOSED-LOOP FINITE ELEMENT DESIGN OF ARRAY ULTRASONIC TRANSDUCERS FOR HIGH FREQUENCY APPLICATIONS", Ph.D diss., (Pennsylvania State University, 2019)

[14] Robert T., "How fingerprint scanners work: optical, capacitive, and ultrasonic variants explained", https://www.androidauthority.com/how-fingerprint-scanners-work-670934/, (Mar 28, 2019)

[15] Qualcomm Technologies Inc., "Samsung Galaxy S10 taps Qualcomm 3D Sonic Sensor for top-notch security and accuracy [video]", https://www.qualcomm.com/news/onq/2019/02/20/samsung-galaxy-s10-taps-qualcomm-3d-sonic-sensor-top-notch-security-and-accuracy, (Feb 20, 2019)

[16] HC-SR04 Specification, https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf

[17] Raspberry Pi 4 model introduction, https://www.raspberrypi.org/products/raspberry-pi-4-model-b/

[18] Analog Discovery 2 reference manual, https://reference.digilentinc.com/reference/instrumentation/analog-discovery-2/reference-manual

[19] Michael Feldman, "Hilbert transform in vibration analysis", *Mechanical Systems and Signal Processing 25* (2011)

[20] Stefan L. Hahn, Hilbert Transform in Signal Processing, Artech House, s.l., p. 305. (1996)

[21] Shan J. X., Yu L., Sook Y., Jucheng Y. and Dong S. P., "Intensity Variation Normalization for Finger Vein Recognition Using Guided Filter Based Singe Scale Retinex", *MDPI Sensors* (2015)

[22] Finger Cross-section Image, http://www.factmonster.com/dk/science/encyclopedia/skin.html

[23] Lester W. Schmerr Jr., *Fundamentals of Ultrasonic Nondestructive Evaluation – A modeling Approach*, Springer International Publishing Switzerland (2016)

[24] Alexander Ng, "Resolution in ultrasound imaging", pp 186–192, *Anaesthesia Critical Care & Pain*, Volume 11, Issue 5 (2011)

Appendix A

Program for Vehicle Machine Control (written in C language)

<Interval_scan.c>

```
/*************************************************
* Penn State University
* On 11/11/2019
* By Geesun Jang
* Program for 1-dimensional scanning with vehicle machine
*************************************************/


//This code is the main source code(interval_scan.c) of the program.
//For more information, please visit my repository.
//Repo address: https://github.com/geesun56/Pi_ultrasoundsensor_motor


#define PWM_RANGE 50 // Range of motor speed
#define MOTOR_SPEED 50 //Speed of motor

#define TRIG  25 //us
#define GROUND  9950 //us
#define SET_INTER  10000 //us Trigger Interval = signal trigger(52us) + usleep
execute(58us) + set time(us), at this time Trigger interval = 10,000us

#define POINTS 77 // Number of points that we measure distance
#define DELAY_S   2333 // delay for sensor trigger in us
#define DELAY_T_1   62400 // delay for motor in s
#define DELAY_T_2   400000 // delay for motor in s
#define TIMES 1 // Number of times that we measure distance from sensor to one
point

int main( void )
{
// Initialization of parameters

    volatile struct io_peripherals *io;
    uint32_t   DLevel;
    bool done = false;
    io = import_registers();
```

```
  if (io != NULL)
  {
    /* print where the I/O memory was actually mapped to */
    printf( "mem at 0x%8.8X\n", (unsigned int)io );

    enable_pwm_clock( io );

    /*  GPIO settings for motor control  */

    io->gpio.GPFSEL1.field.FSEL2 = GPFSEL_ALTERNATE_FUNCTION0;
    io->gpio.GPFSEL1.field.FSEL3 = GPFSEL_ALTERNATE_FUNCTION0;

    io->gpio.GPFSEL1.field.FSEL8 = GPFSEL_OUTPUT;    //set GPIO18 as AI1 (left
motor direction control)
    io->gpio.GPFSEL1.field.FSEL9 = GPFSEL_OUTPUT;    //set GPIO19 as AI2 (left
motor direction control)
    io->gpio.GPFSEL2.field.FSEL2 = GPFSEL_OUTPUT;    //set GPIO22 as BI1 (right
motor direction control)
    io->gpio.GPFSEL2.field.FSEL3 = GPFSEL_OUTPUT;    //set GPIO27 as BI2 (right
motor direction control)

    io->gpio.GPFSEL2.field.FSEL4 = GPFSEL_OUTPUT;
    io->gpio.GPFSEL2.field.FSEL5 = GPFSEL_OUTPUT;


    /*  configure the PWM channels  */
    io->pwm.RNG1 = PWM_RANGE;     /* the range value, 32 level steps */
    io->pwm.RNG2 = PWM_RANGE;     /* the range value, 32 level steps */
    io->pwm.CTL.field.MODE1 = 0;  /* PWM mode */
    io->pwm.CTL.field.MODE2 = 0;  /* PWM mode */
    io->pwm.CTL.field.RPTL1 = 1;  /* not using FIFO, but repeat the last byte
anyway */
    io->pwm.CTL.field.RPTL2 = 1;  /* not using FIFO, but repeat the last byte
anyway */
    io->pwm.CTL.field.SBIT1 = 0;  /* idle low */
    io->pwm.CTL.field.SBIT2 = 0;  /* idle low */
    io->pwm.CTL.field.POLA1 = 0;  /* non-inverted polarity */
    io->pwm.CTL.field.POLA2 = 0;  /* non-inverted polarity */
    io->pwm.CTL.field.USEF1 = 0;  /* do not use FIFO */
    io->pwm.CTL.field.USEF2 = 0;  /* do not use FIFO */
    io->pwm.CTL.field.MSEN1 = 1;  /* use M/S algorithm */
    io->pwm.CTL.field.MSEN2 = 1;  /* use M/S algorithm */
    io->pwm.CTL.field.CLRF1 = 1;  /* clear the FIFO, even though it is not used
*/
    io->pwm.CTL.field.PWEN1 = 1;  /* enable the PWM channel */
    io->pwm.CTL.field.PWEN2 = 1;  /* enable the PWM channel */
```

```
    int a = 0;
    int b = 0;
    int c = 0;
    int d = 0;

   int _interval_time = SET_INTER - 52 - 58;


    /* Set initial state*/

    GPIO_CLR( &(io->gpio), 18);
    GPIO_CLR( &(io->gpio), 19);
    GPIO_CLR( &(io->gpio), 22);
    GPIO_CLR( &(io->gpio), 23);
    GPIO_CLR( &(io->gpio), 24);
    GPIO_CLR( &(io->gpio), 25);

    int _DLevel = MOTOR_SPEED;
    io->pwm.DAT1 = _DLevel;
    io->pwm.DAT2 = _DLevel;

    printf("Commands : s(scan) / m(move forward)/ / q(stop) / b(go backward)
\n");  // s: start to scan objects while moving motors   m,q,b : motor command
    printf("press command...\n");




      while (!done){
          switch (get_pressed_key())
          {

            case 's':   // Scan process automation

                for (int i = 0; i< POINTS; i++){


                    /*----------motor instructions--------------*/

                    GPIO_SET( &(io->gpio), 19);
                    GPIO_SET( &(io->gpio), 23); //Make the car move

                    usleep(DELAY_T_1);

                    GPIO_CLR( &(io->gpio), 19);
                    GPIO_CLR( &(io->gpio), 23); //Make the car stop
                    usleep(DELAY_T_2);
```

```
/*--------------------sensor instructions----------------------*/
                    // trigger signal : 2 pulse 40kHz burst

                    for(int a = 0; a< TIMES; a++){

                        GPIO_SET( &(io->gpio), 24);
                        for (int j = 0; j<DELAY_S; j++){  //Become 12.5 us ->
40kHz


                        }
                          GPIO_CLR( &(io->gpio), 24);
                          GPIO_SET( &(io->gpio), 26);
                        for (int k = 0; k<DELAY_S; k++){

                        }
                          GPIO_SET( &(io->gpio), 24);
                          GPIO_CLR( &(io->gpio), 26);
                        for (int k = 0; k<DELAY_S; k++){

                        }
                          GPIO_CLR( &(io->gpio), 24);
                          GPIO_SET( &(io->gpio), 26);
                        for (int k = 0; k<DELAY_S; k++){

                        }
                          GPIO_CLR( &(io->gpio), 26);

                          usleep(_interval_time);

                        }


                     //printf("POINTS :%d\n", i);
                     sleep(1.0);

                }
                    done = true;
              break;

            /*  Debug modes for debugging motor process

            case 'm':   // start moving : move forward


                GPIO_SET( &(io->gpio), 19);
                GPIO_SET( &(io->gpio), 23);
                 break;
```

```
            case 'q':   // stop moving

                 GPIO_CLR( &(io->gpio), 18);
                 GPIO_CLR( &(io->gpio), 19);

                GPIO_CLR( &(io->gpio), 22);
                GPIO_CLR( &(io->gpio), 23);

              break;

            case 'b': // move backward
                GPIO_SET( &(io->gpio), 18);
                GPIO_CLR( &(io->gpio), 19);

                GPIO_SET( &(io->gpio), 22);
                GPIO_CLR( &(io->gpio), 23);
                break;
            */

            default:
              done = true;
              printf("DLevel: %d\n",_DLevel);
              break;
          }
    }

    GPIO_CLR( &(io->gpio), 23);
    GPIO_CLR( &(io->gpio), 24);
}
  else
  {
    ; /* warning message already issued */
  }

  return 0;
}
```

Appendix B

Program for CNC-Milling Machine Control (written in C language)

<cnc_mill_controller.c>

```
/*****************************************************
* Penn State University
* On 11/11/2019
* By Geesun Jang
* Program for 2-dimensonal ultrasonic scanning experiment
*****************************************************/


//This code is the main source(cnc_mill_controller.c) code of the program.
//For more information, please visit my repository.
//Repo address: https://github.com/geesun56/cnc-mill-controller


#include "starter.h"
#include "pin_config.h"
#include "axis_control.h"
#include "manual_control.h"
#include "operation.h"
#include "trigger.h"

int main( void )
{
    volatile struct io_peripherals *io;
    operation_status op;
    bool done = false;



    io = import_registers();

  if (io != NULL)
  {

    init_GPIO_pins(io); // Set initial state
    reset_controller(io, &op);
    init_operation_status(&op);
```

```
      while (!done){
            printf("Enter commands (m: manual control / s:scan program / q:quit
program / i:set initial state) \n");
            char ch = get_pressed_key();
            int pinno = decode_pin(ch);
            trigger_GPIO_pin(io, OK, QUICK_PUSH ,QUICK_REST, &op);

                if(ch == 'm'){
                      manual_control(io, &op);
                }
                if(ch == 'i'){
                      float x_axis;
                      float y_axis;
                      float z_axis;
                      printf("Type location data : ");
                      scanf("%f %f %f", &x_axis, &y_axis, &z_axis);
                      change_operation_status(&op, x_axis, y_axis, z_axis);
                      print_operation_status(&op);

                }else if(ch=='s'){
                        printf("Initiailize point\n");
                        move_to_start_point(io, &op);

                        bool scan_done = false;

                        printf("Start scanning(y/n)? \n");
                        char st= get_pressed_key();

                        if(st == 'y')     square_range_scan(io, &op);
                        // Minimum unit scan range 0.1mm to 10mm

                }else if(ch=='z'){
                        done = true;

                }else  {}

        }

            printf("Exit machine... \n");
            exit_machine(io,&op);

        }
  else{  printf("Import registers error");}

  return 0;
}
```

Appendix C

Program for Ultrasound Imaging (written in Python)

<main.py>

```
#  Penn State University
#  On 11/11/2019
#  By Geesun Jang
#  Program for Ultrasound Imaging


#This code is the main source(main.py) code of the program.
# For more information, please visit my repository.
# Repo address: https://github.com/geesun56/Ultrasound-imaging


import processing
from data import signalData;
from matplotlib import pyplot as plt
from processing import image_process


# Data directory
folder_path = './data/04142020'

# Configuration
data_rows = 40
data_cols = 40
frequency = 4 # 4MHz - 25 usec period: 25 entries = signal data during 25 usec
mode = "batch"
data_type = "amplitude"
x = 9
y = 3
signal_data = signalData(folder_path)


if mode == 'batch': # batch process: process all data points at once

    image_data = processing.image_process(signal_data, data_rows, data_cols,
frequency, mode, data_type, x, y)
    plt.figure(figsize=(20,20))
```

```
#use imshow to plot the array
    plt.subplot(131)
    plt.imshow(image_data,               #numpy array generating the image
            cmap = 'gray',             #color map used to specify colors
            interpolation='nearest'  #algorithm used to blend square colors;
with 'nearest' colors will not be blended
            )
    plt.xticks(range(data_cols))
    plt.yticks(range(data_rows))
    plt.title('Gray color map, no blending', y=1.02, fontsize=12)

    plt.show()

elif mode == 'sample':
# sample process : process a single data point for testing

    time, ch1, avg, process, envelop  = processing.image_process(signal_data,
data_rows, data_cols, frequency, mode, x, y)


    fig = plt.figure()

    plt.title('original data')
    plt.xlabel('time')
    plt.ylabel('amplitude')
    plt.plot(time, ch1)

    plt.show()

    plt.title('average data')
    plt.xlabel('time')
    plt.ylabel('amplitude')
    plt.plot(avg)

    plt.show()

    plt.title('diff_filter data')
    plt.xlabel('time')
    plt.ylabel('amplitude')
    plt.plot(process)
    plt.plot(envelop)

    plt.show()

else:
    print("Error: specify mode")
```

<data.py>

```
import csv

points = 350

# <signal scan data format>
# sample_file = '/data_r1c1.csv'
# prefix = '/data_r'
# suffix = '.csv'

# Code for data loading

class signalData:
    def __init__(self, folder):
        self.folder = folder
        self.prefix = '/data_r'
        self.suffix = '.csv'

    def data_load(self, path):
        time = []
        ch1 = []
        ch2 = []

        with open(path, encoding='windows-1252') as csvfile:
            reader = csv.DictReader(csvfile)
            i = 0
            for row in reader:
                i+=1
                #print(row)
                if i > 19 and None in row:
                    #wave_data.append([float(row['#Digilent WaveForms
Oscilloscope Acquisition']), float(row[None][0])])
                    ch1.append(float(row[None][0]))
                    ch2.append(float(row[None][1]))

        # truncate data points

        ch1 = ch1[points:]
        ch2 = ch2[points:]

        for i in range(len(ch1)):
            time.append(i)

        return time, ch1, ch2
```

<processing.py>

```python
import numpy as np
import math
from scipy.signal import hilbert

debug = True

# Code for imaging process
# image_process function integrates other functions at once

def period_cal(freq):
    #calculate the period of resonance frequency
    return int(1/freq*100)

def averaging(data, avg_size):
    #average data for every given avg_size
    avg_data = []
    for i in range(0, len(data)-avg_size):
        avg_data.append(math.fsum(data[i:i+avg_size])/avg_size)

    return avg_data

def diff_filter(data, avg_data):
    #filter the data to reduce noise
    diff_data = []
    for i in range(len(avg_data)):
        diff_data.append(avg_data[i] - data[i])

    return diff_data

def hilbert_trans(data):
    #process hilbert transfrom
    envelop = hilbert(data)
    envelop = abs(envelop)

    return envelop

def argmax(data):
    #find the index of maximum value in an array
    max_point = np.argmax(data, axis=0)

    return max_point
```

```
def convertion_to_pixel(data, base=0):
    #convert signal data to pixel

    data = np.array(data)

    if base==0:
        base = np.amax(data)

    ind = np.unravel_index(np.argmax(data, axis=None), data.shape)
    print('# Process base is ', base,ind[0]+1,ind[1]+1)

    ratio_data = np.true_divide(data, base)
    log_data = np.log(ratio_data)*(-1)
    process_data = log_data/np.amax(log_data)

    print("**Log data**")
    print(log_data)

    print("**process data**")
    print(process_data)
    pixel_data = process_data*255


    return pixel_data


def batch_process_timefly(data, avg_size, times=1):
    #batch process the imaging process by timefly

    pre_data = data

    for i in range(times):
        average_data = averaging(pre_data, avg_size)
        process_data = diff_filter(pre_data, average_data)

        pre_data = process_data
        envelop_data = hilbert_trans(process_data)
        argmax_data = argmax(envelop_data)


    return argmax_data
```

```
def batch_process_amplitude(data, avg_size, times=1):
    #batch process the imaging process by amplitude
    pre_data = data

    for i in range(times):
        average_data = averaging(pre_data, avg_size)
        process_data = diff_filter(pre_data, average_data)

        pre_data = process_data

    envelop_data = hilbert_trans(process_data)
    amp_data = max(envelop_data)

    return amp_data

def sample_convertion(data, avg_size, times=1):
    #process sample data point
    pre_data = data

    for i in range(times):
        average_data = averaging(pre_data, avg_size)
        process_data = diff_filter(pre_data, average_data)

        pre_data = process_data

    envelop_data = hilbert_trans(process_data)

    return average_data, process_data, envelop_data, max(envelop_data)


def image_process(signal_data, row, col, freq, mode, data_type, avg_level=1,
x=1, y=1):
    #main function for overall image processing

    process_data = []
    period = period_cal(freq)
    image_data = []

    if mode == 'batch':
        print('# Processing Info #')
        print('- Mode: ', mode)
        print('- Data type: ', data_type)
        print('- Size: ', row, 'x', col)
        print('- Average level: ', avg_level)
        print('==============================')
        print('Image Processing ( 0 /',row,')')
```

```
        for i in range(1,(row+1)):
            if i%2 == 0:
              print('Image Processing (',i,'/',row,')',)
            row_arr = []
            for j in range(1, (col+1)):
                file_path =
signal_data.folder+signal_data.prefix+str(i)+'c'+str(j)+signal_data.suffix
                time, ch1, ch2 = signal_data.data_load(file_path)
                if data_type == 'time':
                    result = batch_process_timefly(ch1, period, avg_level)
                else:
                    result = batch_process_amplitude(ch1, period, avg_level)
                #print(file_path)

                row_arr.append(result)

            process_data.append(row_arr)

        print('================================')
        print('processData result:')
        print(process_data)
        np.savetxt('process_data.txt',process_data)

        #used 2600 for previous experiment

        image_data = convertion_to_pixel(process_data)
        np.savetxt('image_data.txt',image_data)
        return image_data

    elif mode == 'sample':
        print("# Start sample process of point (", x,',', y,')')
        file_path =
signal_data.folder+signal_data.prefix+str(x)+'c'+str(y)+signal_data.suffix
        time, ch1, ch2 = signal_data.data_load(file_path)
        print(file_path)
        avg, pro, env, maxVal = sample_convertion(ch1, period, avg_level)

        return time, ch1, avg, pro, env, print(maxVal)
    else:
        return print('ERROR: select mode for image processing')
```