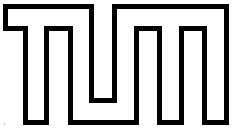


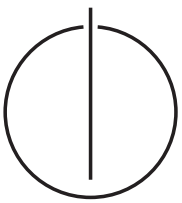
FAKULTÄT FÜR INFORMATIK
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

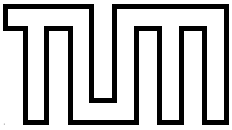
Master's Thesis in Informatik

GPU Ultrasound Simulation and Volume Reconstruction

In collaboration with Siemens Corporate Research Inc., Princeton, USA

Athanasios Karamalis





FAKULTÄT FÜR INFORMATIK
DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatik

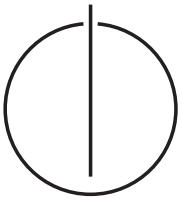
In collaboration with Siemens Corporate Research Inc.,

Princeton, USA

GPU Ultrasound Simulation and Volume Reconstruction

GPU Ultraschall Simulation und Volumen Rekonstruktion

Bearbeiter:	Athanasios Karamalis
Aufgabensteller:	Prof. Dr. Nassir Navab
Betreuer:	Oliver Kutter Dr. Wolfgang Wein
Abgabedatum:	13. Februar 2009



Erklärung

Ich versichere, daß ich diese Master's Thesis selbständig verfaßt und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

Declaration

I assure the single handed composition of this Master's Thesis only supported by declared resources.

München, den 13. Februar 2009 Author

Contents

Abstract

Medical ultrasound imaging has been in clinical use for decades, however, acquisition and interpretation of ultrasound images still requires experience. For this reason, ultrasound simulation for training purposes is gaining importance. Additionally, simulated images can be used for the multimodal registration of Ultrasound and Computed Tomography (CT) images. The simulation process is a computationally demanding task. Thus, in this thesis a simulation method, accelerated by modern graphics hardware (GPU), is introduced.

The accelerated simulation utilizes a ray-based simulation model in order to provide real-time high-throughput image simulation for training and registration purposes. Wave-based simulation methods are computationally even more demanding and have been considered unsuitable for real-time applications. In the scope of this thesis, wave-based models have been investigated, including the Digital Waveguide Mesh and the Finite-Difference Time-Domain method for solving Westervelt's equation. Initial results demonstrate the feasibility of performing near real-time wave-based ultrasound simulation using graphics hardware.

Furthermore, a new algorithm is introduced for volumetric reconstruction of freehand (3D) ultrasound. The proposed algorithm intelligently divides the work between CPU and GPU for optimal performance. The results demonstrate superior performance and equivalent reconstruction quality compared to existing state of the art methods. GPU accelerated ultrasound simulation and freehand volume reconstruction are key components for fast 3D-3D (dense deformable) multimodal registration of Ultrasound and CT images, which is subject of current ongoing work.

Zusammenfassung

Medizinischer Ultraschall (Sonographie) wird schon seid Jahrzehnten im klinischen Umfeld genutzt, wobei für die Aufnahme und Interpretation von Ultraschall immer noch Erfahrung nötig ist. Aus diesem Grund steigt das Interesse an der Ultraschall-Simulation, da sie für Trainingszwecke verwendet werden kann. Ausserdem können simulierte Bilder dazu genutzt werden um eine multimodale Registrierung von Ultraschall und Computertomographie (CT) Bildern durchzuführen. Die Simulation hat einen beachtlichen rechnerischen Aufwand. Deshalb wird in dieser Thesis eine Simulationsmethode vorgestellt, die mit Hilfe von Graphik-Hardware (GPU) beschleunigt ist.

Die beschleunigte Methode verwendet ein strahlenbasiertes Simulationsmodell um die Echtzeit-Simulation von Bildern und einen hohen Datenfluss zu gewährleisten. Wellenbasierte Simulationsmodelle haben rechnerisch gesehen noch höhere Ansprüche und wurden somit als nicht echtzeitfähige Modelle angesehen. Im Verlauf dieser Thesis wurden jedoch wellenbasierte Methoden untersucht. Dazu gehören das 'Digital Waveguide Mesh' und die 'Finite-Difference Time-Domain' Methode zur Lösung der Westervelt Gleichung. Die Resultate demonstrieren die Machbarkeit von wellenbasierter Echtzeit-Ultraschall-Simulation, durch die Verwendung von Graphik-Hardware.

Ausserdem wird ein neuer Algorithmus zur Rekonstruktion von Freihand-3D-Ultraschall vorgestellt. Der vorgeschlagene Algorithmus teilt den Arbeitsaufwand intelligent zwischen CPU und GPU auf, um eine optimal Auslastung zu erreichen. Die Resultate belegen eine höhere Effizienz und eine äquivalente Qualität der Rekonstruktion im Vergleich zu existierenden Methoden, die den jetzigen Stand der Technik ausmachen. GPU-beschleunigte Ultraschall-Simulation und Volumen-Rekonstruktion sind die Hauptkomponenten für eine schnelle 3D-3D ('dense deformable') multimodale Registrierung zwischen Ultraschall und CT Bilder, wobei diese zur Zeit bearbeitet wird.

Acknowledgements

First of all, I would like to thank Wolfgang Wein for his continuous support and the motivation he game me during my entire internship at Siemens Corporate Research (SCR). His valuable advice and his insight into the things which really mattered helped me all the way during this thesis and it was a pleasure working with him. I would also like to thank Oliver Kutter, my mentor back at Munich, for teaching me a lot of things and for supporting me before, during and beyond this thesis.

Furthermore, I would like to thank all the people at SCR who helped me during my internship. Especially, Christoph Vetter and Sylvain Jaume for having time listening to my questions and giving me valuable advice. Patric Ljung, for spending late night office hours in order to find the cause of an interesting problem and possible solutions. Fang Xu and Oliver Fluck, for their support on GPU stuff. And Ilangovane Sinouvassin for helping me getting along with existing software modules.

Additionally, I would like to thank Prof. Nassir Navab for introducing me to his CAMP Chair at Munich, for recommending me to SCR and for the inspiring discussions we shared. Last but not least, I would like to thank my parents for their enduring support and, of course, Marilena for always being there for me.

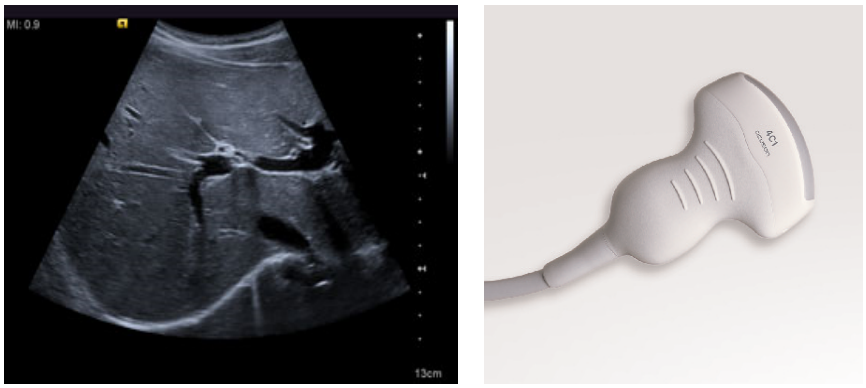
Chapter 1

Introduction to medical ultrasound

In this chapter will be presented the essentials of medical ultrasound imaging, which will be of use for the upcoming chapters on ultrasound simulation. It begins with the absolute basics of medical ultrasound and image formation, in order to give an overview of the imaging modality. Afterwards, the basics of ultrasound physics are presented, which are relevant not only to the imaging modality but also to understand the approaches taken in the chapters referring to the simulation. Next comes a brief introduction to the image acquisition process, the devices (transducers) used for ultrasound transmission and reception and the post processing of the received signals. The following two sections deal with the characteristics of ultrasound images and the artifacts observed, moreover the reasons for these artifacts are discussed. Some of the bioeffects associated with ultrasound are presented, which are relevant to the last chapters of this thesis. Lastly, a brief comparison between medical ultrasound and other imaging modalities is presented. Throughout this chapter we will assume that sound waves are modeled as rays. This is a common simplification that makes an introduction into the topic easier. In the later chapters a wave based modeling will be introduced, which is the physical correct way of modeling sound waves. This chapter is almost entirely based on the books written by Zagzebski [58] and Fisher [13], thus, omitting further references to these books in this chapter.

1.1 Medical ultrasound in a nutshell

Medical ultrasound is a non invasive imaging technique that uses sound pulses in order to visualize biological tissue, and in our case human tissue. An example of an ultrasound image, showing a human liver, is presented in figure 1.1 (a). The general acquisition process for an ultrasound image is schematically presented in figure 1.2. For this the ultrasound transducer device, also shown in figure 1.1 (b), is positioned at the patients skin using a special gel in order to improve the transmission from the transducer into the patient's tissue. In this simple example the transducer sends a single sound pulse which travels into the patient's body. As the pulse travels through the tissue it encounters tissue layers with varying densities. The sound pulse is reflected at the interface/boundary between two tissues having different densities, while the higher the difference the stronger the resulting echoes are. For each reflection the sound pulse energy is reduced, however, the remaining energy contin-



(a) (b)

Figure 1.1: Image (a) shows an ultrasound image from the human liver produced by the Siemens Acuson 2000. Image (b) shows a curved ultrasound transducer (Siemens 4C1).

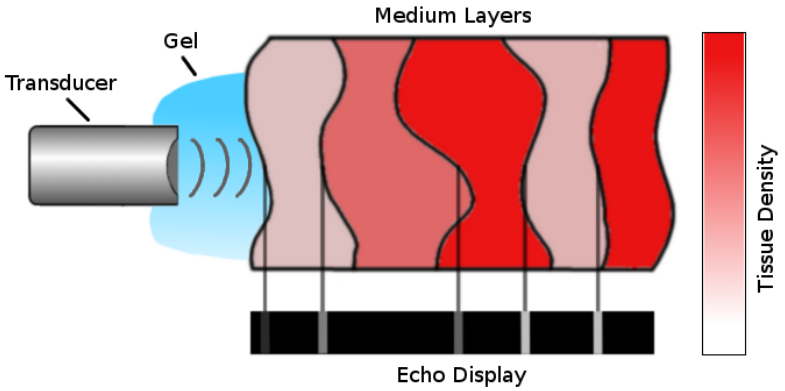


Figure 1.2: Acquisition of ultrasound

ues to travel through the tissue until the initial pulse energy is exhausted. The resulting echoes are detected by the transducer. The depths of the interfaces are estimated based on the time interval between sending the initial pulse and receiving the echoes, which is also known as the pulse-echo principle. The received signal is post processed and the resulting echoes are displayed using the depth information and a brightness value according to the signals strength of the received echoes. More details on the image acquisition process will be presented in the upcoming sections. A comprehensive overview on image formation in real-time ultrasound can be found in Hedrick and Hykes [17].

1.2 Basic physics of (ultra)sound

In this section the basic physics of sound are presented, which are relevant to ultrasound image acquisition and formation. For the mathematical modeling of the physics of sound a

ray-based model is used, which captures most of the relevant effects that produce the final image, but is still quite a simplification of the problem. However, this simplification makes the introduction into this topic a lot easier. Furthermore, ray-based models are commonly used due to their lower computational demands compared to wave-based models.

1.2.1 Sound waves

Generally speaking sound is a disturbance that propagates through a medium without transporting particles but rather oscillating/vibrating them. The reason for the propagation lies in the fact that each medium exhibits elastic restoration forces that move the displaced particles back to their original position. Sound waves belong to the category of mechanical waves, meaning that they need a medium to propagate. On the other hand, electromagnetic waves like light and X-Rays do not need a medium in order to propagate. Furthermore, mechanical waves are generally classified as longitudinal (figure 1.3) and transverse (figure 1.4) waves. Transverse waves can effectively travel through solid materials, like steel and bone. On the other hand, longitudinal sound waves can travel efficiently through tissue, which makes them important for ultrasound imaging.

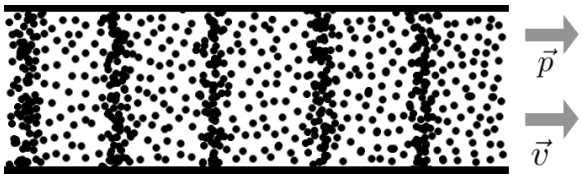


Figure 1.3: 1D Longitudinal Wave, the direction of oscillation $\vec{v}$ is parallel to the direction of propagation $\vec{p}$. The particles oscillate back and forth, compressing and expanding the medium.

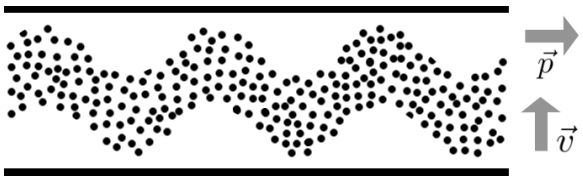


Figure 1.4: 1D Transverse Wave, the direction of oscillation $\vec{v}$ is perpendicular to the direction of propagation $\vec{p}$. The particles oscillate up and down, altering the shear stress of the medium.

A variety of quantities and measurements are closely related to ultrasound and are comprehensively presented. Period and frequency are related since the frequency f is the number of wave cycles in unit time $f = 1/T$, where T is the time for a single wave cycle. A wave cycle refers to the time between the pressure level reaching twice its maximum, or equivalent twice its minimum. The distance between those two pressure levels is known as the wavelength. In medical ultrasound the applied frequencies can range from 1 to 40 MHz, with most ultrasound systems assuming a constant propagation speed of $1540m/s$ for human tissue. The wavelength λ and the frequency are reversely related to each other $\lambda = c/f$. The previous equation can lead to a common misconception which relates the propagation

speed to the wavelength and frequency. However, the propagation speed depends only on the characteristics of the medium and is not affected by changes of the wavelength and frequency. The propagation speed of sound inside a medium depends on the compressibility and density of the medium and is given as follows (Eq. 1.1):

$$c = \sqrt{\frac{K}{\rho}} \tag{1.1}$$

where: c is the speed (m/s) , ρ is the density (kg/m^3) and K is the elastic modulus (stiffness) $(kgm^{-1}s^{-2})$. Some characteristic speed measurements for different mediums can be found in table 1.1. The denser the medium the faster the sound waves can travel and vice versa.

Table 1.1: Speed of sound for different mediums

Material	Speed of sound (m/s)
Air	330
Water	1480
Lead	2400
Aluminium	6400
Lung	600
Fat	1460
Aqueous humor	1510
Liver	1555
Blood	1560
Kidney	1565
Muscle	1600
Lens of eye	1620
Skull bone	4080

Another important quantity is the (acoustic) pressure of a sound wave, which refers to the deviation from the ambient (equilibrium) pressure caused by the sound wave and is measured in Pa (Pascal) 1. As the longitudinal wave propagates through the medium, the particles of the medium are compressed and expended. This is also called condensation and rarefaction respectively, while the maximum condensation occurs at the maximum pressure level and vice versa. A similar quantity is the acoustic intensity, which is the sound power per unit area (W/m^2) 1 and is defined as 1.2:

$$I = \frac{P^2}{2\rho c} \tag{1.2}$$

where again ρ is the density of the medium and c is the speed of sound.

The perceived loudness of a sound is nonlinear in regard to its intensity, meaning that in order to make a sound twice as loud it's intensity has to be multiplied by 10. Therefore, the loudness is expressed in decibels for the sake of convenience. The decibel scale is a

logarithmic scale expressing the acoustic intensity based on a reference, which is usually $I_0 = 1{\cdot}10^{-12}(W/m^2)$. To convert a given intensity I to the dB scale we use $I_{dB} = 10\log_{10} I/I_0$.

This was a comprehensive overview of the most important quantities measured and observed in medical ultrasound. They will come in handy in the upcoming sections and in the chapters referring to the simulation of ultrasound.

1.2.2 Reflections

Sound is reflected at the boundary/interface of two mediums having different densities, which also indicates different propagation speeds for sound waves in the mediums. The angle of the reflection is equal to the angle of incidence, with the incoming and the outgoing sound beams being on the same plane as the normal of the interface. The quantity used to calculate the intensity of the reflections is the characteristic impedance of the mediums, which is given by:

$$Z = \rho c \qquad \text{rayls or } kgm^{-2}s^{-1} \tag{1.3}$$

where ρ is the medium density and c is the propagation speed in the medium. Table 1.2 shows some characteristic impedances for human tissue.

Table 1.2: Acoustic impedances for human tissue

Tissue	Impedance (rayls)
Air	$0.0004 \cdot 10^6$
Lung	$0.18 \cdot 10^6$
Fat	$1.34 \cdot 10^6$
Water	$1.48 \cdot 10^6$
Liver	$1.65 \cdot 10^6$
Blood	$1.65 \cdot 10^6$
Kidney	$1.63 \cdot 10^6$
Muscle	$1.71 \cdot 10^6$
Skull bone	$7.8 \cdot 10^6$

Using the characteristic impedances someone can calculate the intensity reflection coefficient R and intensity transmission coefficient T at the interface between two mediums (M_1, M_2) as below:

$$R = \frac{I_r}{I_i} = \frac{(Z_2 - Z_1)^2}{(Z_1 + Z_2)^2} \tag{1.4}$$

$$T = \frac{I_t}{I_i} = \frac{4Z_1Z_2}{(Z_1 + Z_2)^2} = (1 - R) \tag{1.5}$$

where I_r, I_t, I_i are the reflection, the transmission and the incoming intensity respectively.

Some reflection coefficients for selected tissue boundaries are presented in table 1.3. The intensity reflection coefficient for the muscle-air interface is about 0.98, therefore, the intensity transmission coefficient is about 0.02. This means that almost the entire acoustic energy is reflected at this interface and only a small portion is transmitted into the underlying tissue. The same kind of reflection occurs when the transducer is placed on the patient's skin, since some air is always trapped between the transducer and the skin. To handle this problem, a special gel is used between the transducer and the skin, as previously shown in figure 1.2. This gel has about the same propagation speed as human tissue ($1540m/s$) and is relatively free from air bubbles. Therefore, it prevents initial reflections at the transducer-skin interface and enables the initial sound pulses to efficiently pass into the underlying tissue.

Table 1.3: Reflection coefficients for selected tissue interfaces.

Interface	I_r/I_i
Kidney-liver	0.00004
Liver-muscle	0.0003
Fat-liver	0.01
Muscle-Bone	0.41
Muscle-Air	0.98

The intensity of the received echoes depends on the angle between the incoming sound beam and the interface normal. This can efficiently be modeled with the Lambert's cosine law, which states that the outgoing intensity depends on the incoming intensity and the cosine between the incoming beam direction and the interface normal. Therefore, the smaller the described angle is, the higher is the received echo intensity.

$$I_{out} = I_{in} \cdot cos(\theta) \tag{1.6}$$

1.2.3 Scattering

Most of the echoes from human tissue come from small reflectors called scatterers. The size of scatterers is about the same as the wavelength of the transmitted sound pulses. They are named scatterers because they scatter the incoming sound waves in all directions. Regions having a higher scattering level compared to their surrounding tissue are called hyperechoic, whereas regions having a lower scattering level are called hypoechoic. Echoes from scatterers depend on the following factors:

- The number of scatterers per unit volume
- The acoustic impedance changes at the scatterers interfaces
- The size of the scatterers (Scattering usually increases with increasing radius for very small scatterers)
- The ultrasonic frequency (Scattering usually increases with increasing frequency for very small scatterers)

- The constructive and destructive interference of echoes from scatterers (Details follow in the interference section)

Scatterers that are much smaller than the transmitted wavelength are called Rayleigh scatterers. Red blood cells are Rayleigh scatterers and emit very weak echoes which are undetectable by ultrasound systems. However, the Doppler frequency shift caused by the blood cells motion can be detected and is used for ultrasound flow imaging, which is called Doppler imaging.

1.2.4 Refraction

If a sound beam travels through mediums with different propagation speeds, then the direction of propagation changes at the interface of the two mediums. This effect is called refraction and the angle of refraction is defined by Snell's Law:

$$\frac{\sin\theta_i}{\sin\theta_t} = \frac{c_1}{c_2} \tag{1.7}$$

where θ_i is the angle between the interface normal and the incoming ray, θ_t is the angle between the negative interface normal and the transmitted ray, c_1 is the propagation speed of the medium before the interface and c_2 is the propagation speed of the medium beyond the interface. Human tissue is an inhomogeneous medium, but the changes of propagation speed are generally small and do not result in large scale refractions. However, there are cases where the refraction effect can cause significant artifacts in the acquired images, as for example with muscle-fat and muscle-bone interfaces. This is one of the reasons why ultrasound imaging for obese patients and female breast imaging results in relatively poor image quality.

1.2.5 Interference

Sound waves meeting in space interact with each other based on their phase. If the waves are 'in phase' constructive interference occurs and the waves combine to a stronger wave. On the other hand, if the waves are 'out of phase' destructive interference occurs and the waves cancel out energy from each other, resulting in a weaker wave. Constructive and destructive interferences strongly occur with scatterers and are the main cause for the final echo response from the scattering regions.

1.2.6 Diffraction

Diffraction is defined as the phenomenon were a wave spreads/bends around an object or beyond small openings. The amount of diffraction depends on the size of the object or the size of the opening and the wavelength.

1.2.7 Attenuation

Sound attenuation refers to the reduction of the initial wave energy as the wave travels through different mediums. One form of attenuation is absorption through the medium, which is basically the conversion of the wave's kinetic energy into heat. Through this process the wave intensity becomes lower, the further the wave travels from its source (the transducer). The attenuation depends on the frequency and increases with higher frequencies, while decreases with lower frequencies. This means that there is always a trade-off between high image resolution (high frequencies) and target tissue penetration depth (lower frequencies). That is because for high frequencies the penetration depth decreases due to attenuation, but the image resolution improves since the wavelength becomes smaller and echoes depict structures in more detail, with the opposite being true for lower frequencies. The attenuation does not only depend on the frequency, but also on the tissue. Some attenuation coefficients for different tissues are presented in table (1.4), based on [58] and [57]. Another source of attenuation are reflections at medium interfaces, where the attenuation increases with the difference between the characteristic impedances of the two mediums forming the interface. Lastly, attenuation occurs in form of destructive interferences which are highly present in scattering regions.

Table 1.4: Attenuation coefficients for tissues

Tissue	Attenuation dB/cm/MHz
Water	0.002
Blood	0.18
Kidney	0.50
Liver	0.79
Muscle	1.2
Gall bladder	1.51

1.3 Ultrasound image acquisition, formation and instrumentation

As stated at the beginning of this chapter, the time between sending an initial sound pulse and receiving an echo is used to estimate the depth of a given reflector in the tissue. Thus, the distance D from the transducer to the reflector is calculated using the pulse-echo principle:

$$T = \frac{2D}{c} \tag{1.8}$$

where c is the propagation speed and T is the time interval. Imaging instrumentation always assumes a constant speed of $c = 1540m/s$ (except in special applications like ophthalmology). This necessary assumption leads to image artifacts, which will be discussed later on.

The pulse transmitter generates the sound pulses and includes the pulse-repetition-frequency (PRF) generator. The repetition frequency must be small enough to allow an initially transmitted pulse to travel to the desired depth and back to the transducer. To achieve

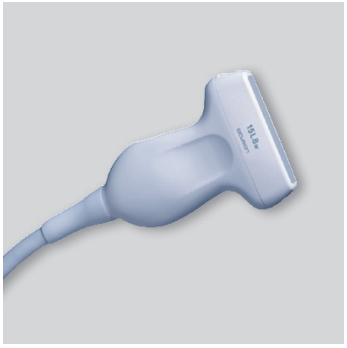
the desired penetration depth one has to adjust the power output of the transducer, which is usually denoted with the MI value on the imaging systems. The MI value is given by the ratio of peak negative pressure (MPa) of an ultrasound wave propagation in a uniform medium by the square root of the transmission frequency (MHz). As the output increases:

- The pulse amplitude increases.
- The transducer emits higher intensity sound waves.
- Echoes appear brighter.
- New echoes from weak reflectors are detected.
- Thermal exposure to tissue increases. High intensity ultrasound will irreparably damage tissue (Used in cancer therapy). [21]

1.3.1 Transducers

The ultrasound transducer devices are used to send the initial sound pulses and receive the echoes from the tissue. For this purpose piezoelectric elements are placed at the transducer surface that will later on lie on the patient's skin. Piezoelectric elements tend to change size (compress and expand) as electric current is applied to them . This property is used by transducers in order to generate sound pulses, since the change in size results in compressing and expanding near particles, which is the basic concept of sound generation itself. By applying altering voltage to the elements the transducer can generate sound pulses of the desired amplitude and frequency. Another useful property of piezoelectric elements is that they generate electric current when external forces are applied, like sound. Therefore, the same elements can also be used to detect echoes, since the electric current generated by the elements is proportional to the intensity of the received echo. Nowadays, ultrasound transducers are made of numerous piezoelectric elements which can individually be controlled for transmission and reception.

There are three main categories for transducers, namely: Linear, Curved and Phased Transducers. In case of linear transducers (figure 1.5 a) the elements are arranged on one line and pulses are transmitted by triggering a group of elements simultaneously. This process continues from one side of the transducer to the other by moving the group of elements one element at a time. The curvilinear transducers (figure 1.5 b) operate in the same way as linear transducers, however, the elements are placed on a curved surface in order to provide better entrance windows (contact area between transducer and skin) and a wider field of view. Phased array transducers (figure 1.5 c) use all the elements to form a single sound beam during transmission and use again all the elements during reception. The sound beam can be electronically steered and multiple scanlines are acquired to form the final image. Scanlines are lines which represent each individual sound beam and contain the depth information and the brightness information for interfaces along the sound beam, as illustrated by the black line in figure 1.2. More details on beam steering will follow in the focusing subsection. Phased array transducers have a small entrance window (1-2cm) and can be efficiently used in electrocardiography and pediatric imaging.



(a)Linear Transducer Siemens 15L8

(b)Curvilinear Transducer Siemens 4C1

(c)Phased Transducer Siemens 4V1C

Figure 1.5: Ultrasound transducers

A relatively new transducer type are 2D-array transducers, which use a 2D-matrix array of individually controlled transducer elements. This element arrangement allows the acquisition of three dimensional ultrasound volumes in real-time. Three dimensional ultrasound is constantly gaining importance in clinical practice offering additional diagnostic information compared to conventional 2D ultrasound imaging [9]. Therefore, 2D-array transducer are slowly but constantly becoming more part of clinical practice.

1.3.2 Focusing

Hyugen's principle states that complex wavefronts can be formed by combining multiple single waves. This principle is used in ultrasound imaging for beam focusing and forming, also known as electrical focusing. Previously it was mentioned, that ultrasound transducers use multiple individually controlled elements. Thus, a specific wavefront can be formed by triggering the elements with different delays, so that the single sound waves emitted by the individual elements form the desired wavefront. Adjusting the wavefront results in actually forming the sound beam in which most of the energy of all the individual waves is present. Phased transducers make use of this technique to form a sound beam that is steered through the tissue in order to acquire each scanline separately. During acquisition the appropriate delays are used to compensate for the initial wavefront delays, before processing the final echo signal.

Focusing refers to the process of forming a wavefront in which the individual waves will constructively interfere at a specific region in the sound beam. The echoes from the target region will be enhanced, because the overall sound intensity will be higher at this region compared to that of the surrounding tissue. Therefore, focusing is mainly used to enhance the echoes from a given region of interest. However, high intensity focusing is used in therapeutic ultrasound applications like cancer treatment. A study on ultrasound surgery of the female breast can be found in Hynynen et. al [21]. Multiple zone focusing is a technique were different focus zones are specified and the resulting echoes are superpositioned to form the final image. The number of focal zones decreases the overall framerate since we have to wait separately for the echoes of each focal zone.

1.3.3 Post processing

At this point the echoes for an initial transmission have been received, but before the signal is ready to be displayed as a 2D grey-scale image, some post processing steps are received or performed to improve the image quality. An initially post processing step is the amplifications of the signal in order to compensate for the losses cased by attenuation. The degree of amplification is called the gain of the receiver and is the ratio of the output signal amplitude to the input signal amplitude, usually in dB. The initially received signal is amplified at different stages. The first amplification is performed at the preamplifier, which is close to the transducer and boosts the weak echoes, protecting, at the same time, the main amplifier from too high signal inputs. The second amplification is performed at the so called main amplifier, which boosts the overall received signal and is adjusted with the overall gain control. Note that there is a difference between the gain and the overall gain control. The first amplifies the transmitted pulse whereas the second amplifies the received signal.

Performing an overall amplification of the signal would result in highly amplified initial echoes that were close to the transducer and not significantly attenuated. Therefore, a mechanism called Time Gain Compensation (TGC) is applied, which boosts echoes based on their reception time, thus, boosting more the delayed and further away from the transducer signals. The TGC can be adjusted for the needs of the examination using different controls. The lateral gain is another amplification control that can amplify each scanline individually and is used in Echocardiography. The number of scanlines used is usually equal to the number of available transducer elements.

The limited input signal amplitude range over which an electronic device will respond effectively is called dynamic range. The received signals are higher than the dynamic range of imaging devices (monitors) and have to be scaled. Therefore, a so called dynamic range compensation is applied which is a logarithmic scaling that boosts low intensity echoes and reduces high intensity echoes. This process does also emphasize scatterers which make up the majority of echoes in an ultrasound image. The dynamic range can be adjusted to emphasize structures of interest, but misadjustment can lead to overlooking important weak echoes.

After this processing steps the compensated and amplified signal is converted into a single pulse for each reflector, which is called demodulation. For this, the radio frequency (RF) signal is rectified (negative components are inversed) and then smoothed. This results in a signal that is an approximation to the envelope of the received signal, so it combines small signal responses into a single response for a reflector.

1.4 Image Characteristics

The interpretation of ultrasound images requires experience with the modality in order to associate the image information with the corresponding anatomical structures. However, all ultrasound images share common image characteristics which are introduced in the following section.

The main echo source in ultrasound images are scatterers that correspond to the numerous bright 'dots' in the image, as marked with (1) in figure (1.6). This dot pattern is also

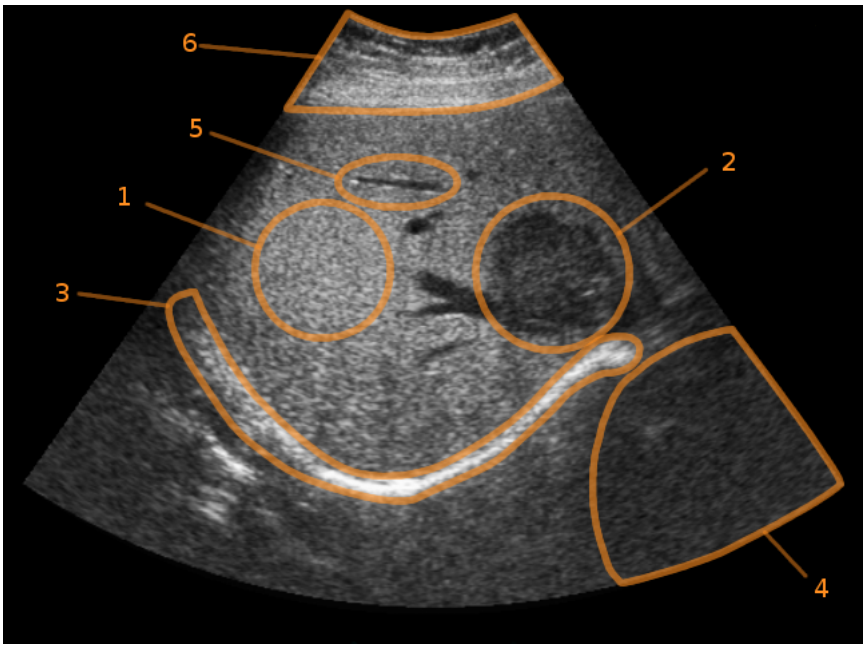


Figure 1.6: Ultrasound image of a human liver - Image Courtesy (Siemens Corporate Research, Princeton, NJ, USA) The labeled regions show: (1) Speckle, (2) Tumor, (3) large scale reflecting interface, (4) shadow region (5) blood vessel and (6) skin layers.

known as speckle or ultrasound texture. The exact tissue interfaces which cause the sound waves to scatter are still investigated, however, it is evident that scatterers are generally numerous and randomly distributed throughout tissue. Scatterers are very small (about the wavelength of the transmitted sound pulse) and are detected as a group by the transducer rather than individual reflectors. The intensity variations of speckle are caused by constructive or destructive interferences at the given scattering region. The position of the transducer does not play a significant role for the resulting echoes, since scatterers are reflecting the incoming waves in all directions. Speckle might initially seem as noise, however, it provides valuable diagnostic information. For example, figure (1.6) shows a dark region in the speckle pattern marked with (2), which is actually a liver tumor that has also been verified with CT imaging. Speckle is also useful for the detection of cysts and organ abnormalities which are identified through abnormal speckle intensity at the region of interest.

Another source of echoes are specular reflectors which are mainly observed at organ/tissue boundaries. The bright horizontal reflection, marked with (3) in figure (1.6), comes from the kidney-lung interface. The high intensity of the reflection is due to the high difference of acoustic impedances between the two tissues. The echo intensity from specular reflectors depends on the angle between the incoming sound beam and the interface normal, as previously described in equaiton 1.6.

Another characteristic of ultrasound images are the dark regions, which result from an effect called shadowing. As the sound waves travel through the tissue they may encounter strongly reflecting interfaces and lose considerable energy. The attenuated sound beam might not be strong enough to travel deeper into the tissue, thus, the regions beyond this depth appear almost echo free as shadows in the image. The region marked with (4) in fig-

ure (1.6) shows a shadowed region resulting from the strong reflection at the tissue-spine interface. Generally, tissue-bone interfaces result in nearly complete reflection of the sound pulse and pose the maximum penetration depth for such scans. Another cause for shadowing are highly attenuating tissue regions like destructively interfering scattering regions.

Ultrasound imaging is a real-time imaging technique. Therefore, one might assume that blood flow would be visible in the images in the form of rapidly moving small reflectors. However, the echoes from blood cells can not be detected by transducers because they are too weak and suppressed by other echoes in the signal. Thus, blood vessels appear as echo free regions in the images and can be detected based on their characteristic shape, marked with (5) in figure (1.6). Furthermore, small reflections from the blood vessel walls are also indicating the existence of a blood vessel in an image. In order to visualize the blow flow, an ultrasound imaging technique called Doppler imaging is used, which detects the Doppler frequency shift caused by the motion of the blood in order to acquire and visualize the flow information.

Another image characteristic varies depending on the transducers used and the pressure applied to the patient's skin. This characteristic is the bright region at the top of the images, marked with (6) in figure (1.6). This region is the result of the combined effects of the following factors. First, the gel used to position the transducer on the patient's skin is not perfect, thus, small reflections occur at the gel-skin interface. Second, after the skin tissue a fat layer is encountered which also results in additional echoes. Lastly, initial echoes might reflect at the transducer itself and travel back into the tissue, resulting in multiple echoes close to the transducer.

1.5 Image artifacts

This section presents an overview of image artifacts found in ultrasound imaging, which basically result from the nature of sound itself and the assumptions made by ultrasound systems. Some of these artifacts may lead to erroneous images and difficulties in examining the anatomical structure of interest, while others provide additional diagnostic information.

Imaging systems assume a constant propagation speed of $1540m/s$ in human tissue, which is an unavoidable assumption in order to estimate the depth of a reflector based on the pulse-echo principle. However, as we know the propagation speed varies throughout tissue and this assumption leads to erroneous depth estimations. Another artifact that originates from the speed variations in human tissue is refraction. The angle of refraction at an interface depends on the difference of propagation speed between the mediums forming the interface. If the difference is high enough, a significant redirection of the initial sound beam can occur, as for example in case of tissue-bone and tissue-fat interfaces. Therefore, echoes that originate from interfaces outside the sound beam might be interpreted as if they are located on the sound beam. This is due to the fact that for each sound beam all received echoes (independent of their origin) are interpreted as interfaces located on the sound beam.

There are cases in which a strong interface (like fat-muscle, tissue-bone) can cause a strong echo which travels back to the transducer and is reflected from it back into the tissue. This second wave is then again reflected at the interface back to the transducer, with this process continuing until the wave energy is exhausted. As a result, this interface would

be displayed multiple times at different depths with subsequent lower brightness. This is because the system will register the echoes multiple times at larger time intervals and estimate the interface at different depths. The brightness will be reduced due to the attenuation of the initial sound pulse. The described effect is called reverberation and can sometimes mask other important/wanted echoes in the image or make imaging of echo free structures, like aortas, difficult. However, there are cases in which reverberation is valuable as a diagnostic tool. Metallic objects such as foreign bodies or biopsy needles cause the sound to ring inside the metallic body, producing a clear pattern that allows to distinct the object from the surrounding tissue. This is also vary valuable for the evaluation of gun-shoot wounds, since bullets appear clearly in ultrasound images which can be used in emergency procedures. However, the most common benefit arises from identifying the tip of a biopsy needle as it can clearly be seen in the ultrasound image, also called the comet-tail effect. Another cause for reverberations are small bubbles or liquids at different locations including the diaphragm and the gallbladder.

It is often observed that after strongly reflecting interfaces - like the diaphragm-lung interface - additional reflectors occur even though they are anatomically not present in that region. For example reflections of the liver can be observed beyond the diaphragm. These are called mirror image artifacts and they are closely related to reverberation artifacts since they follow a similar formation process. Mirror images are formed in the following way: The initial sound pulse is reflected at the region that is later being mirrored. The remaining pulse is then nearly fully reflected at the strong interface. The pulse travels back and is again reflected at the region where in turn the pulse is again reflected at the strong interface. Through this back-and-forth, the region is displayed twice in the final image, where the second echo is temporally further away and is displayed deeper (maybe even after the strong interface) in the image.

Sound beams generated by the simultaneous triggering of individual transducer elements have an actual width and cause echo information from closely positioned reflectors inside the beam to be obliterated. This causes small point-like reflectors to be interpreted as lines in the images, which is also the reason why anechoic structures, like blood vessels, are filled with echoes from surrounding reflectors. Not only has the sound beam a width but also a thickness that varies with depth. This effect is called slice thickness artifact and is analogous to the beam width artifact, where echoes from interfaces close and perpendicular to the imaging plane are included in the image, filling-in again hypoechoic/anechoic structures. Since the slice thickness depends on the beam focusing and on the depth, echoes from similar interfaces can be stronger on a narrow slice thickness and weaker on a broader slice thickness. Contrary to the reverberation effect, the beam width and slice thickness effects do not provide additional clinical value at all and decrease the overall quality of images.

Ultrasound transducers can not be perfectly constructed and transmit pulses off the main beam axis, which are known as lobes. There are two main categories for lobes. Side lobes occur for both single and array transducers and emit sound pulses at a small angle off the main beam axis. The echoes resulting from those are placed on the active beam since ultrasound systems always assume that all echoes (independent of their origin) are from interfaces along the beam. Grating lobes are present for array transducers only and they are caused by the fact that the transducer surface is not continuous but constructed through numerous individual transducer elements. Grating lobes send pulses with a large angle off the main beam and can cause more significant artifacts from false echoes in the image. Fortunately, there

are numerous ways to reduce the effect of side and grating lobes by new transducer designs and by sophisticated transducer element triggering schemes that suppress them greatly.

1.6 Bioeffects

A brief discussion about bioeffects associated with ultrasound is presented in this section, starting with heating. During ultrasound exposure the kinetic energy of the sound waves is partially transformed into heat as it passes through tissue, while the higher the initial acoustic intensity is, the higher the temperature rise in the tissue. Attention is paid especially when it comes to the use of high acoustic power levels with focusing schemes. Focusing the sound beam at a specific region gives rise to the acoustic pressure at those regions, which in turn increases the temperature at the target region. If the power levels are too high, extensive heating can occur which might irreparably damage the tissue [21]. For this reason, ultrasound systems have power limiters to prevent extensive thermal exposure. However, therapeutic ultrasound systems allow high intensity focused ultrasound in order to perform focused ultrasound surgery (cancer treatment, gallstone treatment). Cavitation is an effect that causes the creation of bubbles in the sound field that oscillate with the driving pulse. Stable cavitation only oscillate the bubbles while inertial cavitation causes the bubble to collapse, affecting the surrounding region. The effects of ultrasound can be predicted to some extend for a range of frequencies and mediums, however, a variety of effects are observed which exhibit a nonlinear behavior in regard to frequency and tissue. For example, nonlinear effects are the cause of harmonic frequencies. Harmonic frequencies are detected echo frequencies that are multiples of the fundamental (initial) frequency used for the transmission. Another example of nonlinear effects are shock waves which form because of the nonlinear relationship between acoustic pressure and medium compressibility for high intensity short duration pulses. For this kind of pulses the positive wave cycle travels faster through the medium than the negative cycle, catching up with the negative cycle and resulting in a rapid pressure change.

1.7 Medical Ultrasound compared to other imaging modalities

Nowadays, various medical imaging modalities are at the disposal of physicians for the visualization of the human body. Based on a variety of factors the appropriate imaging modality is used in order to deliver the necessary image information. Available imaging modalities include: X-ray (Film), Fluoroscopy, Angiography, Computed Tomography (CT), Magnetic Resonance Imaging (MRI) and Positron Emission Tomography (PET). There are also extensions of these modalities and possible adjustments in order to deliver the desired images.

Compared to these imaging modalities, medical ultrasound demonstrates some considerable advantages, including its real-time capability and portability. Together with the relatively small instrumentation size, ultrasound is favorable for interventional and emergency procedures. Other advantages come from the fact that no ionizing radiation is used, while all other imaging techniques apply low to high radiation doses to patients and sometimes even physicians. Furthermore, ultrasound demonstrates a high spatial resolution and is relatively

cheaper than the other imaging modalities. On the other hand, the downsides of ultrasound imaging include a low signal to noise ratio and image artifacts like shadowing and noise. Shadowing is the result of a limited depth penetration, which is especially a problem for bone or air interfaces. Therefore, ultrasound can image a subset of the target anatomical regions that the other modalities can image. Lastly, the acquisition and interpretation of the images require considerable experience, so the quality of the examination depends on the skills of the sonographer.

Other than conventional 2D grey-scale images, medical ultrasound can also provide Doppler imaging. This imaging technique allows blood flow visualization (Color Doppler) and blood volume visualization (Power Doppler), without the need of administrating contrast agents, like radioactive agents used in other modalities. Nevertheless, ultrasound contrast agents, also known as microbubbles, have been introduced in the recent years and opened new ways in ultrasound imaging, further increasing its diagnostic value. The interested reader can find comprehensive overviews about ultrasound imaging using contrast agents in Frinking et. al [14] and Correas et. al [7].

Chapter 2

GPU Ray-based Ultrasound Simulation

Part of this chapter has been published at SPIE Medical Imaging 2009 with the title: 'A GPU-Based Framework for Simulation of Medical Ultrasound' [31]

In this chapter the simulation of ultrasound will be presented using a ray-based simulation model and graphics hardware for acceleration purposes. The goal of this work is to provide realistic ultrasound images which can be used for training and image registration applications, ensuring real-time performance in the sense of providing at least 30 simulated images per second (30Hz). The chapter starts with an initial motivation including target applications and a discussion about the required performance goal. In the next section, will be presented the related work on ultrasound simulation in regard to real-time simulation approaches. Currently, no literature was found demonstrating real-time simulation of 2D ultrasound images using wave-based simulation models. Thus, the literature on wave-based approaches will be presented in the next chapter. After the related work, an overview of the new method for GPU ultrasound simulation will be presented, where details on the implementation will follow throughout the chapter. The initial simulation framework is further extended by including additional effects in order to produce more realistic images. Lastly, the results will be presented, which include performance measurements and qualitative comparisons of the simulated images.

2.1 Motivation

In recent years the importance of simulators in medicine has increased, especially in regard to minimally invasive procedures [51]. Simulators for ultrasound guided needle procedures [34], [53] demonstrated the potential of ultrasound simulation in first validation studies. Ultrasound acquisition requires experience and good anatomical knowledge, because of the difficult image acquisition and interpretation. In traditional apprenticeship training the trainee would perform imaging on real patients under the guidance of an experienced sonographer or radiologist in case of minimal invasive procedures. This training method is expensive since an ultrasound system and an expert are required. Furthermore,

in case of minimal invasive procedures like tissue biopsy, abscess drainage and nephrostomy, the trainee might cause pain and complications to the patient. Therefore, ultrasound simulators provide a cost-efficient way of training without the involvement of patients or experts (to some extent). Recent simulators are based on CT volume datasets, as they are commonly available in hospitals, while the availability of ultrasound datasets is more limited. Moreover, reslicing an ultrasound dataset for simulation will not yield the desired imaging since ultrasound acquisition is view dependent. Therefore, in order to provide an ultrasound acquisition suitable to be resliced during simulation the acquisition would have to be performed for multiple views. However, this is rather time consuming and the acquisition would not be reslicable for any possible transducer position and orientation. On the other hand, simulators based on CT datasets allow arbitrary viewing planes and offer a wider range of available pathological cases which can be studied for a variety of patient anatomies (size, age, gender). Combining ultrasound simulation with needle procedure simulation offers the potential of performing offline training of the difficult to master procedure or to perform a preoperative simulation before performing the actual invasive procedure. Needle procedures require psychomotor abilities and good hand-eye coordination which can only be improved through practice, possibly provided by simulators. In addition, the performance of the trainees can be evaluated on the training system in order to assess the performance of a task and determine whether a trainee is ready to perform the task on real patients or not.

A new application domain for ultrasound simulation is the multimodal registration of CT and Ultrasound images. This new approach was introduced by Wein et al. [54], and can fully automatically register freehand ultrasound sweeps with CT volumes. The fusion of the two modalities introduces additional clinical value to the acquisition since anatomical information which is not present in one modality can be found in the other one and vice versa.

2.2 Related Work

In Hostettler et al. [19] an ultrasound simulator is introduced based on CT volume datasets. The method suggests the use of an initial volumetric ray-tracing in order to estimate the acoustic behavior of each pixel in the ultrasound image plane and to detect and label interfaces. The processed data is used to create an echo image, an absorption image and a texture image, where the texture image is blurred using noise. Combining all the images yields the final simulated image. The resulting images seem to cover only basic ultrasound image characteristics, while implementation details and performance measurements have not been published.

In Magee et al. [34] a simulator for ultrasound guided needle placement training is presented. The system comprises a latex mannequin, a mock transducer probe, a standard biopsy needle, two magnetic tracking sensors and the software framework. A first validation study showed positive feedback from radiologists after using the proposed system. Furthermore, the educational value and the potential for error reduction and skill enhancement was pointed out during the evaluation process. The method used for generating simulated ultrasound images will be discussed, where details considering the other system components will be omitted. Initially, the CT volume is manually segmented and each voxel is assigned to the

physical and anatomical property of the tissue in the CT dataset. Thus, different tissue types like the liver, blood vessels, air, and bones are assigned with different labels. Afterwards, the volume is registered to the surface model of the mannequin. Moreover, the volume is smoothed with a Level-Set method in order to avoid the step-like effect present in slices taken from non-axial directions of the CT volume. The method uses a database of volumetric solid textures from real ultrasound images for each classified label (tissue type). A Laplacian image pyramid is used to represent the image and volume statistics in the database. The previous tasks are performed offline, where the simulated image is calculated on-the-fly based on the probe position and orientation. Through raster scanning of the pixels in the ultrasound image plane, each pixel is projected into the 3D volume space in order to determine the corresponding pixel label from the labeled voxels. The label is, in turn, used to access the appropriate texture for each pixel from the texture database. The resulting image still lacks ultrasound-specific artifacts like speckle noise, shadows and attenuation, which are added to the image in a later stage. The speckle is simulated by adding Gaussian distributed noise to the image pixels. The shadowing effect is realized by a 2D ray-casting approach for which straight line segments from the top to the bottom of the image are used to record intersections with air and bone. The more such intersections occur, the darker the pixels become beyond these intersections. This approach is realized during the rasterization process in order to save the computational effort of a separate rasterization pass. The attenuation is realized by using a function to scale the brightness of the image pixels. Thus, the further the pixels are from the transducer, the lower is the intensity. Lastly, a radial blur effect is realized with alpha blending on graphics hardware by blending multiple images rotated along the blurring center. The quality of the simulated ultrasound images is good, especially when the additional ultrasound-specific artifacts are included. However, the main drawback of the approach is the need for manually segmenting and labeling the CT volume in an offline processing step.

In Vidal et al. [53] another simulator is introduced for ultrasound guided needle puncture which uses two haptic devices as user-interface for the transducer and the biopsy needle. The haptic devices enable force feedback to give the feeling of the tissue resistance on the needle and skin resistance on the probe. Of interest is the method used for the simulation of the ultrasound images, where other details considering the simulator will be omitted. The method performs the following six steps for the simulation. In the first step, an MPR slice is acquired from the CT volume dataset by slicing a 3D texture containing the CT volume on the GPU. Furthermore, in a GLSL shader each voxel that is intersected by the needle is filled with the maximum pixel intensity in the resulting MPR slice. In the second step, an image mask is calculated in order to simulate shadowing effects. The authors mention that the mask is calculated on the CPU because of limitations considering the GPU programmability. Later on in this chapter, a method will be presented which overcomes this limitation to increase the performance of the simulation. The third step, involves the generation of a noise image, by taking an MPR from a 3D texture containing procedural noise. In the forth step, the images are combined using multi texturing on the graphics hardware. The combined images are used in the fifth step to simulate large scale reflections using a Sobel filter. In the final step, the image brightness and contrast are adjusted and a Look-Up-Texture (LUT) is used to remap the pixel intensities to more ultrasound-like intensities. The quality of the simulated images is sufficient for the target application, however, it does not really represent ultrasound specific imaging. This may be due to lack of using an explicit simulation model.

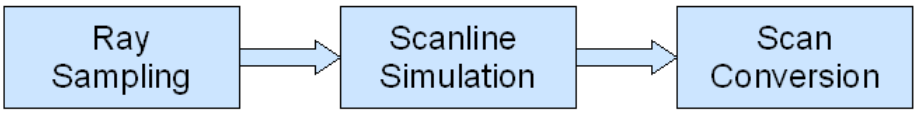


Figure 2.1: Three stage implementation.

In Wein et al. [54] was introduced a simulation method which simulates images suitable for the multimodal registration of CT and Ultrasound image. For the simulation the image pixels are processed in a rasterization scheme which also takes into account the transducer geometry used, emulating a ray-casting through the CT volume. Each of the pixels is transformed into the 3D volume space in order to access the corresponding CT intensity value. The values are used to generate a reflection and an echogeneity image. The reflection image is generated by calculating the reflections along the sound beams as they travel through the tissue (the CT volume). Also the transmission is calculated, which produces the shadowing effect in the final image. The reflection and transmission coefficients are calculated based on an ultrasound physics model, which will be discussed in detail later on together with the modeling of the reflected intensity. The echogeneity image is generated by mapping the CT intensity values sampled to acoustic impedance values, based on a predefined function. The resulting images are then combined into a single image. The simulation quality is good and the images demonstrate the characteristic ultrasound effects of reflection and shadowing. Adding Perlin noise to simulate speckle increases the overall realism of the images and makes them more suitable for training.

The simulation of ultrasound for training purposes requires real-time simulation of the images with a refresh rate of at least 30Hz. In addition, the realism of the images should be sufficient enough to represent the ultrasound specific image characteristics. For the multimodal registration the simulation quality must only be sufficient enough for the registration purpose, where the performance of the simulation becomes more important. Accelerating the simulation of the images would also speed up the registration process. To address both application domains, registration and training, the method proposed by Wein et al. [54] will be used as a basis for GPU ray-based ultrasound simulation. Additionally, the method includes a physical correct modeling of ray-based ultrasound propagation and can be improved by extending the model itself. Literature suggests the use of wave-based models which simulate most of the wave phenomena present with ultrasound. However, the computational demand for this approaches are considered prohibiting for a real-time simulations. More on this topic will be discussed in the chapter referred to the wave-based simulation.

2.3 Method Overview

An overview will be presented before going into the implementation details of the method. The method can be divided into three stages, as shown in figure 2.1.

- **Ray Sampling:** For each transducer element a sound beam (ray) is calculated which propagates through the CT volume. This is similar to the ray-based physics model

presented in the chapter referred to 'Introduction to Medical Ultrasound'. The CT volume intensities are sampled along the sound ray at equidistant sampling points. Afterwards, the sampled intensities are used by the scanline simulation stage.

- **Scanline Simulation:** In this stage, the simulation model is implemented. Each scanline is calculated separately by simulating the ultrasound intensity for the given sampling point along a sound ray. Thus, a scanline image is calculated for which each scanline corresponds to one sound beam from the transducer and each pixel along the vertical axis to one sampling point on the sound ray.

- **Scan Conversion:** The scanline simulation stage results two images, the reflection and the echogeneity image. The images are combined in the scan conversion stage. If a curved transducer is utilized the scanlines must be scan converted from a Polar coordinate system to a Cartesian coordinate system. For linear transducers a mere resampling is performed to account for the physical size of the image.

2.4 Adapting the Wein et al. Model

In this section the modifications on the Wein et al. model will be presented, which were introduced for an efficient GPU implementation. As stated in the chapter referred to 'Introduction to Medical Ultrasound', the cause for reflections is the passing of a sound beam through mediums with different acoustic impedances. Therefore, the reflection at the k-th sample along a sound beam is calculate as follows:

$$I_r^k = I_i^k \frac{(Z_2 - Z_1)^2}{(Z_1 + Z_2)^2} \tag{2.1}$$

where I_r^k is the reflected intensity, I_i^k is the incoming intensity, Z_1, Z_2 is the acoustic impedance at the current and at the next sample respectively. This means that the intensity of the reflection depends naturally on the incoming energy and the difference in acoustic impedance between the mediums. The necessary acoustic impedance values are not available for a CT volume dataset. Therefore, a transfer function is used [54] which maps CT Hounsfield units to acoustic impedance values. The scalar field of the CT dataset is denoted $f(\vec{x})$ were $\vec{x}$ is a position inside the 3D scalar field and $f(x) \in [0..4096]$. The mapping is denoted as $\mu(f\vec{x})$ and the acoustic impedance values from equation 2.1 become:

$$Z_1 = \mu(f(x)), Z_2 = \mu(f(x + \Delta s)) \tag{2.2}$$

where Δs is the distance between the two sampling points along the ray. A reflection results in some of the sound energy being reflected and some being transmitted deeper into the tissue. The transmitted intensity I_t^k at the k-th sample along a sound beam is defined as:

$$I_t^k = I_i^k \frac{4Z_1 Z_2}{(Z_1 + Z_2)^2} \tag{2.3}$$

Taking this into account, the incoming intensity for a given sample k is equal to the transmitted intensity of the previous sample $k - 1$. Subsequently, the incoming intensity along a

ray is calculated recursively as below:

$$I_i^k = \begin{cases} I_t^{k-1} & : k > 0 \\ 1 & : k = 0 \end{cases} \tag{2.4}$$

As discussed in the introduction, the angle between the incoming ray and the interface normal is crucial for the intensity of the reflected echo which is detected by the transducer. This effect is modeled using the Lambertian cosine law, thus, the reflected intensity becomes:

$$I_r^k = I_i^k \frac{(Z_2 - Z_1)^2}{(Z_1 + Z_2)^2} \cdot cos(\theta) \tag{2.5}$$

where θ is the angle between incoming beam vector $\vec{i}_k$ and the interface normal. The angle can be calculated as below:

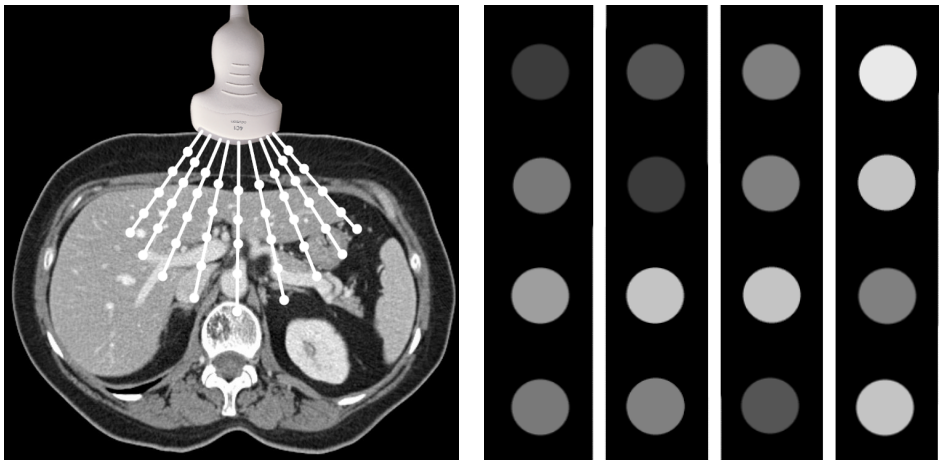
$$cos(\theta) = \frac{\vec{i}_k \cdot \nabla\mu(f(\vec{x}))}{\left\|\vec{i}_k\right\| \cdot \|\nabla\mu(f(\vec{x}))\|} \tag{2.6}$$

The intensity values for the reflection are not in the dynamic range of the imaging devices. For this reason, a log-compression is used to suppress strong echoes and enhance weak echoes. The final reflection intensity is, thus, defined as:

$$I_r^k = \frac{log(\alpha \cdot I_r^k + 1)}{log(\alpha + 1)} \tag{2.7}$$

where $\alpha \in \mathbb{R}$ is the compression parameter. Previously, it was mentioned that two images are used to form the final simulated image, which are the reflection and the echogeneity image. The reflection image contains the log-compressed reflection intensities for the scanlines. Accordingly, the echogeneity image contains the echogeneity values from the CT Hounsfield mapping. A byproduct of the simulation process is the transmission image, which contains the transmission coefficients.

The proposed model is limited in simulating reflection and shadowing effects (based on the transmission). However, compared to the other previously mentioned models, this approach incorporates a physics based simulation model. Additional effects like scattering, interference and diffraction are correctly incorporated only by wave-based models, which are computationally more demanding. A drawback of the method is the use of the transfer function for mapping CT Hounsfield units to acoustic impedance values. Although, the values for this function come from actual measurement, it is still a rough approximation for this mapping which, in turn, introduces errors to the simulation. Nevertheless, the resulting images are realistic enough for multimodal registration and training applications. Lastly, there is a common error source for methods which are based on CT volume datasets. This error results from the fact that CT images can not image all the anatomy that is really present in the imaged regions. This is also the reason why multimodal image fusion is valuable, since anatomy not present in one modality can be found in the other. However, the missing anatomy in the CT datasets results in missing anatomy in the simulated images which is unavoidable.



(a)Ray sampling points (b)Samples in scanlines

Figure 2.2: Image (a) shows schematically how the samples are acquired along sound rays. The sound rays originate from the transducer elements and are sampled at equidistant sampling points. Image (b) shows a schematic snapshot from four scanlines containing four intensity values sampled from the CT dataset along the sound rays.

2.5 Method

In the following sections the implementation of a GPU ray-based ultrasound simulation will be presented. To fully comprehend this section, the reader should be familiar with the basics of GPGPU programming concepts. For additional information please refer to Rost [46], Luebke et al. [33] and the website of the GPGPU community [1]. The interested reader is encouraged to take a look at the GPU Gems book series ([12], [42] and [38]) which include a large amount of GPU algorithms for visualization and general purpose computation. The books are currently available for free at the Nvidia developer homepage [2]. The proposed method is implemented using C++, OpenGL [40] and GLSL (OpenGL Shading Language) [46].

2.5.1 Ray-Sampling

The ray sampling is the first stage of the method in which the required CT intensity values are sampled from the volume dataset, as demonstrated in figure 2.2 (a). The ray sampling process is quite similar to ray-casting used for visualizing 3D datasets. An efficient and widely used algorithm for ray-casting using graphics hardware has been proposed by Krüger and Westermann in [30], where subsequent improvements resulted in better quality and performance, as presented in Scharsach [49]. The interested reader is encouraged to read the paper as it will provide more insight into ray-casting, making it easier to comprehend this section. In ray-casting each ray is modeled using a starting position $\vec{p_n}$ and a direction vector $\vec{d_n}$, where n is the total number of rays used. The positions and directions of the rays are encoded as two 2D 3-channel (RGB) textures and uploaded in the GPU memory. Afterwards, the blending equation for the ray-casting is computed inside a fragment

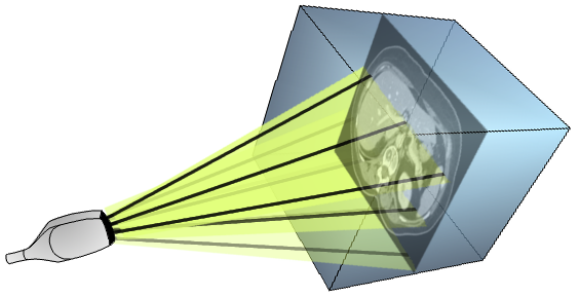


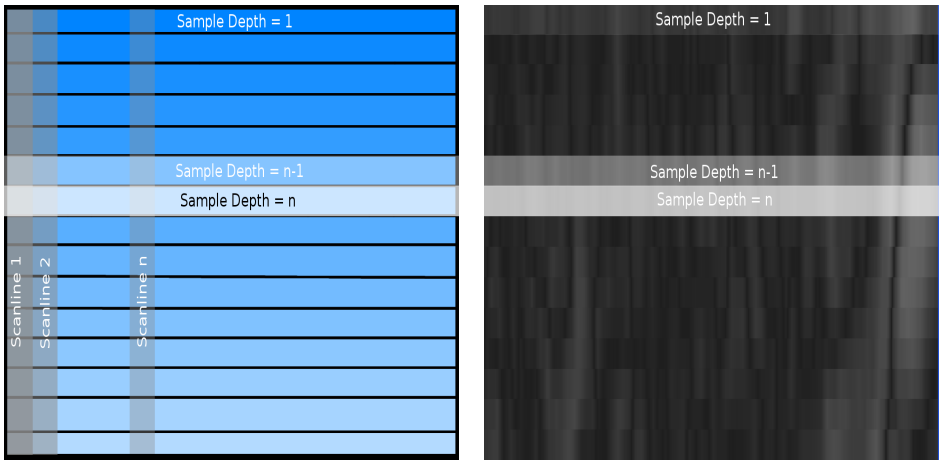
Figure 2.3: Schematic representation of ray sampling for a 2D transducer. The planes from the transducer represent different rows of the 2D element matrix, while the lines represent sound beams from elements on the transducer. The sampling is performed for each sound beam independently.

shader program [46] using samples from equidistant sampling points along the ray. Finally, for each ray a single pixel intensity is returned, which is saved in a 2D RGBA texture for visualization.

For the ultrasound simulation the ray-casting approach was modified. The key difference between ray-casting and the GPU ultrasound simulation is that ray-casting needs an RGBA tuple for each ray, in contrary, the ultrasound simulation needs a value for each sampling point during the simulation. This key difference leads to various modifications of the initial ray-casting approach, including different sample processing steps, texture layouts and shaders.

Initially, the starting points and direction vectors are computed for each element of the transducer. These depend on the characteristics of the transducer including the number of transducer elements, the Field-Of-View (curved transducers), the size of the transducer (linear transducers), the small radius from the transducer center to the transducer elements, and the long radius from the transducer center to the maximum penetration depth. Accordingly, the starting points and ending points for each sound ray are calculated on a default coordinate system. Subtracting a starting point and ending point yields the direction vector. Based on the orientation and position of the transducer, a transformation matrix is calculated which transforms the starting points and the direction vectors from the initial coordinate system to the CT coordinate system. Additionally, a transformation matrix is used to transform the points and vectors to the GPU volume coordinate system. This is necessary since the CT volume is stored as a 3D texture on the GPU. Valid texture coordinates for accessing the 3D volume are, thus, in the range of $[0..1]$ for the OpenGL implementation. The initial starting points and direction vectors for a given transducer geometry are calculated once and uploaded in the GPU as 1D RGB textures. If the orientation of the transducer changes during the simulation the points and vectors are multiplied with the transformation matrix, which is performed on the GPU to increase the simulation performance.

The sampling is performed for each scanline in a fragment shader and the ray position and direction are read from the textures for a given scanline. Therefore, the sample position $\vec{s_n}$ for a given scanline n is calculated as $\vec{s_n} = \vec{p_n} + u \cdot \vec{d_n}$, where u is the sampling depth along the vector. The sampling depth u is equal to the ratio of the length from the transducer element to the desired depth by the number of samples to be taken. At the point $\vec{s_n}$ the voxel



(a)Scanline scheme (b)Scanline reflection intensities

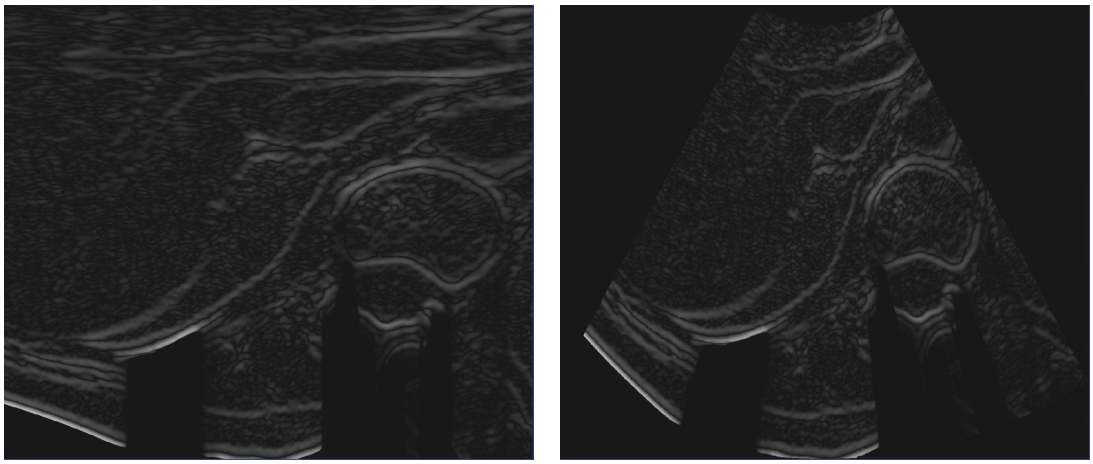
Figure 2.4: Image (a) shows the scanline setup that is implemented for the simulation. The white region in the middle represents the currently rendered line primitive, which calls the simulation for the given depth samples of the scanlines. Image (b) shows the simulation results for the reflection image. Only 15 samples are taken for the image, which explains the pixelated appearance.

intensity of the CT volume is accessed from the 3D textures, where trilinear interpolation is also enabled on the graphics hardware. Therefore, for each scanline, a number of samples is accessed, as schematically demonstrated in figure 2.2 (b). Increasing the number of samples results in a better resolution and a higher image quality. However, each additional sample requires additional processing since the simulation is performed for each sample. Furthermore, increasing the number of samples will not necessarily result in better image quality. The number of samples that provide better quality is bounded by the actual resolution of the CT dataset. Increasing the samples over the available CT resolution results in oversampling, which is unnecessary and might even introduce errors in the simulation.

In this section, the ray sampling process was described for transducers with 1D element arrays. The same process is also utilized to model 2D array transducers. This is done by simply specifying sound rays for all elements of the 2D element matrix, demonstrated in figure 2.3. Afterwards, the same sampling process is used since each scanline, thus, each ray, is evaluated independently of the others.

2.5.2 Scanline Simulation

During the sampling process the sampled values are used to generate the reflection, the transmission and the echogeneity image. The sampling is performed using the previously calculated starting point texture and direction texture for the sound beams. The echogeneity image is simply generated by mapping each of the sampled CT intensities to acoustic impedance values based on a predefined function presented in Wein et al. [54]. This function is implemented as a 1D Look-Up-Texture (LUT) on the GPU , while the mapping is performed in a GLSL shader program. The resulting image is saved in a 2D 1-channel texture



(a)Scanline scheme (b)Part of a reflection image

Figure 2.5: Image (a) shows the reflection image for 128 scanlines with 128 samples. The shadowing effects result from utilizing the calculated transmission for attenuating the initial sound energy. Image (b) shows the scan conversion of the reflection image.

using the OpenGL Framebuffer Object (FBO) Extension [3]. The reflection and transmission images are calculated based on the simulation model previously presented. Looking at the model it becomes imminent that for each sample we need the transmitted energy from the previous sample. However, the ray-casting algorithm, returns only a RGBA tuple for each ray, with previous values along the ray being unavailable. The problem was also mentioned in Vidal et al. [53]. Next, an efficient solution to this problem will be presented.

The goal of the method is to calculate the transmitted intensity for a sample $k-1$ and make it available for the calculation of the next sample k, as described in the model. This means that we would like to calculate the transmission for all scanlines only at a given sample depth. The scanlines can be saved as columns in a 2D 1-channel texture. Thus, the values along the columns correspond to the simulated intensities. As an example, for a 128 element transducer for which 128 samples are calculated a 128×128 2D texture would be utilized. With this texture layout in mind we now want to call the GLSL shader that samples the CT volume and calculates the transmission, based on the proposed model, for a specific sample depth. This is done by rendering a horizontal line primitive which only covers the pixels which correspond to a given sample depth in the described texture, schematically illustrated in figure 2.4 (a). An example of the resulting reflections for a 128-element transducer is presented in figure 2.4 (b). Thus, the transmission is calculated only for the k-th samples along the rays, where the result is rendered at the fragments covered by the line. In the same manner, the transmission at the k-th sample is calculated by using the transmission of the previous sample $k-1$, which is stored one row above the current transmission in the transmission texture. Note that the initial transmission at sample $k=0$ is set to 1.0. This is simply done by filling the first row of the transmission texture with this intensity value. The described process continues in this recursive manner until all samples for all scanlines are evaluated. Two textures are used to calculate the transmission. One is used to write in the current result, and the other one to read the previous values. This represents a ping-pong

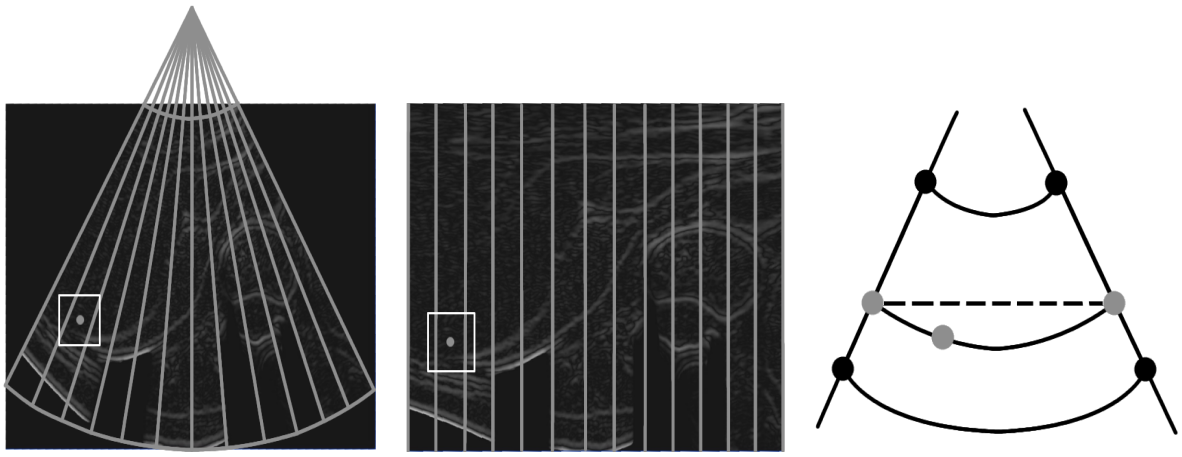
scheme [46], which is used to avoid simultaneous reads and writes on the same texture, because that would cause errors due to graphics hardware limitations.

At this point note that actually the reflection, the transmission and the echogeneity image are calculated by this line rendering scheme at the same time. This is done by using a single shader for the calculations and the multiple render target feature of the FBO to redirect the shader outputs to the appropriate textures. This also improves the performance of the method by reducing the overall overhead (less texture binding operations, shader calls etc.) The transmission is applied on the reflection and echogeneity intensities by multiplying those with the transmission coefficient. This results in the characteristic shadowing effects in ultrasound, where the transmission is also used to calculate the reflection. At the end of the simulation, two textures (because of the ping-pong scheme) with transmission intensities are available but not further used, therefore, deleted. The reflection image is calculated based on the model proposed using the transmission from the previous sample as the incoming sound beam intensity for a current sample. The reflection and echogeneity images are saved in a single 2D texture since only write operations will be performed on these textures. Image 2.5 (a) shows the reflection image simulated for a 128-element transducer taking 128 samples along the scanlines.

Attention has to be paid at the following issues for the implementation. First, the line primitive must be rendered exactly at the texture row that contains the samples to be evaluated. Otherwise, it will interfere with the other samples stored in the texture. Second, note that texel intensities are stored in the center of each pixel in the texture, based on the texture coordinate system. Therefore, when accessing the texture values in the shader, appropriate offsets must be used to access the values exactly at the pixel center. Initially, the render target textures for the reflection, echogeneity and transmission images are set to nearest filtering. For the scan conversion the filtering is set to linear in order to interpolated between scanlines. Thus, the correct offsets must be used to access the desired texture values, taking the interpolation into account.

2.5.3 Scan conversion

In this stage the reflection and echogeneity image are combined to form the final image and, if required, the image is scan converted. The images are combined using a coefficient for each image. Thus, a pixel intensity $I(x, y)$ in the final image is defined as $I(x, y) = \alpha \cdot R(x, y) + \beta \cdot E(x, y) + \gamma$, where R is the reflection image, E is the echogeneity image, $\alpha, \beta, \gamma \in [0..1]$ are the coefficients and (x, y) the pixel position in the 2D image. The γ coefficient is used to increase the brightness of the image. The coefficients can manually be adjusted, where for multimodal registration the coefficients are automatically adjusted during the registration process. Scan conversion refers to the process of mapping the individual scanlines into a scan converted image representation which is based on the transducer design. A scan converted image of the previously presented reflection image is showed in image 2.5 (b). For linear transducers a scan conversion is unnecessary, since the simulated scanlines already correspond to the transducer design. Thus, the images are only combined and the resulting image resized to keep the physical aspect ratio. On the other hand, a scan conversion is necessary for curved transducers, which maps the scanlines into the characteristic fan shaped image. The scan conversion is realized with a backward warping scheme.



(a)Scan converted scanlines (b)Initial scanlines (c) Scanline interpolation

Figure 2.6: Image (a) shows the scan converted scanlines (illustrated with grey lines) in the transducer Polar coordinate system. The intensity of a pixel (marked with a white square) is calculated from the initial scanlines shown in image (b). The pixel intensity has to be interpolated if the pixel is not lying on a sample of the initial scanlines. This is done by interpolating between the samples of the two scanlines that lie on each side of the pixel, as demonstrated in image (c). First, the intensity is linearly interpolated along the scanlines for a given sample depth. Second, the resulting intensities are linearly interpolated using the arc length between the scanlines. Using the direct length between the interpolated samples, marked with the dotted line, would result in erroneous interpolation results. Note that this is a schematic illustration for which the size and number of the scanlines are exaggerated for demonstration purposes.

Subsequently, a Polar coordinate system is defined for the transducer geometry, taking into account the transducers Field-of-View, the number of transducer elements, the small and the long radius from the transducer center to the elements and the maximum sample depth respectively. Afterwards, for each pixel in the Polar coordinate system that belongs to the scan converted image, the corresponding intensity from the simulated scanline is used. If the pixel lies between two scanlines then an interpolation between the two scanlines is performed, taking into account the arc length between the scanlines. A schematic illustration of the scan conversion process is shown in figure 2.6 (a-c). The combination of the reflection and echogeneity images is performed in a shader for both linear and curved transducer geometries. Moreover, for curved transducers the reflection and echogeneity image are scan converted and combined in the same shader to yield the final image and improve the performance by avoiding separate render passes for scan conversion and image combination.

2.5.4 Simulating multiple images

The proposed method can be extended to simulate multiple images at once by applying slight changes to the framework. The sampling process and scanline simulation remain generally the same, since scanlines are interpreted independently in the framework until

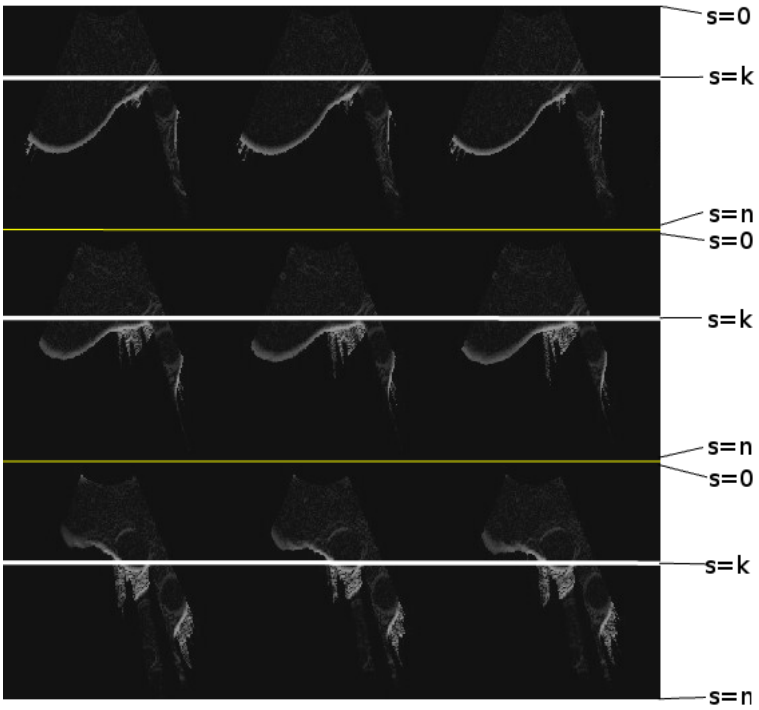


Figure 2.7: The image demonstrates the simulation of multiple images. The bright white lines represent the line primitives drawn to evaluate the k-th sample in each image respectively. The first sample and the last sample are stored horizontally at the top and the bottom of the individual images.

they reach the scan conversion stage.

Simulating multiple images suggests that the transducer position and orientation will be different for most of the images. Thus, initially, the starting positions and direction vectors for the rays for the different transducer orientations and positions are calculated and stored in 2D RBG textures.

The previous texture layout suggested that for each simulated image one texture is used. For multiple images a single texture is used to store multiple images, as demonstrated in figure 2.7. The number of simulated images that fit into a single texture is limited by the number of elements used and the number of samples taken. Current graphics hardware supports a maximum texture size of 8192^2, thus, for example, if 128 elements are used then $8192/128 = 64$ images can be placed horizontally in the texture. If more than 64 images are simulated then the other images are placed below the first ones as shown in figure 2.7. The same scheme applies for the subsequent rows. To continue the example, if 256 samples are taken along the scanlines then $8192/256 = 32$ images can be placed in a vertical tile of this texture. For this texture layout example, a maximum of $64 \cdot 32 = 2048$ images could be stored and simulated at the same time in one texture. To call the simulation at the same time for all images, a horizontal line primitives is drawn for the k-th sample in all images, as shown in figure 2.7. The simulation shader is called for the k-th samples for all scanlines and the

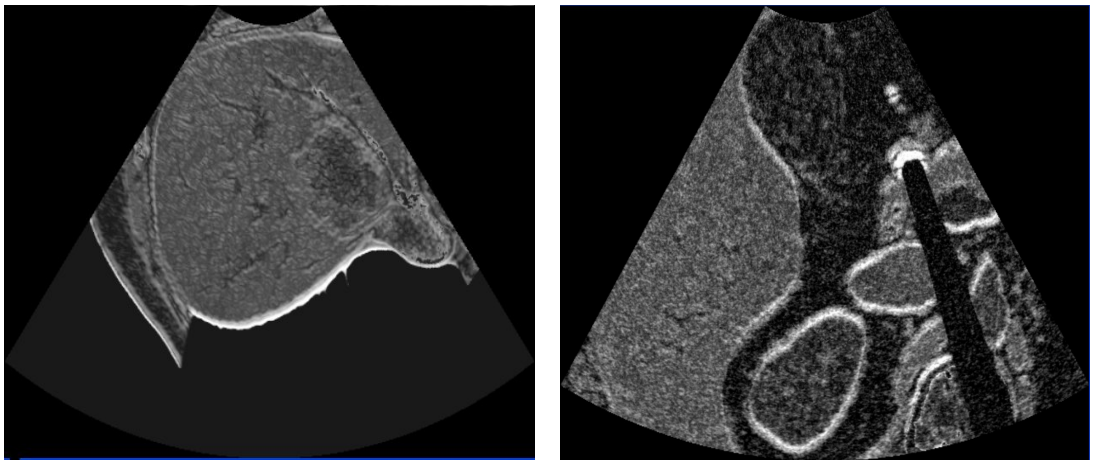
previously calculated 2D textures with the ray starting positions and directions are used for the sampling process.

The scan conversion for the multiple image simulation, is performed similar to the scan conversion of a single image. The difference is that the target scan conversion texture will contain multiple images, relative to the previously calculated reflection and echogeneity textures. Therefore, the scan conversion shader is called separately for each image by drawing a quadrilateral covering only one image in the target scan conversion texture. Using the appropriate offsets the scan conversion shader will only access the scanlines of the subimage covered by the quadrilateral from the reflection and echogeneity textures, in order to convert and combine them into the final subimage of the target texture.

Simulating multiple images at the same time introduces the possibility of simulating entire 3D ultrasound volumes. A 2D array transducer can be modeled by defining an image for each array row. For example, for a 128×128 element matrix, 128 images with 128 scanlines would be simulated. The starting positions and direction vectors for the sound beams are defined, as previously, in 2D textures, taking into account the geometry of the 2D array transducer. After the images have been simulated they are combined into a volume in a 3D scan conversion stage. This is done by rendering scan converted 2D image planes into a 3D texture volume, using the render to volume feature of the OpenGL FBO extension. For each layer that is rendered, the corresponding voxels are scan converted using a 3D backward warping scheme. The difference with the 2D case is that if no direct correspondence exists between a voxel and a scanline sample, then the voxel intensity is interpolated from a four scanlines neighborhood. Again, the arc lengths among the scanlines are taken into account for the interpolation.

2.6 Additional Effects

To improve the realism of the simulation the use of Perlin noise was already suggested in Wein et al. [54]. Perlin noise, named after its inventor Ken Perlin, is a pseudo-random noise in $\mathbb{R}^3$ that is visually isotropic and band limited. The noise function is calculated for a point $\vec{p} \in \mathbb{R}^3$ and yields always the same result for this point. This is an important attribute that makes this noise suitable to mimic ultrasound speckle, since the same noise pattern is desired whenever the same anatomy is imaged. A guide for implementing the improved Perlin noise on the GPU is found in [41]. The noise can be calculated for the entire volume at the beginning of the simulation or on-the-fly during the simulation. The first approach is computationally more efficient since the noise function has to be computed only once for all voxels of the volume. However, additional GPU memory is required to store the volume, which, nowadays, does not pose a real problem given the GPU memory available on current graphic cards (about 1 GB). The second approach is computationally more demanding and significantly impacts the simulation performance, thus, it should be avoided if the required memory is available. Post processing the simulated images with a Gaussian smoothing filter increases the realism of the images by smoothing the reflections. However, the kernel size should be kept small enough to avoid extensive blurring of the image. The Gaussian filter can efficiently be implemented on the GPU [22] and is applied on the scanline reflection image. In Shams et al. [50] the authors suggest the use of a Hanning window to efficiently simulate the beam width effect. The effect is simulated by performing a convolution on the



(a) Simulated image using Wein et al. model

(b) Simulated image with additional effects

Figure 2.8: Image (a) shows the final simulated image using the original model described in Wein et al. [54].The anatomy displayed is from the liver with a tumor on the right side. Image (b) shows the final simulated image with additional effects (Perlin Noise, Hanning Window, Gaussian smoothing). The anatomy displayed is from the kidney region.

scanline image in the horizontal axis using a Hanning window. The effect of this convolution is that the values along the scanline are affected by the values of neighboring scanlines, which is, in principle, the same cause for the beam width effect in real ultrasound imaging.

2.7 Results

For the simulation CTA (Computed Tomography Angiography), volume datasets were used. The datasets include pathological anatomy (tumors) and are generally available in clinical practice. A simulated image using the Wein et al. model is presented in figure 2.8 (a). The same kind of images are used to register the real ultrasound with the simulated ultrasound. A simulated image using the previously described additional effects is presented in figure 2.8 (b). This kind of images are more suitable for training purposes. Additionally, the simulated beam width effect gives rise to strong reflectors as in real ultrasound. In the appendix in figure 1, a comparison is presented for CT, real and simulated ultrasound for the same anatomy.

The performance of the GPU implementation was evaluated on an Intel Xeon 3.2 GHz CPU with 2GB RAM and a GeForce 8800GTX 768MB graphics card. The performance measurements for different image batch sizes are presented in figure 2.9. The image batch sizes refers to the number of images simulated simultaneously in a single texture. A table with the exact measurement can be found in the appendix (2). The large difference in performance for higher batch sizes is due to a number of different factors related to the graphics hardware. First and foremost, simulating a single image at a time increases the overhead on the graphics hardware by increasing the number of texture binding operations, FBO calls and

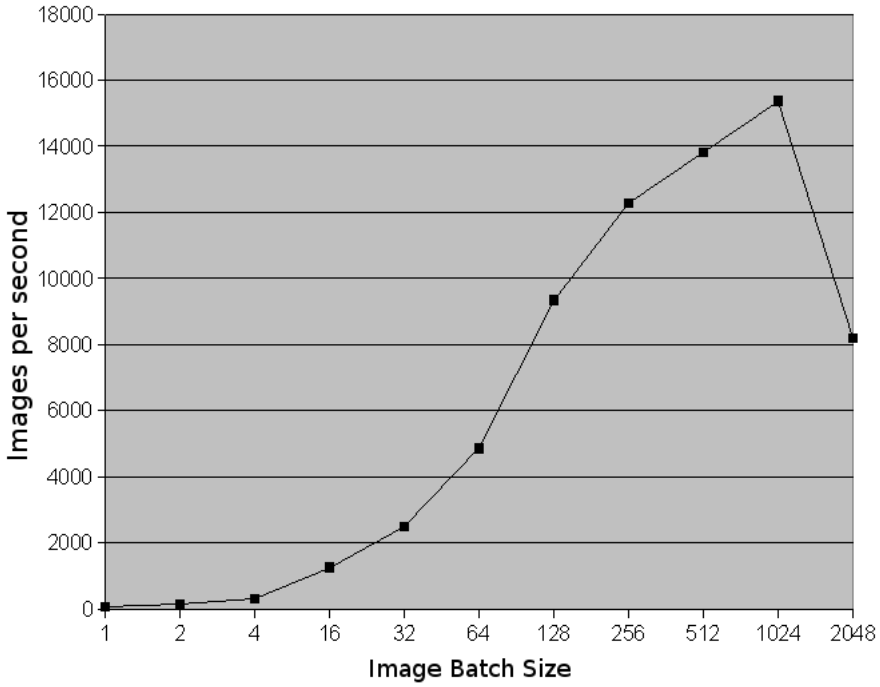


Figure 2.9: The performance graph shows the number of simulated images per second for different image batch sizes (number of images simulated simultaneously). For the simulated images were calculated 128 scanlines with 128 samples.

shader program calls. FBO binding operations are especially costly operations on graphics hardware, compared to other operations, and have a negative impact the overall performance. Therefore, by using a single texture for simulating multiple images the overhead is reduced and the image throughput increased. Second, simulating multiple images at once increases the thread throughput by sending more fragments to be processed at the numerous processing units on the graphics hardware. Therefore, the overall performance increases by reducing the idle time on the GPU cores, thus, hiding texture memory access latency. More information on this topic can be found in [39] and [32]. For an image batch size of 2048 the performance decreases dramatically. The reasons for this hardware related behavior is still investigated, where the continuous driver improvements for graphics hardware might solve this issue in the imminent future.

2.8 Discussion

In the results section, the high performance of the proposed method was demonstrated, exceeding the requirements of real-time 2D ultrasound simulation for a 30Hz refresh rate. The available performance allows the integration of additional effects (Speckle Noise, Gaussian smoothing, Hanning Window), preserving the real-time simulation.

Furthermore, the available performance allows the simulation of 3D ultrasound as men-

tioned in the section referred to 'Simulating Multiple Images'. Current 2D array transducers have element matrices of up to 128×128 elements. To simulate such a transducer geometry 128×128 scanlines would have to be simulated or equivalently 128 images with 128 scanlines (with 128 samples). Subsequently, a total of $30 \times 128 = 3840$ images per second would have to be simulated to perform real-time simulation of 3D ultrasound. The measurements show that this can easily be achieved since a total of 9344 could be simulated for such a transducer design on the current system. The remaining computational resources can, thus, be utilized to perform a DVR of the volume to visualize it during training. This has been demonstrated in a prototype using an existing DVR framework, where details can be found in [31].

The high performance of the method can also be utilized to accelerate the multimodal registration of CT and Ultrasound images. For this the images of an ultrasound sweep (usually 200-500 images) are registered to a CT volume. The registration process involves the simulation of ultrasound images for a subset (20-30) of real ultrasound images from the original sweep. Afterwards, the similarity between simulated and real images is evaluated. This process of simulating images and calculating the similarity is repeated multiple times during the optimization in the registration in order to find the correct transformation between the ultrasound sweep and the CT volume. The simulation of an image (128×96) takes 3.5ms on a 2.2 GHz mobile Core 2 Duo processor and is sufficient to provide an adequate registration time, when a subset of images is used. However, the registration time would be noticeably reduced using the proposed method. This would especially be true if all the images for a given sweep were simulated, since the proposed method offers enough performance to accomplish this task in real-time. Using all the images for the registration would improve it's accuracy. The CPU implementation uses a subset of images, which contribute most to the registration, for the sake of performance. However, using all the images would result in shifting the bottleneck from the simulation to the evaluation of the similarity measure. Therefore, in order to perform a fast registration process, the calculation of the similarity measure would also need to be ported on the GPU.

The implementation of the proposed method was realized using C++, OpenGL and GLSL. Recently graphic card vendors have introduced C-like programming languages for their hardware. These include CUDA (Compute Unified Device Architecture) for Nvidia cards and CTM (Close To Metal) for AMD/ATI cards. The languages make programming GPGPU applications a lot easier and allow scatter reads and writes from and to GPU memory. This was previously the limiting factor of the GPU API that needed to be circumvented with the line rendering approach. Thus, one would guess that using CUDA or CTM would be the best way to implement the GPU ray-based simulation. However, there are some fact to be considered. First, using one of the languages restricts the approach to a single GPU vendor and makes it platform dependent. This is not the case for OpenGL and GLSL which is supported by both major GPU vendors. Second, CUDA and CTM are incompatible with each other, thus, porting the simulation from one hardware to another is time consuming and might introduce unexpected errors. Therefore, the benefits of programmability come at the cost of platform dependency.

2.9 Conclusion

Conclusively, this chapter presented a new method for ray-based ultrasound simulation using graphics hardware. The method is suitable for both registration and training applications, where the supreme performance guarantees real-time simulation. Future work will involve modifications of the noise function in order to make the speckle effect more realistic. Furthermore, additional effects like the slice width effect, can be included by using a Hanning window for multiple scanlines parallel to the simulated image plane. However, improving the ray-based model will most like result in approximations of most of the characteristic ultrasound effects. Therefore, the real focus lies in exploring the possibilities of wave-based methods in light of the continuous graphics hardware innovations and their increasing computational capabilities.

Chapter 3

GPU Wave-based Ultrasound Simulation

In the previous chapter it was mentioned that ray-based simulation might be beneficial in terms of performance, but does not model a variety of wave phenomena that contribute largely to the final echo response. In light of the continuous advantages in graphics hardware, especially in regard to computational capabilities, real-time (30Hz) wave-based ultrasound simulation seems to be at reach in the imminent future, at least for 2D image simulation. This chapter focuses on investigating wave-based sound simulation approaches which are suitable for GPU implementation and could yield real-time ultrasound simulation results. Initially, an overview of the investigated methods will be provided, where two methods will be explained in more detail, namely the Digital Waveguide Mesh (DW-Mesh) and the Finite-Difference Time-Domain (FDTD) method for solving the full nonlinear wave equation, known as the Westervelt equation. The related work will be presented for each method together with the initial GPU implementations and first results.

This chapter is a proof of concept that shows the possibility of using wave-based simulation models for real-time ultrasound simulation. During the course of this chapter it will become clear that DW-Meshes are currently unsuitable for ultrasound simulation as they introduce a high number of errors. On the other hand, the FDTD scheme solving the Westervelt equation seems, currently, to be the best candidate for real-time wave-based ultrasound simulation on the GPU.

3.1 Overview of investigated approaches

One of the most referenced ultrasound simulations is the Field II program introduced by Jensen [26]. The publication did not include details on the simulation itself, but it was rather an introduction of the program. The program is freely available as a precompiled package that can be executed using Matlab. Field II has been in use now for more than a decade for ultrasound research and transducer development, providing realistic simulation of the signal response for different transducer types and beam forming schemes. Furthermore, the signal response can be post-processed to create realistic ultrasound images. In a later publication [25] the method was accelerated by performing the beam forming after the spatial

impulse response has been received for each point in the field. Thus, the received response from each element is calculated and then the responses are subsequently focused. The additional advantage of this approach is that the image can be focused after the field simulation. Therefore, the same simulation data can be used to test different focusing strategies, without the time consuming re-simulation of the element responses. A method for generating artificial volumes for the Field II simulation is presented in [27]. The main idea is to define scatters as randomly distributed scatter points in the volume using a Gaussian distribution as amplitude. The relative amplitude among different scatterers is determined by a scatterer map of the structures to be scanned, generated from CT or MRI images. The blow flow can also be simulated with this method by modeling the blood cells, causing the Doppler frequency shift, as point scatterers. The flow is simulated based on either a parametric flow model or through finite element modeling. Thus, the scatterers are propagated using the flow model during the signal response evaluation in order the create the detectable frequency shift. The simulation of non-linear ultrasound fields based on the Field II program was introduced in [24]. For this, the Westervelt equation was modified, more specifically the attenuation term, and introduced into the Field II framework using the Operator Splitting Method for the calculations. The Operator Splinting Method refers to solving the different effects (diffraction,attenuation,nonlinearity) separately by numerical integration over a small spatial region, and then combine them to form the final simulation response. The simulation has been compared to measurements from a hydrophone in a water tank using a convex array probe. The results indicate an accurate predication of the pulse shape, with slight amplitude deviations. Additionally, predictions for the rise of harmonics with depth are close to real measurement for second and third harmonics. The current Field II program is still used by a large number of research groups and provides the most extensive realistic ultrasound simulation framework at this time, even suited for the development of new ultrasound systems and transducers. However, the main drawback of Field II are the immense computational requirements which result in processing times in the range of hours even when using clusters of PC workstations [23], [50]. Therefore, this approach seems far from real-time capable in the sense of allowing an interactive ultrasound simulation with a refresh rate of 30Hz.

The other two approaches investigated are DW-Meshes and FDTD schemes for solving the Westervelt equation. DW-Meshes were introduced as an alternative to FDTD schemes as they have lower processing and memory requirements. They have mainly been used for the simulation of room acoustics and musical instruments, where an initial GPU implementation for room acoustics simulation [44] showed promising results in regard to performance. On the other hand, FDTD schemes are known to electrodynamics for decades and have, thus, been studied extensively for solving partial differential equations (PDE)s. In recent publications FDTD schemes for solving the Westervelt equation demonstrated good predictions for ultrasound simulations, using measurements from water tanks as comparison. Furthermore, FDTD schemes are highly suitable for GPU implementations as they involve a large amount of parallel computations. The efficiency of GPU FDTD implementations was also demonstrated in Baron et al. [5] and Adams et al. [4].

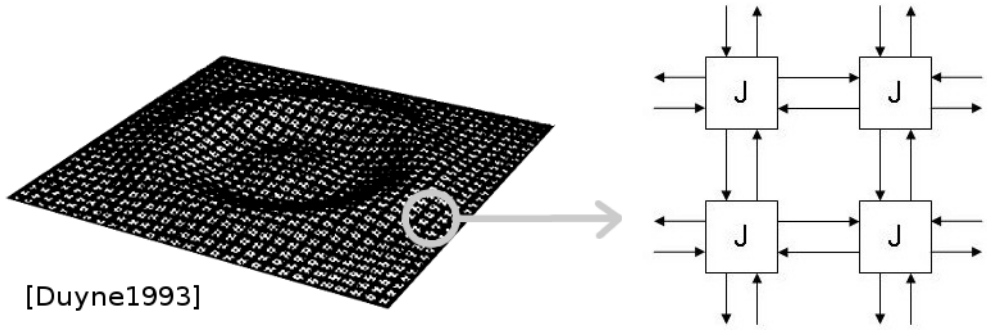


Figure 3.1: The image on the left shows a disturbance in the DW-Mesh. The image on the right shows some of the scattering junctions used for the mesh.

3.2 Digital Waveguide Meshes

This section will, initially, present the theory of DW-Meshes. Afterwards, the GPU implementation will be discussed and slight improvements are suggested to make the method more efficient. The result will briefly be discussed, whereas emphasis will be given to the reasons which make the approach unsuitable for ultrasound simulation.

3.2.1 Theory

The 2D digital waveguide mesh was introduced by Van Duyne and Smith [10] for the numerical calculation of the 2D lossless wave propagation. Additionally, the method was extended to 3D using a tetrahedral mesh structure in Van Duyne and Smith [11]. The authors proved that their formulation was equivalent to the second order difference equation of the wave equation for the lossless case. The method introduces the use of so called 'scattering junctions', which scatter the incoming wave components to their neighboring junctions. Figure 3.1 shows the wave propagation in such a mesh and how the scattering junctions are connected to each other. The wave amplitude is numerically evaluated at a discretized grid of the original domain, where the grid points correspond to the scattering junctions. Furthermore, the wave is evaluated at specific, usually equally long, time intervals (timesteps). The wave amplitude at a junction p_c for the timestep t is calculated as:

$$p_c(t) = \frac{1}{N} \sum_{2N}^{l=1} p_l(t-1) - p_c(t-2) \tag{3.1}$$

where p_c is the wave amplitude at the scattering junction of interest, p_l is the wave amplitude of a neighboring/connected scattering note, N is the dimension (1,2 or 3) and t is the timestep. Accordingly, the wave amplitude at a given scattering junction for the current timestep t can be evaluated from the wave amplitudes at the direct neighboring junctions of the previous timestep $t-1$ and the amplitude at the given junction of the timestep $t-2$. The simplicity of this formulation is the main advantage of this method. Thus, to calculate the wave amplitude at one scattering junction (grid point) for the 2D case, only four additions,

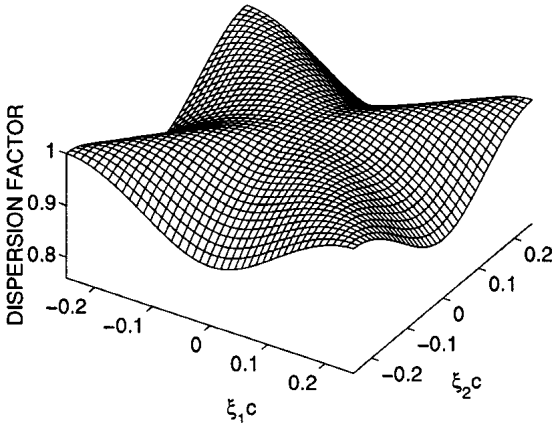


Figure 3.2: 2D Dispersion Error. The ratio of ξ_1 and ξ_2 (with the sign) determines the propagation direction, where the distance from the center is directly proportional to the frequency. The value 1 represents the ideal speed of the propagation that is achieved on the diagonal directions. Image from [48]

one division and one substation are necessary. Furthermore, the method only needs the values of two previous timesteps in order to calculate the current timestep, independently of the dimension. Later on the differences will be discussed between the DW-Mesh and the FDTD scheme considering the computational demands and the memory requirements.

3.2.2 Dispersion Error

The equation 3.1 which was presented previously, neither contains a term for the propagation speed of the waves nor for the frequency. The frequency is implicitly defined by the number of disturbances applied on the mesh for a given time interval. Thus, sound sources are modeled by disturbing the mesh through explicitly setting the wave amplitude at given scattering junctions. This also means that the initial condition for the DW-Mesh is a zero wave amplitude for all junctions, except for junctions where disturbances are applied. The propagation speed is implicitly defined by the formulation of the DW-Meshes. However, the implicitly defined propagation speed is different for diagonal propagations and horizontal/vertical propagations in the grid. This difference in propagation speed is known as dispersion error and additionally depends on the frequency, as illustrated in figure 3.2. More informations on the propagation speed in DW-Meshes can be found in [10] and [11].

In order to handle the dispersion error an interpolated waveguide was introduced by Savioja and Välimäki in [47], which demonstrated improved dispersion characteristics compared to the initially proposed DW-Mesh. In their follow up publication [48] the interpolation schemes was further refined and the dispersion further reduced through frequency-warping techniques. Nevertheless, the dispersion error was reduced significantly but not completely.

3.2.3 Boundary Conditions

The primary goal of acoustics simulation is the prediction of echo responses from reflectors in the simulation domain. A reflection occurs in a DW-Mesh whenever the wave disturbance reaches a junction marked as a boundary. As an example, the simulation of the echo response from a concert hall would require the modeling of the concert hall in the DW-Mesh. Based on the required accuracy a number of scattering junctions would be used to propagate the wave in the simulated concert hall. The walls, ceiling, seats, stage and etc. would be modeled by specifying boundary junctions at their position in the simulation grid. Therefore, a sound wave would travel through the modeled concert hall, reflect on the specified junctions and the response would be observed at a given junction in the DW-Mesh. Specifying the reflectors is not the only issue, but specifying the intensity of the reflection based on the material of the reflector is also important. To handle this issue, a new type of boundary junction was introduced by Murphy and Mullen [36]. Furthermore, the evaluation of the wave amplitude at the boundary junction was reformulated, including a reflection coefficient. The coefficient can, thus, easily be used to specify the amplitude of the reflection at the boundary, allowing the modeling of differently reflecting materials. Thus, the wave amplitude at a boundary junction can be calculated as:

$$P_c(t) = (1 + r) \cdot p_l(t - 1) - r \cdot p_c(t - 2) \tag{3.2}$$

where r is the reflection coefficient. Another important goal of the proposed boundary scheme was to provide anechoic boundaries for the grid edges.

One property of DW-Meshes is that waves are reflected at the edges of the DW-Mesh. This poses quite a problem since waves reflected at the mesh edges travel back into the mesh and thus, interfere with other waves. To clarify the problem lets assume the following example. A simulation is performed to get the echo response from an outdoor music stage. The stage and its imminent surroundings are modeled in the same way as the previously described concert hall. However, since it is an outdoor stage no walls are present around it. During the simulation an echo from the stage would be reflected at the DW-Mesh edge and would interfere with other sounds in the mesh. This would be equivalent to an invisible wall surrounding the stage, which would definitely introduce unexpected echoes. Therefore, the goal is to introduce boundaries that simulate a wave propagation into infinity, so that waves reaching the DW-Mesh edges do not interfere anymore with the simulation domain.

One would initially assume that setting the reflection coefficient of the previous amplitude formulation to zero would result in anechoic boundaries. However, this is not the case and waves are still reflected back into the mesh. The same problem exists for the FDTD scheme, but there are solutions which define Absorbing Boundary Conditions (ABC)s. As the name implies, these boundaries absorb the incoming wave in order to prevent it from traveling back into the grid. Inspired by this approach Murphy and Mullen [36] introduced boundary junctions based on Taylor Series expansions. Thus, a series of neighboring junctions is used to diminish the reflection at the target boundary. Usually, the higher the order of the series the better the accuracy is. However, after testing the new formulation the first order approximation would yield the same results as higher order approximations. Nevertheless, the new formulation proves to be a better approximation of an ABC for DW-meshes reducing the reflections at the grid edges to a great extend.

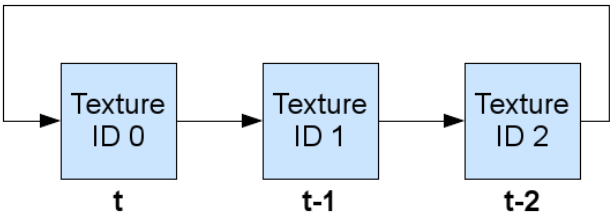


Figure 3.3: DW-Mesh Texture Queue. The queue can be shifted to the right and thus, the last element would become the first.

A joined effort to improve further the boundary conditions for DW-Meshes has been published by Kelloniemi, Murphy, Savioja and Välimäki in [29]. Taking into account the proposed mesh for decreasing the dispersion error [48] and the previous mentioned Taylor Series expansion for boundaries resulted in improved boundary conditions. The new formulation decreased considerably errors from boundaries and could model ABCs with less reflections. Nevertheless, the boundaries still exhibit significant errors especially for low reflection coefficients and frequencies. The presence of these errors is the strongest point against the use of DW-Meshes for ultrasound simulation. The exact reason for this will become clear during the next sections where the implementation and the initial results will be discussed.

3.2.4 GPU Implementation

A GPU implementation of the DW-Mesh for 3D room acoustics simulation is presented in Röber et al. [44]. The method was reimplemented using C++, OpenGL [40] and GLSL [46], where it was adapted for 2D simulation in order to validate first results and then proceed with the 3D simulation. For the GPU implementation the DW-Mesh is modeled as a texture where each texel corresponds to one scattering junction. In the initial proposal three 3D 3-channel textures were used, however, this was rather a waste of GPU memory as some of the channels contained redundant data. Therefore, in this implementation four 2D 1-channel textures were used, where equivalent for the 3D case four 3D 1-channel texture would be used. One texture is used to store the wave amplitudes of the scattering junctions to be evaluated for the current timestep t. Two textures are used to store the previous timesteps $t-1$, $t-2$ and one texture is used to encoded the position of the boundaries. For the simulation, the texture for timestep t is bounded to the FBO as a render target and a quad filling the entire texture is drawn. When the quad is drawn the simulation program is called for each pixel/texel that the quad covers, in this case, the entire texture. The simulation program is implemented in a fragment shader program and evaluates the wave amplitude (eq. 3.1) at each texel, accessing the textures of the previous timesteps in the shader.

The textures for the DW-Mesh are processed in a queue structure, as illustrated in figure 3.3. Each element of the queue contains a texture id, where at the beginning of a timestep simulation the texture id in the first element (on the left of the queue) is used to bind the corresponding texture as a render target to the FBO. The texture ids in the other two elements are subsequently used to bind the textures containing the results of the previous timesteps. After calculating the current timestep t, we might want to continue the simulation and evaluate the next timestep $t+1$. For this, the result of the current timestep t, saved in the corre-

sponding texture, becomes the timestep $t-1$ for the next simulation run $t+1$. Accordingly, the texture containing the results of timestep $t-1$ and $t-2$ would become $t-2$ and $t-3$ respectively. However, the timestep $t-3$ is not required for the simulation. Therefore, the texture containing timestep $t-3$ can be used to write in the new results for the timestep $t+1$. This behavior is achieved by shifting the texture queue one element to the right, as illustrated in figure 3.3. Thus, only the texture ids are changed for each simulation run, reducing the overall overhead and increasing the performance. For most simulation schemes (including ultrasound simulation) we are interested in the final echo response rather than the intermediate simulation results. However, if the intermediate results are required, then readbacks from the textures to host memory / host RAM can be performed to save the subsequent simulation results, at a high performance cost.

3.2.5 Results

Different simulation runs are presented in the appendix in figure 3. The performance of the GPU implementation was evaluated on an Intel Xeon 3.2 GHz CPU with 2GB RAM and a GeForce 8800GTX 768MB graphics card. For all simulations the mesh size (number of horizontal and vertical scattering junctions) was set to 512×512, where our implementation could calculate an average of 1130 timesteps per second. This performance allows the real-time visual observation of the wavefront propagation during the simulation by rendering the texture of the simulated timestep t after each timestep evaluation. Characteristic wave effects like constructive and destructive interference, diffraction and (diffuse) reflection are thus, observed during the simulation as demonstrated in the images of figure 3. The first two simulation runs (images (a-f)) used similar boundaries as described in [44].

The next step was to evaluate the use of this implementation for ultrasound simulation. For this purpose a slice from a CT phantom dataset (figure 4 (a)) was used to model the boundaries in the mesh. The slice is directly placed in the boundary texture in order to be accessed during the simulation. Intensities which correspond to bone structures in the CT image are thus classified as boundaries in the simulation shader and handled appropriately by using the previous boundary formulation (eq. 3.2), with a reflection coefficient of 0.98 (approximated reflection coefficient for tissue-bone interfaces). However, the definition of these boundaries is the restricting factor that prevent DW-Mesh from being used in ultrasound simulation, with details being discussed in the next section.

3.2.6 DW-Meshed unsuitable Ultrasound Simulation

One of the most problematic aspects is the lack of a definition for sufficient Absorbing Boundary Conditions (ABC)s for this scheme. The fact that waves are reflected at the edges of the mesh introduces the need to define boundary conditions for the edges of the computational grid that are either completely absorbing, or emulate a wave propagation to infinity. Nevertheless, the boundary conditions should only allow insignificant amounts of wave energy to be returned back into the grid. One way to deal with this issue, and thus, obtain a signal response free from 'boundary noise', is to extent the computational mesh. Making the mesh large enough increases the time for reflections from mesh edges to reach the region of interest in the mesh. Subsequently, the reflections from the mesh edges do not

interfere with a signal if the time interval between transmission and reception of a signal is small enough. However, this approach introduces additional computational demands and memory requirements that should not be necessary. Apart from this, another goal of ultrasound simulation is the research of continuous ultrasound, which uses multiple pulses for the transmission. Additionally, different instrumentations and settings, like phased arrays or multiple zone focusing, increase the number of transmitted pulses and thus, increase the necessary time to complete the simulation. To perform such a simulation the reflections from the mesh edges would need to be delayed even further by extending immensely the size of the mesh, even if only a time interval of a few seconds would be required for transmission and reception. This would dramatically increase the necessary simulation time. One aspect that makes ultrasound simulation so interesting is to observe the distribution of acoustic energy inside the body during a prolonged sound exposure and thus, investigate the interaction between different pulses in human tissue. Therefore, limiting the simulation only to short exposures is restricting the possibilities of the simulation. Conclusively, the current lack of sufficient ABCs for DW-Meshes makes the approach unsuitable for continuous ultrasound simulation, where increasing the mesh size makes the approach unsuitable for real-time simulations.

Another crucial aspect are the errors introduced with boundary conditions. As stated in the theory subsection, the currently proposed schemes for boundary conditions are not error free and introduce significant errors for small reflection coefficients. The proposed improvements make the scheme definitely more sophisticated and introduce less errors than the initially proposed scheme 15 years ago, however, the presence of such error can not be neglected in the case of ultrasound simulation. For room acoustics and music instrument simulation the definition of the boundaries is rather simple, compared to defining boundaries for human tissue. Rooms can simply be modeled by defining boundaries for walls and furniture, whereas the outlines of music instruments are used as boundaries. Furthermore, the percentage of boundaries in the mesh is small, since most scattering junctions are defined as air for the propagation of sound. On the other hand, when it comes to modeling human tissue things become rather complicated. In this case, nearly the entire mesh would be filled with varying boundaries. The reason for this lies in the fact that human tissue is inhomogeneous and includes numerous reflecting interfaces. So all the interfaces must be classified and modeled with boundary junctions of varying reflection coefficients. A method that would accurately perform such a classification for entire CT and MRI datasets is unknown, whereas manual classification would also be inaccurate. This is due to the fact that even small pixel intensity variations in CT and MRI datasets indicate reflecting interfaces in the tissue, and would thus be unrealistic for manual segmentation. This further suggests that most of the pixels of a CT or MRI image would be modeled as boundary conditions in the DW-Mesh scheme. The same would be true for modeling speckle in terms of defining a procedural artificial speckle pattern by multiple boundary junctions very close to each other. The practical infeasibility of this boundary definition procedure is not the only factor which limits this approach. Additionally, the error introduced by boundary conditions would accumulate due to the high number of boundary junctions in the mesh and result in erroneous simulation results. Taking into account the dispersion error and the fact that reflection coefficients are most of the time small for interfaces inside human tissue (resulting in higher errors for boundaries), leads to the conclusion that DW-Meshes are unsuitable for ultrasound simulation.

3.2.7 Conclusion

This section presented the 2D Digital Waveguide Mesh for the purpose of ultrasound simulation. The simple formulation of the method and the resulting good performance of the GPU implementation are encouraging for it's use in acoustics simulation. However, as extensively discussed, the method is rather unsuitable for ultrasound simulation due to erroneous boundary conditions and the problem in defining the boundaries. Nevertheless, the method is a good introduction to the topic of numerically solving the wave equation for acoustics simulation, demonstrating strong similarities with FDTD schemes presented in the next section.

3.3 Finite-Difference Time-Domain Ultrasound Simulation

From a physics point of view sound is a wave phenomenon. Therefore, the most correct way to model sound propagation and interaction with matter is with a wave related mathematical formulation. One of the most wide spread methods for numerically solving the wave equation is the the Finite-Difference Time-Domain (FDTD) method. The FDTD method is a grid-based method which was initially introduced by Yee [56] in the field of computational electrodynamics. Initially, the method was used for numerically solving Maxwell's time-dependent equations which are fundamental in electromagnetism. Over the past decades the method has been extensively used for various wave related simulation purposes, with numerous FDTD publications appearing each year. Recent publications ([16], [20] ,[43], demonstrated the potential of the FDTD method for numerically solving the Westervelt Partial Differential Equation(PDE) for ultrasound simulation purposes. Comparing simulated signal predictions with measurements from water tanks showed coincidental results with slight deviations, encouraging the use of the Westervelt PDE for nonlinear ultrasound simulation. Additionally, the FDTD method is highly suitable for parallel computation, therefore, also preferable for GPU implementation ([4], [5]). In the remaining section the basics of the FDTD method will be presented, accompanied by examples. Furthermore, the evaluation of the Westervelt equation using the FDTD method will be discussed and the initial GPU implementation will be presented. Similarly to the DW-Mesh section, the results will be briefly discussed and emphasis will be given to the suitability of the approach for real-time ultrasound simulation.

3.3.1 FDTD Theory

A short description of the FDTD method would be as follows:

> The FDTD method substitutes the partial derivatives of PDEs with their equivalent finite differences approximations and evaluates the PDE for a finite number of grid points in the simulation domain at specific times.

More details on the FDTD method can be found in the book by Taflove and Hagness [52]. The simplest grid scheme is the rectilinear grid for which the grid points are equidistantly spaced. The rectilinear grid can be applied to 1D, 2D and 3D domains. Additionally, there are other grid types including hexagonal and polar grids which are utilized according to the given problem to be modeled. Throughout this section, a 2D rectilinear grid will be assumed. Finite differences for partial derivatives are derived from Taylor expansions. Thus, the more terms from the Taylor expansion are used, the more accurate the approximation of the partial derivative is. However, more terms result in more operations to be performed, resulting in a compromise between accuracy and performance. To clarify the concept an example of an FDTD solution for the 1D lossless wave equation 3.3 will be presented.

$$\frac{\partial^2 u(x,t)}{\partial x^2} = \frac{1}{c^2}\frac{\partial^2 u(x,t)}{\partial t^2} \tag{3.3}$$

where u is the wave amplitude, x is the spatial variable, t is the temporal variable and c is the propagation speed. To solve the equation numerically the partial derivative terms are

substituted with the approximate finite differences, which would be:

$$\frac{\partial^2 u(x,t)}{\partial x^2} \cong \frac{u(x+\Delta x,t) - 2u(x,t) + u(x-\Delta x,t)}{\Delta x^2}$$
$$\frac{\partial^2 u(x,t)}{\partial t^2} \cong \frac{u(x,t+\Delta t) - 2u(x,t) + u(x,t-\Delta t)}{\Delta t^2}$$

where Δx is the distance between the grid points and Δt is the time between the timesteps for the evaluation of the PDE. The term of interest in these equations is $u(x, t + \Delta t)$, since it will provide the wave amplitude at the timestep to be evaluated. Substituting the equations and solving for this term yields to:

$$u(x,t+\Delta t) = c^2 \frac{\Delta t^2}{\Delta x^2} \left(u(x+\Delta x,t) - 2u(x,t) + u(x-\Delta x,t)\right) + 2u(x,t) - u(x,t-\Delta t) \quad (3.4)$$

This equation states that the wave amplitude for a given grid point for the next timestep can be calculated from the known wave amplitudes in the grid at the previous two timesteps, using the imminent neighboring grid points. Performing this computation multiple times yields the wave propagation in the FDTD grid. An example of the FDTD method used for numerically solving the 2D lossless wave equation is presented in the appendix 1.

Initial Condition

Since the finite differences for second order derivatives involve values from previous timesteps, the initial conditions have to be provided for these timesteps at the beginning of the simulation. For example, a grid free of disturbances is specified by setting the initial condition to $u(x, 0) = u(x, 1) = 0$, which states that for the independent spatial variable x for the initial two timesteps $t = 0, t = 1$ the wave amplitude is zero, thus, the wave amplitude is zero for all grid points. Initial conditions are also used to introduce the wave disturbance into the grid, where the initial disturbance will evolve over the next timesteps.

Boundary Conditions

The previously presented DW-Mesh method demonstrated reflections at the mesh edges, which introduces various disadvantages and limitations in regard to its applicability for ultrasound simulation. The FDTD method shares the same property as waves are also reflected at the edge of the computational grid. However, various approaches have been introduced that successfully solved this issue. Contrary to the DW-Mesh, boundary conditions for the FDTD method are not used to cause reflections, but rather to handle the reflections at the boundaries of the computational grid. Thus, boundary conditions for the FDTD method specify the values of the solution that the PDE will take at the boundaries of the computational grid (domain). Boundary conditions include:

- Dirichlet boundary condition: Specifies the values the solution will take at the boundary.
- Neumann boundary condition: Specifies the value of the derivative of the solution at the boundary.

- Cauchy boundary condition: Combines the Dirichlet and Neumann boundary.
- Absorbing Boundary Condition (ABC): This type of boundary condition is considered to be the most important one, since a lot of problems are calculated for an unbound/open domain. This means that the wave must be absorbed at the boundaries of the domain, otherwise the resulting reflections would interfere with the simulation. A way to achieve this behavior would be to enlarge the computational domain, however, as previously discussed, this approach is computationally insufficient and improved approaches have been suggested. One fundamental approach for an ABC is the Perfectly Matched Layers (PML) introduced by Berenger [6]. The PML approach allows waves to travel from non-PML regions into PML regions without producing reflections, where the waves are decayed in the PML layers. The approach has successfully been used in computational electrodynamics and in general wave simulations to model ABCs, whereas an improved reformulation (UPML) has been proposed by Gedney [15]. A quick historical overview of PMLs can be found in Ziolkowski [59].

Stability

Stability is an important issue for the FDTD method. Different spatial Δx and temporal Δt discretization steps can lead to an unstable solution. For a given simulation the stability criteria should be defined, for example by performing a von Neumann stability analysis. Numerically unstable solutions lead to the evolution of initially small errors and compromise the entire simulation as the timesteps increase. For the current implementation the discretization steps were chosen empirically through trial-and-error, whereas a more sophisticated analysis is a subject to future work.

3.3.2 Related Work

Recent publications demonstrated the potential of using the Westervelt equation to simulate ultrasound together with nonlinear effects. As initially mentioned at the beginning of this chapter, in the publication of Jensen et al. [24] a slightly modified version of the Westervelt equation was used together with the Field II program to simulate nonlinear ultrasound responses. The comparison with real measurements from a water tank demonstrated slight deviations of the simulated prediction and the actual signal response. The original Westervelt equation is given as below:

$$\nabla^2 p - \frac{1}{c_0^2}\frac{\partial^2 p}{\partial t^2} + \frac{\delta}{c_0^4}\frac{\partial^3 p}{\partial t^3} + \frac{\beta}{\rho_0 c_0^4}\frac{\partial^2 p^2}{\partial t^2} = 0 \tag{3.5}$$

where the first two terms $\nabla^2 p - \frac{1}{c_0^2}\frac{\partial^2 p}{\partial t^2}$ are identical to the lossless wave equation. The third term $\frac{\delta}{c_0^4}\frac{\partial^3 p}{\partial t^3}$ is the loss term, mostly due to thermal conduction. The fourth term $\frac{\beta}{\rho_0 c_0^4}\frac{\partial^2 p^2}{\partial t^2}$ describes the nonlinearity. The last term is very important since it models the nonlinear behavior of ultrasound inside complex mediums (like human tissue). The equation coefficients are: p [Pa] Acoustic Pressure, c_0 [ms^{-1}] propagation speed, ρ_0 [kgm^{-3}] Ambient Density, δ [$m^2 s^{-1}$] Diffusivity of Sound, β Coefficient of nonlinearity, usually takes values in the range

of 6-10 for biological tissue [24]. In Yongchen et al. [57]is presented a collection of some of these parameters for human tissue.

In the publication of Hallaj and Cleveland [16] the Westervelt equation was used to evaluate the pressure field in a homogeneous thermoviscous fluid. Using the pressure field the bioheat equation was evaluated to determine the heating of tissue in the acoustic field. High intensity pulses generate harmonics due to nonlinear ultrasound effects, where higher harmonics enhance the heating of tissue. Thus, the Westervelt equation was chosen to simulate the nonlinear effects, with the simulation utilizing the FDTD method to numerically solve the Westervelt and the bioheat equation. The simulation results indicate a steep rise of temperature with increased acoustic pressure.

In a later publication by Huigssen et al. in [20] the Westervelt equation was compared with the Khokhlov-Zabolotskaya-Kuznetsov (KZK) equation. Again, the FDTD method was utilized, where the comparison included real measurements using an unfocused single element transducer in a water tank. The comparison shows a good agreement between the numerical prediction of the Westervelt equation and the real measurements, whereas errors were present more in the near field of the transducer. Furthermore, the accuracy of the Westervelt prediction outperformed the one for the KZK prediction. The simulations were performed on a desktop workstation and took about one hour for the nonlinear simulation, while only some minutes for the linear simulation. The possibility of using the method for 3D ultrasound simulation would only be limited by availability of computational and memory resources.

The good agreement between the prediction and measurement are encouraging for the use of the FDTD-Westervelt method for 2D ultrasound simulation. Additionally, the fact that the simulation was performed in a rather short time on a desktop workstation 6 years ago is rather encouraging, since it might now be realizable in real-time using current graphics hardware for accelerating the process. Therefore, in the remaining chapter an initial GPU implementation of the FDTD method for solving the Westervelt equation will be presented and the results will be discussed.

3.3.3 GPU Implementation

The FDTD method for solving the Westervelt equation will shortly be presented, before going into details of the GPU implementation. Similarly to the previous publication ([16],[20]), the following finite differences are used to substitute the partial derivatives of the equation. Forth-order central differences are used for the first two terms, second-order central differences for the third term and second-order backward differences for the forth term. Thus, the temporal partial derivatives were calculated using:

$$\frac{\partial^2 p}{\partial t^2} \cong \frac{p_{i,j}^{n+1} - 2p_{i,j}^{n} + p_{i,j}^{n-1}}{\Delta t^2} \tag{3.6}$$

$$\frac{\partial^3 p}{\partial t^3} \cong \frac{6p_{i,j}^{n} - 23p_{i,j}^{n-1} + 34p_{i,j}^{n-2} - 24p_{i,j}^{n-3} + 8p_{i,j}^{n+4} - p_{i,j}^{n-5}}{(2\Delta t)^2} \tag{3.7}$$

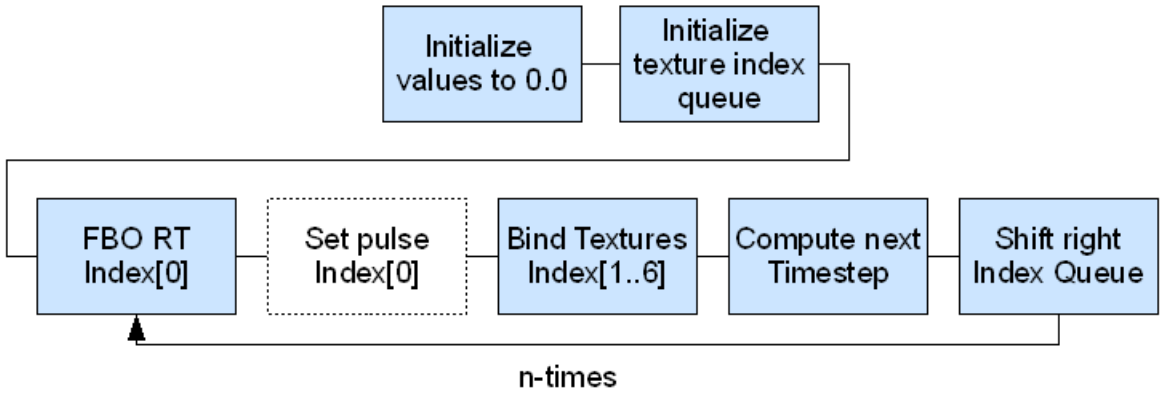


Figure 3.4: FDTD 2D Rectalinear Grid

where $p^n_{i,j}$ is the acoustic pressure at a grid point (i, j) for the timestep n. The spatial partial derivatives were calculated using:

$$\frac{\partial^2 p}{\partial x^2} \cong \frac{-p^n_{i+2,j} + 16p^n_{i+1,j} - 30p^n_{i,j} + 16p^n_{i-1,j} - p^n_{i-2,j}}{12\Delta x^2} \tag{3.8}$$

$$\frac{\partial^2 p}{\partial y^2} \cong \frac{-p^n_{i,j+2} + 16p^n_{i,j+1} - 30p^n_{i,j} + 16p^n_{i,j-1} - p^n_{i,j-2}}{12\Delta y^2} \tag{3.9}$$

The expansion of the forth term is given as below:

$$\frac{\partial^2 p^2}{\partial t^2} = 2 \cdot \left(\frac{\partial p}{\partial t}\right)^2 + 2 \cdot p \cdot \frac{\partial^2 p}{\partial^2 t}$$

To give an overview of the GPU implementation the different steps are presented in figure 3.4. Initially, the values of the textures used for the simulation are explicitly initialized to zero. Then, a texture queue is initialized, similar to the one used for the DW-Mesh GPU implementation. The element at the first position of the queue is, thus, used to store the result of a simulation run. New disturbances to the grid are applied before performing the simulation. The other textures in the texture queue contain the wave pressure for the grid points at the previous timesteps, whereas they are used for the simulation of the next timestep in a fragment shader program. After the simulation is finished, the queue is shifted to the right and if it is required the simulation is continued. Detail will follow on the implementation.

For a 2D FDTD grid 2D textures are utilized, where each texel of the textures corresponds to a single grid point storing the wave pressure at this grid point for a given timestep. A 3D FDTD grid simulation would, thus, require 3D textures. The textures are defined as single channel 32-bit floating point (32F_EXT) textures, since only a single float value is needed to save the pressure for a given grid point. Current graphics hardware do only support 32-bit floating point precision, which introduces numerical errors compared to CPU methods. The finite differences for the numerical solution of the Westervelt equation suggest that 6 timesteps are needed in order to calculate the next timestep. Therefore, 7 textures are needed to save the previous timesteps and the result of the simulation. The maximum grid size that

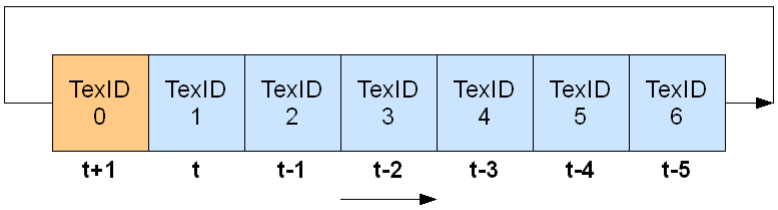


Figure 3.5: FDTD-Westervelt 2D Texture Queue

can be simulated on the GPU is limited by the maximum supported texture size on graphics hardware, which is currently 8192^2 equivalently about $6 \cdot 10^7$ grid points. After creating the textures, their id's are pushed onto the queue in random order, while an ascending order will mostly be the case for the sake of simplicity. Each element of the queue corresponds to a given timestep as shown in the figure 3.5.

Before starting the simulation the ids in the queue are used to bound the textures and set their texels values to 0.0. This also represents the initial condition of the simulation. Disturbances (sound pulses) are applied just before the simulation by setting the wave amplitude at specific grid points, by specifying values at the corresponding texels. For the simulation the output is saved in the texture that corresponds to the id stored in the first element (left) of the queue, whereas the other ids are used to bind the textures which contain the wave amplitude of the previous timesteps, as illustrated in figure 3.5. In a fragment shader program the numerical solution for the Westervelt equation is evaluated for the next timestep $t + 1$, using the bound textures to access the needed grid points. After the simulation is finished the texture queue is shifted to the right, thus, the texture containing the timestep $t - 5$ during the simulation will become the texture that will store the next timestep $t + 1$ at the next simulation run. Overwriting this texture is possible since only the previous 6 timesteps are needed for the simulation. Therefore, if the simulation is repeated n times in order to get the wave amplitude for the grid point at the timestep $t + n + 1$, then the previous timesteps $[t + n..t + n - 5]$ are still available, whereas timesteps bigger than $t + n - 5$ are lost due to the texture queue approach. Usually, only the final timestep is required to get the signal response for an ultrasound simulation. Nevertheless, if all timesteps are acquired then readbacks from GPU memory to the RAM are required to store them. However, readbacks are expensive operations and slow down the simulation considerably.

Previously, regarding the DW-Mesh, each reflector in the tissue needed to be specified as a boundary in the DW-Mesh. In contrast, the FDTD Westervelt simulation interacts with the modeled domain through adjusting the coefficients of the equation. Therefore, specifying a different propagation speed for varying regions of the domain results in reflections of the sound waves, since a change in propagation speed indicates a change in the density of the medium, which is considered a reflecting interface. This is implemented by using additional textures for each non-constant coefficient of the Westervelt equation (c_0,ρ_0,δ,β) which is varying for different grid points. Thus, each grid point can have different coefficients to model the medium it represents.

3.3.4 Results

For the simulation of ultrasound the coefficients for each grid point need to be specified in order to model the medium/tissue they represent. For the evaluation of the implementation only the propagation coefficient is varying, whereas the other coefficients are set to constant values. To specify different propagations speeds for an ultrasound simulation an image of the anatomy is needed. For this purpose, a CT phantom dataset was used, which is shown in the appendix in figure 4 (a). Five different intensities of the dataset were selected in order to label these intensities with different propagation speeds, as shown in figure 3.3.5. Therefore, during the simulation for each texel that is evaluated the corresponding propagation speed is accessed from the labeled texture and inserted into the numerical solution of the Westervelt equation to yield the new pressure. The change of speed gives rise to reflection and refraction effects in the simulation, whereas diffusion and interference effects are also present since a wave-based simulation model is used. The images (a-f) in figure 3.6 show a non focused wavefront propagating through the modeled anatomy. The images can be rendered during the simulation with insignificant performance penalty, since the simulation result for each timestep is already available as a texture to be displayed. An additional example using the same settings is presented in the appendix in figure 4 utilizing a focused wavefront. The labeling for the propagation speed was not included in order to make the changes of the wavefront more visible. The performance of the GPU implementation was evaluated on an Intel Xeon 3.2 GHz CPU with 2GB RAM and a GeForce 8800GTX 768MB graphics card and demonstrated an average of 1078 timestep calculations per second for a 512^2 grid, without using the nonlinearity term of the Westervelt equation for the sake of performance.

3.3.5 Discussion

The initial GPU implementation of the FTDTD method for the Westervelt equation showed promising results in regard to performance, computing a complete simulation (from pulse transmission to reception) in about 8 seconds. However, implementing PMLs on the GPU will also affect the performance, whereas additional computational effort is required to perform the signal processing of the received echo response in order to create a final 2D greyscale ultrasound image from the simulated data. Nevertheless, the current implementation was not optimized, thus, a faster simulation can be achieved. Furthermore, the continuous advantages in graphics hardware suggests more powerful devices in the imminent future. Comparing the used GeForce 8800GTX 768MB GPU with the new generation of Nvidia graphic cards the GeForce 285GTX 1GB, indicate that nearly twice as many GPU cores are available on the new device, with all other specifications being also improved (for more details refer to the Nvidia homepage www.nvidia.com). Therefore, the simulation time will most likely decrease due to imminent hardware advantages.

3.4 Conclusion

In this chapter two approaches were evaluated for the wave-based simulation of ultrasound, namely the Digital Waveguide Meshes and the FDTD method for solving the Westervelt equation. Considering the performance, the DW-Mesh needs only few operations for the

calculation of the wave amplitude and only three 2D 1-channel textures to store the timestep results. On the other hand, the numerical solution to the Westervelt equation needs to performed considerably more operations and requires seven 2D 1-channel textures to store the timestep results. Therefore, DW-Meshes are more efficient in terms of computational demand and memory requirements. However, during the process of this evaluation the DW-Meshes are demonstrated to be unsuitable for ultrasound simulation due to lack of sufficient Absorbing Boundary Conditions and the restriction of explicitly defining each reflector in the simulation domain as a boundary, which introduces errors to the simulation. The errors accumulate even more for ultrasound simulations, as a high number of interfaces are present in human tissue which need to be modeled as boundaries in the DW-Meshes.

On the other hand, the related work on solving the Westervelt equation with the FDTD method for ultrasound simulation indicated very promising results, as the simulated predictions coincided considerably with real measurements. Additionally, ABCs for the FDTD method have extensively been studied and proven solutions are available for implementation. Another advantage of the FDTD-Westervelt method is the use of coefficients in order to model the medium properties at specific grid points. Therefore, reflections result from the change of propagation speed, instead of modeling explicitly interfaces in the tissue like with the DW-Meshes.

Future work will, therefore, concentrate on the FDTD-Westervelt method implemented on the GPU, thus, extending this proof of concept into an ultrasound image simulation. Initially, PMLs or UPMLs for the GPU method would be implemented to provide ABCs. Afterwards, simple transducer models (single element) would be used to compare the simulation results to results from literature. After this step, more complex transducer models would be used, followed by differed focusing schemes. Finally, the signal processing pipeline of ultrasound instrumentation would be implemented to produce the final simulated images. An important issue that might restrict the feasibility of this approach is the mapping of CT or MRI image intensities to sound propagation speed in tissue and the mapping of the other medium specific coefficients used in the Westervelt equation. Thus, an appropriate approximation of this mapping is crucial for the realism of the simulation results.

Another application domain for real-time wave-based ultrasound simulation is Focused Ultrasound Surgery (FUS). The authors of [16] already mentioned the increasing interest towards FUS and the value of accurate simulations for the interaction of High Intensity Focused Ultrasound (HIFU) with human tissue. The applicability of FUS for treating fibroadenomas in the breast, using MR imaging for guidance, has already been demonstrated by Hynynen et al. [21]. The high intensity of the pulses used gives rise to nonlinearly generated harmonics, which subsequently enhance the tissue heating. Predicting this effect by the means of simulation and defining safety regions for the application of HIFU would benefit the FUS treatment. However, this application domain will most likely not be addressed in the imminent future.

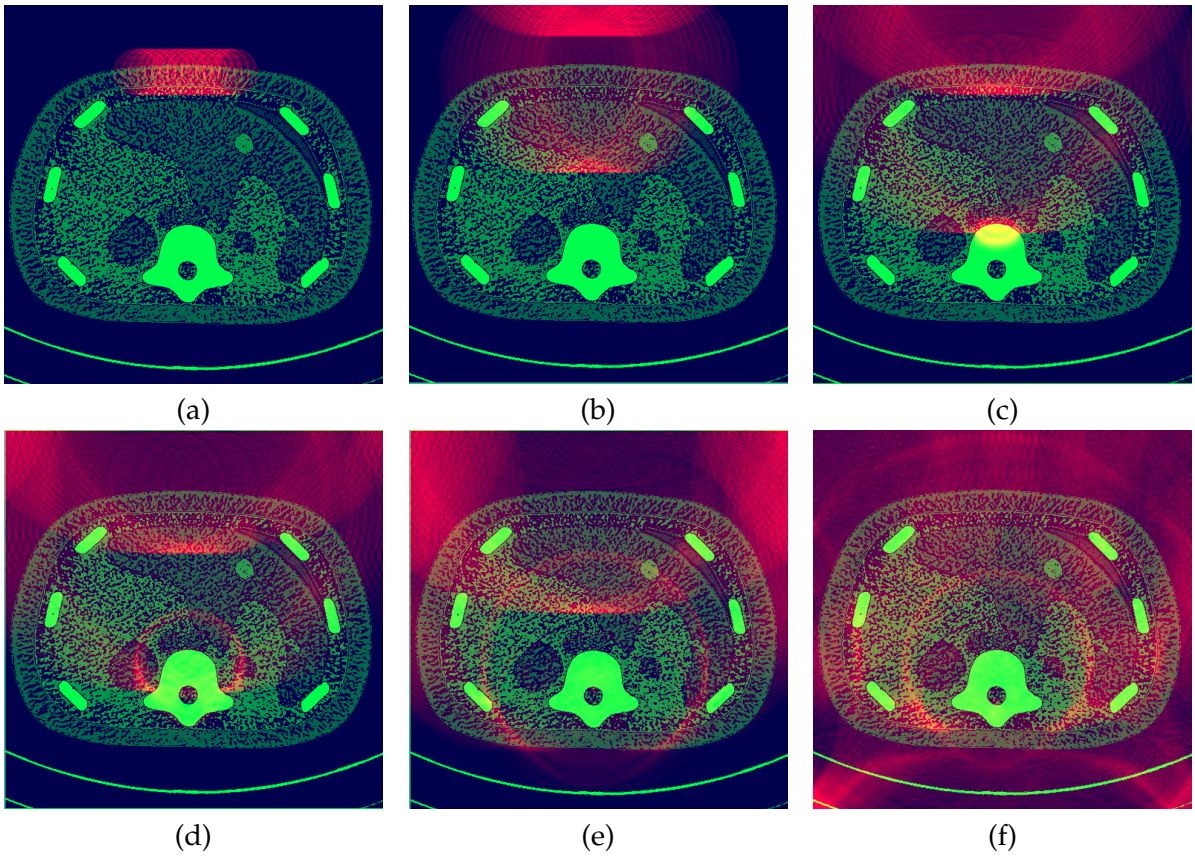
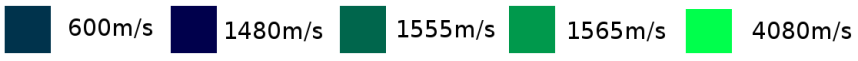


Figure 3.6: 2D FDTD Westervelt simulation using a manually segmented phantom CT dataset with an unfocused wavefront propagation. Reflections at the grid boundaries are the result of lacking ABCs implementation. The images (a-f) show the evolution of the initial wavefront and its interaction with the modeled phantom dataset. The speed of the wavefront propagation increases visibly when the spine is reached, where continuously reflections occur due to the differences in the speed of sound. Furthermore, the decrease in speed of the wavefront is visible in the modeled lung part (600 m/s for middle right section). The equation coefficients were set to ρ_0 (Ambient Density) = 800.0, δ (Diffusivity of Sound) = $1 \cdot 10^{-6}$



Chapter 4

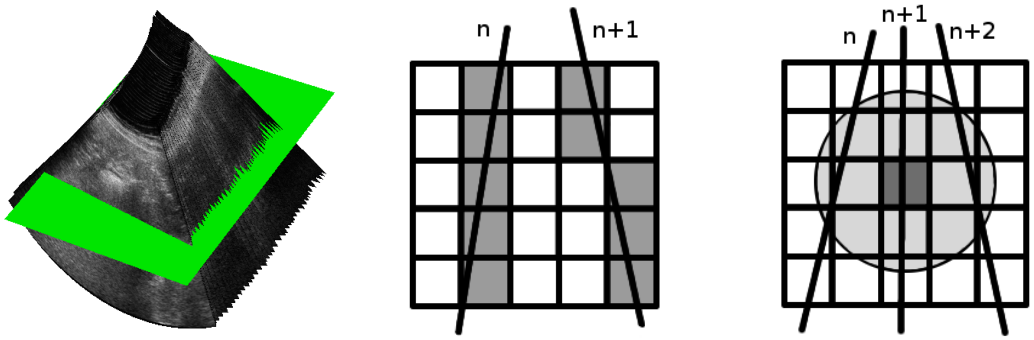
GPU Freehand Ultrasound Volume Reconstruction

Part of this chapter has been published at SPIE Medical Imaging 2009 with the title: 'Fast Hybrid Freehand Ultrasound Volume Reconstruction' [28]

The volumetric reconstruction of a freehand ultrasound sweep, also called compounding, introduces additional diagnostic value to the ultrasound acquisition by allowing 3D visualization and fast generation of arbitrary MPR (Multi-Planar-Reformatting) slices. Furthermore, reconstructing a sweep adds to the general availability of the ultrasound data since volumes are more common to a variety of clinical applications/systems like PACS. Generally there are two reconstruction approaches, namely forward and backward with their respective advantages and disadvantages. In this chapter a hybrid reconstruction method is presented that is partially implemented on the GPU, combining the forward and backward approaches to efficiently reconstruct a continuous freehand ultrasound sweep while ensuring at the same time a high reconstruction quality. The main goal of this work was to significantly decrease the waiting time from sweep acquisition to volume reconstruction in order to make an ultrasound examination more convenient for both the patient and the sonographer. Testing the algorithm demonstrated a significant performance gain by an average factor of 197 for simple interpolation and 84 for advanced interpolation schemes, reconstructing a 256^3 volume in 0.35 seconds and 0.82 seconds respectively.

4.1 Introduction

Three dimensional ultrasound is constantly gaining importance in clinical practice offering additional diagnostic information compared to conventional 2D ultrasound imaging [9]. One advantage of 3D ultrasound compared to 2D ultrasound is obviously its ability to create 3D ultrasound volumes. Using 2D ultrasound, the operator looks at a number of 2D images in order to reconstruct a mental representation of the anatomy. On the other hand, 3D ultrasound can directly represent the exact relationship of anatomic structures. Furthermore, the possible viewing planes which can be acquired with 2D ultrasound are limited by the patient's anatomy and the difficulty of placing/orienting the transducer at specific locations.



(a)Ultrasound sweep with MPR slice (green)

(b)Pixel-Nearest Neighbor

(c)Voxel-Nearest Neighbor

Figure 4.1: Image (a) shows part of an ultrasound sweep of a liver, whereas the green slice represents an MPR slice. Image (b) shows the principle of the Pixel-Nearest Neighbor interpolation for a simplified 2D case. For each pixel on the B-scans $(n, n+1)$ the nearest voxel is filled with its intensity value (marked with grey). Image (c) shows the principle of the Voxel-Nearest Neighbor interpolation using a fixed spherical neighborhood. For each voxel (for example marked with grey) the B-scans $(n, n+1, n+2)$ passing through the sphere are used to fill the voxel intensity value.

Three dimensional ultrasound can acquire unlimited viewing planes, provided that an initial entrance window is available. The long-term evaluation of follow-up examinations using 2D ultrasound is difficult and ill suited, since it is not always clear if the observed changes are due to change in the image acquisition or due to tissue changes. Thus, 3D ultrasound offers improved evaluation accuracy by comparing entire volumes for the evaluation. Furthermore, ultrasound volumes can be used by existing clinical tools including registration, segmentation, visualization and volume estimation.

A variety of methods exist for the acquisition of 3D ultrasound [37] including integrated position sensors, external transducer fixation devices, external position sensing devices and 2D transducer arrays. The work of this thesis focuses on acquisitions with external position sensing devices, which are performed with optic or electromagnetic tracking. For this kind of setup, the transducer position and orientation is recorded for each acquired B-scan, providing spatial and temporal information for the entire acquisition. This set of recorded B-scans is called an ultrasound sweep, demonstrated in figure 4.1 (a).

Mapping the acquired sweep into a regular Cartesian volume and filling the gaps between the B-scans is called freehand ultrasound volume reconstruction and is the main concern of this chapter. The free motion of the transducer during the acquisition results in arbitrary positioned and oriented B-scans in space. This makes the volume reconstruction of scattered B-scans challenging, since only limited assumptions can be made about the acquisition. Limited assumption do not include for example a regular spacing between the acquired B-scans or a constant change in their relative orientation. This assumptions would largely simplify the reconstruction problem, especially in regard to interpolating between the B-scans in order to fill the gaps between them. However, some assumptions can be made, which are in turn depend on the sweep performed to image the target anatomical re-

gion of interest. This assumptions include an implicit ordering of the slices and a relatively continues and uniformly oriented trajectory.

4.2 Related Work

Various methods have been proposed for the volume reconstruction of 2D freehand ultrasound sweeps [45], which can generally be categorized into forward and backward reconstruction/compounding methods. Generally speaking forward methods map 2D ultrasound image pixels into the reconstruction volume, while backward methods use for each voxel the related B-scan pixels. The most straightforward reconstruction method is the Pixel Nearest-Neighbor (PNN) interpolation method which belongs to the forward methods. The first step of this method is to project the B-scans into the volume by mapping each B-scan pixel to the nearest voxel, see figure 4.1 (b). The resulting volume will most likely have considerable gaps between the B-scans. Therefore, in the second step of the method these gaps are filled based on various approaches, including averaging of local neighborhoods and interpolation between the two nearest filled voxels. The method is computationally the least expensive compared to the other methods and can be used for fast direct (meaning, without reconstructing first the entire volume) MPR slice reconstruction, especially when the second step is not performed for the sake of performance. However, the method itself demonstrates visible artifacts in the reconstructed volume. Note that MPR slices refer to reconstructing a 2D slice image with arbitrary position and orientation inside the sweep using the available B-scans. The orientation of a possible MPR slice in a given sweep is demonstrated in figure 4.1 (a).

The following methods belong to the backward reconstruction category and share the fact that they perform operations for each voxel of the volume to be reconstructed. Therefore, the computational requirements of these methods depend strongly on the size of the volume. The Voxel Nearest-Neighbor (VNN) interpolation method assigns for each voxel the intensity value of the nearest B-scan pixel and avoids gaps between the B-scans. However, the resulting reconstruction will still show artifacts. A similar method is the Distance-Weighted (DW) interpolation method which assigns for each voxel the weighted average from a selected set of pixel intensities. Different approaches are used to specify the weight function and the voxel neighborhood for the selection. The most straightforward approach is realized by using a sphere neighborhood with a predefined fixed radius, demonstrated in figure 4.1 (c). Gaps might result if the radius is chosen too low and blurring of the reconstruction can occur when the radius is chosen too high, where local shape properties are not reproduced. Another approach uses non-uniformly shaped neighborhoods in form of a truncated 3D Gaussian kernel. Again, gaps might be formed if the Gaussian smoothing window is too small. Lastly, the DW method can also be realized by interpolating only between the two B-scans that lie on each side of a given voxel. This approach offers a gap free reconstruction and retains the resolution of the original B-scans. However, depending on the acquisition it is not always clear which B-scan pairs should be used for the interpolation, as for example when the trajectory of the acquisition is discontinuous.

An improved version of the previous DW method, which only used a pair of B-scans for the reconstruction of a given voxel, is proposed in Coupé et al. [8]. The method includes the probe trajectory for defining the weight function and demonstrates superior reconstruction

quality compared to the other approaches. However, the method shares the same limitations as it's original proposal, namely the need for an explicit B-scan acquisition order. Additionally, it is limited by assuming a constant speed along the probe trajectory during the sweep acquisition.

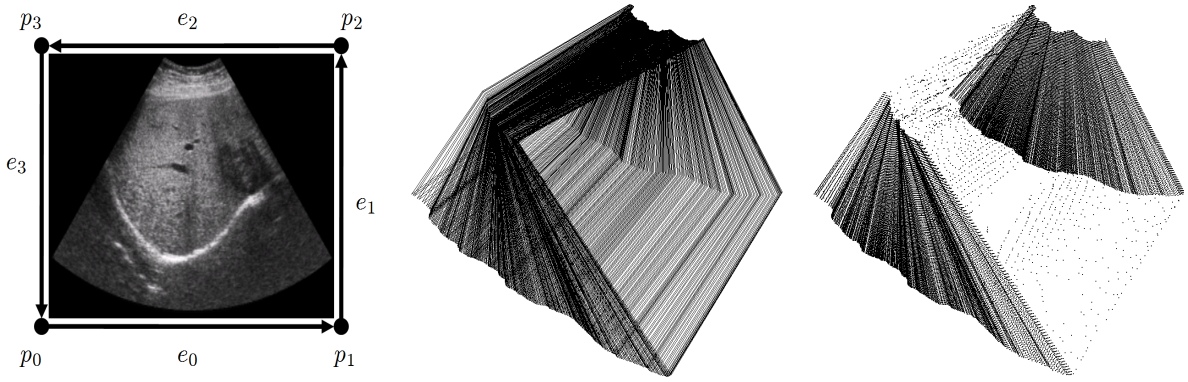
A reconstruction method that does not belong neither to the forward nor to the backward method is introduced in Rohling et al. [45] and is referred to as Radial Basis Function (RBF) interpolation method. The method is based on functional interpolation and reconstructs the volume without any visible artifacts. Furthermore, the reconstruction quality surpasses that of PNN and VNN methods, with a statistical comparison of the reconstruction quality between the DW and the RBF method showing equivalent results. The downside of the approach is the high computational demand. Other methods can be computed in a matter of a few minutes, whereas the RBF method takes up to a few hours.

As stated, the main concern of this work is to reduce the waiting time from acquisition to volume reconstruction, which especially becomes irritating when the acquisition has to be repeated multiple times, e.g. in the context of whole organ imaging for guiding interventional oncology procedures [54], or registration to a pre-operative plan for prostate radiotherapeutic delivery. An initial attempt to improve the overall performance of the described backward methods is presented in Wein et al.[55] which utilizes a fast slice selection and an efficient voxel traversal algorithm, allowing fast direct MPR slice reconstruction. Our goal is to further decrease the waiting time aiming for near real-time reconstruction, where the reconstruction quality should at least be comparable with existing methods.

4.3 Acceleration potential of available approaches

The previously mentioned methods, except the RBF method, had been implemented for an existing framework using a spherical neighborhood to limit the influence of B-scans. Additionally, the proposed acceleration techniques presented in Wein et al. [55] were utilized to improve the performance. However, as we will also see later on, the accelerated methods still require about a minute for the reconstruction of a 256^3 volume. In recent years, the use of the GPU for solving time-consuming problems has been proven as an efficient solution [1]. Therefore, in order to significantly decrease the reconstruction time and achieve real-time reconstruction, the implementation of the previous algorithms on the GPU seemed a promising approach. Even though GPUs demonstrate higher performance, in terms of Giga Floating point Operations Per Second (GFLOPS), compared to CPUs there is still an issue of when the problem can be parallelized and when it is suitable for GPU implementation. The authors of [8] also pointed out the prospect of using the GPU for the acceleration of the volume reconstruction. However, no literature was found that explicitly dealt with this topic.

Considering the suitability of the previous methods, the PNN method seems favorable for GPU implementation because of the available GPU 'render-to-volume' feature which would accelerate the first step of projecting the B-scans into the volume. The second step of the PNN method could also be accelerated using GPUs since they demonstrated high acceleration potential for signal-processing-like applications [22]. However, the aim was to deliver a good reconstruction quality which has been demonstrated by the DW methods



(a)Image edge lines (b)Sweep edge lines (c)Intersection points

Figure 4.2: The left image demonstrates the edge line setup, which is performed for all B-scans. In the center image all edge lines are rendered for an entire sweep. The right images shows the intersection points between sampling planes and edge lines.

[45]. Implementing DW methods on the GPU poses a problem considering the fact that for each voxel at least a pair of two 2D textures (containing the B-scans) has to be bound on the GPU. The problem lies in the time consuming task of reading from textures on the GPU, which is magnitudes slower than performing calculations in shaders. Thus a direct port of the proposed DW methods to the GPU would not result in the desired real-time performance due to a bottleneck caused by the delays from texture reads. To deal with this problem a new method is introduced which is similar to the DW method using B-scan pairs.

4.4 Methods

In this section the proposed algorithm will be presented starting with the required modeling of the input. Initially the algorithm requires a geometrical representation for each B-scan in form of defining a line for each B-scan edge, as shown in figure 4.2 (a). Figure 4.2 (b) shows a visualization of B-scan edge lines for a given ultrasound sweep.

The edge lines are defined as:

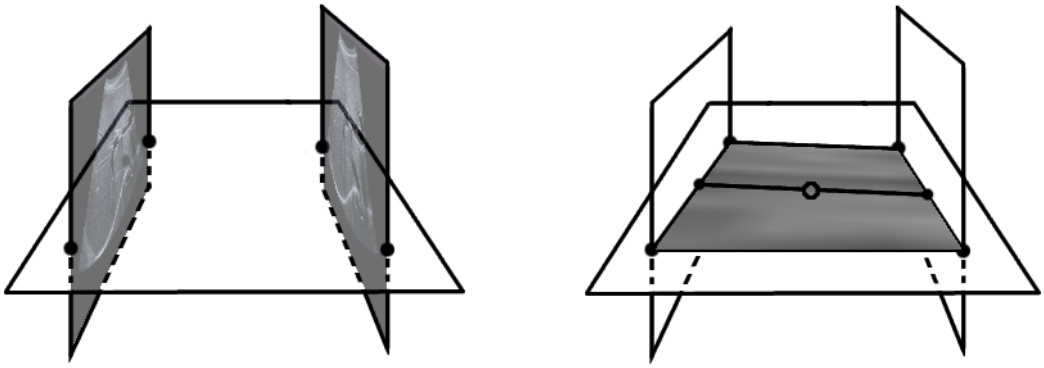
$$p = p_a + u \cdot \vec{D_l} = p_a + u \cdot (p_b - p_a) \tag{4.1}$$

where p_a, p_b are the points defining the line segment and u is the line parameter. The necessary B-scan image points $p_{0..3}$ for this representation can be determined by utilizing the recorded tracking information and the actual physical size of the recorded images. The next step is to find the optimal sampling direction, which is basically done by taking the average of all the B-scan edge line direction vectors $\vec{D_l} = p_0 - p_3$. The reason for this selection will become clear later on in the discussion section.

A reconstructed volume can be interpreted as a stack of 2D MPR slices of the target sweep, which are stored along a desired default axis (x,y or z) in the target volume. Based on this interpretation if the size of the target volume dimension is M, then also a number of

Algorithm 1 Fast Hybrid Freehand Ultrasound Volume Reconstruction

Require: Sweep image edge lines
1: Find optimal sampling direction
2: Define sampling layers
3: **for all** sampling layers **do**
4: **for all** slices **do**
5: **if** edge line not parallel to sampling layer **then**
6: find valid line-plane intersections
7: calculate texture coordinates
8: **end if**
9: **end for**
10: render quadrilaterals
11: **end for**



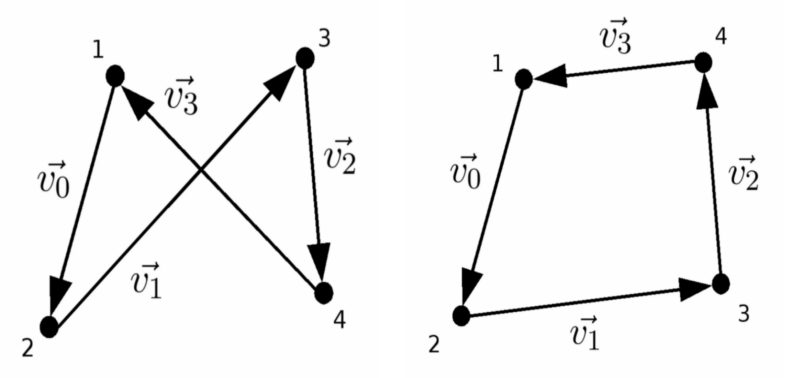
(a) Edge-lines plane intersection (b) Quadrilateral interpolation

Figure 4.3: Forming the quadrilaterals. The left image shows the intersection between two B-scans with a sampling plane and the resulting intersection points along the line segments. The right image shows the rendering of the interpolated quadrilateral utilizing the previous intersection points which lie on the sampling plane.

M sampling layers are created. These sampling layers are defined by planes $\vec{n} \cdot (x - x_m) = 0$, where $\vec{n}$ is the previously selected sampling direction and x_m a point that lies on the plane of sampling layer m. Therefore, each sampling plane corresponds to one MPR slice. After those initial steps the main algorithm is performed, which is explained in more detail in the following subsections. At this point, note that only step 10 (render quadrilaterals) of the algorithm is performed on the GPU, where the other steps are entirely performed on the CPU.

4.4.1 MPR generation

For the next steps of the algorithm we first have to find the intersection points between the sampling layers and the set of B-scan edge lines, demonstrated for two B-scans in figure 4.3(a). For each sampling layer we check if the given B-scan edge lines are intersected by



(a)Twisted Quad (b)Correct Quad

Figure 4.4: Image (a) a twisted quad which would be visible as are reconstruction artifact. Image (b) shows the corrected quad by exchanging point 3 with 4.

the sampling plane through testing if $\vec{n} \cdot \vec{D_l} \neq 0$. Additionally, a B-scan plane intersection is conditionally valid if two of the four image edge lines are intersected. Otherwise, any given intersection point for this B-scan is discarded. After passing the initial test, the actual intersection point for a given edge line is calculated, firstly by solving the line parameter u using the equation below:

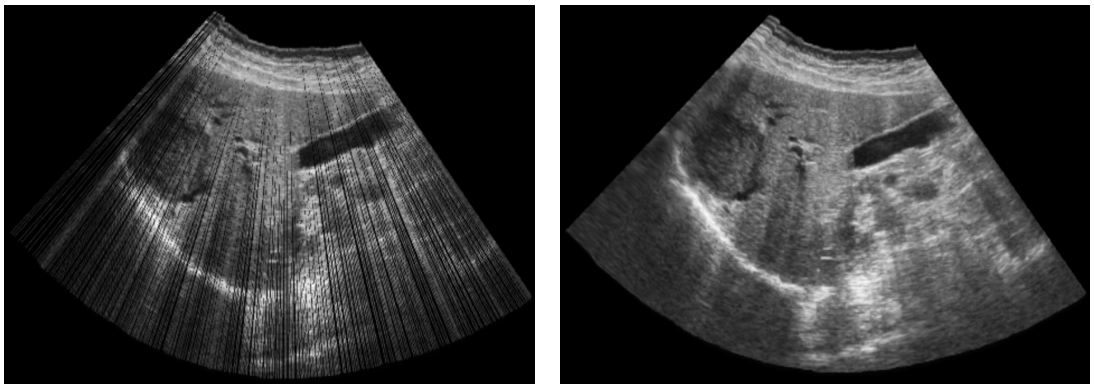
$$u = \frac{\vec{n} \cdot \vec{D_p}}{\vec{n} \cdot \vec{D_l}} = \frac{\vec{n} \cdot (x_m - p_a)}{\vec{n} \cdot (p_b - p_a)} \tag{4.2}$$

The B-scan plane intersection becomes finally valid if the two intersection points lie on the respective edge line segments. This is simply checked by $0 \leqq u \leqq 1$. Now substituting the u parameter into the line equations yields the final pair of intersection points for a given B-scan. The same steps are performed for all the B-scans and sampling planes resulting in a set of intersection points for the given sweep, as shown in figure 4.2 (c).

In the next step for each sampling layer we render quadrilaterals (quads) utilizing the intersection points of adjacent B-scans by using common graphic Application Programming Interfaces (API)s like OpenGL (in our case) or DirectX. However, the order in which the intersections points are passed to the APIs for rendering is important, because for some cases twisted quads may be formed, as demonstrated in figure 4.4 (a). Therefore, these cases have to be detected and the ordering of the two points lying on one of the B-scans exchanged. This is simply done by checking the following condition:

$$(\vec{v_0} \times \vec{v_1}) \cdot (\vec{v_2} \times \vec{v_3}) = c \tag{4.3}$$

If $c < 0$ then the ordering has to be changed by switching points 3 and 4. Otherwise, if $c > 0$ then the ordering is by default correct. When $c = 0$ then slice n and slice $n + 1$ are perpendicular and the default ordering is used since changing the ordering would not change the outcome of the quadrilateral rendering. Note that the entire rendering is performed using orthogonal projection. The rendered quads are textured using the texture coordinates at the intersection points on the edge lines. These can simply and efficiently be calculated by knowing the edge of intersection, which results in assigning one of two texture coordinates (s,t) to 0.0 or 1.0. The second coordinate is calculated by dividing the Euclidean distance of



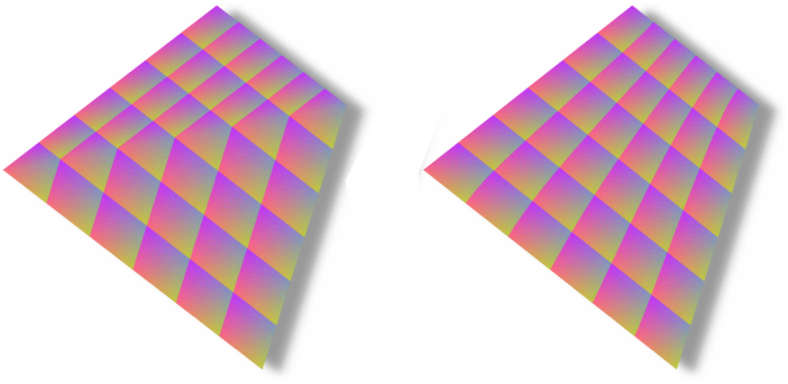
(a)Line MPR (b)Quad MPR

Figure 4.5: Image (a) shows a reconstructed MPR slices for which the B-scans are projected as lines onto the MPR. Image (b) shows the same reconstructions using the hybrid methods for which quads are used to connect adjacent B-scans.

an intersection point to an image edge point, by the length of the edge it lies on, eventually compensating to map it to the default texture coordinate system. Forming quadrilaterals for adjacent B-scans produces an MPR slice of the sweep for a given sampling layer, where the same steps are performed for all MPR slices that form the reconstructed volume. Using quadrilaterals to connect adjacent B-scans results in avoiding gaps that might arise, as in the case of the PNN method (first step). An example of this, is presented in figures 4.5 (a) and 4.5 (b). Image (a) shows the direct projection of the textured edge lines onto the reconstructed MPR slice. This would be similar to the result of the PNN methods first step. The second image (b) shows the same MPR slice but now the hybrid method is used which utilizes quadrilaterals to fill the previously visible gaps. Furthermore, this also results in using the hardware interpolation of the GPU in order to bilinearly interpolate inside the B-scan and linearly interpolating between the pair of B-scans, which is similar to the DW method. However, the method can not be considered as a forward method nor as a backward method, since the first steps are similar to the forward methods, rendering the B-scans into the volume, however, the next steps perform an interpolation between the slices which is a characteristic of backward methods. Therefore, the method is referred to as a hybrid reconstruction method.

If the acquisition images the same anatomy twice, the most recent sweep imaging data will be reconstructed due to the successive drawing of slices. Repeatedly imaging the same anatomy with freehand ultrasound usually does not yield the exact same data, due to the highly dynamic imaging process (speckle noise, tissue deformation etc.); hence this behavior is desirable, unless explicit averaging is needed (similar to aperture compounding on ultrasound transducers).

In case of discontinuous sweeps, a distance restriction can be employed that limits the acceptable distance between the edges of the quads which connect intersection points of adjacent B-scans. If the distance between the B-scans is higher than the specified threshold, then the connecting quad is not rendered. This prevents from rendering large quads which connect discontinues B-scans across the reconstructed volume. Such a threshold is



(a)Triangularization and Interpolation

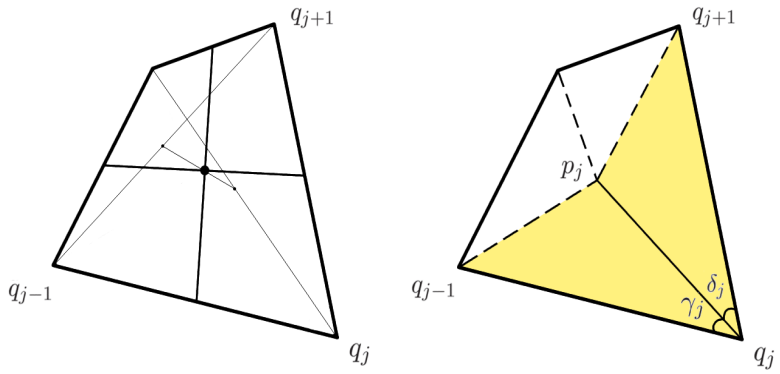
(b)Barycentric Interpolation

Figure 4.6: The irregular quadrilaterals in the images are textured with a regular pattern. In image (a) we can clearly see the interpolation artifact caused by splitting the quadrilateral into two triangles which are separately interpolated using barycentric coordinates. On the other hand, the pattern in image(b) is continuous using an alternative barycentric interpolation. Images from [18].

also advisable to prevent misinterpretation of reconstructed data, which is the case for reconstructions that might visually seem correct but anatomically be false due to the lack of available B-scans for reconstructing the desired region of interest [45]. The optimal distance threshold, in regard to clinical safety, is outside the scope of this thesis.

4.4.2 Quadrilateral interpolation

The interpolation of the quadrilateral is performed inside a GPU fragment shader program (shader) [46] using the respective pair of B-scan textures. Inside the shader the B-scan pixel intensities are accessed along the respective quad edge using the GPUs bilinear texture filtering. The resulting intensities of both edges intersecting the B-scans are linearly interpolated inside a shader along the entire quad surface. Therefore, each pixel on the quadrilaterals is the result of a linear interpolation between the bilinear interpolated values of the adjacent B-scans, as shown in figure 4.3 (b). However, even modern GPU hardware does not support direct rendering of quadrilaterals and decomposes them into two triangles which are processed separately. This results in a discontinuous attribute interpolation inside the quadrilateral [18], which is highly visible since we use irregular convex quadrilaterals for rendering. To demonstrate this issue a quadrilateral texture with a pattern is presented in figure 4.6 (a) and (b), using different interpolation schemes. In order to deal with this problem, we used the following two approaches. The first approach approximates an improved interpolation by splitting the quadrilateral into four triangles using the center point inside the quadrilateral, see figure 4.7 (a). The centroid can simply be calculated from the midpoint of one of the bimedians, since the intersection of the bimedians is the centroid, with the bimedian bisecting each other based on Varignon's Theorem. Alternatively, the centroid point can be calculated form the midpoint of the line segment connecting the midpoints of the



(a)Quadrilateral Centroid

(b)Barycentric Coordinate Calculation Scheme

Figure 4.7: Image (a) shows the centroid for the quadrilateral calculated from the bimedians or from the diagonals. Image (b) shows the efficient barycentric coordinate calculation scheme proposed by Meyer et al. [35].

diagonals, also shown in figure 4.7 (a). Generally, the visual continuity of the interpolation improves with the increase of splits made, however, the processing time increase with the increasing number of triangles to handle.

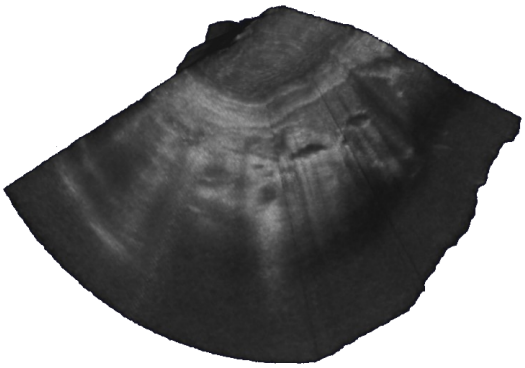
The second approach is more sophisticated and defines a continuous attribute interpolation inside the quadrilateral by implementing generalized barycentric coordinates for irregular convex polygons in a fragment shader program, which makes this approach computationally more demanding. In Meyer et al. [35] an easy way was introduced to calculate the barycentric coordinates for a point p inside an irregular N-sided convex polygon, with $j \in [1..N]$. Thus, the non-normalized weight ω_j and the normalized weight $\hat{\omega_j}$ for each of the points q_j which form the polygon is calculated by:

$$\omega_j = \frac{\cot(\gamma_j) + \cot(\delta_j)}{\|p - q_j\|^2} \qquad \hat{\omega_j} = \frac{\omega_j}{\sum_N^1 \omega_j} \tag{4.4}$$

The calculation process is schematically presented in figure 4.7 (b). To improve the accuracy, the authors suggest to calculate the cotangent not with trigonometric function calls, but with a division between the dot product and the cross product of the triangles involved. For our example, the involved triangles are (p, q_j, q_{j-1}) and (p, q_j, q_{j+1}), also marked in figure 4.7 (b). The cotangents are, therefore, given as:

$$\vec{a} = p - q_j \qquad \vec{b} = q_{j-1} - q_j \qquad \vec{c} = q_{j+1} - q_j \tag{4.5}$$

$$\cot(\gamma_j) = \frac{\vec{b} \cdot \vec{a}}{\left\|\vec{b} \times \vec{a}\right\|} \qquad \cot(\delta_j) = \frac{\vec{c} \cdot \vec{a}}{\|\vec{c} \times \vec{a}\|} \tag{4.6}$$



(a) Volume Rendering

Figure 4.8: Image shows the volume rendering of a reconstructed volume from an ultrasound sweep of the liver.

Therefore, for the interpolation of the quadrilateral we calculate the barycentric coordinates for each fragment of the quadrilateral and interpolate using the weights for the respective vertices forming the rendered quadrilateral.

4.4.3 Volume generation

Nowadays most graphic cards provide the ability to 'render-to-volume' which allows redirecting the rendering from the display to an internal volume on the graphics card, without much effort. This volume can directly be used to generate arbitrary MPR slices or for 3D rendering of the entire volume, see figure 4.8 (a) , where trilinear interpolation for volume access is supported by the GPU hardware without any performance penalty. In order to render the sampling layers (MPRs) into the volume, a transformation matrix is calculated which co-aligns the sampling direction (common quad normal) with a volume axis (default the z-axis). Based on the extents of the sweep, a translation is included to center the volume, while calculating an Oriented Bounding Box for the sweep could be used to perfectly fit the sweep inside the volume, saving unnecessary voxel space. The final volume, can, then be read back into the RAM in order to be used by external tools or stored on the hard drive for later reference.

4.5 Results

The method we tested against a variety of established backward compounding approaches, which have been implemented in a separate framework for the CPU and which enforce various acceleration techniques including fast slice selection and efficient voxel traversal [55]. The Maximum, the Gaussian, the Weighted Median and the Inverse method belong the backward methods and use a fixed spherical neighborhood. More details on these methods can be found in Wein et al. [55]. We also tested our method against a non-accelerated forward

compounding implementation (not included in the mentioned average factors). The performance was evaluated on an Intel Xeon 3.2 GHz CPU with 2GB RAM and a GeForce 8800GTX 768MB GPU. The tests were performed for a dataset containing 293 slices and for a target reconstruction volume size of 256^3. The resulting measurements are presented in table 4.1. For a readback from GPU to host memory 0.275 seconds would be required for the described volume.

Table 4.1: Performance measurements with execution times for each method and the performance gain factor compared to the quadrilateral split and barycentric coordinate interpolation method. The proposed hybrid method outperforms considerably the existing methods.

Hybrid Method	Trajectory	Maximum	Gaussian	Weighted Median	Inverse	Forward
-	72.72s	60.80s	69.78s	75.60s	66.52s	471.88s
Quad Split 0.35s	208x	174x	199x	216x	190x	1348x
Barycentric 0.82s	88x	74x	85x	92x	81x	575x

As initially stated, the main goal of the proposed algorithm is to deliver real-time volume reconstruction of freehand ultrasound. Additionally, the approach must ensure a sufficient reconstruction quality compared to available methods. For this purpose we compared the directly generated MPR slices using the hybrid method (quad split and barycentric coordinates) with those generated by other methods. For the comparison, we calculated the absolute differences between the pixels values of the hybrid MPR slice and those generated by the other test methods. The difference images resulting between the hybrid (quad split) MPR reconstruction and the nearest and the trajectory method are presented in figure 4.9. The absolute difference images for all methods compared to both the quad split and the barycentric coordinate implementation of the hybrid method are presented in the appendix in figures 6 and 7 respectively. Furthermore, the Mean Absolute Differences (MAD) for all reconstruction methods are presented in table 4.2, with the MAD is calculated as:

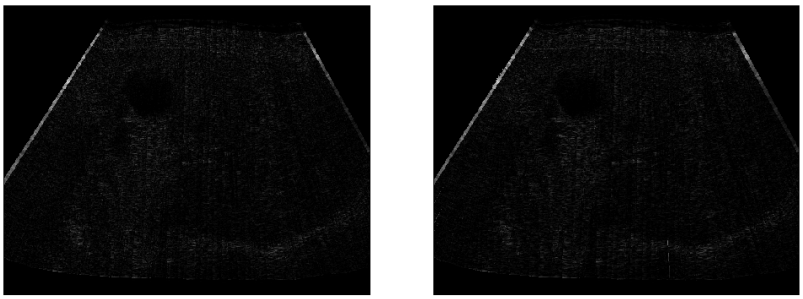
$$MAD = \frac{1}{M} \sum_{i=1}^{M} |p_i - u_i| \tag{4.7}$$

where p_i is the MPR slice pixel intensities using the hybrid method, u_i is the MPR slice pixel intensities from the other test methods and M is the total number of image pixels.

Table 4.2: Mean Absolute Differences for reconstruction methods

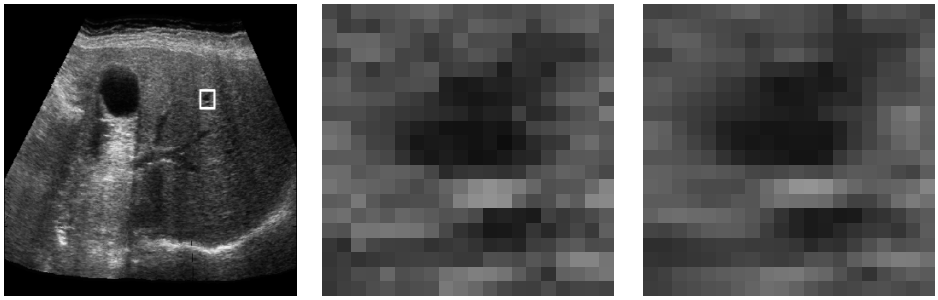
Hybrid Method	Nearest	Maximum	Trajectory	Inverse	Gaussian	Weighted Median
Quad Split	8.16	12.68	7.66	7.43	7.68	8.07
Barycentric	10.39	13.83	9.87	9.72	9.61	10.11

The presented MAD values suggest a good similarity between the hybrid quad split method and the trajectory method, which is encouraging since the trajectory method delivers generally high quality reconstruction results. The barycentric coordinate interpolation



(a)Nearest (c)Trajectory

Figure 4.9: Images of absolute differences between the hybrid (quad split) method and the nearest and the trajectory method. The intensity values are scaled to [0..255], the common intensity range used for B-scan imaging.



(a) Sweep MPR slice (c)Trajectory (d)Hybrid (Quad Split)

Figure 4.10: Image (a) shows a reconstructed slice from a liver sweep. The marked ROI shows a small vessel. The other images show the zoomed ROI for MPR slices using different reconstruction methods.

yields slightly higher differences compared to the quad split interpolation. This is rather unexpected and still investigated. Additionally, to the MAD comparison, the Mean Square Error (MSE) values were also calculated and are presented in the appendix in table 3.

The high differences on the left and right of the images in figure 4.9, and in the appendix in figures 6 and 7, are due to the fact that the backward compounding is realized with a fixed spherical neighborhood, meaning that for each voxel only B-scans that pass through the spherical neighborhood are taken into account for the reconstruction. Hence the hybrid method will reconstruct pixels only between the first and the last B-scan, while the backward method will also fill pixel intensities slightly outside this range. A visual comparison between the approaches is presented in figure 4.10, showing a zoomed region of interest from reconstructed MPR slices. Again, the trajectory and the hybrid (quad split) method appear similar, however the trajectory method slightly outperforms our method by preserving better the native texture pattern of the B-scan. The visual comparison for all available methods is presented in the appendix in figure 5.

4.6 Discussion

The performance of the proposed method is ensured through the simple calculation of intersection points and the hardware accelerated interpolation of the rendered quadrilaterals. The method shares the same limitation as the DW B-Scan pair interpolation method and trajectory method, namely assuming a continuous trajectory for the sweep. The main limitation of this approach is the orientation of the sampling plane used to calculate the intersection points. This is due to the fact that the sampling plane is not always positioned perpendicular to the B-scans, which would be the most correct orientation for the interpolation. Therefore, the more parallel the sampling plane to the B-Scans is, the higher becomes the interpolation error for the voxels between the B-scans. The error depends on the distance between the originally acquired B-scans, resulting naturally in smaller errors for dense sweeps and larger errors for sparse sweeps. The worst case scenario for the reconstruction would be sampling planes near or completely parallel to the B-scans, resulting in a very small number of intersections and erroneous interpolation.

However, these cases can be detected and handled by taking directly the nearest available B-scan for a given sampling plane. Furthermore, for these cases one could interpolate between the adjacent B-scans by orthogonally projecting the pixels of the adjacent B-scans onto the reconstructed MPR slice, which is similar to interpolating between pairs of B-scans as described for the DW methods.

4.7 Conclusion

In this chapter, an efficient method for ultrasound volume reconstruction using graphics hardware was presented, which demonstrated a significant performance gain compared to other methods. The general simplicity of the approach leads to a fast re-implementation time and further encourages its use. The proposed technique yields on-the-fly volumetric reconstruction, which can be of use in a great variety of clinical settings for visualization, registration and interventional navigation. The trade-off between quality and performance depends on the sweep to be reconstructed. Therefore, uniformly oriented slices along a continues probe trajectory are favorable for this approach and demonstrate high reconstruction quality.

The method can be extended to accommodate more sophistic geometric modeling of a sweep. In particular, taking the probe trajectory into account [8], can be implemented in the GPU fragment shader. Likewise, a two-dimensional curvilinear scan-conversion can be implicitly computed, allowing for high-fidelity reconstruction from raw US data, or efficient simulation of ultrasonic volumes [50].

Future work will focus on handling the worst case scenario (parallel sampling plane), initially by the means of the proposed approaches. Additionally, it is worth pursuing approaches to minimize the sampling plane orientation dependent interpolation error, which would make the method available to a larger group of problems.

Chapter 5

Conclusion

The GPU ray-based ultrasound simulation demonstrated real-time and high-throughput simulation results, making the method suitable for both training and registration purposes. The available performance was further utilized to apply convolution operations on the simulated scanlines, in order to introduce additional effects. These include, a horizontal Hanning window convolution to simulate the beam width effect and a 2D Gaussian convolution to smooth the reflections. Furthermore, the combination of Perlin noise with the Hanning window resulted in more realistic images for training applications. Additionally, the method is able to simulate multiple images at once and allows the real-time simulation of 3D ultrasound volumes.

The chapter which referred to wave-based ultrasound simulation was focused on investigating possible methods which would lead to real-time wave-based simulation using graphics hardware. During the course of this investigation the Digital Waveguide Mesh (DW-Mesh) was demonstrated rather insufficient for ultrasound simulation, because it introduces errors for each boundary junction defined in the simulation domain. However, boundaries are necessary to model reflectors in the simulation domain. Therefore, modeling the numerous interfaces in human tissue with boundaries in the DW-Mesh, based on CT or MRI datasets, would also result in defining numerous error sources. What's more, explicitly defining all ultrasound reflecting interfaces for a CT or MRI dataset is rather impractical and restricts even further to applicability of this method for ultrasound simulation. On the other hand, the FDTD method for solving the Westervelt equation is the most promising candidate for real-time wave-based ultrasound simulation. To begin with, literature has already pointed out the suitability of the FDTD-Westervelt method for ultrasound simulation, through demonstrating good agreements between predicted signal responses and measurements from real ultrasound in water tanks. Furthermore, the GPU has already demonstrated it's potential for accelerating FDTD simulations. The initial implementation of a GPU-based FDTD-Westervelt simulation, in this thesis, demonstrated the potential of the approach.

Lastly, a fast freehand ultrasound volume reconstruction algorithm was presented, which mostly runs on the CPU and performs a small number of computationally demanding steps on the GPU. The demonstrated performance exceeded by far available approaches, reconstructing a sweep into a 256^3 volume in 0.35 seconds. Furthermore, initial evaluations indicated a good reconstruction quality for continuous freehand sweeps.

5.1 Future Work

Future work for the ray-based ultrasound simulation includes the integration of the GPU implementation into a multimodal registration framework. Since the GPU implementation demonstrated equivalent results with it's CPU counterpart, the integration should result in an accelerated registration process. The realism of the simulation can further be increased by integrating more complex simulation models, as described in [31]. Nevertheless, additional effort is needed to further increase the realism of the simulation, especially for training applications. For this, investigating further the noise function is necessary, in order to find an appropriate statistical noise distribution and appropriate intensity amplitudes, which would match speckle more realistically. Another important factor is the use of the transfer function for mapping CT intensities to acoustic impedance values. Optimizing the transfer function to improve the accuracy of the mapping would result in more realistic simulation results.

Major part of the future work will focus on the FDTD-Westervelt simulation method. As previously mentioned, a lot of work is required to move from the initial implementation to the simulation of 2D ultrasound images. This includes, the implementation of PMLs or UPMLs on the GPU, where literature is already available which suggests possible implementations. Furthermore, validation phases would be very important in order to verify the simulation results with existing results in literature and in order to extend further the simulation. Thus, initially single element transducers will be tested and the predicted signal responses compared with actual measurements (available in literature). Afterwards, more sophisticated setups would be evaluated, including multi-element 1D transducers and different focusing schemes.

Considering the freehand volume reconstruction, future work will aim at making the method available to a wider group of freehand ultrasound acquisitions. Currently, a good reconstruction quality is demonstrated for continuous sweeps with the orientation of the images being relatively uniformly along the acquisition trajectory. Previously in the chapter referred to the volume reconstruction, a special case was mentioned which restricted the reconstruction method to certain acquisitions. However, this special case can be detected and handled accordingly. On the other hand, the reconstruction dependent interpolation error will further be investigated in order to determine the error range and find alternative interpolation schemes.

Appendices

Relevant SI Units

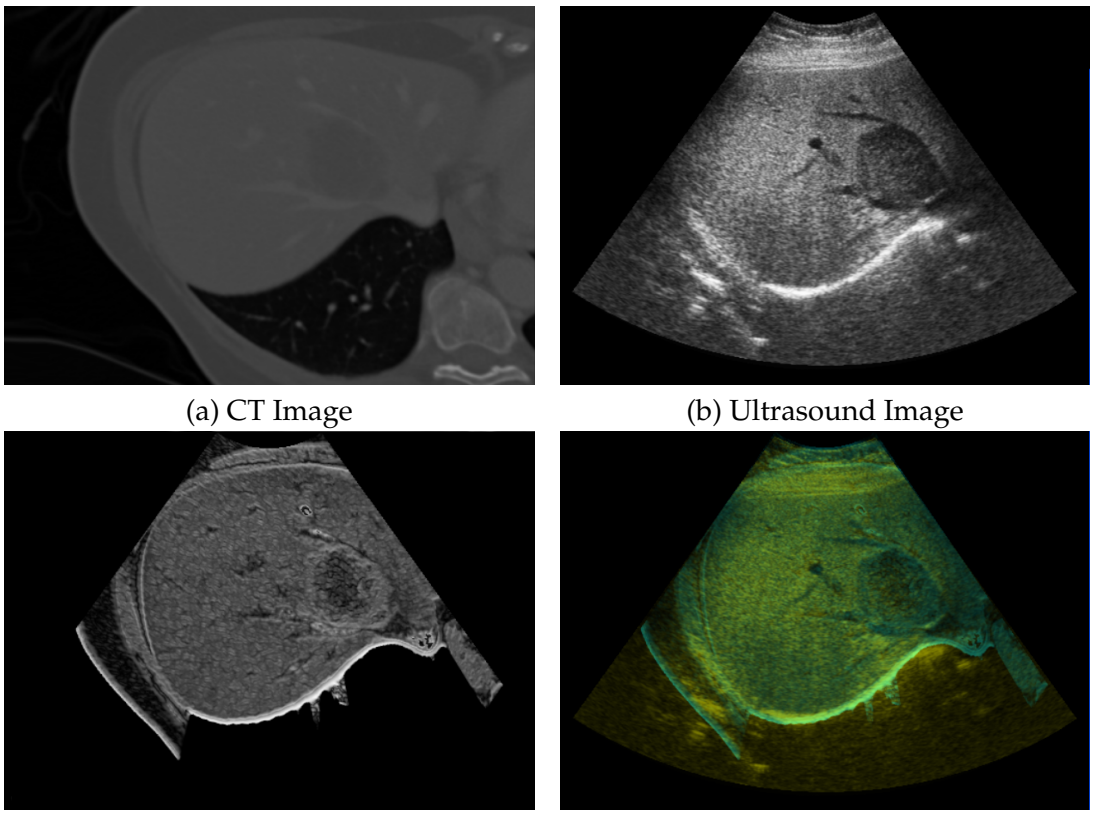
Table 1: SI

Derived Quantity	Name	Symbol	Expression in terms of other SI units	Expression in terms of SI base units
force	newton	N	-	$m \cdot kg \cdot s^{-2}$
energy, work, quantity of heat	joule	J	$N \cdot m$	$m^2 \cdot kg \cdot s^{-2}$
power, radiant flux	wattt	W	J/s	$m^2 \cdot kg \cdot s^{-2}$
pressure, stress	pascal	Pa	N/m^2	$m^{-1} \cdot kg \cdot s^{-2}$
heat flux density, irradiance	-	-	W/m^2	-

Ray-based Simulation

Table 2: Performance table for GPU ray-based ultrasound simulation. Batch size is the number of simultaneously simulated images. An image consists of 128 scanlines with 128 samples.

Batch Size	1	2	4	16	32	64	128	256	512	1024	2048
Images/sec	80	160	320	1264	2496	4864	9344	12288	13824	15360	8192



(a) CT Image (b) Ultrasound Image

(c) Simulated Image (d) Simulated and real image blended

Figure 1: The images (a-d) are taken for the same target anatomy. Image (a) shows the CT volume MPR which corresponds to the sound beams. Image (b) shows a real ultrasound image for this target anatomy. Image (c) shows the simulated image and image (d) blends the real and the simulated image for comparison purposes. The similarities between real and simulated image are enough in order to register them.

Wave-based Simulation

FDTD Method for the 2D Wave Equation

In the following example the 2D wave equation will be solved using the FDTD method. The grid setup used is presented in figure 2. A similar approach would be taken for 2D wave-based ultrasound simulation.

$$\frac{\partial^2 u(x,y,t)}{\partial t^2} = c^2 \nabla^2 u(x,y,t) \Rightarrow \frac{\partial^2 u(x,y,t)}{\partial t^2} = c^2 \left(\frac{\partial^2 u(x,y,t)}{\partial x^2} + \frac{\partial^2 u(x,y,t)}{\partial y^2} \right) \tag{1}$$

To make the formulation more compact the following conversions are used:

$$\begin{aligned} u(x \pm \Delta x, y, t) &= u^t_{x\pm1,y} \\ u(x, y \pm \Delta y, t) &= u^t_{x,y\pm1} \\ u(x, y, t \pm \Delta t) &= u^{t\pm1}_{x,y} \end{aligned}$$

The finite differences for the partial derivatives are defined as below, using second order accurate Taylor expansions.

$$\begin{aligned} \frac{\partial^2 u}{\partial t^2} &= \frac{u^{t-1}_{x,y} - 2u^t_{x,y} + u^{t+1}_{x,y}}{\Delta t^2} \\ \frac{\partial^2 u}{\partial x^2} &= \frac{u^t_{x-1,y} - 2u^t_{x,y} + u^t_{x+1,y}}{\Delta x^2} \\ \frac{\partial^2 u}{\partial y^2} &= \frac{u^t_{x,y-1} - 2u^t_{x,y} + u^t_{x,y+1}}{\Delta y^2} \end{aligned}$$

Substituting the finite difference into the initial equation results in:

$$\frac{u^{t-1}_{x,y} - 2u^t_{x,y} + u^{t+1}_{x,y}}{\Delta t^2} = c^2 \left(\frac{u^t_{x-1,y} - 2u^t_{x,y} + u^t_{x+1,y}}{\Delta x^2} + \frac{u^t_{x,y-1} - 2u^t_{x,y} + u^t_{x,y+1}}{\Delta y^2} \right)$$

The term of interest is $u^{t+1}_{x,y}$, which gives the wave amplitudes for the new timestep. Solving this term results in the final difference equation.

$$u^{t+1}_{x,y} = \frac{c^2 \Delta t}{\Delta x^2 \Delta y^2} \left[\Delta y^2 \left(u^t_{x-1,y} - 2u^t_{x,y} + u^t_{x+1,y} \right) + \Delta x^2 \left(u^t_{x,y-1} - 2u^t_{x,y} + u^t_{x,y+1} \right) \right] + 2u^t_{x,y} - u^{t-1}_{x,y}$$

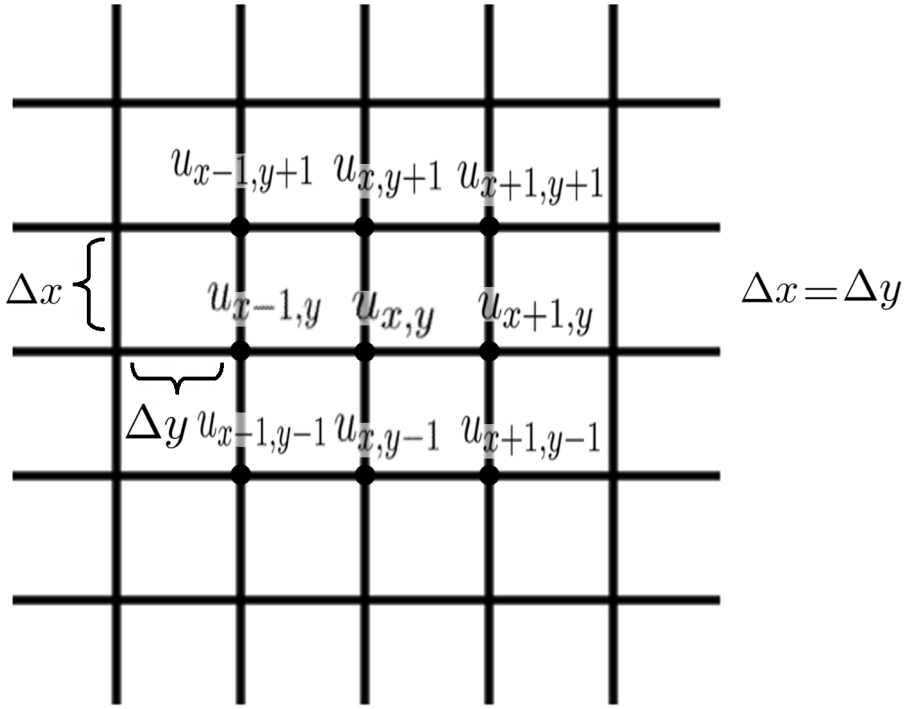


Figure 2: FDTD 2D rectilinear grid. The wave is evaluated at the grid points.

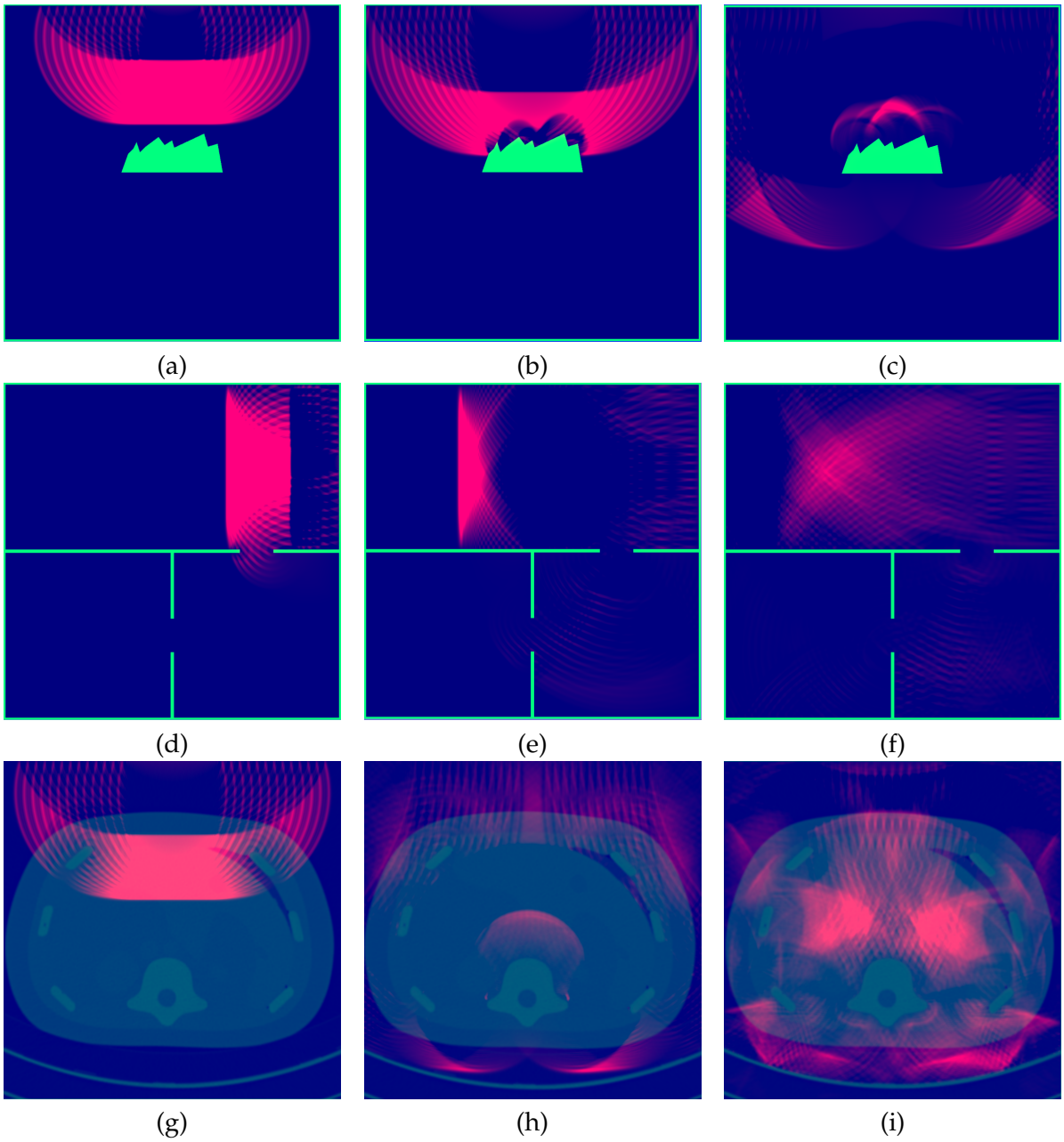


Figure 3: DW-Mesh simulations. *First row:* Images (a-c) show an initial wavefront being reflected at a diffuse reflector. The subsequent echo responses are presented at increasing timesteps in images (b-c). *Second row:* Images (d-f) show the initial wavefront propagating in a room model (as initially demonstrated in the publication of Röber et al. [44]). Diffraction effects are clearly visible at the opening between the rooms. *Third row:* Images (g-i) show an initial wavefront inside a manually segmented CT phantom dataset model. Only the bones in the dataset are classified as reflectors. The simulations are observed in increasing timesteps.

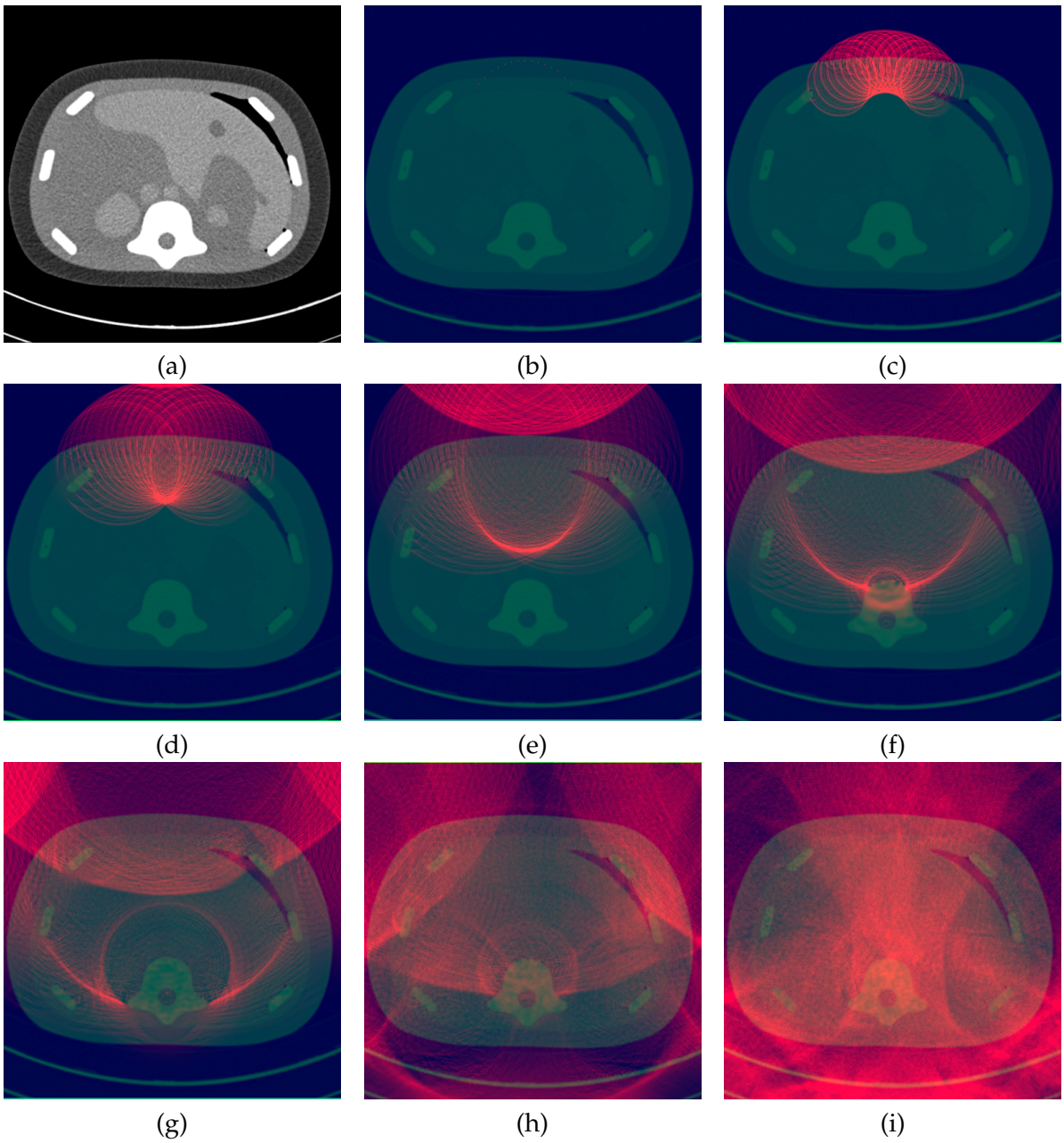


Figure 4: 2D FDTD Westervelt simulation using a manually segmented phantom CT dataset with a focused wavefront propagation.

Table 3: The following tables present the comparisons between the Hybrid - Quad Split and the Hybrid - Barycentric method with the other available methods. For each comparison the Mean Absolute Differences (MAD) and the Mean Square Error (MSE) are presented.

Quad Comparison

	MAD	MSE
Nearest	8.16	219.87
Maximum	12.68	408.88
Trajectory	7.66	211.05
Inverse	7.43	194.80
Gaussian	7.68	202.02
Weighted	8.07	212.80

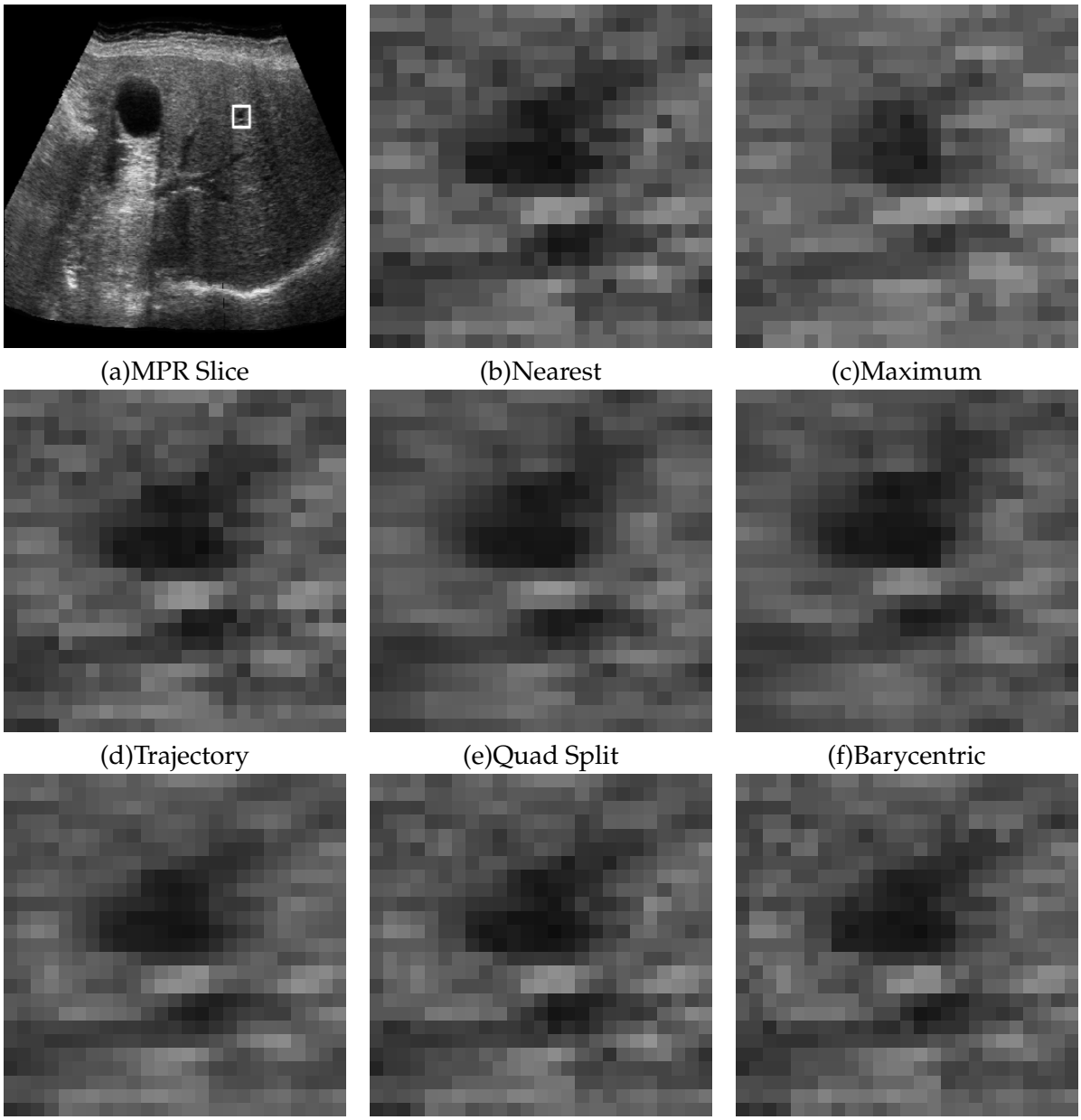
Barycentric Comparison

	MAD	MSE
Nearest	10.39	306.53
Maximum	13.83	472.80
Trajectory	9.87	291.31
Inverse	9.72	277.66
Gaussian	9.61	274.38
Weighted	10.11	291.09

$$MAD = \frac{1}{M}\sum_{i=1}^{M}|p_i - u_i| \tag{2}$$

$$MSE = \frac{1}{M}\sum_{i=1}^{M}(p_i - u_i)^2 \tag{3}$$

where p_i is the MPR slice pixel intensity using the hybrid method, u_i is the MPR slice pixel intensity from the other test methods and M is the total number of image pixels. The size of the images in the tests were 512×392.

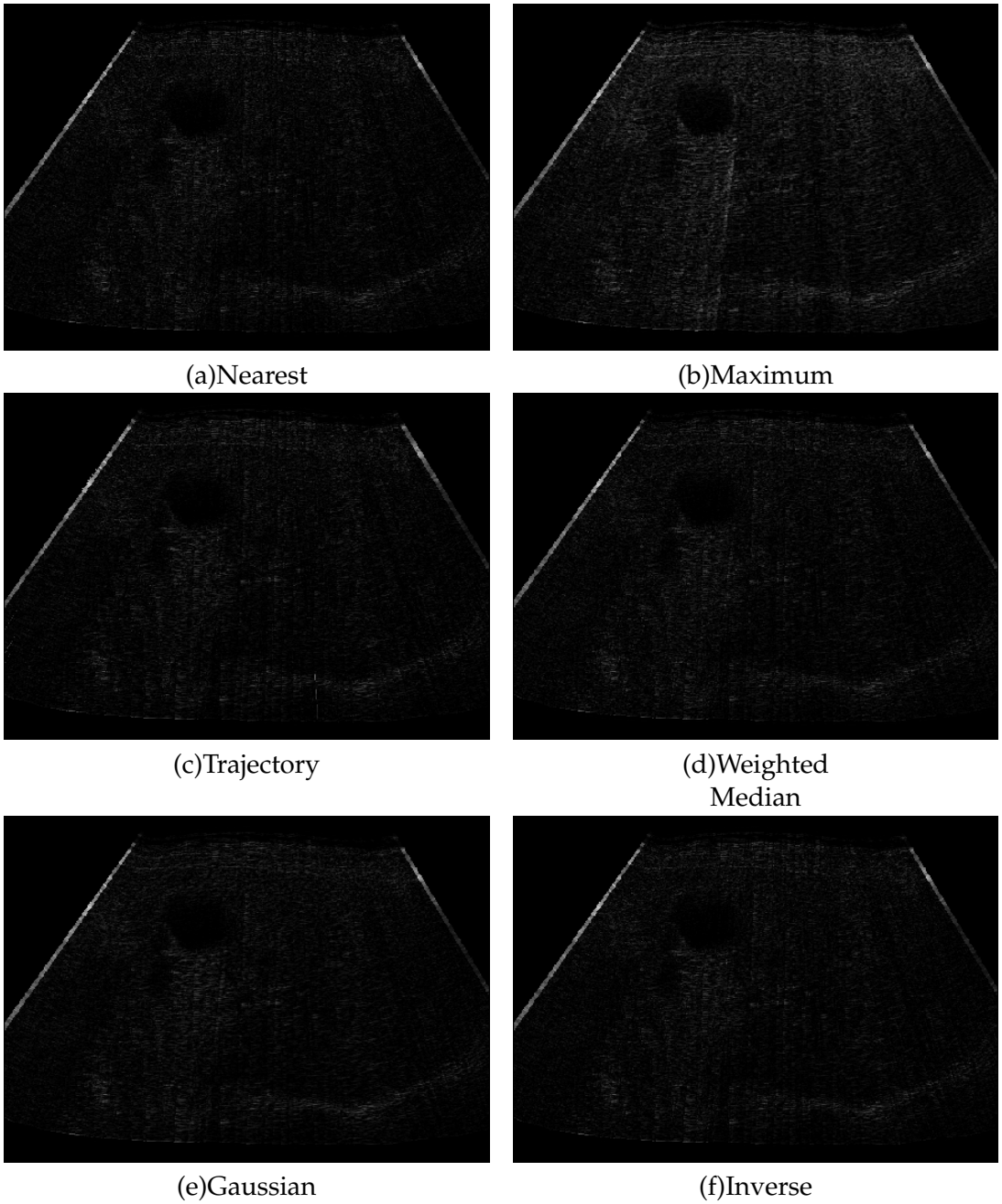


(a)MPR Slice (b)Nearest (c)Maximum

(d)Trajectory (e)Quad Split (f)Barycentric

(g)Gaussian (h)Inverse (i)Weighted Medien

Figure 5: Image (a) shows a reconstructed MPR slice with a region of interest (ROI) marked with a white square. The same MPR slice has been reconstructed using all available methods and the zoomed ROI for all methods is presented in images (b) - (i).

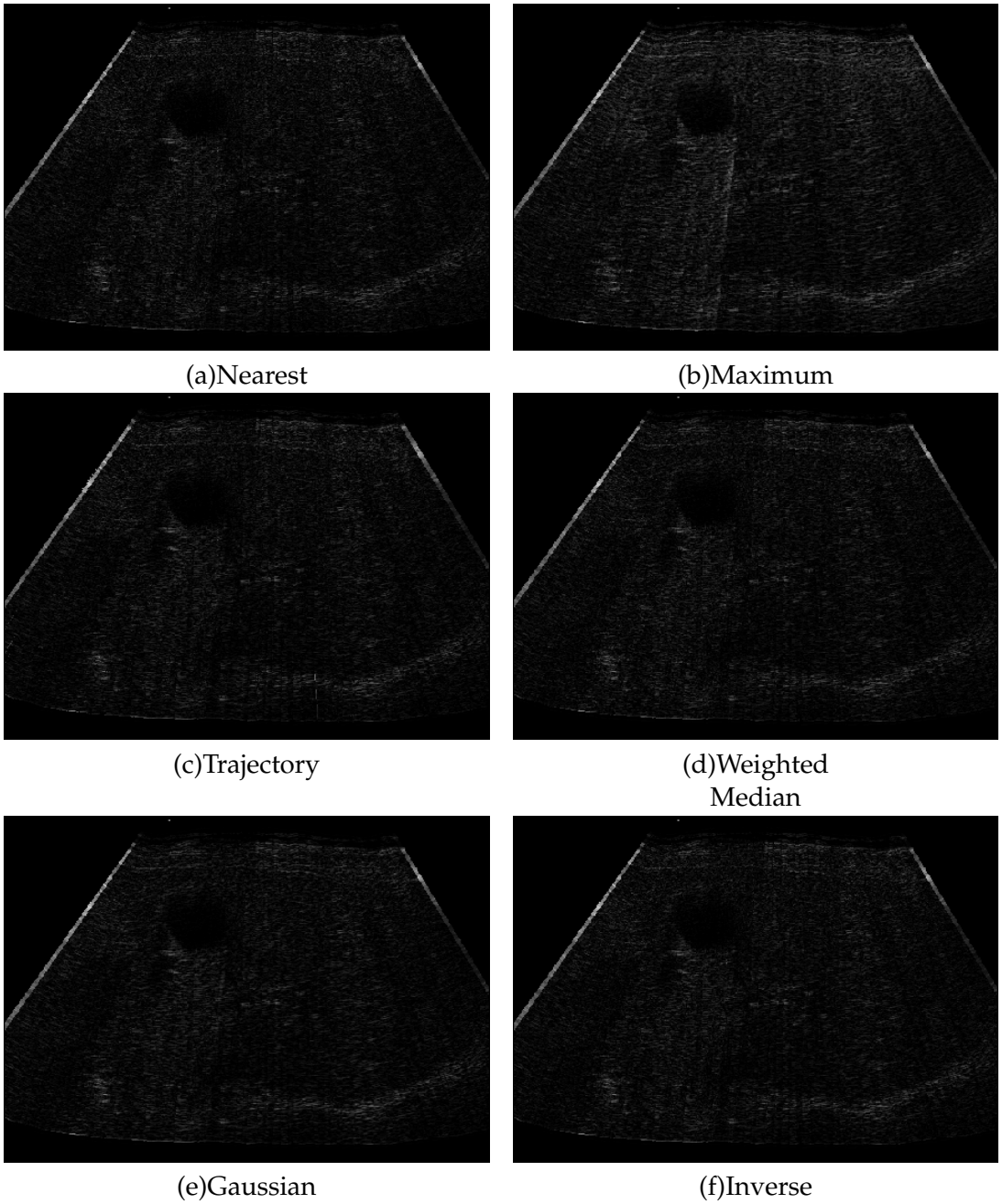


(a)Nearest (b)Maximum

(c)Trajectory (d)Weighted Median

(e)Gaussian (f)Inverse

Figure 6: The images (a)-(f) show the absolute differences between an MPR slice reconstructed with the Hybrid - Quad Split Method and the same MPR slice reconstructed with the other methods. The intensity range is [0..255] which is common for ultrasound imaging.



(a)Nearest (b)Maximum

(c)Trajectory (d)Weighted Median

(e)Gaussian (f)Inverse

Figure 7: The images (a)-(f) show the absolute differences between an MPR slice reconstructed with the Hybrid - Barycentric Method and the same MPR slice reconstructed with the other methods. The intensity range is [0..255] which is common for ultrasound imaging.

Abbrevations

ABC:	Absorbing Boundary Condition
CPU:	Central Processing Unit
CT:	Computed Tomography
CTA:	Computed Tomography Angiography
DW:	Distance-Weighted
DW-Mesh:	Digital Waveguide Mesh
FBO:	Framebuffer Object
FDTD:	Finite-Difference Time-Domain
GLSL:	OpenGL Shading Language
GPGPU:	General-Purpose computing on Graphics Processing Units
GPU:	Graphics Processing Unit
MAD:	Mean of Absolute Differences
MPR:	Multi Planar Reformatting
MRI:	Magnetic Resonance Imaging
PDE:	Partial Differential Equation
PML:	Perfectly Matched Layer
PNN:	Pixel Nearest-Neighbor
RAM:	Random Access Memory
RBF:	Radial Basis Function
RGB:	Red Green Blue (3-channel texture on graphic hardware)
ROI:	Region Of Interest
UPML:	Uniaxial Perfectly Matched Layer
VNN:	Voxel Nearest-Neighbor

Bibliography

[1] General-purpose computation using graphics hardware - community web site. http://www.gpgpu.org.

[2] Nvidia developer home page. http://developer.nvidia.com/.

[3] OpenGL framebuffer object extension. http://www.opengl.org/registry/.

[4] ADAMS, S., PAYNE, J., AND BOPPANA, R. Finite difference time domain (FDTD) simulations using graphics processors. In *IEEE HPCMP Users Group Conference* (2007).

[5] BARON, G. S., SARRIS, C. D., AND FIUME, E. Fast and accurate time-domain simulations with commodity graphics hardware. In *IEEE Antennas and Propagation Society International Symposium* (2005).

[6] BERENGER, J.-P. A perfectly matched layer for the absorption of electromagnetic waves. *J. Comput. Phys. 114*, 2 (1994), 185–200.

[7] CORREAS, J.-M., BRIDALM, L., LESAVRE, A., MÉJEAN, A., CLAUDON, M., AND HÉLÉNON, O. Ultrasound contrast agents: properties, principles of action, tolerance, and artifacts. *European Radiology 11* (2001), 1316–1328.

[8] COUPÈ, P., HELLIER, P., AZZABOU, N., AND BARILLOT, C. 3D freehand ultrasound reconstruction based on probe trajectory. In *Lecture Notes in Computer Science* (2005), vol. 3749, pp. 597–604.

[9] DOWNEY, D. B., FENSTER, A., AND WILLIAMS, J. C. Clinical utility of three-dimensional US. *Radiographics 20* (2000), 559–571.

[10] DUYNE, S. A. V., AND SMITH, J. O. The 2-D digital waveguide mesh. In *Proc. IEEE Workshop on App. of Sig. Proc. to Audio and Acoust.* (1993).

[11] DUYNE, S. V., AND SMITH, J. The tetrahedral digital waveguide mesh. In *IEEE Applications of Signal Processing to Audio and Acoustics* (1995).

[12] FERNANDO, R., Ed. *GPU Gems*. Addison-Wesley Professional, 2004.

[13] FISH, P. *Physics and Instrumentation of Diagnostic Medical Ultrasound*. John Wiley and Sons Ltd, 1996.

[14] FRINKING, P. J., BOUAKAZ, A., KIRKHORN, J., CATE, F. J. T., AND JONG, N. D. Ultrasound contrast imaging : Current and new potential methods. *Ultrasound in Med. & Biol. 26* (2000).

[15] GEDNEY, S. An anisotropic perfectly matched layer-absorbing medium for the truncation of FDTD lattices. In *IEEE Transactions on Antennas and Propagation* (1996), vol. 44, pp. 1630–1639.

[16] HALLAJ, I. M., AND CLEVELAND, R. O. FDTD simulation of finite-amplitude pressure and temperature fields for biomedical ultrasound. *The Journal of the Acoustical Society of America 105* (1999), L7–L12.

[17] HEDRICK, W. R., AND HYKES, D. L. Image formation in real-time ultrasound. *Journal of Diagnostic Medical Sonography 11* (1995), 246–251.

[18] HORMANN, K., AND TARINI, M. A quadrilateral rendering primitive. In *HWWS '04: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware* (New York, NY, USA, 2004), ACM, pp. 7–14.

[19] HOSTETTLER, A., FOREST, C., FORGIONE, A., SOLER, L., AND MARESCAUX, J. Real-time ultrasonography simulator based on 3D CT-scan images. *Medicine Meets Virtual Reality 13: The Magical Next Becomes the Medical Now 2005* (2005), 191–193.

[20] HUIJSSEN, J., BOUAKAZ, A., VERWEIJ, M., AND DE JONG, N. Simulations of the nonlinear acoustic pressure field without using the parabolic approximation. In *IEEE Symposium on Ultrasonics* (2003).

[21] HYNYNEN, K., POMEROY, O., SMITH, D. N., HUBER, P. E., MCDANNOLD, N. J., KETTENBACH, J., BAUM, J., AND ANDFERENC A. JOLESZ, S. S. MR imaging-guided focused ultrasound surgery of fibroadenomas in the breast: A feasibility study. *Radiology 219* (2001), 176–185.

[22] JARGSTORFF, F. *GPU Gems*. Addison-Wesley Professional, 2004, ch. 27- A Framework for Image Processing.

[23] JENSEN, J. Simulation of advanced ultrasound systems using Field II. In *IEEE International Symposium on Biomedical Imaging: Nano to Macro* (2004), vol. 1, pp. 636– 639.

[24] JENSEN, J., FOX, P., WILHJELM, J., AND TAYLOR, L. Simulation of non-linear ultrasound fields. In *IEEE Ultrasonics Symposium* (2002).

[25] JENSEN, J., AND NIKOLOV, I. Fast simulation of ultrasound images. *IEEE Ultrasonics Symposium 2* (2000), 1721–1724.

[26] JENSEN, J. A. Field: A program for simulating ultrasound systems. In *10th NordicBaltic Conference on Biomedical Imaging* (1996), pp. 351–353.

[27] JENSEN, J. A., AND MUNK, P. Computer phantoms for simulating ultrasound B-mode and CFM images. In *Acoustical Imaging* (1997), pp. 75–80.

[28] KARAMALIS, A., KUTTER, O., WEIN, W., AND NAVAB, N. Fast hybrid freehand ultrasound volume reconstruction. In *SPIE Medical Imaging* (2009).

[29] KELLONIEMI, A., MURPHY, D. T., SAVIOJA, L., AND VÄLIMÄKI, V. Boundary conditions in a multi-dimensional digital waveguide mesh. In *Acoustics, Speech, and Signal Processing (ICASSP04)* (2004).

[30] KRÜGER, J., AND WESTERMANN, R. Acceleration techniques for GPU-based volume rendering. In *Proceedings of the 14th IEEE Visualization (VIS'03)* (2003).

[31] KUTTER, O., KARAMALIS, A., WEIN, W., AND NAVAB, N. A GPU-based framework for simulation of medical ultrasound. In *SPIE Medical Imaging* (2009).

[32] LINDHOLM, E., NICKOLLS, J., OBERMAN, S., AND MONTRYM, J. Nvidia Tesla: A unified graphics and computing architecture. In *IEEE Mirco* (2008), pp. 39–55.

[33] LUEBKE, D., HARRIS, M., KRÜGER, J., PURCELL, T., GOVINDARAJU, N., BUCK, I., WOOLLEY, C., AND LEFOHN, A. GPGPU: general purpose computation on graphics hardware. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Course Notes*. ACM, New York, NY, USA, 2004, p. 33.

[34] MAGEE, D., ZHU, Y., RATNALINGAM, R., GARDNER, P., AND KESSEL, D. An augmented reality simulator for ultrasound guided needle placement training. *Medical and Biological Engineering and Computing 45* (2007), 957–967.

[35] MEYER, M., LEE, H., BARR, A., AND DESBRUN, M. Generalized barycentric coordinates on irregular polygons. *Journal of graphics tools 7*, 1 (2002), 13–22.

[36] MURPHY, D. T., AND MULLEN, J. Digital waveguide mesh modelling of room acoustics: Improved anechoic boundaries. In *Proc. of the 5th Int. Conference on Digital Audio Effects (DAFX-02)* (2002).

[37] NELSON, T. R., AND PRETORIUS, D. H. Three-dimensional ultrasound imaging. *Ultrasound in Med. & Biol 24* (1998).

[38] NGUYEN, H., Ed. *GPU Gems 3*. Addison-Wesley Professional, 2007.

[39] NVIDIA. Nvidia CUDA compute unified device architecture. Tech. rep., 2008.

[40] OPENGL ARCHITECTURE REVIEW BOARD, SHREINER, D., WOO, M., NEIDER, J., AND DAVIS, T. *OpenGL Programming Guide: The Official Guide to Learning OpenGL*. Addison-Wesley Professional, 2005.

[41] PERLIN, K. *GPU Gems*. Addison-Wesley Professional, 2004, ch. 5 - Implementing Improved Perlin Noise.

[42] PHARR, M., AND FERNANDO, R., Eds. *GPU Gems 2*. Addison-Wesley Professional, 2005.

[43] PINTON, G. F., AND TRAHEY, G. E. A comparison of time-domain solutions for the full-wave equation and the parabolic wave equation for a diagnostic ultrasound transducer. *IEEE Transactions on Ultrasonics, Ferroelectrics and frequency control 55* (2008), 730–733.

[44] RÖBER, N., SPINDLER, M., AND MASUCH, M. Waveguide-based room acoustics through graphics hardware. In *Proceedings of ICMC06* (2006).

[45] ROHLING, R., GEE, A., AND BERMAN, L. A comparison of freehand three-dimensional ultrasound reconstruction techniques. *Medical Image Analysis 3* (1999), 339–359.

[46] ROST, R. J. *OpenGL Shading Language*, 2nd ed. Addison-Wesley Professional, 2006.

[47] SAVIOJA, L., AND VÄLIMÄKI, V. Improved discrete-time modeling of multi-dimensional wave propagation using the interpolated digital waveguide mesh. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing (ICASSP97)* (1997), vol. 1, pp. 459–462.

[48] SAVIOJA, L., AND VÄLIMÄKI, V. Reducing the dispersion error in the digital waveguide mesh using interpolation and frequency -warping techniques. *IEEE Transactions on Speech and Audio Processing 8* (2000), 184–194.

[49] SCHARSACH, H. Advanced GPU raycasting. Tech. rep., VRVis Research Center, Vienna, Austria, 2005.

[50] SHAMS, R., HARTLEY, R., AND NAVAB, N. Real-time simulation of medical ultrasound from CT images. In *Medical Image Computing and Computer Assisted Intervention* (2008).

[51] SOLER, L., AND MARESCAUX, J. Patient-specific surgical simulation. *World Journal of Surgery 32* (2007), 208–212.

[52] TAFLOVE, A., AND HAGNESS, S. C. *Computational Electrodynamics: The Finite-Difference Time-Domain Method*, 3rd ed. Artech House Publishers, 2005.

[53] VIDAL, F. P., HEALEY, A. E., GOULD, D. A., AND JOHN, N. W. Simulation of ultrasound guided needle puncture using patient specific data with 3D textures and volume haptics. *Computer Animation and Virtual Worlds 19* (2008), 111–127(17).

[54] WEIN, W., BRUNKE, S., KHAMENE, A., CALLSTROM, M. R., AND NAVAB, N. Automatic CT-ultrasound registration for diagnostic imaging and image-guided intervention. *Medical Image Analysis 12* (2008), 577–585.

[55] WEIN, W., PACHE, F., ROEPER, B., AND NAVAB, N. Backward-warping ultrasound reconstruction for improving diagnostic value and registration. In *Lecture Notes in Computer Science* (2006), vol. 4191, pp. 750–757.

[56] YEE, K. Numerical solution of inital boundary value problems involving Maxwell's equations in isotropic media. *IEEE Transactions on Antennas and Propagation 14* (1966), 302 – 307.

[57] YONGCHEN, S., YANWU, D., JIE, T., AND ZHENSHENG, T. Ultrasonic propagation parameters in human tissues. In *IEEE Ultrasonics Symposium* (1986), pp. 905 – 908.

[58] ZAGZEBSKI, J. A. *Essentials of Ultrasound Physics*. Mosby Inc, 1996.

[59] ZIOLKOWSKI, R. W. FDTD absorbing boundary conditions: From global lookback schemes to metamaterial PMLs. In *IEEE Antennas and Propagation Society International Symposium* (2008).

List of Figures