GPU Ultrasound Simulation and Volume Reconstruction

Athanasios Karamalis
Supervisor: Nassir Navab
Advisor: Oliver Kutter, Wolfgang Wein

Computer Aided Medical Procedures (CAMP), Technische Universität München, Germany

Siemens Corporate Research (SCR), Princeton, USA
Outline

• (3D) Freehand ultrasound volume reconstruction / MPR generation

• Ray-based ultrasound simulation

• Wave-based ultrasound simulation
What is 3D freehand ultrasound?

- 1D Transducer tracked (optic or electromagnetic)
- Transducer position and orientation is arbitrary
- Multiple 2D US images recorded together with spatial information

Multiple US Image Acquisitions
Motivation

• **MPR Generation**
  – Additional clinical value
  – Multi-modal registration
  – Image-based calibration

• **Volume Reconstruction**
  – Make sweep available to existing clinical tools, including: volume visualization, segmentation, registration, etc.
  – Multi-modal registration
  – Improved compatibility (e.g. PACS)
Volume Reconstruction

- Reconstruct Scattered 2D images (slices) into regular 2D/3D grid
- Data interpolation (between slices)
  - Limited assumptions about the acquisition can be made

*US Sweep, MPR (green)*
Available methods

- Forward reconstruction (Pixel-based) – Fast, Low quality
- Backward reconstruction (Voxel-based) – Slow, High quality
  - Distance-weighted, Probe Trajectory weighted
- Advanced methods
  - Radial Basis Function Interpolation (RBF) – Very Slow, High quality

MPR (without filling gaps)

MPR (interpolating between slices)
Goals

- Fast volume reconstruction, On-the-fly MPR
- Develop new approach suitable for GPU implementation
  - Hardware accelerated interpolation
- At least same visual quality as existing, slower methods
  - Focusing at Distant-Weighted Methods

Approach:
Fast MPR reconstruction on GPU to reconstruct entire Volume
Method – 2D US Image representation

• Using point coordinates of each ultrasound image

\[ p_0, p_1, p_2, p_3 \in \mathbb{R}^3 \]

• Define line for each edge

\[ p = p_n + u \cdot \vec{v} \]

Sweep edge lines
Method – MPR Representation

- Each MPRs is represented by a sampling plane: $\vec{n} \cdot (x - x_0) = 0$
- Multiple MPRs/Planes for volume reconstruction
  - One MPR for each volume layer

Single MPR/Plane  3D Volume  Multiple MPRs/Planes
Method – MPR Reconstruction

• Find all intersections between image edge lines and sampling plane(s)
• For each image pair
  – Use four points to form a connecting quadrilateral
  – Bilinear interpolate at quadrilateral edges along the images
  – Linear interpolate the values at the edges along the quadrilateral
• MPR from multiple quadrilaterals
Method – Interpolation Problem

- GPU hardware does not support continuous interpolation of irregular quadrilaterals
- Solution
  - Split quadrilateral
  - Or generalized barycentric coordinates

[Image of quadrilateral split with comparison to default and barycentric methods]

[Reference: Horman and Tarini]
Results - Performance

- **System:** 8800 GTX 768 MB - Xeon 3.2GHz – 2 GB RAM
- **Reconstructing 293 slices in a 256^3 volume**

<table>
<thead>
<tr>
<th>GPU Reconstruction</th>
<th>Backward Trajectory</th>
<th>Backward Maximum</th>
<th>Backward Gaussian</th>
<th>Forward (non optimized)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.35s</td>
<td>72.72s</td>
<td>60.80s</td>
<td>69.78s</td>
<td>471.88s</td>
</tr>
<tr>
<td>Gain</td>
<td>208x</td>
<td>174x</td>
<td>199x</td>
<td>1348x</td>
</tr>
</tbody>
</table>

* *Barycentric 0.82s*

- **MPR of 293 slices**
  - GPU: 124 FPS
  - CPU: <= 1FPS
Results - Quality

![Images of MPR, Maximum, Trajectory, Quad Split]

<table>
<thead>
<tr>
<th>Method</th>
<th>Nearest</th>
<th>Maximum</th>
<th>Trajectory</th>
<th>Inverse</th>
<th>Gaussian</th>
<th>Weighted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quad Split</td>
<td>8.16</td>
<td>12.68</td>
<td>7.66</td>
<td>7.43</td>
<td>7.68</td>
<td>8.07</td>
</tr>
<tr>
<td>Barycentric</td>
<td>10.38</td>
<td>13.83</td>
<td>9.87</td>
<td>9.72</td>
<td>9.61</td>
<td>10.11</td>
</tr>
</tbody>
</table>

Mean of Absolute Differences, Pixel Intensities [0..255]
Future Work

• Make approach available for discontinuous sweeps (continuous sweeps most common case in our clinical scenario)

• Averaging of coincident images
  – Does not always yield better reconstruction results

• MPR planes parallel to US images are sampled insufficiently
  – Detect case and use alternative method. For example, projecting adjacent slices onto MPR
Outline

• (3D) Freehand ultrasound volume reconstruction / MPR generation

• Ray-based ultrasound simulation

• Wave-based ultrasound simulation
What is it about?

• Simulation of medical ultrasound images from CT datasets
• GPU accelerated, Real-time simulation
• Model sound physics with rays
  – Rays: Faster simulation, decreased realism
  – Waves: Considerably slower simulation, increased realism
Motivation

• 2D/3D medical ultrasound training
  – Experience through training
  – CT datasets commonly available in clinical setups
    – Variety of pathological cases available
  – Requires: Real-time simulation and visualization

• Accelerating multi-modal image registration
  – Registration of 2D US with CT
  – Fast ultrasound simulation beneficial (might improve accuracy)

• Future: Fast 3D-3D deformable multi-modal CT-US registration
  – Requires: Fast simulation
Method-Basics

- One ray for each transducer element
- The CT volume is sampled at equidistant points on the rays
- Samples used to calculate
  - reflection coefficient
    \[ I_r^k = I_i^k \frac{(Z_2 - Z_1)^2}{(Z_1 + Z_2)^2} \]
  - transmission coefficient
    \[ I_t^k = I_i^k \frac{4Z_1Z_2}{(Z_1 + Z_2)^2} \]
Method – Scanline Computation

• Initially, method seems similar to ray-casting, however:
  – Ray-casting returns only a single value and
  – Performs only a single render pass

• Scanline computation:
  – Returns multiple values along the rays
  – Performs recursive calculation of reflection and transmission

\[
I^k_i = \begin{cases} 
I^k_{i-1} & : k > 0 \\
1 & : k = 0
\end{cases}
\]

• Solution: Adjust initial ray-casting algorithm, introduce multiple render passes
Method - Scanline computation

Scanline Texture Layout and computation
Method – Scan Conversion

- From polar to Cartesian coordinate system
- Based on transducer geometry
- Interpolate between samples and scanlines
Method – Multiple Image Simulation

- Method extentable for simulating multiple images
- Number of multiple images limited by maximal texture size (currently 8192x8192)

Multiple Reflection Images
Results - Performance

• Higher batch size → Higher throughput

• Less GPU hardware overhead

• Real-time 3D ultrasound simulation (128 images x 30 FPS = 3840 Images/sec)
Results - Images

- Additional real-time effects (single 2D image simulation)
  - Gaussian blurring
  - Horizontal Hanning Window
  - On-the-fly 3D Perlin Noise

*Wein et al. model*  
*Additional Effects*
Future Work

• Additional effects, like
  – Multiple echoes (ray-tracing)
  – Tissue absorption (Look-Up-Tables)
  – Refraction

• Integrate framework into multi-modal registration pipeline

• Improve speckle simulation to add more realism
Outline

• (3D) Freehand ultrasound volume reconstruction / MPR generation

• Ray-based ultrasound simulation

• Wave-based ultrasound simulation
Motivation

• Ray-based models are rough approximations

• Wave-based models are more realistic, modeling:
  – Diffraction, Refraction, Interference and Scattering effects

• Current wave-based implementations show good results
  – Signal response simulation and ultrasound image simulation
  – Transducer and ultrasound system design

• But, simulations can take up to hours even on PC clusters, for example Field II simulation program
Goals

• Investigate GPU „friendly“ approaches
  – Lack of literature on GPU accelerated wave-based ultrasound simulation
  – Digital waveguide meshes
  – Finite-difference time-domain method

• Demonstrate feasibility of near real-time wave-based ultrasound simulation

Digital Waveguide Mesh Unsuitable for Ultrasound Simulation
Outline

• (3D) Freehand ultrasound volume reconstruction / MPR generation

• Ray-based ultrasound simulation

• Wave-based ultrasound simulation
  – Digital Waveguide Mesh
  – Finite-Difference Time-Domain Method
Digital Waveguide Mesh

- Introduced for room acoustics and music instrument simulation
- Low computational demands and memory requirements

\[ p_c(t) = \frac{1}{N} \sum_{l=1}^{2N} p_l(t - 1) - p_c(t - 2) \]

[duyne1993]

Waveguide

Scattering Junctions
Results

Room acoustics

Segemented Phantom Dataset

Avg. 1130
Time-steps/sec
512x512 grid
Issues and Problems

- Dispersion error in the grid
- Boundaries are defined explicitly
- For each boundary additional errors are introduced
- Reflections at computational grid edges
  - Lack of sufficient absorbing boundary conditions (ABC)
- Considerable improvements have been suggested

**However:** Errors are still high! Especially for low reflection coefficients.
Approach unsuitable for Ultrasound Simulation

• For CT or MRI dataset
  – Numerous tissue interfaces
  – All tissue interfaces must be defined
  – Each interface must be defined as a boundary junction

• Therefore, numerous error sources are introduced

• Accumulation of errors during simulation

• Lack of sufficient ABCs further restricts approach
Outline

• (3D) Freehand ultrasound volume reconstruction / MPR generation

• Ray-based ultrasound simulation

• Wave-based ultrasound simulation
  – Digital Waveguide Mesh
  – Finite-Difference Time-Domain Method
Finite-Difference Time-Domain (FDTD) Ultrasound Simulation

• FDTD method introduced in computational electrodynamics, decades ago

• Rich literature, including ABC modeling
  – Perfectly Matched Layers (PML)

• Suitable for GPU acceleration

• Proven suitable for ultrasound simulation
FDTD Basics

- Solve Partial Differential Equation (PDE) by:
  - Discretizing simulation domain
  - Numerically approximating partial derivatives with finite differences
  - Solving for term of interest
- Example 1D wave equation

\[
\frac{\partial^2 u(x, t)}{\partial x^2} = \frac{1}{c^2} \frac{\partial^2 u(x, t)}{\partial t^2}
\]

\[
\frac{\partial^2 u(x, t)}{\partial x^2} \approx \frac{u(x + \Delta x, t) - 2u(x, t) + u(x - \Delta x, t)}{\Delta x^2}
\]

\[
\frac{\partial^2 u(x, t)}{\partial t^2} \approx \frac{u(x, t + \Delta t) - 2u(x, t) + u(x, t - \Delta t)}{\Delta t^2}
\]
GPU FDTD Solver

- Each grid point corresponds to one texel of a 2D/3D texture
- Textures are used for the current computations and to store the previous results
- For our example, the wave amplitude for the next time-step is:

\[
u(x, t + \Delta t) = c^2 \frac{\Delta t^2}{\Delta x^2} (u(x + \Delta x, t) - 2u(x, t) + u(x - \Delta x, t)) + 2u(x, t) - u(x, t - \Delta t)\]

- Only the two previous timesteps are needed
- Calculations are performed in a GPU fragment shader
Initial Results

- Single Point source
- Two mediums with propagation speeds of 1200 and 1600 [m/s]
- Difference in propagation speed causes reflection
- Grid size 512x512

GPU FDTD Solver
2D Lossless Wave Equation
Initial Results

- Multiple sources possible
- Performance independent of number of sources
- Different wavefront / focusing schemes can be simulated

*Linear Wavefront*  
*Focused Wavefront*
Westervelt Equation

• Full nonlinear wave equation for acoustic simulation

• Applications in Ultrasound Simulation
  – Signal response simulation
  – Harmonic Frequency simulation
  – Temperature field simulation

• First results indicate that simulated predictions
  – Coincidence with real measurements from water tanks
  – Are equivalent, and sometimes, superior to other methods
The equation itself

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

- First two terms identical to lossless wave equation
- Third term, attenuation
- Fourth term, nonlinearity
- Terms:

  - \( p \) : Pressure
  - \( c_0 \) : Propagation speed
  - \( \rho_0 \) : Ambient density
  - \( \delta \) : Diffusivity of sound
  - \( \beta \) : Coefficient of nonlinearity
GPU Implementation of FDTD-Westervelt Method

- Substitute partial derivatives with finite difference
- Solve for next time-step and use formulation in GPU fragment shader
- The previous 6 time-steps are required, thus, 6 textures are required
- Utilizing textures for time-steps and coefficient specification
  - All coefficients are set constant, except propagation speed
- Texture queue used to reduce GPU overhead

Texture Queue for GPU FDTD-Westervelt
Results

• Manually segmented CT phantom dataset
• Propagation speed varying for different pixels
• Nonlinear term omitted
• 512x512 grid
• Avg. 1078 Time-Steps/sec
Future Work

- FDTD-Westervelt method promising for:
  - Real-time simulation
  - Accurate predictions
- Absorbing Boundary Conditions required (CPML)
- Segmentation/labeling of CT or MRI datasets
- Test different excitation pulses (currently Dirac)
- Different focusing strategies
- Post processing pipeline to transform pressure signal into B-scan ultrasound image
Distant Future

- Therapeutic applications using High Intensity Focused Ultrasound (HIFU)
- MRI guided ultrasound surgery already demonstrated potential
- First temperature field simulations for thermoviscous fluids showed potential
- Simulating focusing regions and defining safety regions would be extremely valuable

Focused US Temperature field
[Hallaj and Cleveland]
THANK YOU FOR YOUR ATTENTION!

Special Thanks to:
• Wolfgang Wein
• Oliver Kutter
• Christoph Vetter
• Sylvain Jaume
• And a lot of other people from SCR and CAMP

akaramal@mytum.de