Summer Research

Ting Zhou Jr.

August 14, 2015

Objective and Significance of the Project

Simply put, the project is to begin with a 3D Computed Tomography (CT) volume and from that generate a realistic, simulated 3D ultrasound (US). Converting CT, a high-resolution imaging technique, to US, a low-resolution, noisy imaging technique, may seem counter-intuitive; however, algorithms for analyzing US images for applications such as organ segmentation and tissue tracking require the ground truth provided by CT. Simulating US from CT and having both allows for significantly simplified validation of these algorithms.

Mapping out the Project

The project began with a reading of the Shams, Hartley, and Navab paper, as well as learning about how ultrasound works, primarily a study of the physics of ultrasound. Shams et al. (2008) provided several necessary equations and discrete steps, but did not explain all of the methods used to go from one step to the next. This led us to explore the methods other papers used and proposed to simulate US from CT.

The research led to the conclusion that most algorithms had a binary approach to simulating US. On one side was the scattering image, generated with a software package such as Field II or Fusk 3D or done independently. On the other side was the reflection image, based on the tissue's echogenicity. A simple flowchart as well as examples of each side of the simulation, scattering and reflection, are shown below.

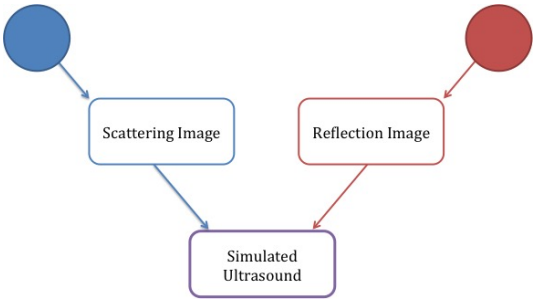

Figure 1: The Standard Method

1 Researching Algorithms

1.1 Shams, R., Hartley, R. & Navab, N.: Real-time simulation of medical ultrasound from CT images (2008)

Shams, Hartley, and Navab (2008) propose a novel method for simulating ultrasound from a CT volume. Their method is broken down into two parts, a pre-processing computation and a run-time computation.

Pre-Processing - Scattering Image using Field II, shown on the left half of the flowchart.

Run-Time - Reflection Image using Echogenicity, shown on the right half of the flowchart.

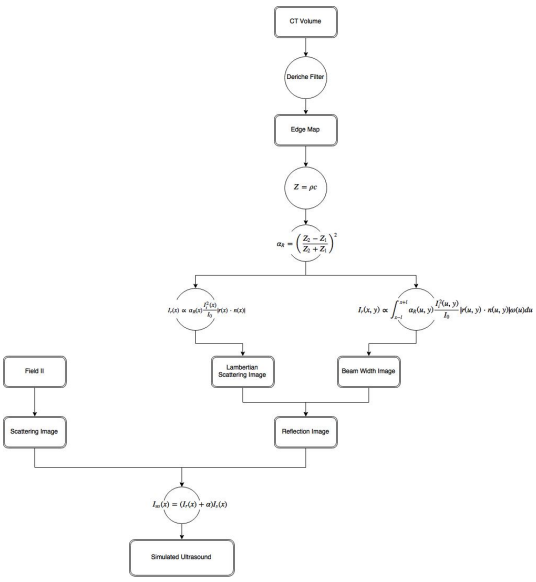

Figure 2: Shams, Hartley, Navab (2008) FlowChart

1.2 Wein, W., Brunke, S., Khamene, A., Callstrom, M. R. & Navab, N.: Automatic CT-ultrasound registration for diagnostic imaging and image-guided intervention (2007)

In this paper, Wein et al. (2007) worked both on simulation of Medical Ultrasound as well as Registration, with the former being directly relevant. They use the same linear acoustics models for calculating reflection and transmission as Shams et al. (2008), but their method of measuring intensity differs. They first derive an incremental acoustic intensity reflection $\Delta r(x, d)$, which in turn is used to integrate and calculate $I(x)$. This represents the Intensity at each depth along a scanline. To finalize the reflection image, a log-compression is applied to fine-tune smaller reflections, analogous to the Dynamic Range knob on ultrasound machines. Finally, scattering is added in the form of Perlin noise. In a similar manner as above, the flowchart is divided in half, with the left side showing scattering and the right side reflection.

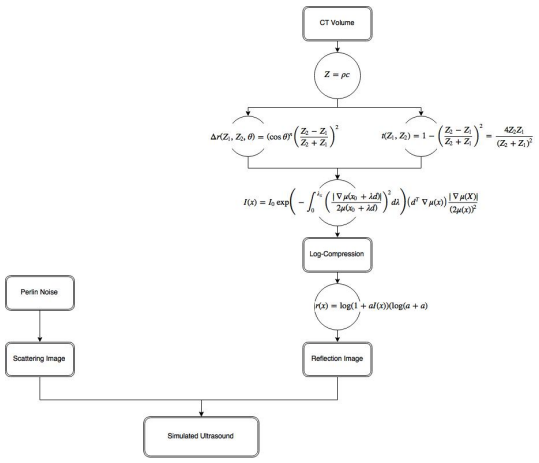

Figure 3: Wein et al. (2007) FlowChart

1.3 Schneider, U.: The calibration of CT Hounsfield units for radiotherapy treatment planning. (1996)

In the paper, Schneider (1996) starts with CT Hounsfield Units and provides resulting Densities and Velocities.

2 The Algorithm

2.1 Functions from File Exchange

imshow3D by Maysam Shahedi ⇒ Displays 3D Image Volumes slice by slice, which is useful for visualization.

bresenham_line3D by Jimmy Shen ⇒ Returns a vector of the voxels passed through as calculated by Bresenham's Line Drawing Algorithm, which is used while calculating attenuation

along each scan line.

2.2 Load in the 3D CT Volume

loadfile.m loads in the 3D CT file (.mat format) as an a x b x c matrix.

- Inputs
  - CT Volume (.mat file)
- Outputs
  - CT Volume (matrix)
  - Dimensions of matrix

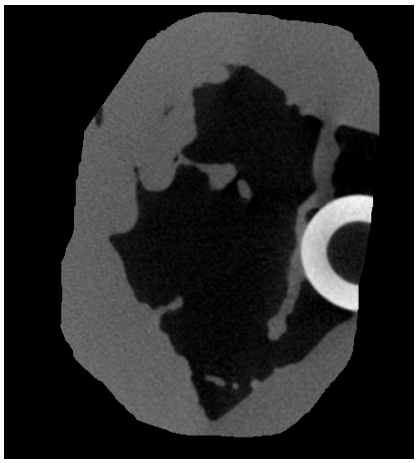

Figure 4: Slice of the original CT Image

2.3 Adjust the Data Values to Standard Hounsfield Units

redistribute.m converts the data to standard Hounsfield Units. The inputted .mat file created with Harvard's MicroCT has data values ranging from $[0, 1000]$, but the standard range is $[-1000, 1000]$, with the endpoints representing air and bone, respectively, and the midpoint representing water.

- Inputs
  - CT Volume
- Outputs
  - Adjusted CT Volume

2.4 Model the Relation between Hounsfield Units (HU) and Density

HU_density_model.m uses HU-Density values from Schneider et al. (1996) to model their relation. A 'smoothingspline' fit was used, and the model produced showed a linear piece-wise relation. Finally, the values were retrieved from the model.

- Inputs
  - Vector of Schneider HU Values
  - Vector of corresponding Schneider Density Values
- Outputs
  - Vector of fitted HU Values
  - Vector of corresponding fitted Density Values

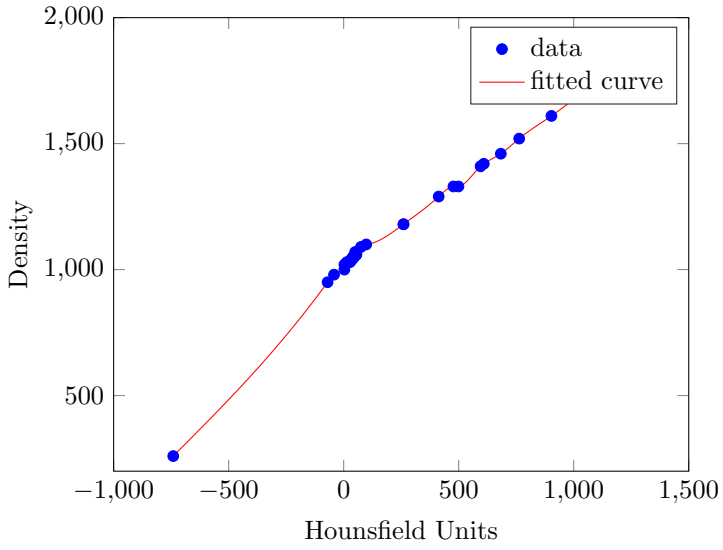

Figure 5: The Modeled Hounsfield-Density Relation

2.5 Convert HU Data to Density Data

interpolate_density.m interpolates each pixel's density based on its HU from the fitted model. In terms of the algorithm, knnsearch was used to find the two nearest neighbors, and since the model was assumed to be linear between every consecutive point the density could be interpolated from the HU.

- Inputs
  - Adjusted CT Volume

  - Density and HU model values
  - Dimensions of CT Volume
- Outputs
  - CT Volume in Density units

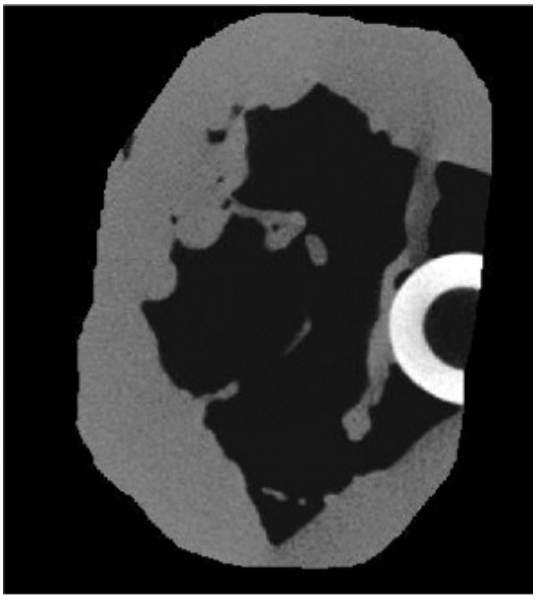

Figure 6: Slice of Density Map

2.6 Compute a Threshold Map

compute_thresholds.m computes a ternary threshold map based on the original CT data. This allows for simpler edge detection and attenuation computation. Image values between 0 and 250 were set to 0 (air), values between 250 and 750 were set to 500 (soft tissue), and values greater than 750 were set to 1000 (bone). If mapped to the density data, the respective ranges and threshold values between -1000 and -500 were set to -1000 (air), values between -500 and +500 were set to 0 (soft tissue), and values greater than +500 were set to +1000 (bone).

- Inputs
  - CT Volume
  - Threshold limits
  - Ternary values - air(0), soft tissue(500), and bone(1000)
- Outputs
  - 3D Threshold Map

Figure 7: Slice of Threshold Map

2.7 Compute an Edge Map

Use MATLAB's built-in edge() function to compute a logical (binary) edge map based off of the threshold map. These edges are the locations where reflection will take place.

- Inputs
  - 3D Threshold Map
  - Edge Detection Method
  - Sensitivity Threshold
- Outputs
  - 3D Logical Edge Map

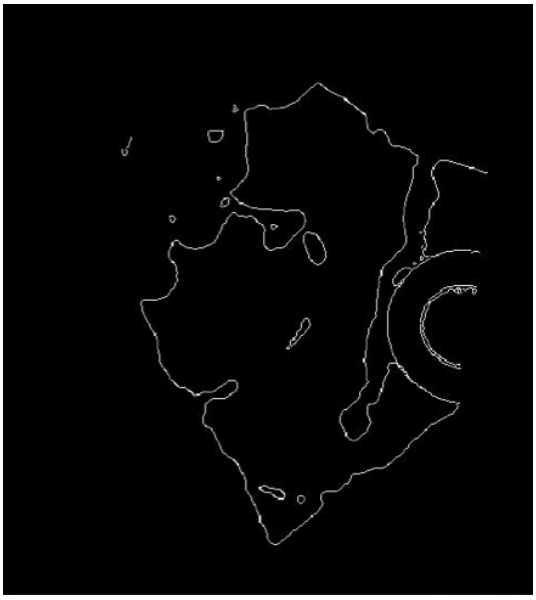

Figure 8: Slice of Edge Map

2.8 Compute the Normal Vector Map

To find the angle of incidence for the incident ultrasound beam, the normal vectors of the surface and the incident ray vectors are required. normal_vector_map.m calculates the gradient at each point along the edge map, returning unit and non-unit vectors normal to the surface.

- Inputs
  - 2D Threshold Map
  - 2D Edge Map
  - Dimensions of Slice
- Outputs
  - x and y components of normal vectors
  - x and y components of unit normal vectors

2.9 Compute the Ray Vector Maps for each Transducer

ray_vector_map.m calculates the vectors of the incident rays from an origin and is called once per transducer.

- Inputs
  - Transducer Coordinates
  - 2D Edge Map

- Dimensions of Slice

- Outputs
  - Vector length to each point in image
  - x and y components of ray vectors
  - x and y components of unit ray vectors

2.10 Compute the Angles of Incidence for each Point along the Edge

The reflection coefficient is dependent on the angle of incidence and the impedances of the initial and final medium. The angle of incidence is computed in incident_angle.m using the dot product of each point's ray and normal vectors and is called once per transducer.

- Inputs
  - x and y components of ray vectors
  - x and y components of normal vectors

- Outputs
  - Incident Angle Values in Degrees

2.11 Compute the Initial and Final Impedances for each Ray at each Point along the Edge

mean_impedances.m computes the impedances on either side of each edge by taking a $N \times N$ block centered around the edge point and finds the mean density of the values on either side of the edge. This is used because each image's edge is not a hard edge but rather a gradient. Taking a $N \times N$ block allows the computation to ignore this gradient. One note: This method is isotropic, as it does not attribute different impedances based on the direction but rather is set to one value for a specific point, regardless of ray propagation.

- Inputs
  - 2D Edge Map
  - Dimensions of Block
  - Dimensions of Slice
  - Threshold Map
  - Density Map
  - Ternary values - air, soft tissue, and bone

- Outputs
  - Mean Densities on Either Side of each Edge Point
  - $N \times N$ Block for each Edge Point from Threshold Map (only for figure)
  - $N \times N$ Block for each Edge Point from Density Map (only for figure)

Figure 9: The Region of Interest - Threshold Map

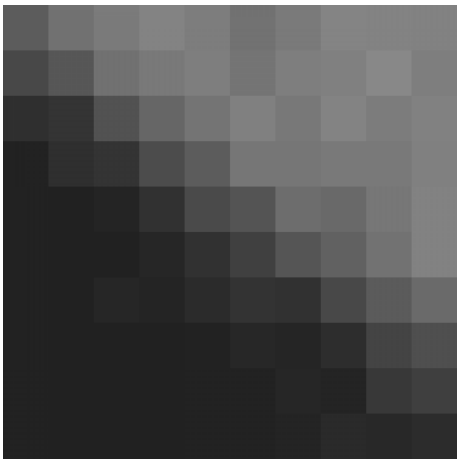

Figure 10: The Region of Interest - Density Map

2.12 Model the Relation between Density and Velocity

density_velocity_model.m uses Density-Velocity values from Schneider et al. (1996) to model their relation. A 'smoothingspline' fit was used, and then values were retrieved from the model in an identical matter to the HU-Density Model.

- Inputs
  - Vector of Schneider Density Values
  - Vector of Schneider Velocity Values
- Outputs
  - Vector of fitted Density Values
  - Vector of corresponding fitted Velocity Values

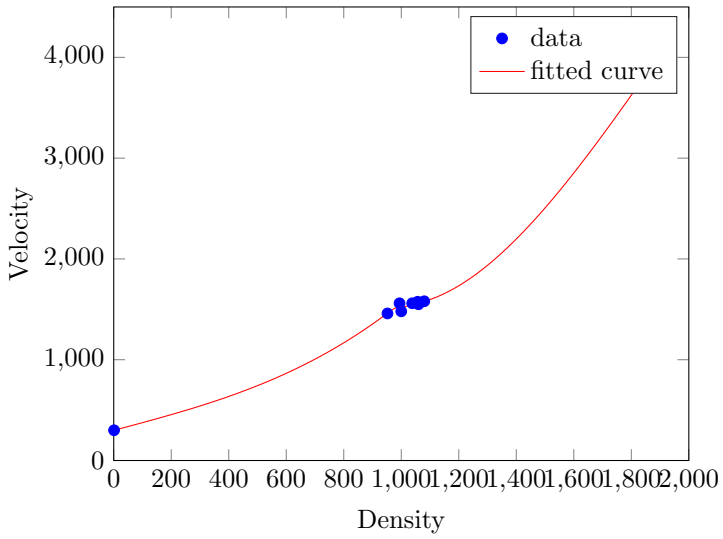

Figure 11: The Modeled Density-Velocity Relation

2.13 Compute Velocities from the Density-Velocity Model

interpolate_velocity.m interpolates velocities from the mean densities generated in mean_impedances.m based on the Density-Velocity Model. This must be run twice, once each for the initial impedance and the final impedance.

- Inputs
  - Mean Densities on Either Side of each Edge Point
  - Vector of fitted Density Values
  - Vector of corresponding fitted Velocity Values
- Outputs
  - Velocity Data on Either side of each Edge Point

2.14 Compute the Reflection Coefficient at each Edge Point

reflection_coeff_vals.m computes the reflection coefficient at each edge using:

$$\alpha_R = cos(\theta)^2(\frac{Z_2 - Z_1}{Z_2 + Z_1})^2$$

- Inputs
  - Mean Densities on Either Side of each Edge Point
  - Mean Velocities on Either Side of each Edge Point
  - Edge Map

  - Incident Angle Values in Degrees

- Outputs
  - Reflection Coefficient Values

2.15 Compute the Reflected Intensity

compute_ir_x.m computes the reflected intensity received back at the transducer.

- Inputs
  - Reflection Coefficient Values
  - Unit Normal Vectors
  - Unit Ray Vectors
  - Edge Map
  - Vector length to each point in image
- Outputs
  - Reflection Image from Edge Points

2.16 Compute the Attenuation

The attenuation is based on:

$$I(D) = I_0 e^{-\alpha D}$$

where α is the attenuation coefficient of the wave, with units of $dB/length$

2.16.1 Method

Use Bresenham's Line Drawing Algorithm to trace each voxel the beam passes through.

2.17 Load the Constants

loadconstants.m loads the constant values for the simulation.

3 Concerns

4 Sources

Shams, R., Hartley, R. & Navab, N.: Real-time simulation of medical ultrasound from CT images. *Medical image computing and computer-assisted intervention: MICCAI - International Conference on Medical Image Computing and Computer-Assisted Intervention* 11, 734741 (2008).