Modeling and Simulation schemes in Radiography/ CT

Koushik Viswanathan
Dr. Krishnan Balasubramanian

Center for Non-Destructive Evaluation
Indian Institute of Technology, Madras

January 12, 2010
Were going to look at two schemes pertaining to X-ray radiography and CT
Were going to look at two schemes pertaining to X-ray radiography and CT

▶ Data reconstruction involving Computed Tomography, mainly dealing with reconstruction of 2D cross-sections from X-ray projections
Were going to look at two schemes pertaining to X-ray radiography and CT

▶ Data reconstruction involving Computed Tomography, mainly dealing with reconstruction of 2D cross-sections from X-ray projections

▶ Forward simulation to simulate radiography/ tomographic imaging on CAD models
Were going to look at two schemes pertaining to X-ray radiography and CT

- Data reconstruction involving Computed Tomography, mainly dealing with reconstruction of 2D cross-sections from X-ray projections
- Forward simulation to simulate radiography/ tomographic imaging on CAD models
- All of this is done ENTIRELY on commodity graphics hardware!
Simulations on GPUs

- Graphics Processing Units are capable of large-scale parallel processing
Simulations on GPUs

- Graphics Processing Units are capable of large-scale parallel processing
  - Low-end GPUs contain around 32 shader cores
Simulations on GPUs

- Graphics Processing Units are capable of large-scale parallel processing
  - Low-end GPUs contain around 32 shader cores
  - Available high end cards feature up to 480 cores
Simulations on GPUs

- Graphics Processing Units are capable of large-scale parallel processing
  - Low-end GPUs contain around 32 shader cores
  - Available high end cards feature up to 480 cores
  - Intel’s next-gen processor will have 16 cores
Simulations on GPUs

- Graphics Processing Units are capable of large-scale parallel processing
  - Low-end GPUs contain around 32 shader cores
  - Available high end cards feature up to 480 cores
  - Intel’s next-gen processor will have 16 cores
- Two approaches to parallelization, namely multi-core processing (reconstruction) and traditional GPGPU techniques (forward simulation)
Simulations on GPUs

- Graphics Processing Units are capable of large-scale parallel processing
  - Low-end GPUs contain around 32 shader cores
  - Available high end cards feature up to 480 cores
  - Intel’s next-gen processor will have 16 cores

- Two approaches to parallelization, namely multi-core processing (reconstruction) and traditional GPGPU techniques (forward simulation)

- Drawbacks with GPUs
Simulations on GPUs

- Graphics Processing Units are capable of large-scale parallel processing
  - Low-end GPUs contain around 32 shader cores
  - Available high end cards feature up to 480 cores
  - Intel’s next-gen processor will have 16 cores
- Two approaches to parallelization, namely multi-core processing (reconstruction) and traditional GPGPU techniques (forward simulation)
- Drawbacks with GPUs
  - No double precision support (though that is changing)
Simulations on GPUs

- Graphics Processing Units are capable of large-scale parallel processing
  - Low-end GPUs contain around 32 shader cores
  - Available high end cards feature up to 480 cores
  - Intel’s next-gen processor will have 16 cores
- Two approaches to parallelization, namely multi-core processing (reconstruction) and traditional GPGPU techniques (forward simulation)
- Drawbacks with GPUs
  - No double precision support (though that is changing)
  - Low clock speeds (unlikely to change soon)
Image reconstruction schemes
Most commonly used techniques include Filtered Backprojection and Algebraic Reconstruction algorithms.
Most commonly used techniques include Filtered Backprojection and Algebraic Reconstruction algorithms.

Typical implementations on a standard PC require approx. 20 seconds for around 0.5 degree projections.
Most commonly used techniques include Filtered Backprojection and Algebraic Reconstruction algorithms.

Typical implementations on a standard PC require approx. 20 seconds for around 0.5 degree projections.

Reconstruction done layer-by-layer, full 3D Feldkamp reconstruction takes much longer due to larger datasets.
Our implementation

Our Reconstruction algorithms are implemented using a large-scale multi-threading scheme with NVidia's CUDA API.
Our implementation

Our Reconstruction algorithms are implemented using a large-scale multi-threading scheme with NVidia’s CUDA API

- 2D FFT is done initially on the CPU due to problems with double precision handling
Our implementation

Our Reconstruction algorithms are implemented using a large-scale multi-threading scheme with NVidia’s CUDA API

- 2D FFT is done initially on the CPU due to problems with double precision handling
- After pre-filtering on the CPU, the image is transferred onto the GPU for backprojection
Our implementation

Our Reconstruction algorithms are implemented using a large-scale multi-threading scheme with Nvidia's CUDA API

- 2D FFT is done initially on the CPU due to problems with double precision handling
- After pre-filtering on the CPU, the image is transferred onto the GPU for backprojection
- For a large number of projections, Backprojection is computationally the most intensive step
Our implementation

Our Reconstruction algorithms are implemented using a large-scale multi-threading scheme with NVidia’s CUDA API

- 2D FFT is done initially on the CPU due to problems with double precision handling
- After pre-filtering on the CPU, the image is transferred onto the GPU for backprojection
- For a large number of projections, Backprojection is computationally the most intensive step
- Done massively in parallel, spawn one thread per projection
Our implementation

Our Reconstruction algorithms are implemented using a large-scale multi-threading scheme with NVidia’s CUDA API

- 2D FFT is done initially on the CPU due to problems with double precision handling
- After pre-filtering on the CPU, the image is transferred onto the GPU for backprojection
- For a large number of projections, Backprojection is computationally the most intensive step
- Done massively in parallel, spawn one thread per projection
- This technique scales very well with modern GPUs
Quick comparisons

<table>
<thead>
<tr>
<th>No of projections</th>
<th>CPU implementation (sec)</th>
<th>GPU implementation (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>3.623</td>
<td>0.132</td>
</tr>
<tr>
<td>200</td>
<td>7.609</td>
<td>0.197</td>
</tr>
<tr>
<td>300</td>
<td>11.630</td>
<td>0.263</td>
</tr>
<tr>
<td>400</td>
<td>15.567</td>
<td>0.323</td>
</tr>
<tr>
<td>500</td>
<td>19.388</td>
<td>0.381</td>
</tr>
</tbody>
</table>
Forward simulation schemes
Use Ray tracing, more commonly termed ray-casting, since only primary rays are followed through the medium.
Fundamental algorithm

- Use Ray tracing, more commonly termed ray-casting, since only primary rays are followed through the medium
- Spawn one ray per detector pixel, going from the source to the detector, through the medium
Fundamental algorithm

- Use Ray tracing, more commonly termed ray-casting, since only primary rays are followed through the medium
- Spawn one ray per detector pixel, going from the source to the detector, through the medium
- Depending on the requirements, a triangulated mesh or a voxel mode is used
Fundamental algorithm

\[ N(E) = N_0(E) \Delta \Omega \prod_i \exp[-\mu_i(E)x_i] \]

\[ = N_0(E) \Delta \Omega \exp \left[ \sum_i -\mu_i(E)x_i \right] . \]
Possible optimizations

Ray-tracing optimization has been treated extensively, several techniques have been proposed.
Possible optimizations

Ray-tracing optimization has been treated extensively, several techniques have been proposed

- Spatial sub-division techniques esp. Octrees and KD Trees
Possible optimizations

Ray-tracing optimization has been treated extensively, several techniques have been proposed

▶ Spatial sub-division techniques esp. Octrees and KD Trees
▶ Intersection tests can be reduced using Bounding Volume Hierarchies
Possible optimizations

Ray-tracing optimization has been treated extensively, several techniques have been proposed:

- Spatial sub-division techniques esp. Octrees and KD Trees
- Intersection tests can be reduced using Bounding Volume Hierarchies
- Parallel processing involving parallel ray-tracing algorithms
Parallel Processing tests

![Graph showing parallel processing tests]

- Serial execution
- Coarse Grain
- Fine Grain

Time (s) vs. Faces
Our algorithm

We went back to the drawing board and reworked the basic law evaluation
Our algorithm

We went back to the drawing board and reworked the basic law evaluation

- Developed a new algorithm to evaluate the Beer-Lambert law
Our algorithm

We went back to the drawing board and reworked the basic law evaluation

- Developed a new algorithm to evaluate the Beer-Lambert law
- Instead of following ray-tracing based techniques, we developed a rasterization based algorithm
Our algorithm

We went back to the drawing board and reworked the basic law evaluation

▶ Developed a new algorithm to evaluate the Beer-Lambert law
▶ Instead of following ray-tracing based techniques, we developed a rasterization based algorithm
▶ Intensity of every detector pixel computed in parallel, we make GPUs do what they were meant to do!
Very promising results

- Other groups around the world have tackled this problem, but our algorithm works faster than all of the results reported so far.
Very promising results

- Other groups around the world have tackled this problem, but our algorithm works faster than all of the results reported so far.
- Works on any consumer-grade hardware, only OpenGL 2.0 support needed!

<table>
<thead>
<tr>
<th>Triangle count</th>
<th>CPU computation (ms)</th>
<th>GPU computation (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>53582</td>
<td>70.07</td>
<td>0.41</td>
</tr>
<tr>
<td>72353</td>
<td>79.56</td>
<td>0.44</td>
</tr>
<tr>
<td>96967</td>
<td>87.58</td>
<td>0.44</td>
</tr>
<tr>
<td>154349</td>
<td>116.65</td>
<td>0.52</td>
</tr>
<tr>
<td>178690</td>
<td>156.71</td>
<td>0.58</td>
</tr>
<tr>
<td>871415</td>
<td>470.83</td>
<td>1.34</td>
</tr>
</tbody>
</table>

Koushik Viswanathan, Dr. Krishnan Balasubramanian
Modeling and Simulation schemes in Radiography/CT
XRaySim

- Open source Radiography/Computed Tomography simulation tool
- Developed from scratch at the Center for Non-Destructive Evaluation, IIT Madras
- Seamless integration with commercial CAD packages through standard data exchange formats
Thank you for your attention!
Questions/ enquiries are welcome