Faster 3D CT Reconstruction using CUDA and OpenCL

Saoni Mukherjee, Nicholas Moore, James Brock and Miriam Leeser

April 24, 2012

Northeastern
Introduction to 3D Computer Tomography Scan

A sinogram is a line for every angle of the reconstructed cross-sectional slice data. The process involves taking 2D X-ray images from different angles and reconstructing a 3D volume from these projections.

Klaus Mueller, Introduction to Medical Imaging, Lecture 6: X-Ray Computed Tomography, Computer Science Department, Stony Brook University
FDK Cone beam CT reconstruction

- Feldkamp, Davis and Kress (FDK)\(^1\) developed in 1984.

- Most commercial CT scanners use FDK.

- The raw projections individually weighted and ramp filtered. Weighting includes cosine weighting and short-scan weighting.

- Reconstructions of filtered projections for the final volume.

\(^1\) [http://www.eecs.umich.edu/~fessler/irt/irt](http://www.eecs.umich.edu/~fessler/irt/irt)
FDK Cone beam CT reconstruction

• Feldkamp, Davis and Kress (FDK)\(^1\) developed in 1984.

• Most commercial CT scanners use FDK.

• The raw projections individually weighted and ramp filtered. Weighting includes cosine weighting and short-scan weighting.

• Reconstructions of filtered projections for the final volume.

\(^1\) http://www.eecs.umich.edu/~fessler/irt/irt
FDK Cone beam CT reconstruction

• Feldkamp, Davis and Kress (FDK)\(^1\) developed in 1984.

• Most commercial CT scanners use FDK.

• The raw projections individually weighted and ramp filtered. Weighting includes cosine weighting and short-scan weighting.

• Reconstructions of filtered projections for the final volume.

\(^1\) [http://www.eecs.umich.edu/~fessler/irt/irt](http://www.eecs.umich.edu/~fessler/irt/irt)
Feldkamp CT reconstruction geometry

filtered projections

reconstructed 3D volume

X-ray source
Feldkamp CT reconstruction geometry - 2

Backprojection:

\[ F(x, y, z) = \frac{1}{2\pi t} \sum_{i=1}^{t} W_2(x, y, i) Q_i(u(x, y, i), v(x, y, z, i)), \]

\[
\begin{align*}
    u(x, y, i) &= \frac{d'(x \sin \theta_i + y \cos \theta_i)}{d_i - x \cos \theta_i - y \sin \theta_i}, \\
    v(x, y, z, i) &= \frac{d' z}{d_i - x \cos \theta_i - y \sin \theta_i},
\end{align*}
\]

Weight value,

\[ W_2(x, y, i) = \frac{d_i}{d_i - x \cos \theta_i - y \sin \theta_i}. \]
What’s the problem?

The long time it takes to reconstruct the volume!

- Interruption in treatment/diagnosis
- Capturing data takes ~9 seconds.
Time spent in single-threaded code

<table>
<thead>
<tr>
<th>Programming paradigm</th>
<th>Time to run</th>
<th>Total time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Backprojection</td>
<td></td>
</tr>
<tr>
<td>MATLAB</td>
<td>2h 20m 40s</td>
<td>2h 20m 43s</td>
</tr>
<tr>
<td>C</td>
<td>1h 32m 36s</td>
<td>1h 32m 39s</td>
</tr>
</tbody>
</table>
GPUs provides faster way to compute

GPU computing key ideas:

- Massively parallel
- Hundreds of cores
- Thousands of threads
- Cheap
- Highly available
Goal - GPU as an accelerator in CBCT

• Backprojection most computationally intensive part taking most of the time, but highly parallelizable.

• Independent different voxels to be processed simultaneously.

• Fessler’s image reconstruction toolbox\(^1\) an implementation of Feldkamp CBCT in MATLAB. Widely used in Academia.

• Our goal is to implement faster Feldkamp CT.

\(^1\): http://www.eecs.umich.edu/~fessler/irt/irt
GPU implementation of Feldkamp CBCT

- Processing divided into three steps: weighting, filtering and backprojection.
- Each step executed in each kernel.
- Non-blocking kernel calls, but executed in series.
- Minimization of expensive memory transfers
GPUs used to test the implementations

**NVIDIA TESLA C2070**

- Maximum 1536 resident threads in each multiprocessor
- 14 streaming multiprocessors
- Theoretical limit on the number of threads in flight at once is 21,504
- Memory size 6GB

**AMD Radeon HD 5870**

- Can run up to 31,744 threads concurrently
- Memory size 1GB

Similar generation
Sample Projections

Mathematical phantom
Input: 64 × 60 pixels with 72 projections
final volume: 64 × 60 × 50 voxels
Size : 1MB + 1MB

Mouse scan
Input: 512 × 768 pixels with 361 projections
final volume: 512 × 512 × 768 voxels
Size : 542 MB + 768 MB > 1GB
## Architectures and Languages used

<table>
<thead>
<tr>
<th>Host</th>
<th>Device</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel Core i7 quad-core processor with @ 3.4 GHz</td>
<td></td>
<td>MATLAB MATLAB PCT</td>
</tr>
<tr>
<td>Intel Xeon W3580 quad-core processor @ 3.33 GHz</td>
<td>NVIDIA Tesla C2070</td>
<td>C C with OpenMP CUDA</td>
</tr>
<tr>
<td>Intel Xeon CPUs E5520 @ 2.27GHz</td>
<td>AMD Radeon HD5870</td>
<td>OpenCL</td>
</tr>
</tbody>
</table>
Roadmap for Results

- Total runtime
- Speed up
- Runtime for each kernel: Nvidia vs. AMD
Results on phantom data

<table>
<thead>
<tr>
<th>Programming paradigm</th>
<th>Time to run</th>
<th>Total time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Backprojection (sec)</td>
<td></td>
</tr>
<tr>
<td>MATLAB</td>
<td>17.02</td>
<td>17.09</td>
</tr>
<tr>
<td>C</td>
<td>1.36</td>
<td>1.44</td>
</tr>
<tr>
<td>C with OpenMP (4 thrds)</td>
<td>0.32</td>
<td>0.33</td>
</tr>
<tr>
<td>OpenCL (AMD)</td>
<td>0.10</td>
<td>0.16</td>
</tr>
<tr>
<td>OpenCL (NVIDIA)</td>
<td>0.01</td>
<td>0.11</td>
</tr>
<tr>
<td>CUDA</td>
<td>0.01</td>
<td>0.10</td>
</tr>
</tbody>
</table>
### Speedups for phantom data

<table>
<thead>
<tr>
<th>Programming Paradigm</th>
<th>Speedup over single threaded MATLAB</th>
<th>Speedup over single threaded C</th>
<th>Speedup over multi-threaded C</th>
</tr>
</thead>
<tbody>
<tr>
<td>C with OpenMP</td>
<td>50x</td>
<td>4x</td>
<td>-</td>
</tr>
<tr>
<td>OpenCL (AMD)</td>
<td>170x</td>
<td>13x</td>
<td>3x</td>
</tr>
<tr>
<td>OpenCL (NVIDIA)</td>
<td>1700x</td>
<td>136x</td>
<td>32x</td>
</tr>
<tr>
<td>CUDA</td>
<td>1700x</td>
<td>136x</td>
<td>32x</td>
</tr>
</tbody>
</table>

Comparisons are based on the time taken by Backprojection.
## Results – comparing NVIDIA vs. AMD

<table>
<thead>
<tr>
<th>GPU</th>
<th>Kernel</th>
<th>Time (millisecond)</th>
<th>Total time (millisecond)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMD</td>
<td>Weighting</td>
<td>14.70</td>
<td>157.61</td>
</tr>
<tr>
<td></td>
<td>Filtering</td>
<td>123.23</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Backprojection</td>
<td>19.68</td>
<td></td>
</tr>
<tr>
<td>NVIDIA</td>
<td>Weighting</td>
<td>2.25</td>
<td>105.94</td>
</tr>
<tr>
<td></td>
<td>Filtering</td>
<td>89.62</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Backprojection</td>
<td>14.07</td>
<td></td>
</tr>
</tbody>
</table>
Results on mouse scan data

<table>
<thead>
<tr>
<th>Programming paradigm</th>
<th>Hardware</th>
<th>Backprojection (sec)</th>
<th>Total time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATLAB</td>
<td>Intel Core i7</td>
<td>2h 20m 40s</td>
<td>2h 20m 43s</td>
</tr>
<tr>
<td>MATLAB PCT (8thrds)</td>
<td>Intel Core i7</td>
<td>1h 32m 36s</td>
<td>1h 32m 39s</td>
</tr>
<tr>
<td>C</td>
<td>Intel Xeon W3580</td>
<td>1h 14m 37s</td>
<td>1h 14m 43s</td>
</tr>
<tr>
<td>C with OpenMP (4thrds)</td>
<td>Intel Xeon W3580</td>
<td>32m 09s</td>
<td>32m 12s</td>
</tr>
<tr>
<td>OpenCL</td>
<td>NVIDIA Tesla 2070</td>
<td>01m 07s</td>
<td>01m 31s</td>
</tr>
<tr>
<td>CUDA</td>
<td>NVIDIA Tesla 2070</td>
<td>42s</td>
<td>55s</td>
</tr>
</tbody>
</table>

Memory was an issue for AMD GPU!
## Speedups for mouse scan data

<table>
<thead>
<tr>
<th>Programming Paradigm</th>
<th>Speedup over single threaded MATLAB</th>
<th>Speedup over multi-threaded MATLAB</th>
<th>Speedup over single threaded C</th>
<th>Speedup over multi-threaded C</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATLAB PCT</td>
<td>1.5x</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>C with OpenMP</td>
<td>4x</td>
<td>-</td>
<td>2x</td>
<td>-</td>
</tr>
<tr>
<td>OpenCL (NVIDIA)</td>
<td>125x</td>
<td>80x</td>
<td>70x</td>
<td>30x</td>
</tr>
<tr>
<td>CUDA</td>
<td>200x</td>
<td>130x</td>
<td>100x</td>
<td>45x</td>
</tr>
</tbody>
</table>

Comparisons are based on the time taken by Backprojection.
Future Work

- Optimize other GPU kernels
- More configurations to be tested with auto-tuning
- Streaming for bigger datasets
- Overlapping computation and communication
- Improve performance on AMD device
Conclusions

• Faster 3D cone beam reconstruction using GPU.
• Compatible with Fessler’s image reconstruction tool box.
• Compared CUDA and OpenCL, to serial and multithreaded C and MATLAB implementations.

- Tested on two types of hardware

  - CUDA code takes 43 sec to backproject mouse scan.
    ➢ 200x faster than single-threaded MATLAB,
    ➢ 100x faster than single-threaded C,
    ➢ 45x faster than multi-threaded C with OpenMP.
Saoni Mukherjee, saoni@coe.neu.edu

Software can be downloaded from:
http://www.coe.neu.edu/Research/rcl/projects/CBCT.php

Acknowledgments

Award Number
EEC-0946463