CUDA and OpenCL Implementations of 3D CT Reconstruction for Biomedical Imaging

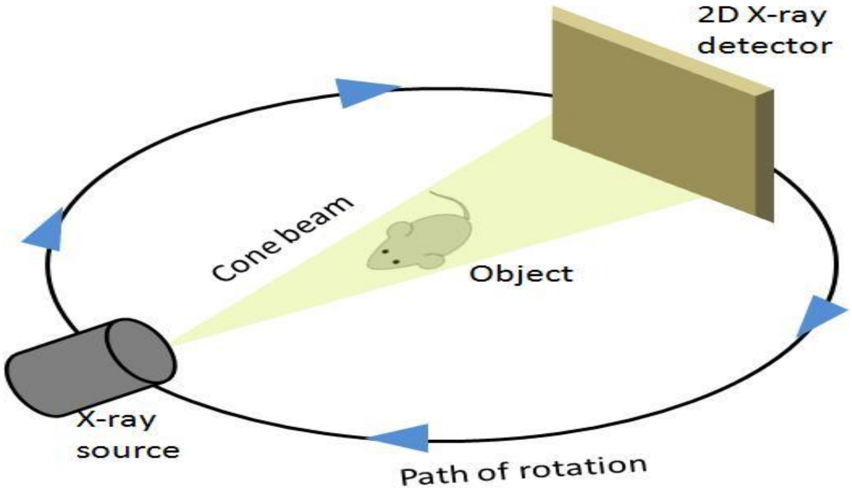
Saoni Mukherjee, Nicholas Moore, James Brock
and Miriam Leeser

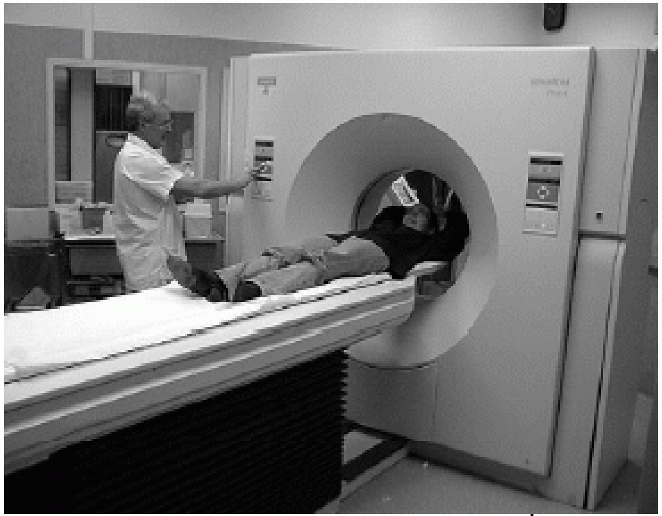
September 12, 2012

Outline

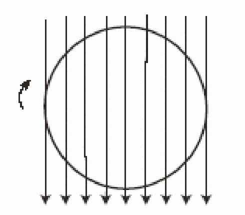
- ✓ Introduction to CT Scan, 3D reconstruction
- ✓ Algorithm for CT reconstruction- Feldkamp Algorithm
- ✓ Pros and Cons of the reconstruction method
- ✓ How we resolved the issues?
- ✓ Results
- ✓ Future Work
- ✓ Conclusions

Introduction to 3D Computer Tomography Scan

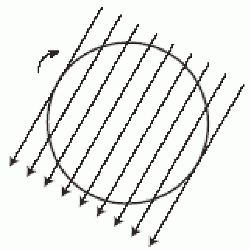




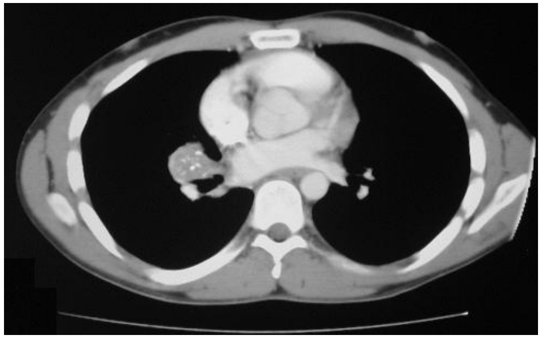
data



reconstruction routine



sinogram: a line for every angle



reconstructed cross-sectional slice

3D reconstructed volume

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

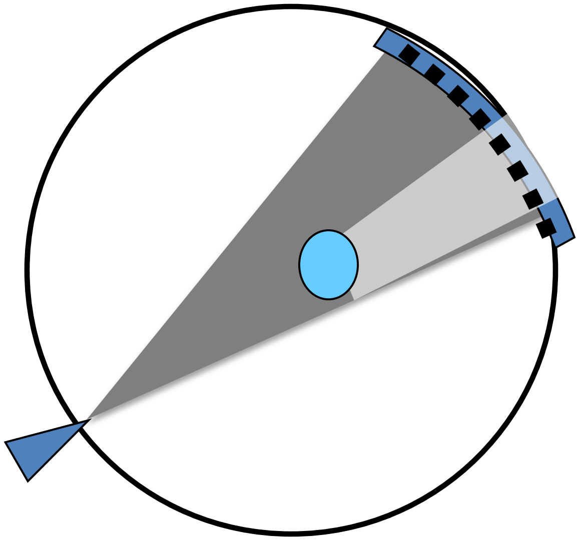
Feldkamp Cone beam CT reconstruction

- Feldkamp, Davis and Kress (FDK)[1] developed in 1984.

- Most commercial CT scanners use FDK.

- The raw projections P_1, P_2,..., P_K are individually weighted and ramp filtered. Weighting includes cosine weighting and short-scan weighting.

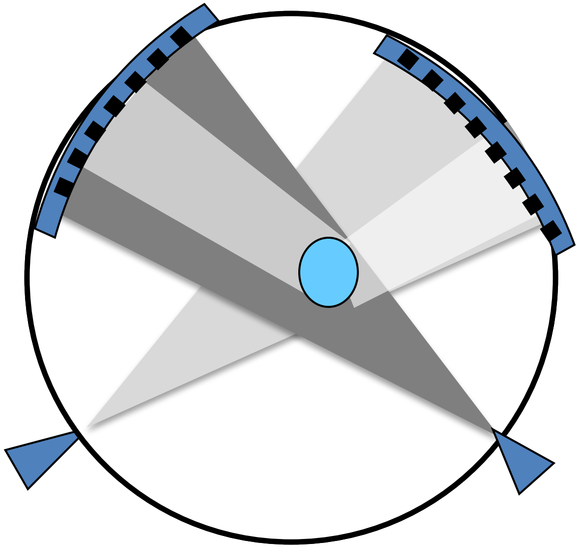
- The filtered projections are reconstructed to get the final volume.



[1]: http://www.eecs.umich.edu/~fessler/irt/irt

Feldkamp Cone beam CT reconstruction

- Feldkamp, Davis and Kress (FDK)[1] developed in 1984.
- Most commercial CT scanners use FDK.
- The raw projections P_1, P_2,..., P_K are individually weighted and ramp filtered. Weighting includes cosine weighting and short-scan weighting.
- The filtered projections are reconstructed to get the final volume.



[1]: http://www.eecs.umich.edu/~fessler/irt/irt

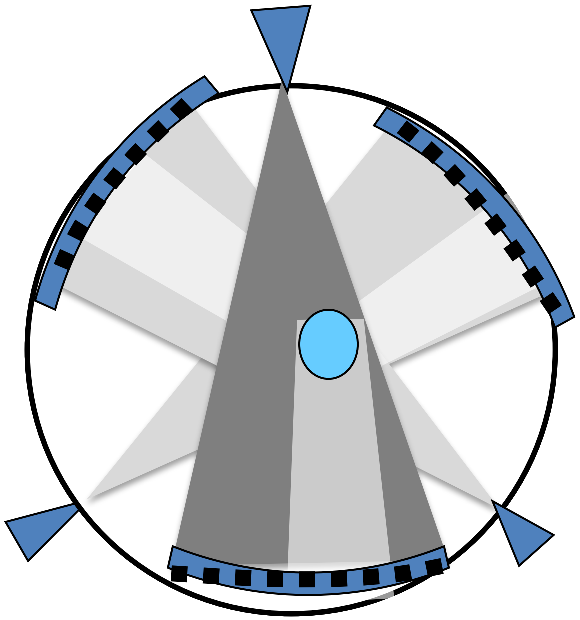
Feldkamp Cone beam CT reconstruction

- Feldkamp, Davis and Kress (FDK)[1] developed in 1984.

- Most commercial CT scanners use FDK.

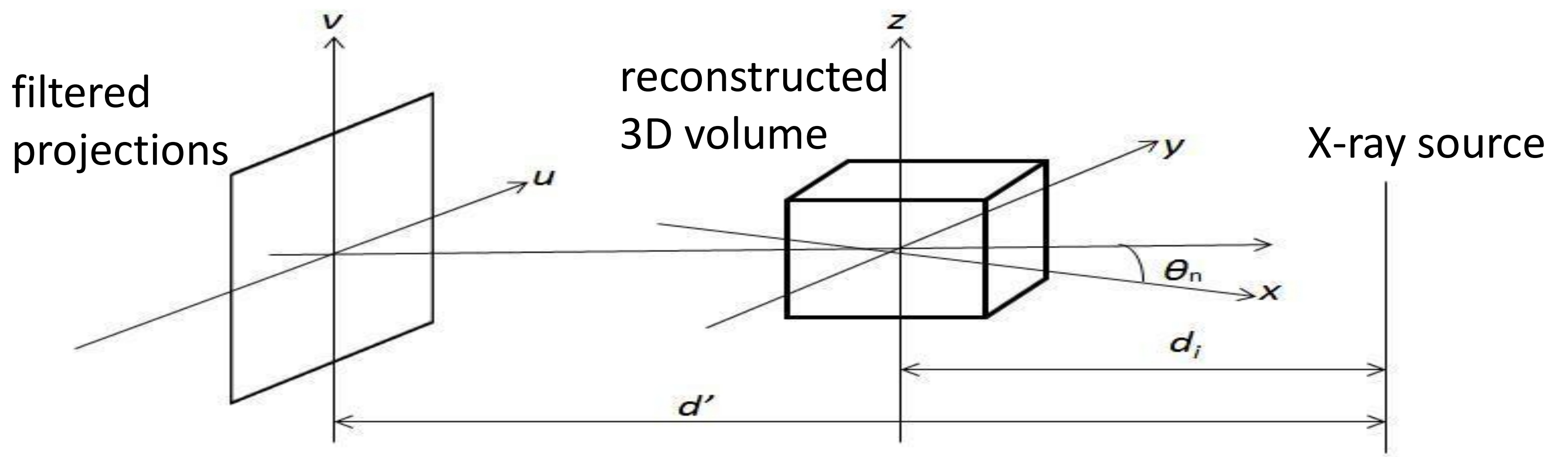
- The raw projections P_1, P_2,..., P_K are individually weighted and ramp filtered. Weighting includes cosine weighting and short-scan weighting.

- The filtered projections are reconstructed to get the final volume.



[1]: http://www.eecs.umich.edu/~fessler/irt/irt

Feldkamp CT reconstruction geometry- 1



1. **Weighted Projection:** Weighted and ramp filtered raw data produce filtered projections $Q_1, Q_2, \ldots, Q_K$, collected at an angle ϑn where $1 \leq n \leq K$.
   d_i = distance between the volume origin and the source.
   $F(x, y, z)$ = value of voxel (x, y, z) in volume F
   Volume F in xyz space and Projections are in *uv* space.

Feldkamp CT reconstruction geometry- 2

2. Backprojection: The volume F is reconstructed using the following equations:

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

Co-ordinates

$$u(x,y,i) = \frac{d'(-x\sin\theta_i + y\cos\theta_i)}{d_i - x\cos\theta_i - y\sin\theta_i)},$$

$$v(x,y,z,i) = \frac{d'z}{d_i - x\cos\theta_i - y\sin\theta_i)},$$

Weight value,

$$W_2(x,y,i) = \frac{d_i}{d_i - x\cos\theta_i - y\sin\theta_i)}.$$

Pros and Cons of cone beam CT

Advantage

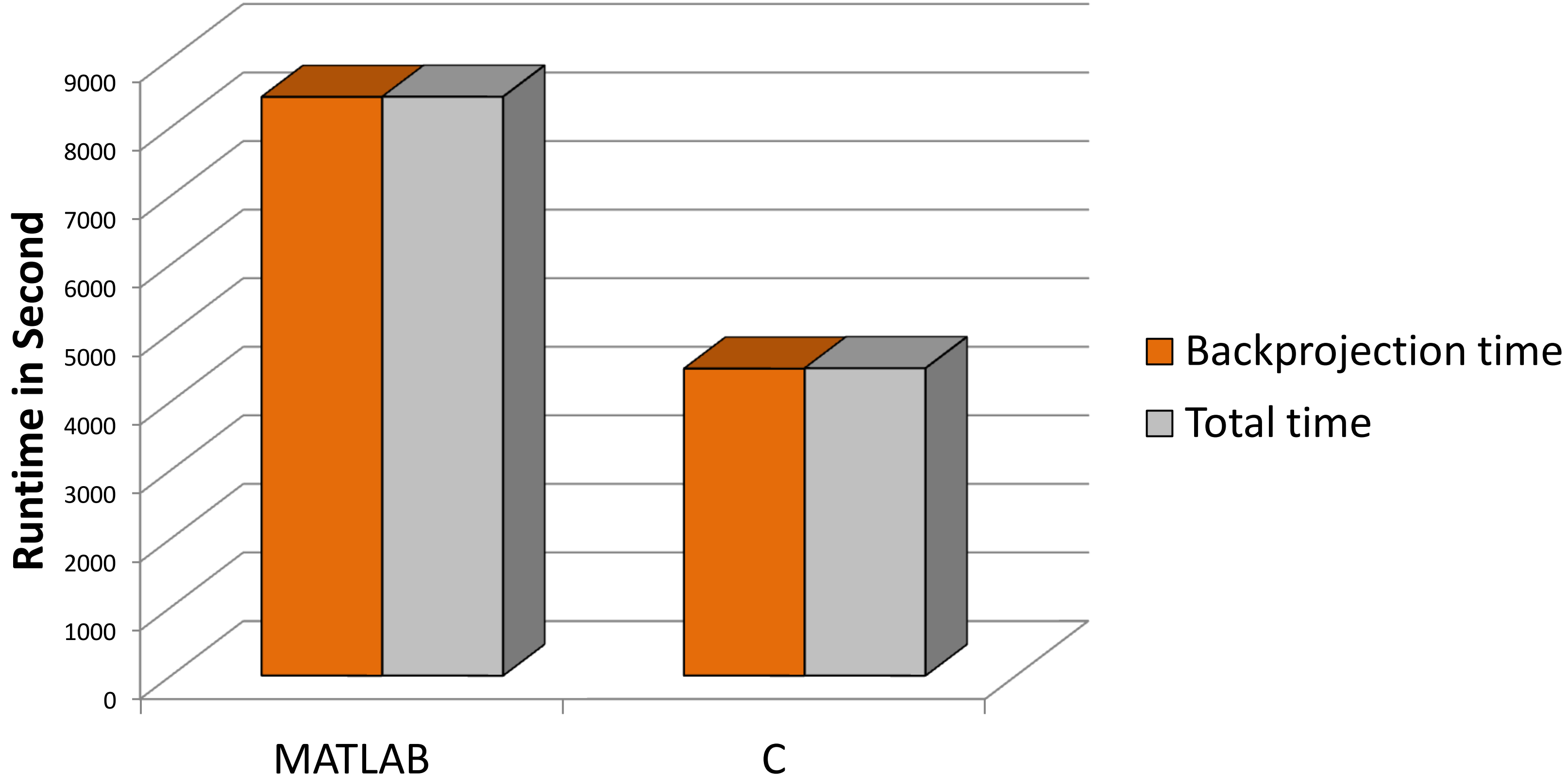
- Reduced X-ray exposure
- Image accuracy
  - more accurate than MRI!

Philips Brilliance CT Scanner

Disadvantage

- **The longer time it takes to reconstruct the volume!**
  - Interruption in treatment/ diagnosis.

Time spent in single-threaded code

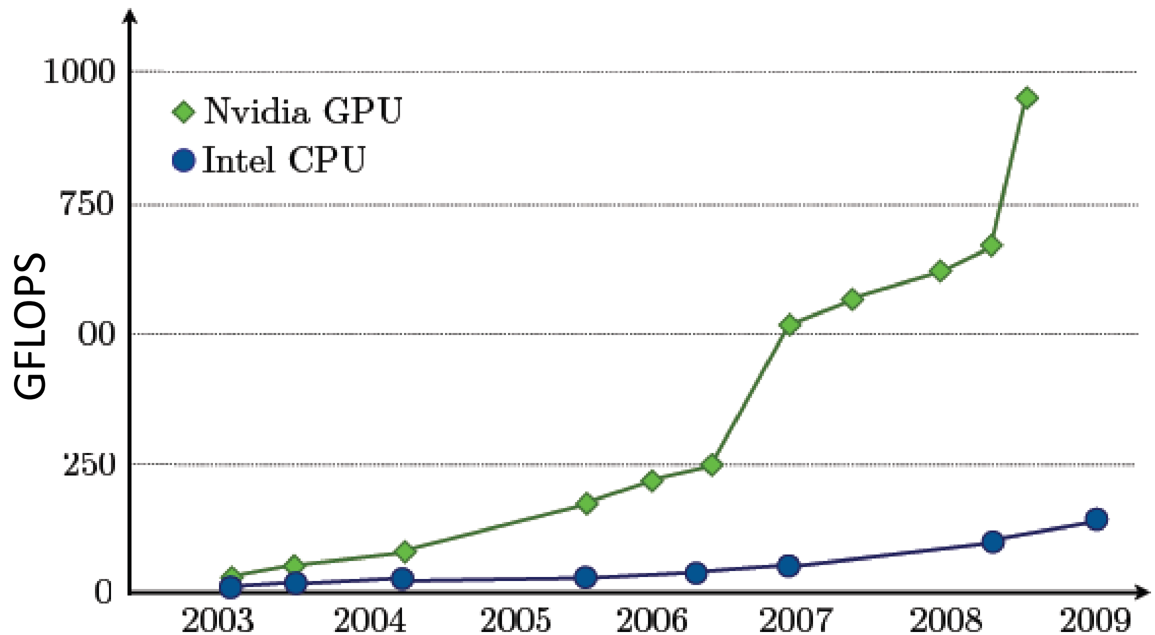


Programming paradigm	Time to run Backprojection	Total time
MATLAB	2h 20m 40s	2h 20m 43s
C	1h 32m 36s	1h 32m 39s

GPUs provides faster way to compute

GPU computing key ideas:

- Massively parallel
- Hundreds of cores
- Thousands of threads
- Cheap
- Highly available



Felipe A. Cruz, Tutorial on GPU computing with an introduction to CUDA, University of Bristol, Bristol, United Kingdom

Goal - GPU as an accelerator in CBCT

- Backprojection is the *most computationally intensive* part and takes the most of the time, but it is *highly parallelizable*.

- Different voxels are independent and can be *processed simultaneously*.

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

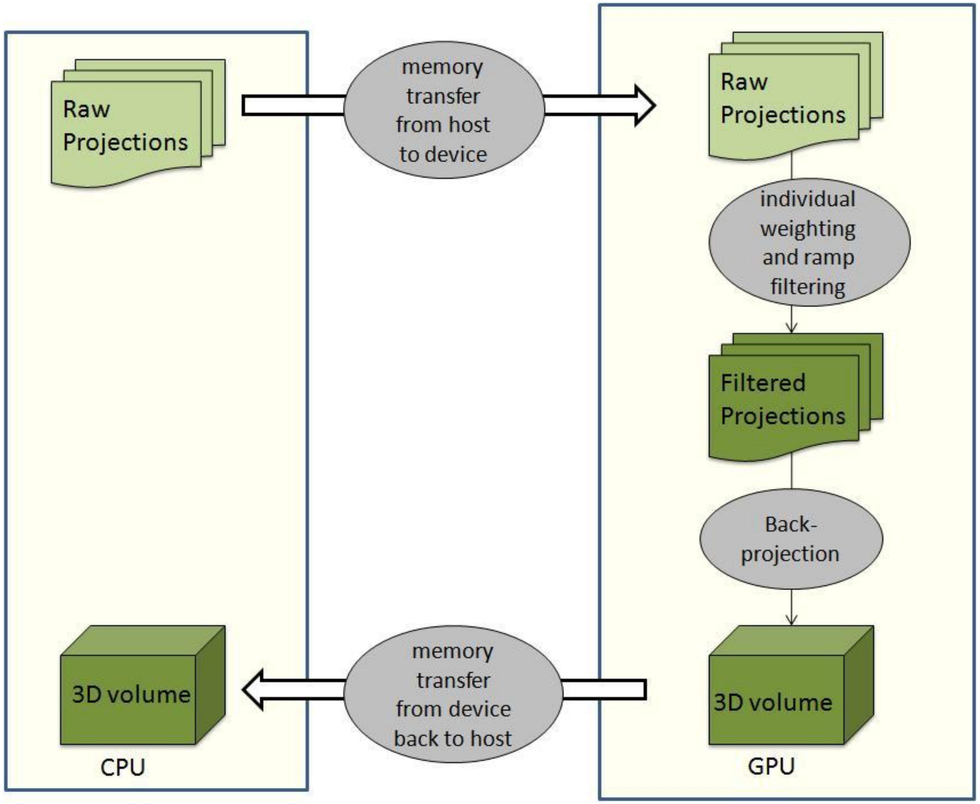
- *Fessler's image reconstruction toolbox*[1] provide an implementation of Feldkamp CBCT in MATLAB. Widely used in Academia.

- Our goal is to implement *Feldkamp CT in a faster way* that is compatible with the toolbox.

[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. Each step finishes before the next can begin.
- *Minimization of expensive memory transfers* by transferring the whole data to GPU before start of computation and transferring back after final volume reconstruction.



GPUs used to test the implementations

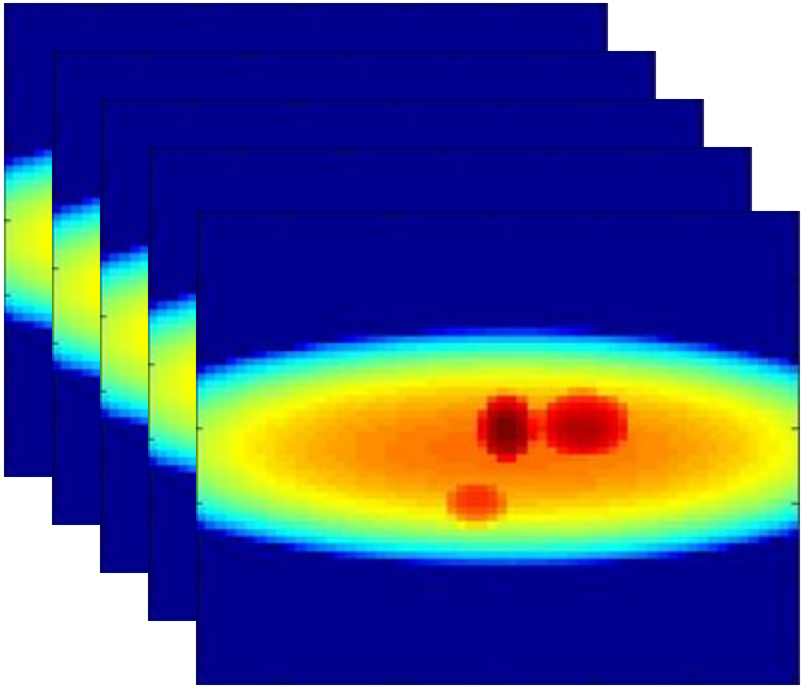
NVIDIA TESLA C2070

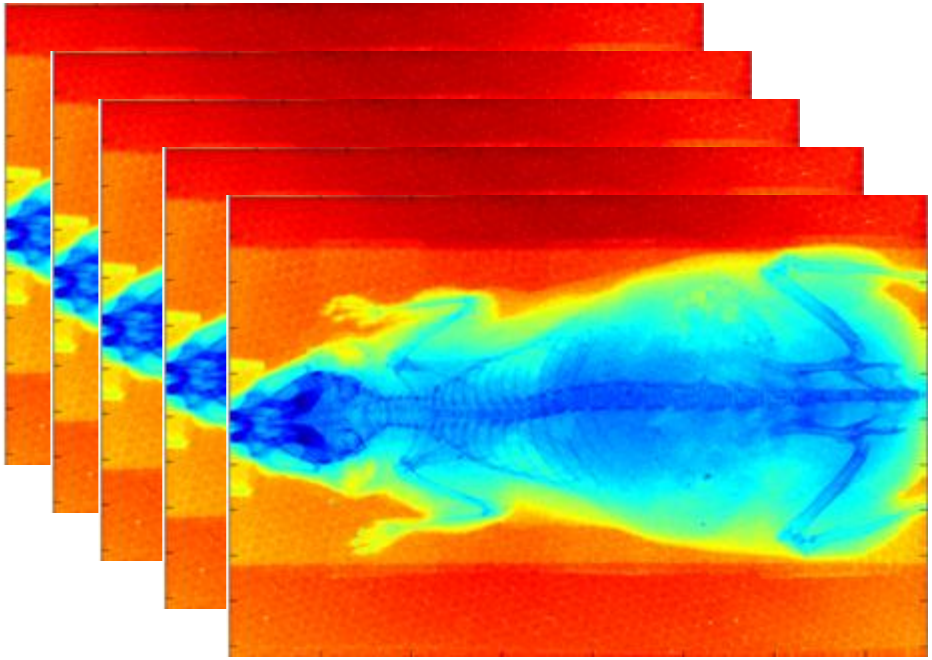
- Maximum 1536 resident threads in each multiprocessor
- 14 streaming multiprocessors
- Theoretical limit on the number of threads in flight at once is 21,504.

AMD Radeon HD 5870

- Can run up to 31,744 threads concurrently
- Similar generation as Tesla C2070.

Sample Projections





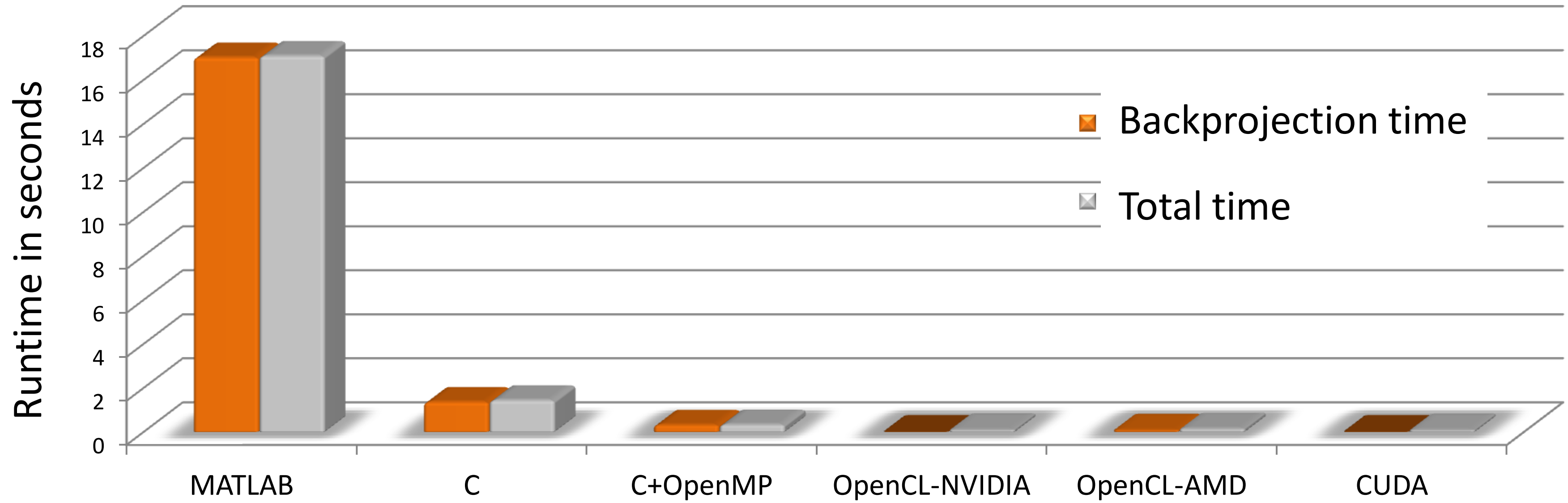
Mathematical phantom
Input: 64 × 60 pixels with 72 projections
final volume: 64 × 60 × 50 voxels

Mouse scan
Input: 512 × 768 pixels with 361 projections
final volume: 512 × 512 × 768 voxels

Architectures and Languages used

Host	Device	Language
Intel Core i7 quad-core processor with @ 3.4 GHz		MATLAB MATLAB PCT
Intel Xeon W3580 quad-core processor @ 3.33 GHz	NVIDIA Tesla C2070	C C with OpenMP CUDA
Intel Xeon CPUs E5520 @ 2.27GHz	AMD Radeon HD5870	OpenCL

Results on phantom data



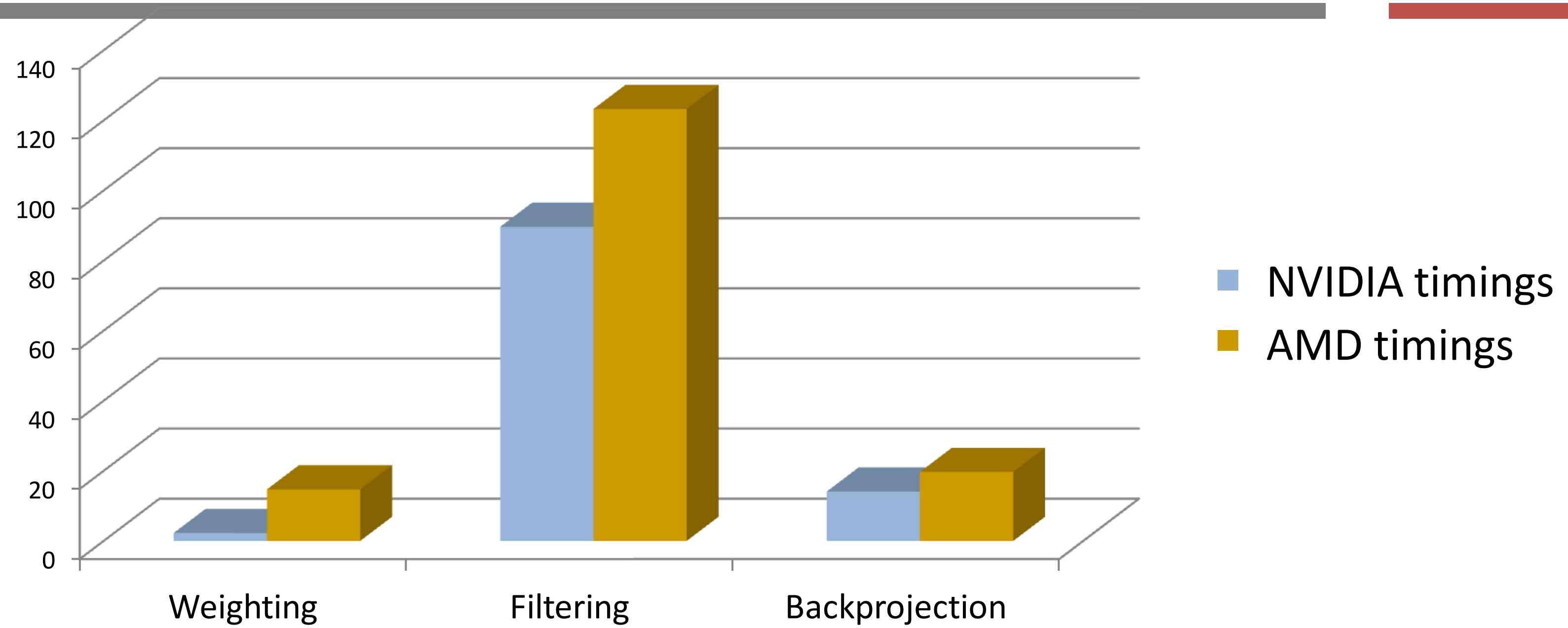
Programming paradigm	Time to run Backprojection (sec)	Total time (sec)
MATLAB	17.02	17.09
C	1.36	1.44
C with OpenMP (4 thrds)	0.32	0.33
OpenCL (NVIDIA)	0.01	0.11
OpenCL (AMD)	0.1	0.16
CUDA	0.01	0.1

Speedups for phantom data

Programming Paradigm	Speedup over single threaded MATLAB	Speedup over single threaded C	Speedup over multi-threaded C
C with OpenMP	50x	4x	-
OpenCL (NVIDIA)	**1700x**	**136x**	**32x**
OpenCL (AMD)	170x	13x	3x
CUDA	**1700x**	**136x**	**32x**

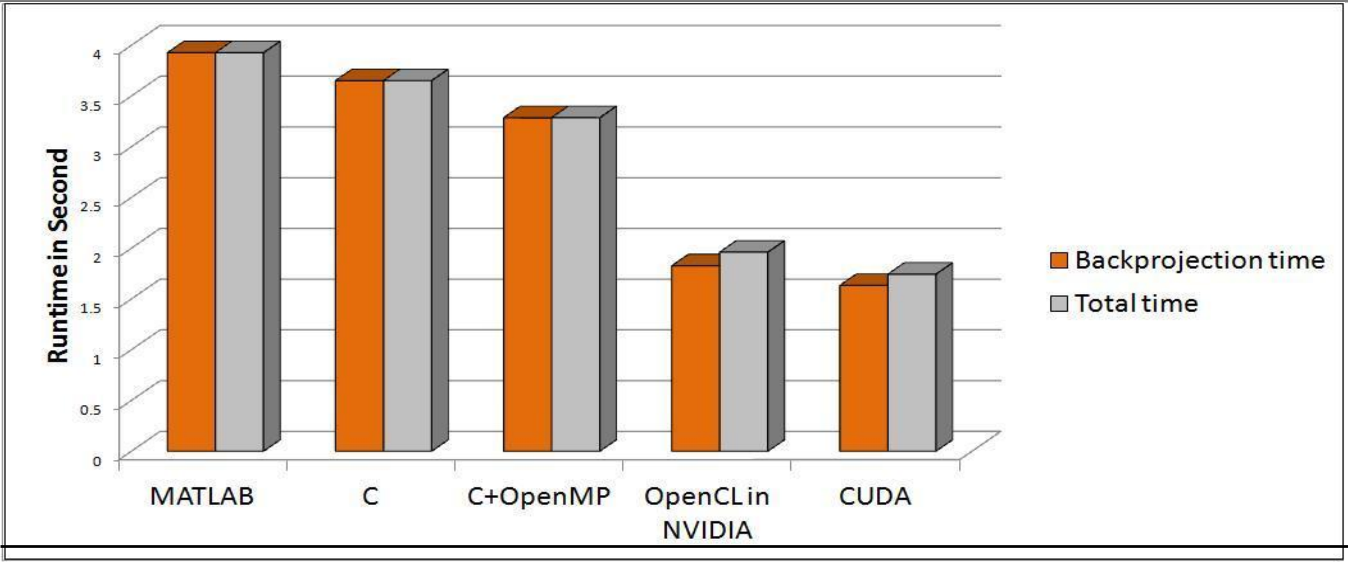
Comparisons are based on the time taken by Backprojection

Results – comparing NVIDIA vs. AMD



GPU	Kernel	Time (millisecond)	Total time (millisecond)
NVIDIA	Weighting Filtering Backprojection	2.25 89.62 14.07	105.94
AMD	Weighting Filtering Backprojection	14.70 123.23 19.68	157.61

Results on mouse scan data



Memory was an issue for AMD GPU!

Programming paradigm	Hardware	Time to run Backprojection (sec)	Total time (sec)
MATLAB	Intel Core i7	2h 20m 40s	2h 20m 43s
MATLAB PCT (8thrds)	Intel Core i7	1h 32m 36s	1h 32m 39s
C	Intel Xeon W3580	1h 14m 37	1h 14m 43s
C with OpenMP (4thrds)	Intel Xeon W3580	32m 9s	32m 12s
OpenCL	NVIDIA Tesla 2070	1m 7s	1m 31s
CUDA	NVIDIA Tesla 2070	42s	55s

Speedups for mouse scan data

Programming Paradigm	Speedup over single threaded MATLAB	Speedup over multi-threaded MATLAB	Speedup over single threaded C	Speedup over multi-threaded C
MATLAB PCT	1.5x	-	-	-
C with OpenMP	4x	-	2x	-
OpenCL (NVIDIA)	125x	80x	70x	30x
CUDA	200x	130x	100x	45x

Comparisons are based on the time taken by Backprojection

Future Work

- The next bottleneck- Weighted Filtering. Was *not* earlier!

- More configurations to be tested with auto-tuning- number of kernels to be launched, number of threads.

- Streaming for bigger datasets.

- Overlapping computation and communication.

Conclusions

- A faster way to 3D reconstruct cone beam projections in a GPU-enabled system based on the FDK method.
- Compatible with Fessler's image reconstruction tool box.
- Compared the performance of CUDA and OpenCL, to serial and multithreaded C and MATLAB implementations.
  - Tested on two types of hardware platforms: CPU and a combination of CPU and GPU, two types of GPUs- NVIDIA and AMD.
  - CUDA code takes 43 seconds to backproject mouse scan.
    - ➢ around **200x** faster than the single-threaded implementation in MATLAB,
    - ➢ around **100x** faster than the single-threaded implementation in C,
    - ➢ around **45x** faster than the multi-threaded implementation C + OpenMP.

This work was supported in part by the National Science Foundation Engineering Research Centers Innovations Program, Biomedical Imaging Acceleration Testbench (Award Number EEC-0946463).

This work is funded in part by a gift from Mathworks.

We thank Drs. Ralph Weissleder and Sarit Sekhar Agasthi, Massachusetts General Hospital for providing the mouse scan data.

Thanks to NUCAR group at Northeastern University for letting us use the Medusa cluster.

Saoni Mukherjee, saoni@coe.neu.edu

RCL lab, http://www.coe.neu.edu/Research/rcl/index.php