Real-Time Multi-Plane Tomosynthesis Using GPUs

Tobias Funk, Oleg Konings
Triple Ring Technologies
Conventional fluoroscopy
Interventional Procedures

Many Application

- Cardiology
  - Stents
  - Valve replacement
  - Congenital heart disease
- Radiology
  - Peripheral artery disease
  - Embolization
  - Emergency care
  - Orthopedics

Advantages

- Minimally invasive
- Fast recuperation times
- High success rate

Disadvantages

- High radiation dose
- No 3d information
Scanning beam digital X-ray (SBDX) system

Conventional

SBDX

Detector

Patient

X-ray source

© 2014 Triple Ring Technologies

19 March
System Comparison

**Conventional**

Simple shadowgram image

**SBDX**

Many small images captured rapidly
Images reconstructed in real time
Large entrance area – High efficiency detector
Image generation

Source array

Detector

Row n
Row n+1
Row n+2
Every source position illuminates the object from a slightly different angle
Reconstruction

Detector

Focal spots at collimator
Tomosynthesis: reconstruction of focal planes
Tomosynthesis
Image comparison
Philips FD10 vs. SBDX

4-fold exposure reduction
How to address the interventional radiology market?

Interventional cardiology
Interventional radiology

X-ray source
Detector
Interventional cardiology SBDX
Dual-detector SBDX system

- Detector 1
- Detector 2
- Collimator
- X-ray source
Results

Cardiology

Radiology

Bottom Imaging Volume
\[ z = 34.2 \text{ cm} \]
Results

Cardiology

Radiology

Top Imaging Volume
\[ z = 53.4 \text{ cm} \]
Single detector SBDX
Dual detector SBDX

Collimator

Detector 1

Detector 2
Data complexity

- Detector size 160x80 pixel
- 100x100 detector images
  - (50x100) from each detector
- Reconstruction into planes of 1000x2000 pixels
- Post processing
- 32 planes
- Plane selection

In 133 ms
Dataflow in the dual-detector system

Detector 1
- Data receiver alternates between detectors (FPGAs)

Pre-processing GPU

GPU Reconstructs planes in real-time

Display

Detector 2
- Pre-processing GPU

Current Single Plane SBDX System

Data receiver alternates between detectors (FPGAs)

Stage 1: Pre-Process GPU

Stage 2: Single Plane Reconstruction

Display

K20

Titan
Detectors and Chassis Hardware
Stage 1: Pre-processing GPU

RDMA:
- In coming data: 4x100x160x80 Blocks

PCIe:
- K20: Reshape Accumulate
- Ilh-data 160x80x100x100

Timing is continuous

Timing burst

To Reconstruction GPU
Stage 2: SBDX image reconstruction

*Using GPU #1, GTX Titan Black*

- A total of 8 GPU kernel launches in reconstruction and post-processing
- Must reconstruct full set to image buffer within time window
- Reconstruction kernel takes over 80% of the running time
- All kernels in process have dependencies and must be launched in serial order
Reconstruction

Focal plane (at distance z from X-ray source)

Position in the focal plane is a function of:
- Coordinates of the detector pixel
- Collimator hole position

Collimator hole position

Detector

X-direction

Y-direction

Y_{det}
Reconstruction Kernel

Detector Raw Data, 160x80x100x100

X-dim detector, 160x80

Y-dim detector, 160x80

Flux value sums, 100x100

Current Focal Plane (Z-dim)

1000x1800 Raw Image
Algorithm: Dual-detector SBDX image reconstruction

Inputs: 4-d detector array `llhdata[][][][]`, `height × width × rows × cols`,  
2-d `x_pixel_locs[][]`, `height × width`,  
2-d `y_pixel_locs[][]`, `height × width`,

Kernel Launch: 3-d, `(height × width)/work_per_thread, rows, cols)`

Output: 2-d image array `recon_imag[][]`, 1000 × 1800

Running Time: 19-23 ms, dependent on nature of inputs
CUDA reconstruction objectives:

- To coalesce global memory reads and writes
- To maximize 32-bit GFLOPS
- To maximize occupancy of GPU cores

Achieved by determining optimal:

- Launch configuration
  - Dimensionality of launch (3D)
  - Numbers of threads per block
- Work performed by each thread
  - Global memory operations
  - __shared__ memory and register operations
  - 32-bit floating point compute
Dual detector SBDX

Collimator

Detector 1

Detector 2
Combine Dual-Detector Data into Single Image

1 \( \text{row} \leftarrow \text{blockIdx}.y, \)
   \( \text{col} \leftarrow \text{blockIdx}.z, \)
   \( \text{Collimator}_\text{index} \leftarrow \text{threadIdx}.x + \text{blockIdx}.x*\text{blockDim}.x \)

2 \textbf{if} \ \text{threadIdx}.x == 0 \textbf{then} /* fill in \texttt{__shared__} memory for broadcast */

3 \textbf{if} \ \text{blockIdx}.y&1 \textbf{then}
   \( \text{image offset to account for left detector} \)
\textbf{else}
   \( \text{image offset to account for right detector} \)
Point of intersection of the back-projected ray

Weights (Bilinear coefficients)

\[
\begin{align*}
(I_x, I_y) & : (1-f_x) \times (1-f_y) \\
(I_x+1, I_y) & : f_x \times (1-f_y) \\
(I_x, I_y+1) & : (1-f_x) \times f_y \\
(I_x+1, I_y+1) & : f_x \times f_y \\
\end{align*}
\]
The output image 2x2 region per single llhdata input value is determined by the values in the current input set, starting with the ‘base’ row and column indices.

Generate ‘base’ row and column in output image:

\[
\text{row\_index} = \text{floor}(x_{\text{det}[k]}*z\_value) - 1 + \text{collimatorRow}*m + \text{offset};
\]
\[
\text{col\_index} = \text{floor}(y_{\text{det}[k]}*z\_value) - 1 + \text{collimatorCol}*m + \text{offset};
\]

Then write weighted values to the square region in pattern:

\((\text{row\_index}, \text{col\_index}), (\text{row\_index}, \text{col\_index}+1), (\text{row\_index}+1, \text{col\_index}), (\text{row\_index}+1, \text{col\_index}+1)\)
Floating Point Computation Optimization

- Utilize GPU FMA capability for “lerp”
  - Reduce two floating point operations to one FMA
  - Increase in performance and accuracy

Use fmaf(a, b, c), which calculates a*b+c

\[
\text{val} = v \times (1.0f - dx); \quad //\text{two floating point operations}
\]

\[
\text{val} = \text{fmaf}(-dx, v, v); \quad //\text{one operation with better accuracy}
\]
Primary reconstruction bottleneck:

- Image Reconstruction is “Memory Bound”
- Patterns of memory writes determined by input data
- 2x2 write pattern may cause region overlap between blocks

Solutions:

- Vectorized of loads input data
- reorder input data to target writes within same output region
- Pre-accumulate sums within block before atomic updates
//Big load of 16 values as uint4 of llhData array

uint4 temp_load_val = D_llhdata[((((k<<1)<<4)+baseIdx)>>4];

//16 values stored in 128 bits (or 16 bytes, 1 value per byte)

//breakdown into 4 groups of 4 bytes

uchar4 group0;

group0 = reinterpret_cast<uchar4 *>(&temp_load_val.x);

//cast 32 bit uint4 down to 4 8 bit unsigned chars
(which will be cast to floats)
Total number of worker threads = \( 400 \times 100 \times 100 = 4,000,000 \)

Each thread loads 64 values from x and y detector data (total), and 32 values from llhdata, for a load total of 96 input values.

Per each 32 llhdata loads there are 128 global memory updates

Each thread loads \( 64 \times 4 = 256 + 32 = 288 \) bytes, and writes out \( 128 \times 4 = 512 \) bytes, operating on 800 bytes worth of memory

Total memory operations done per reconstruction = 
\( 4,000,000 \times 800 = 3.2 \) GBs,

it takes 19-23 ms which is \textbf{139-168 GBs} of utilized memory bandwidth
Detector features become patterns with regular spacing.
Post Processing of Reconstructed Image

- Apply Normalization
- Alpha Correction to remove periodic artifacts
  - Box filter, 2D Separable Convolution
  - Comb filter, 2D Separable Convolution
  - Apply to Filters to raw image
- Transpose
Raw Image Reconstruction Output
Reconstruction Output after Alpha Correction
Runtime Custom Features for Users

- Ability to dynamically adjust current focal plane (z dimension) in real time
- Ability to dynamically adjust image offset spacing in real time
- Ability to toggle enhancement features during imaging
Real-time Imaging

Frame: 1 of 171, Downsampling Factor: 2
Future Improvements and Challenges

- Improve global memory write access patterns (main bottleneck)
- Multiple Planes reconstructed concurrently (4 per GPU)
- Plane Selection
- Add additional features based on user input
- Take advantage of scalability of system
Next Generation Multi Plane SBDX

Detector 1

Data receiver alternates between detectors

Pre-Process GPU

K20

Plane Selection GPU

Reconstruction GPUs

Titan

Titan

Titan

Titan

Real time Display

Detector 2
Acknowledgements

- Paul Kahn
- Jamie Ku
- Augustus Lowell
- Christopher Ellenor

- NIH SBIR grant: 5 R44 EB015910-03