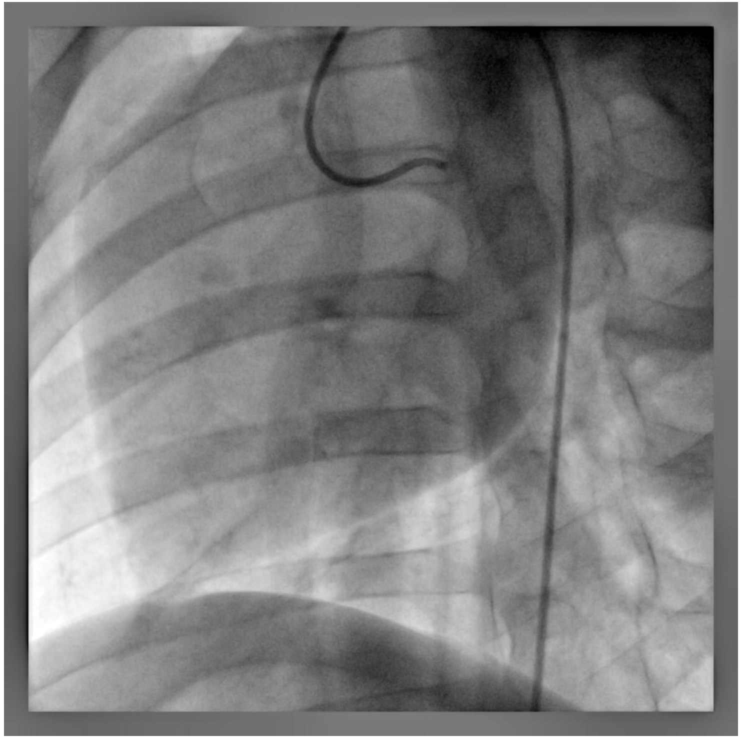
Real-Time Multi-Plane Tomosynthesis Using GPUs

Tobias Funk, Oleg Konings

Triple Ring Technologies

www.tripleringtech.com

Conventional fluoroscopy



Interventional Procedures

Many Application

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

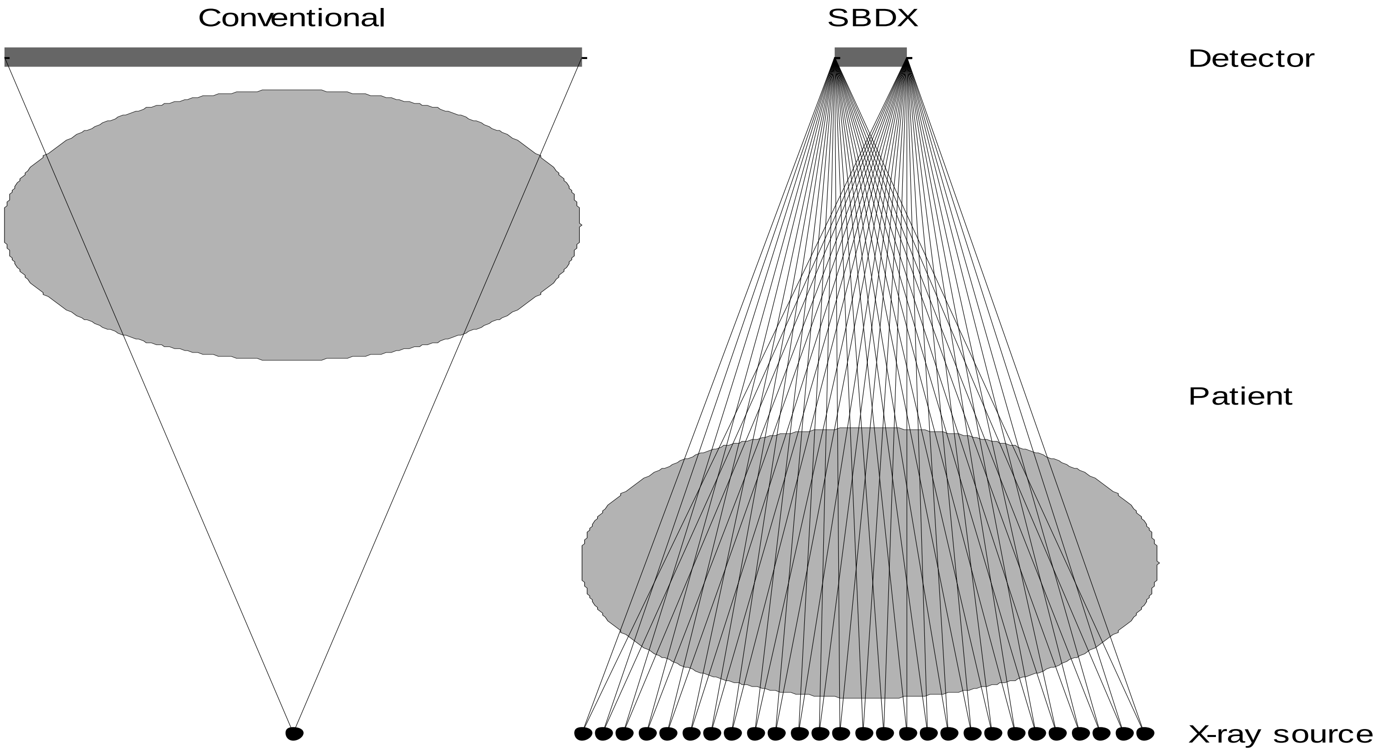
Advantages

- Minimally invasive
- Fast recuperation times
- High success rate

Disadvantages

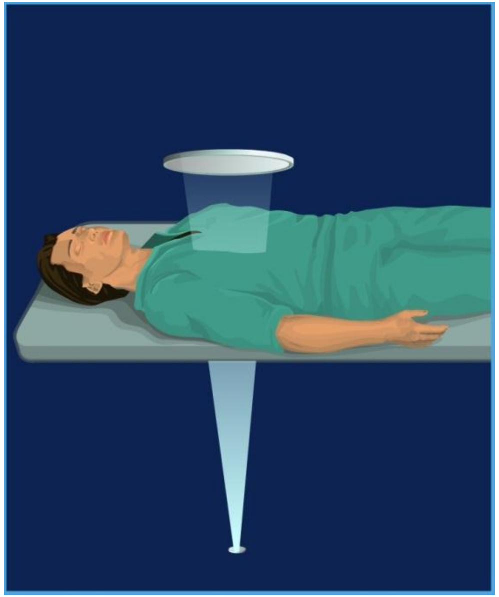
- High radiation dose
- No 3d information

Scanning beam digital X-ray (SBDX) system



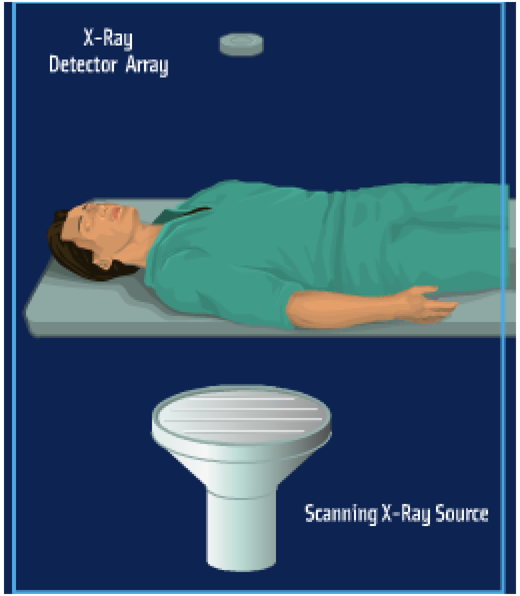
System Comparison

Conventional



Simple shadowgram image

SBDX



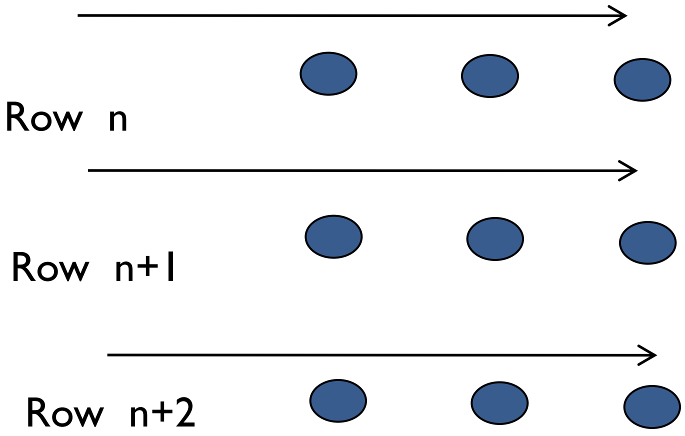
Many small images captured rapidly

Images reconstructed in real time

Large entrance area – High efficiency detector

Image generation

Source array

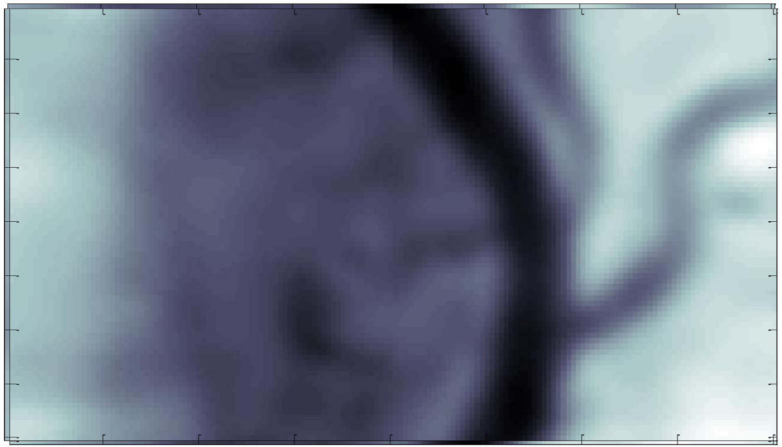


Row n

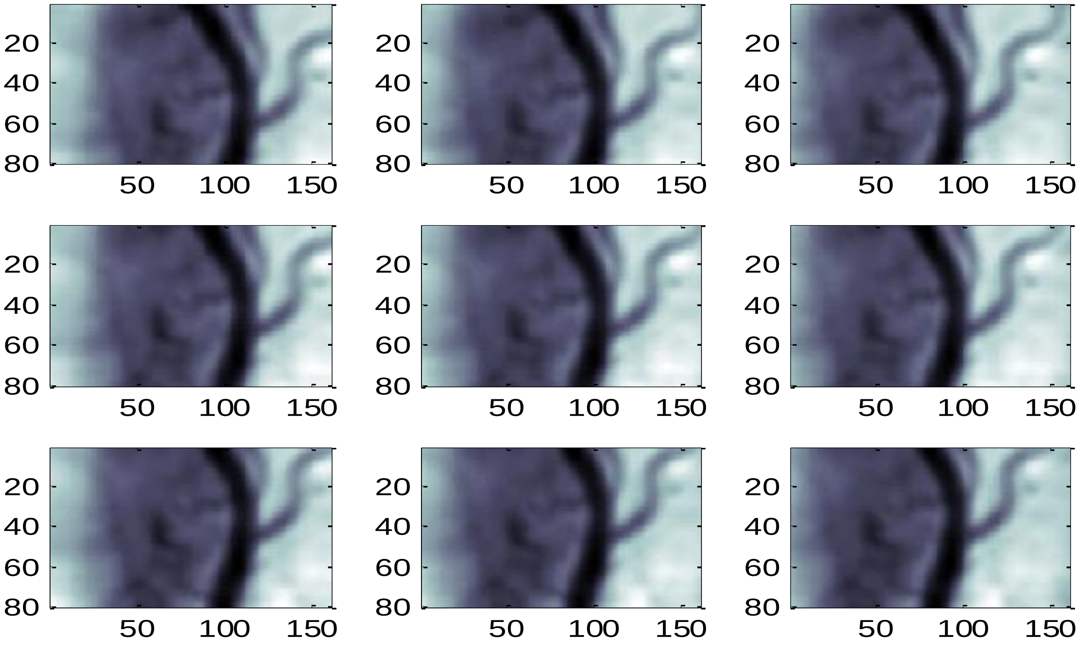
Row n+1

Row n+2

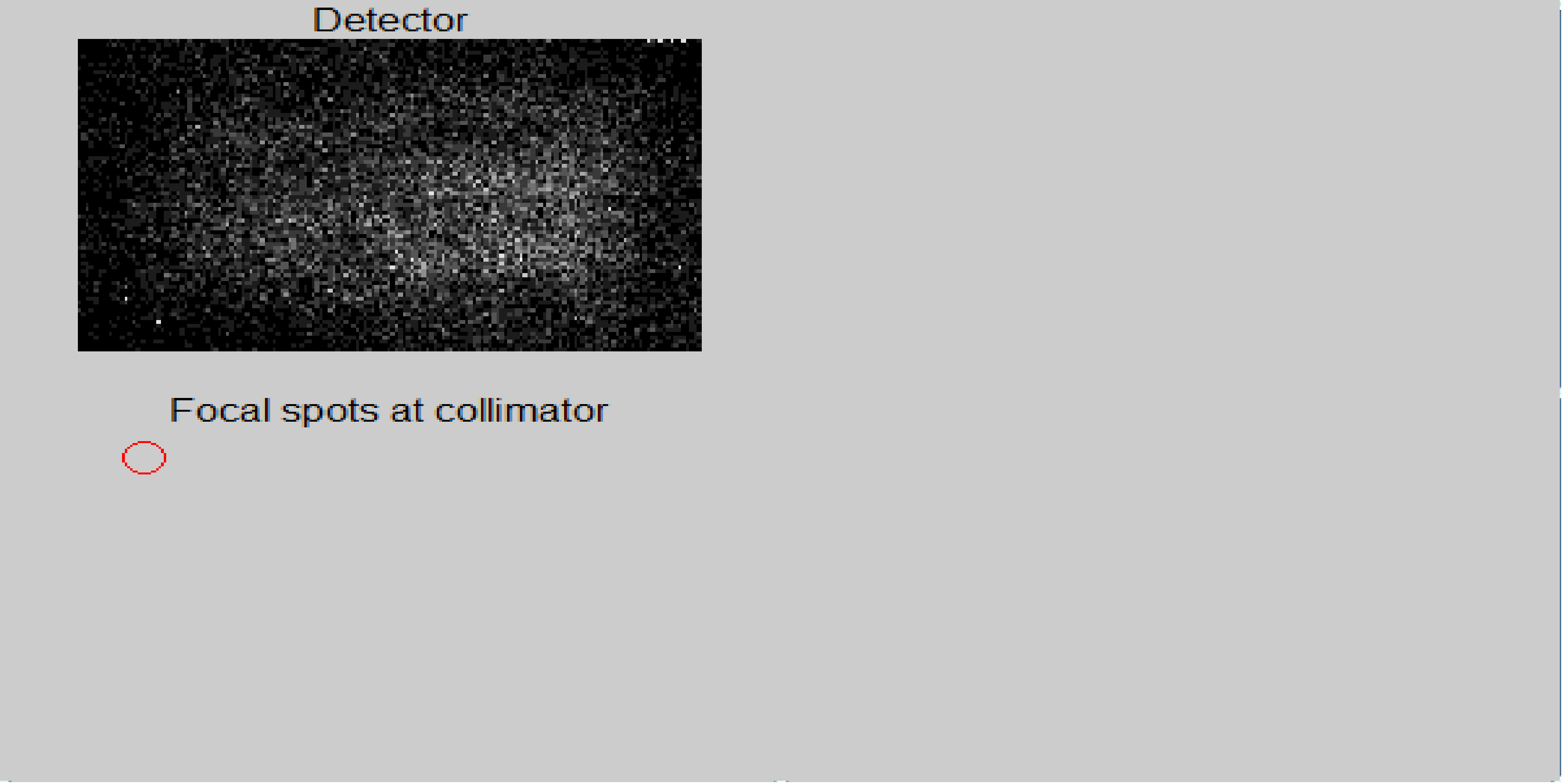
Detector



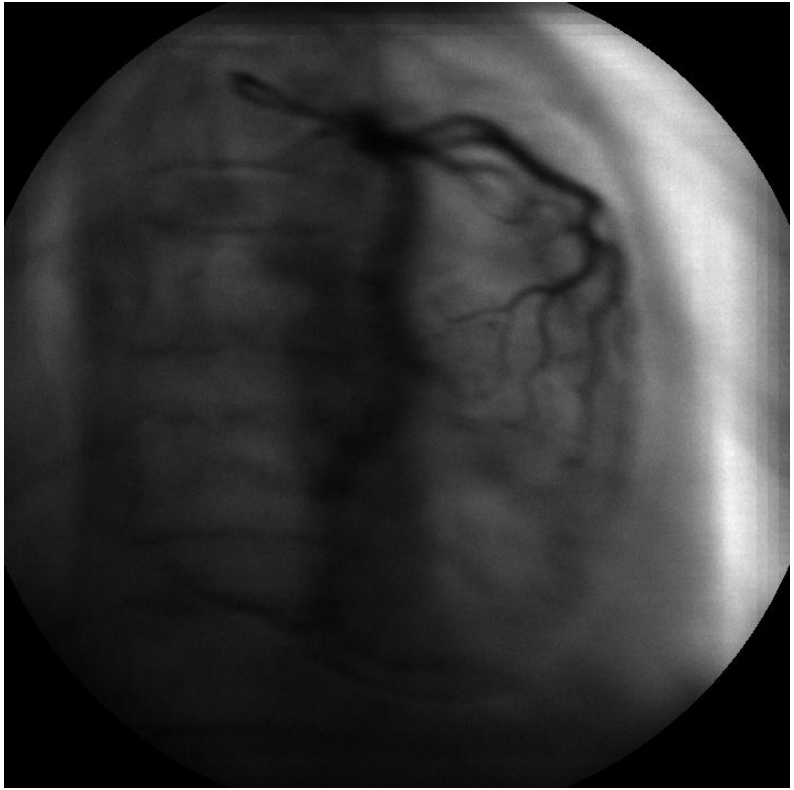
Every source position illuminates the object from a slightly different angle



Reconstruction



Tomosynthesis: reconstruction of focal planes



www.tripleringtech.com

© 2014 Triple Ring Technologies

Tomosynthesis

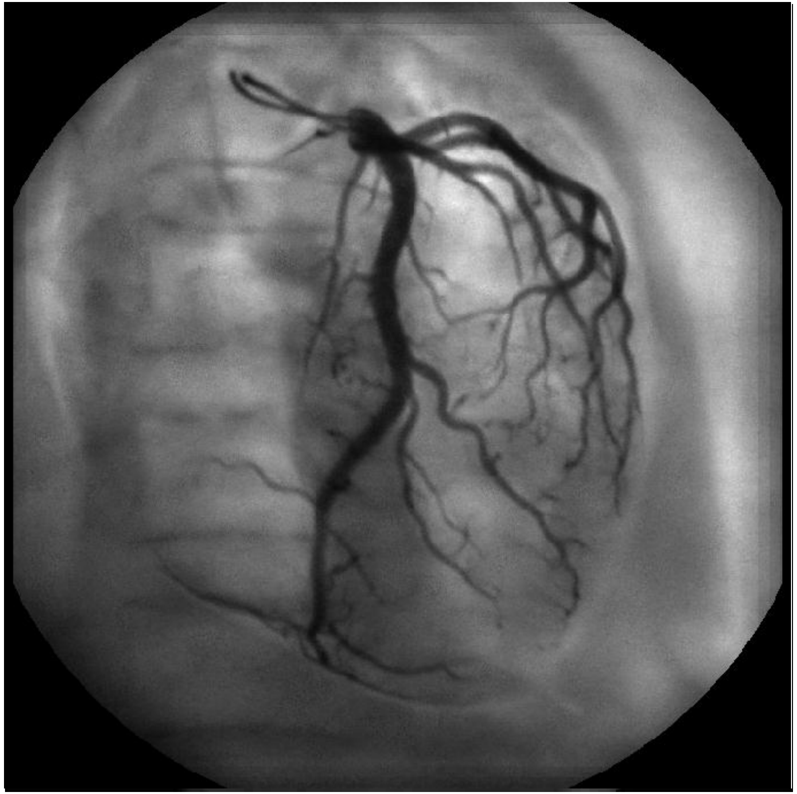
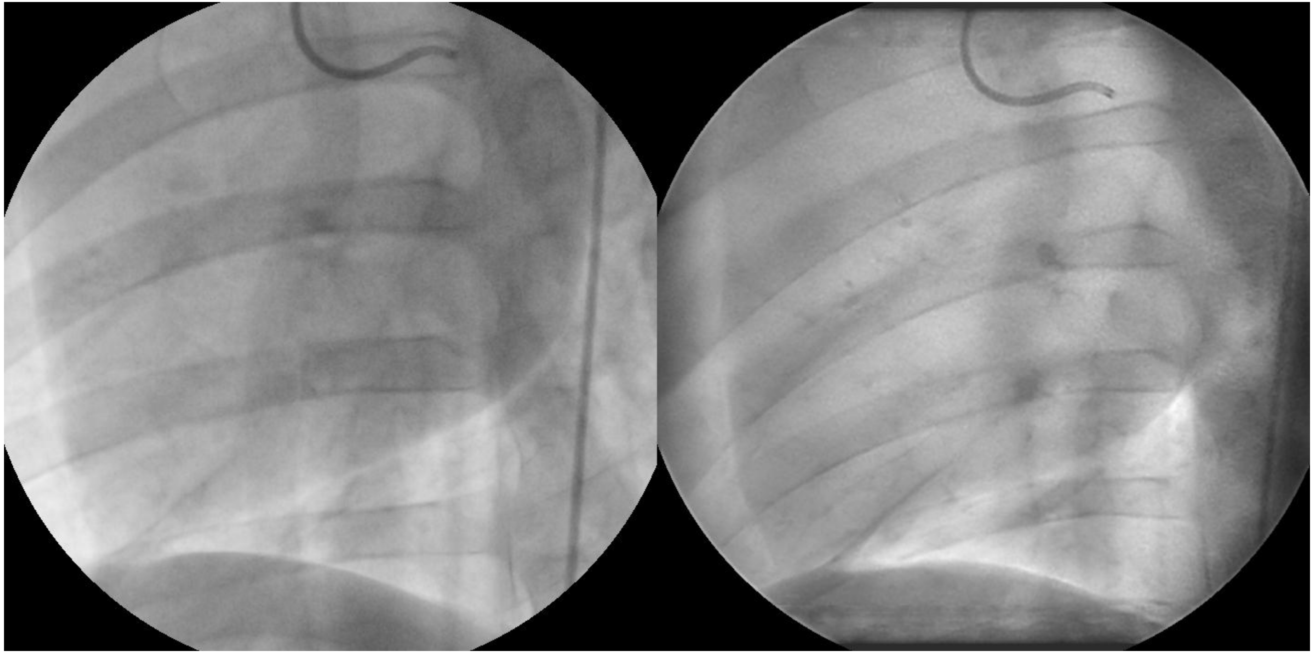


Image comparison
Philips FD10 vs. SBDX

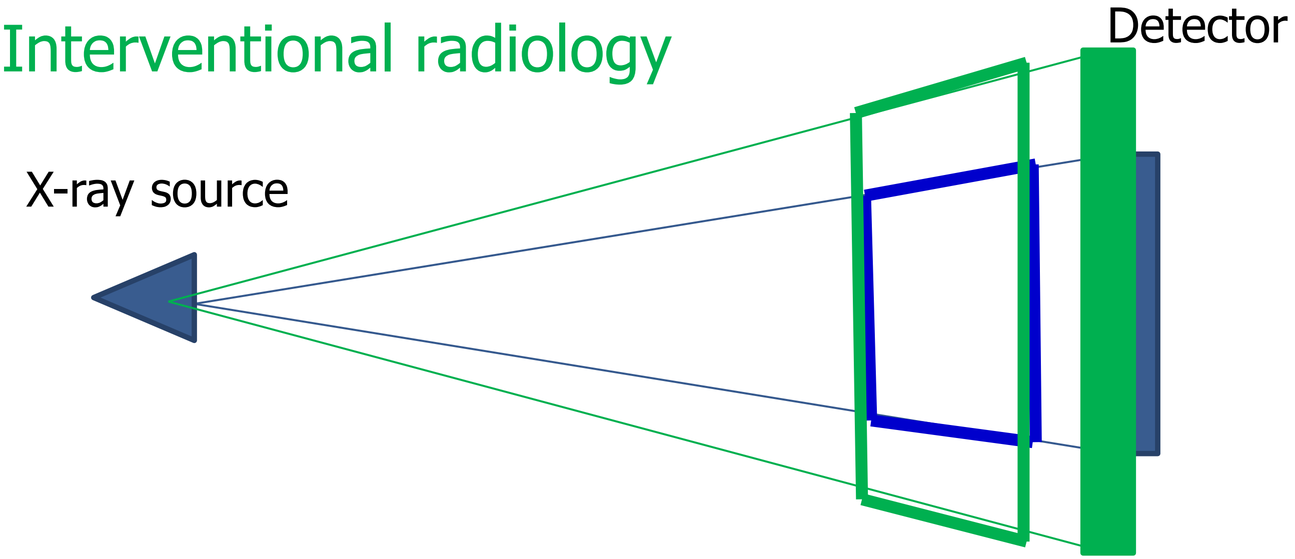


4-fold exposure reduction

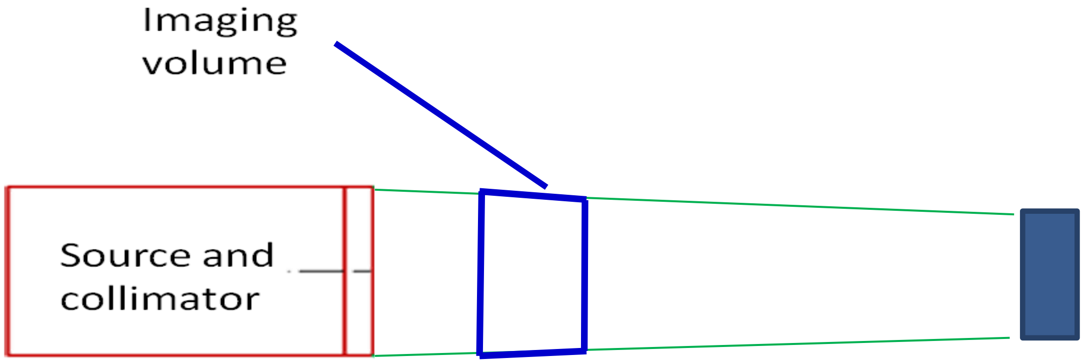
How to address the interventional radiology market?

Interventional cardiology

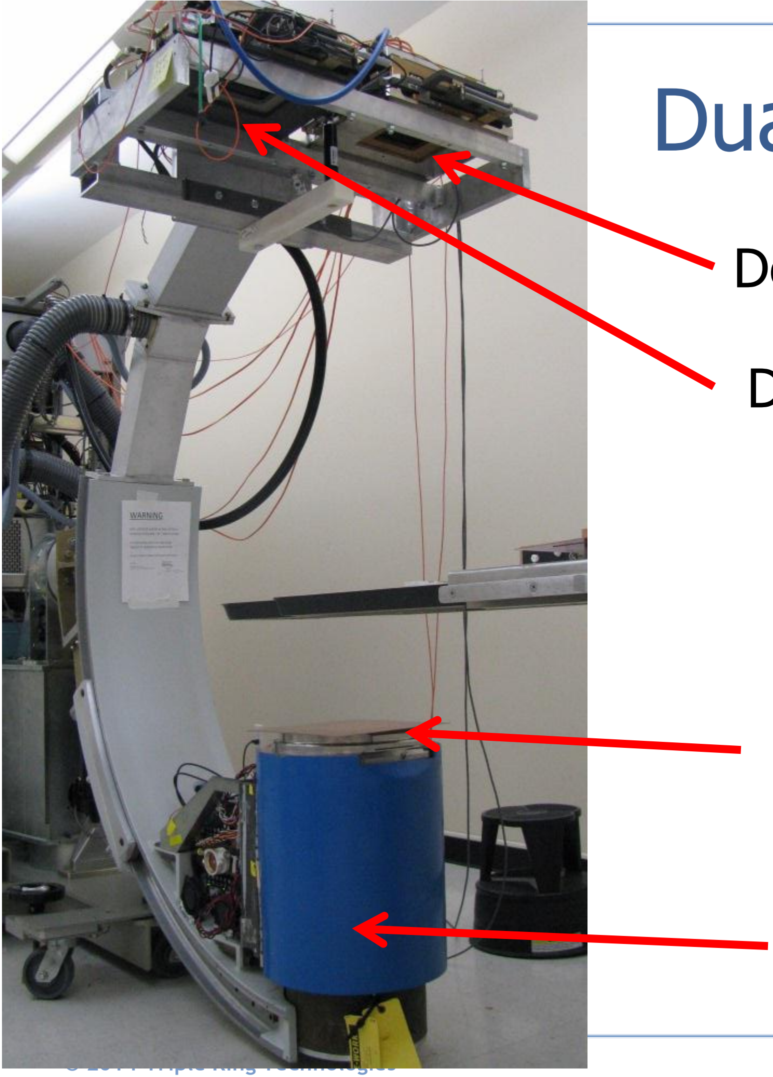
Interventional radiology



Interventional cardiology SBDX



Dual-detector SBDX system



Detector 1

Detector 2

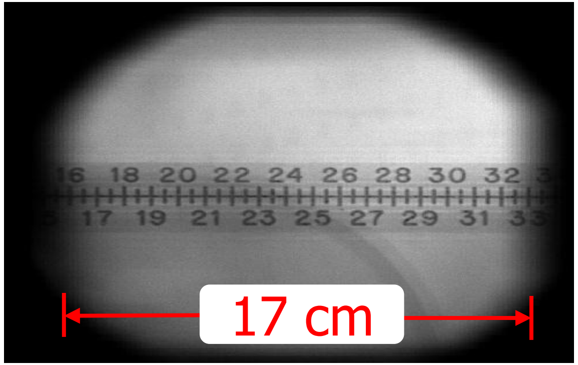
Collimator

X-ray source

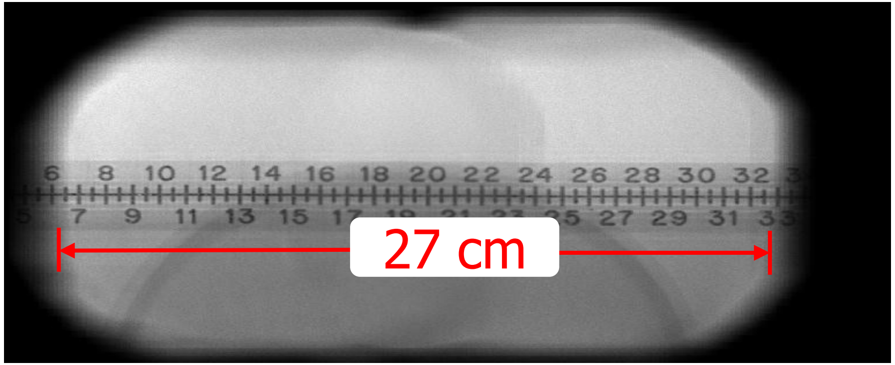
Results

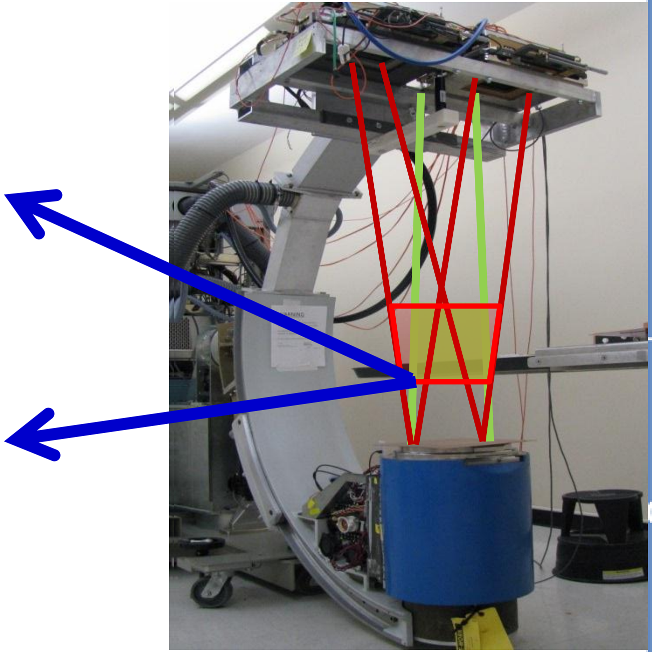
Bottom Imaging Volume
z = 34.2 cm

Cardiology



Radiology

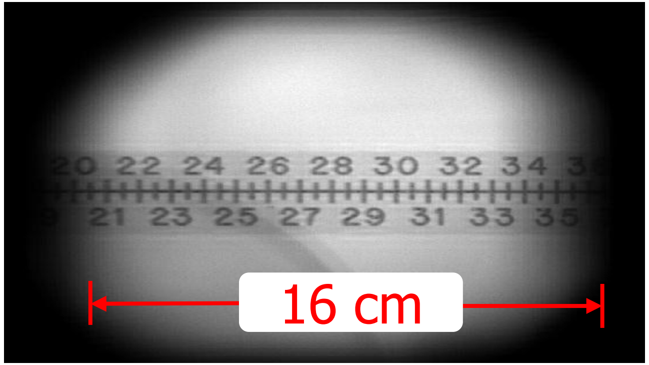




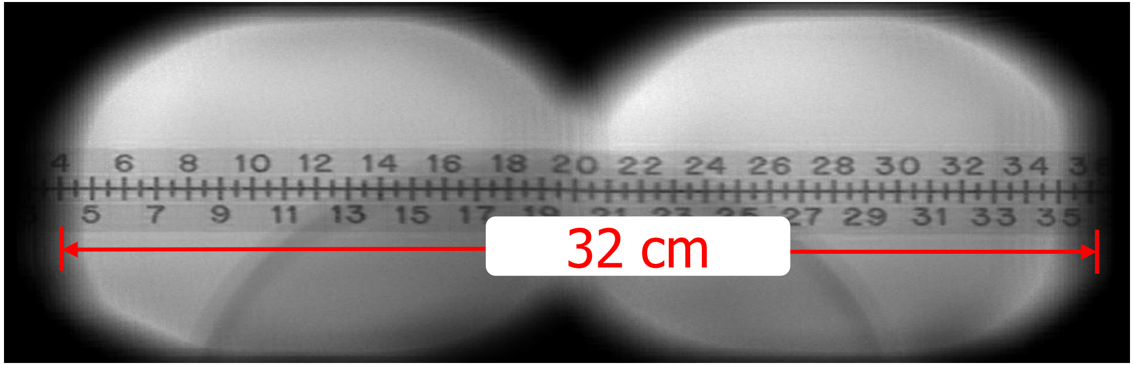
Results

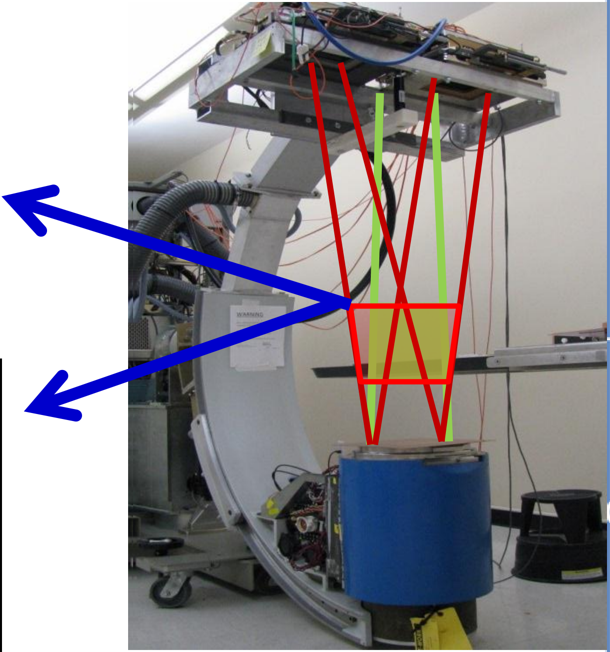
Top Imaging Volume
z = 53.4 cm

Cardiology

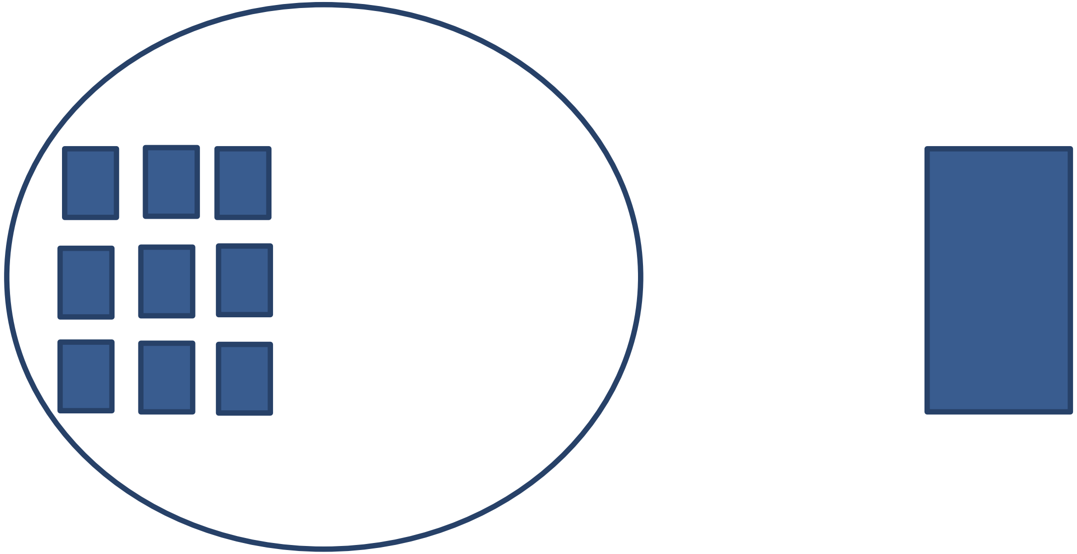


Radiology





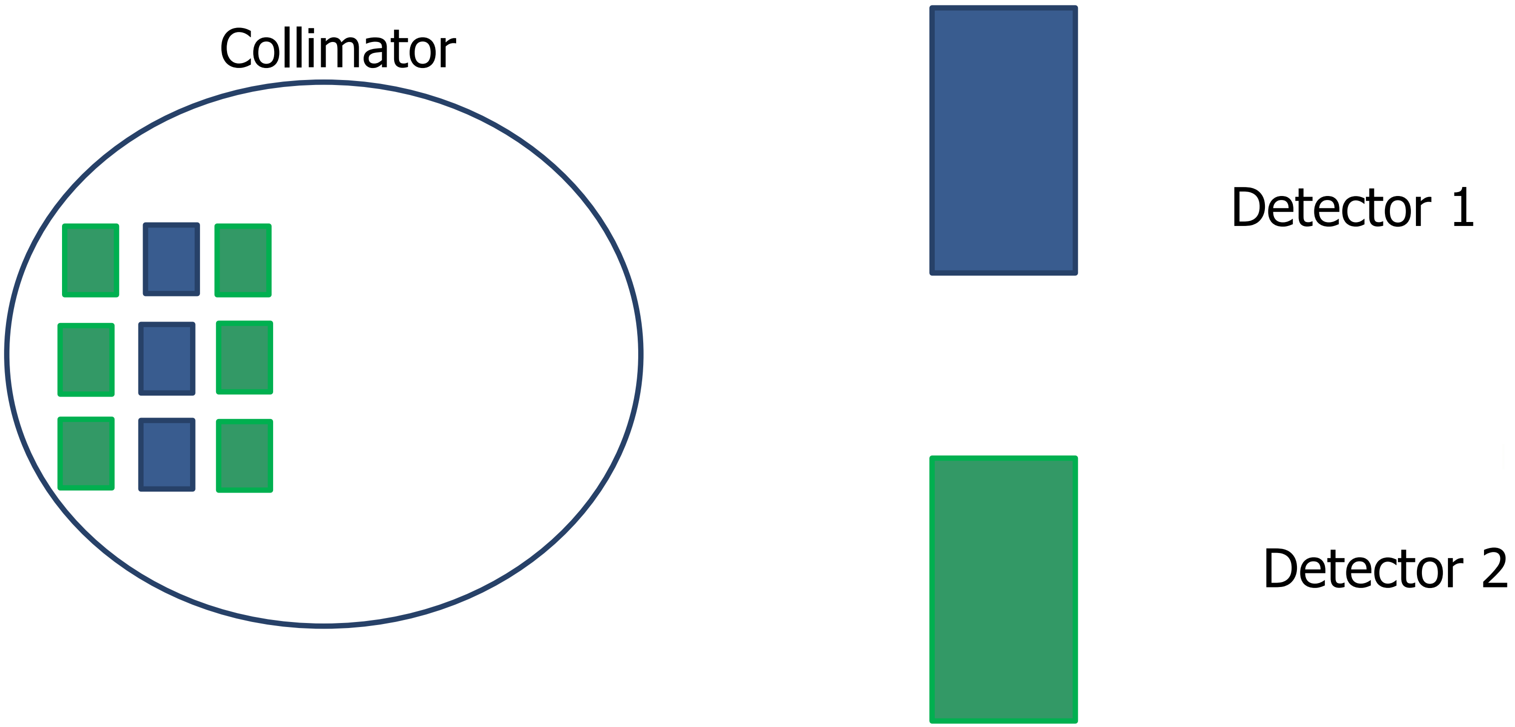
Single detector SBDX



Collimator

Detector

Dual detector SBDX

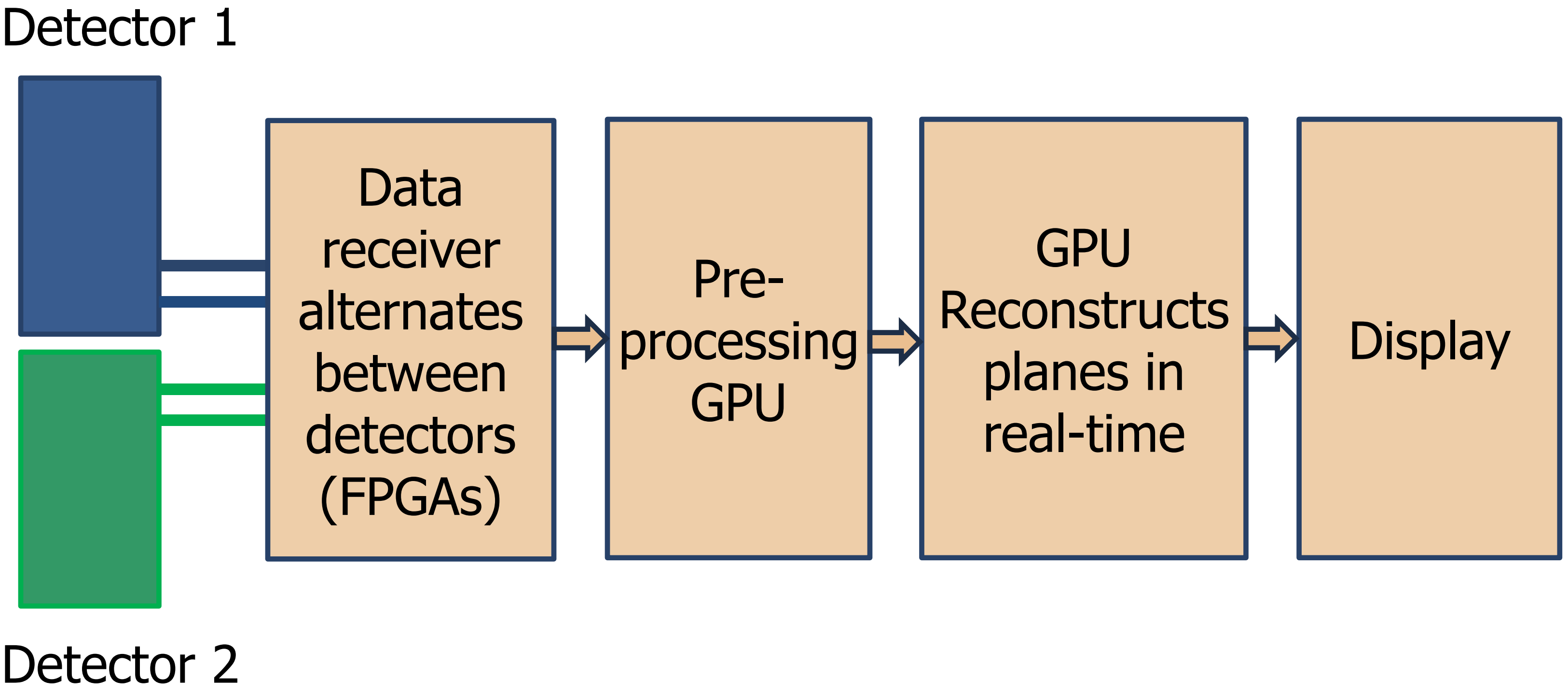


Data complexity

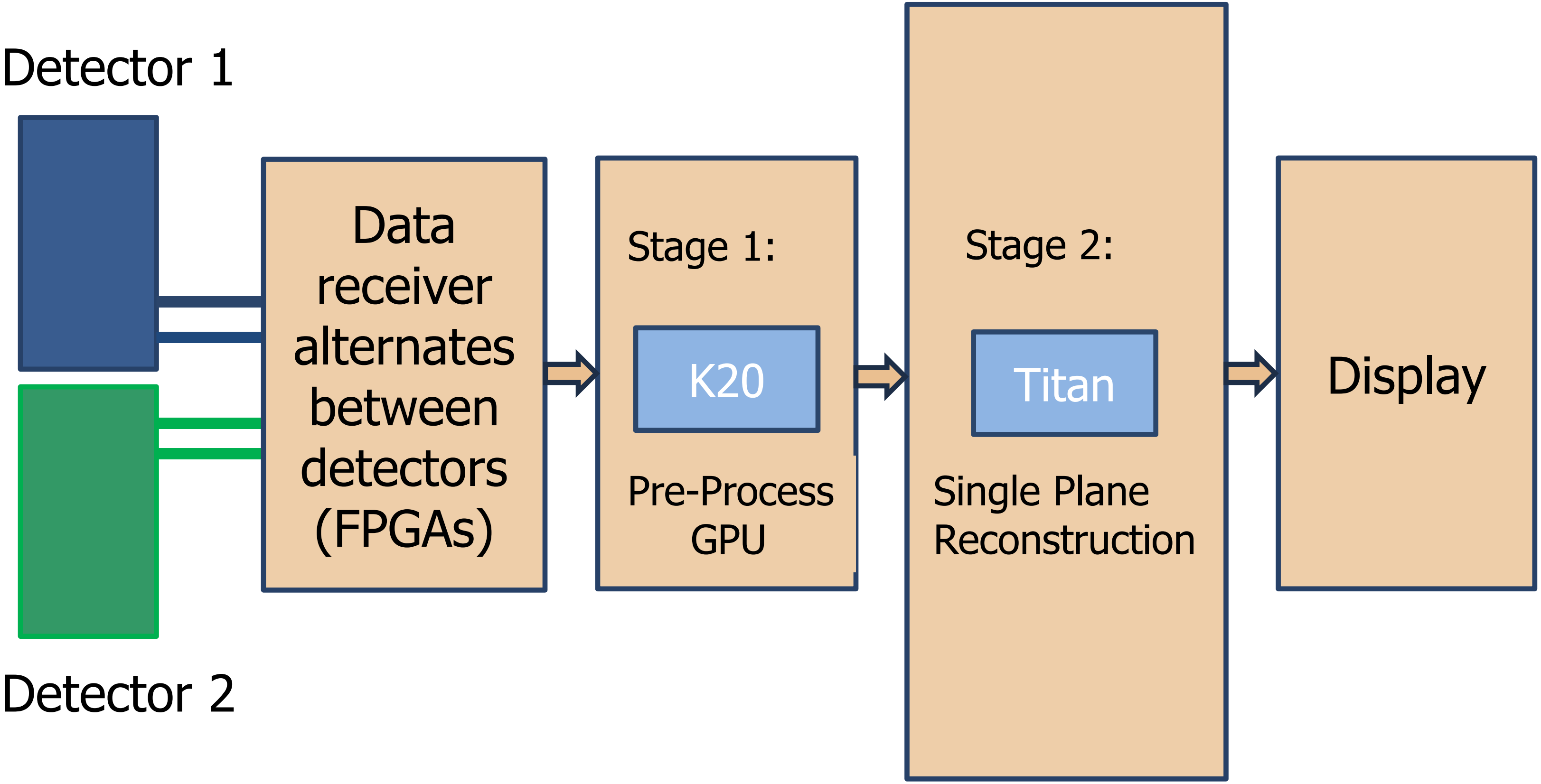
- Detector size 160x80 pixel
- 100x100 detector images
  - (50x100) from each detector
- Reconstruction into planes of 1000x2000 pixels
- Post processing
- 32 planes
- Plane selection

In 133 ms

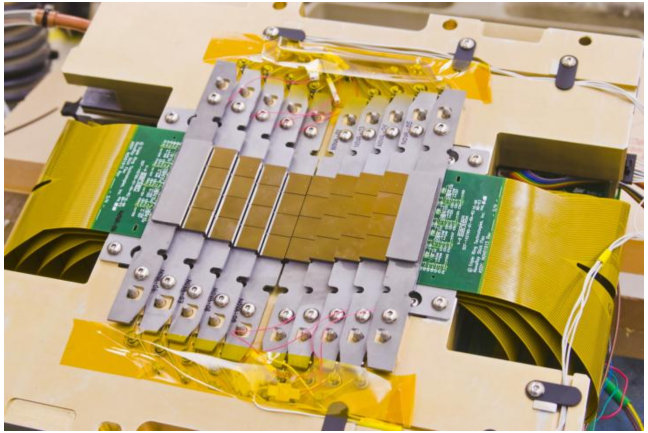
Dataflow in the dual-detector system

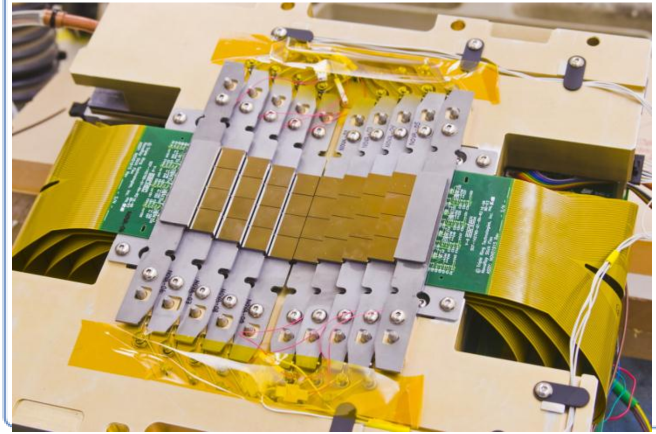

Detector 1
Data receiver alternates between detectors (FPGAs)
Pre-processing GPU
GPU Reconstructs planes in real-time
Display
Detector 2

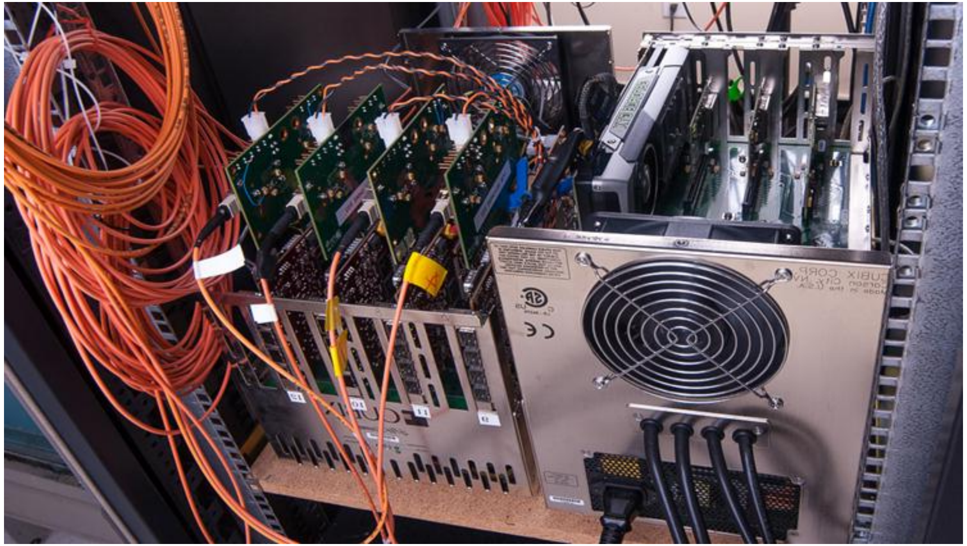
Current Single Plane SBDX System


Detector 1
Data receiver alternates between detectors (FPGAs)
Stage 1:
K20
Pre-Process GPU
Stage 2:
Titan
Single Plane Reconstruction
Display
Detector 2

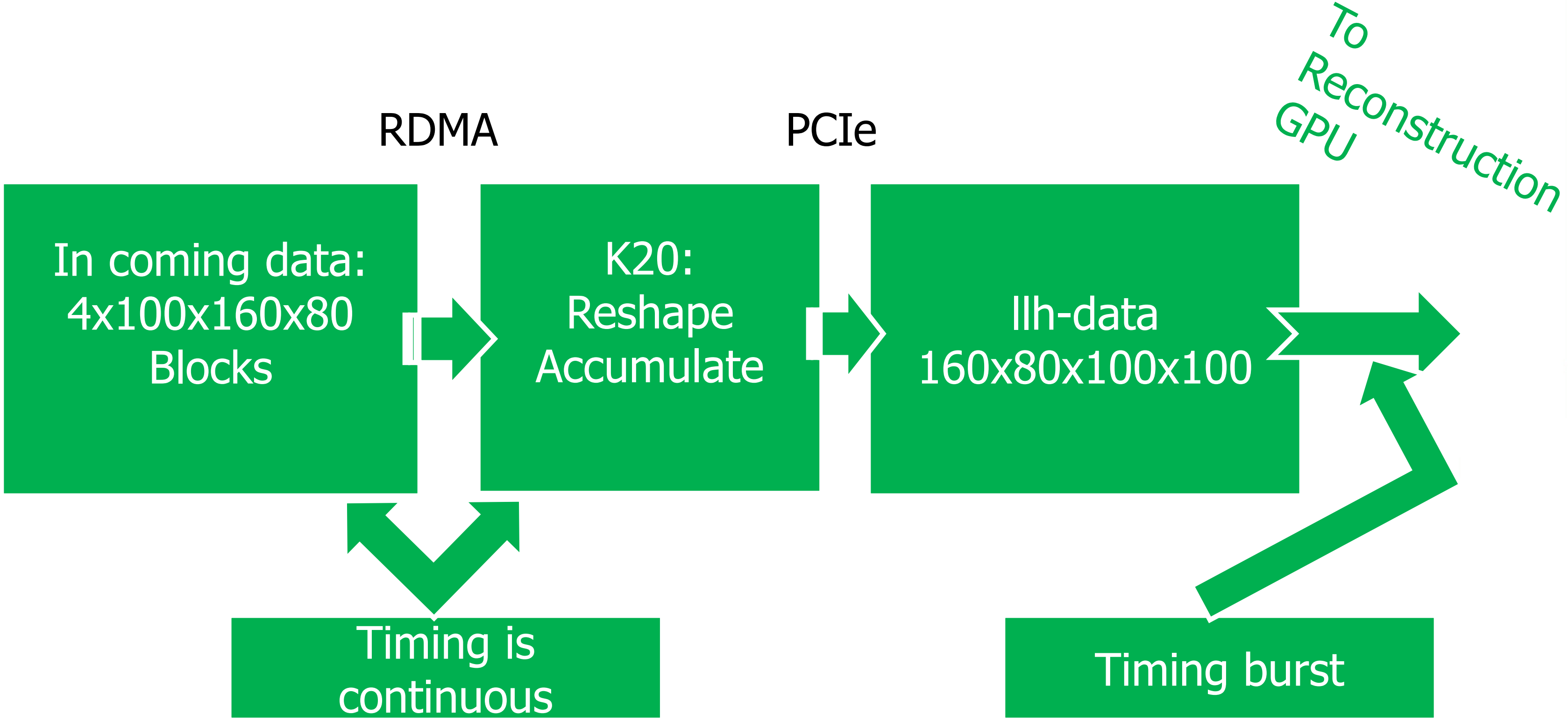
Detectors and Chassis Hardware







Stage1: Pre-processing GPU



RDMA

PCIe

In coming data:
4x100x160x80
Blocks

K20:
Reshape
Accumulate

llh-data
160x80x100x100

To Reconstruction GPU

Timing is continuous

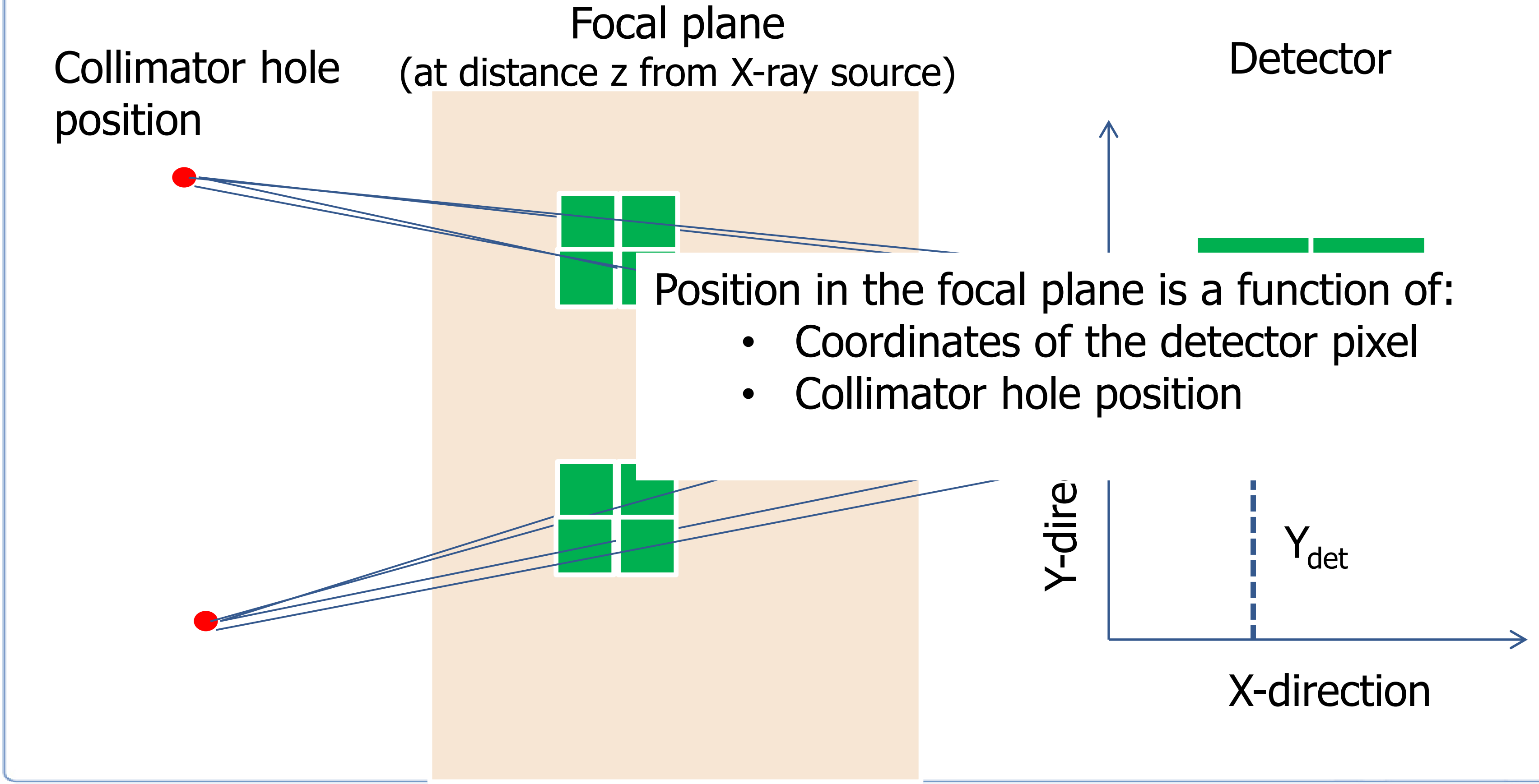
Timing burst

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

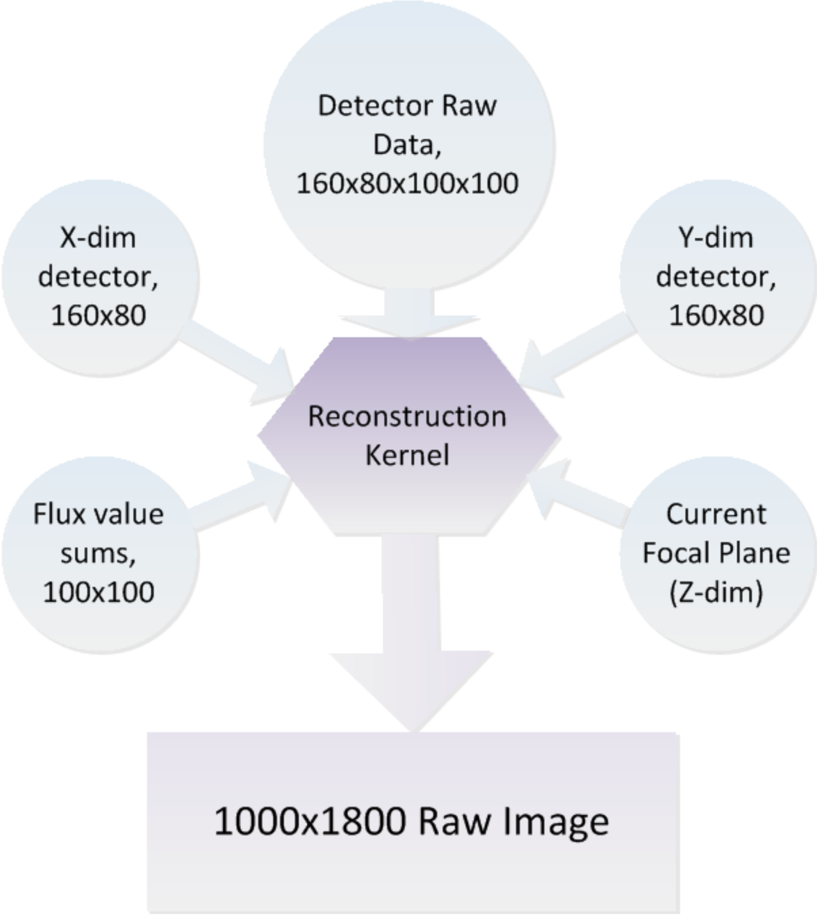
Reconstruction



Position in the focal plane is a function of:

- Coordinates of the detector pixel
- Collimator hole position

Reconstruction Kernel



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:

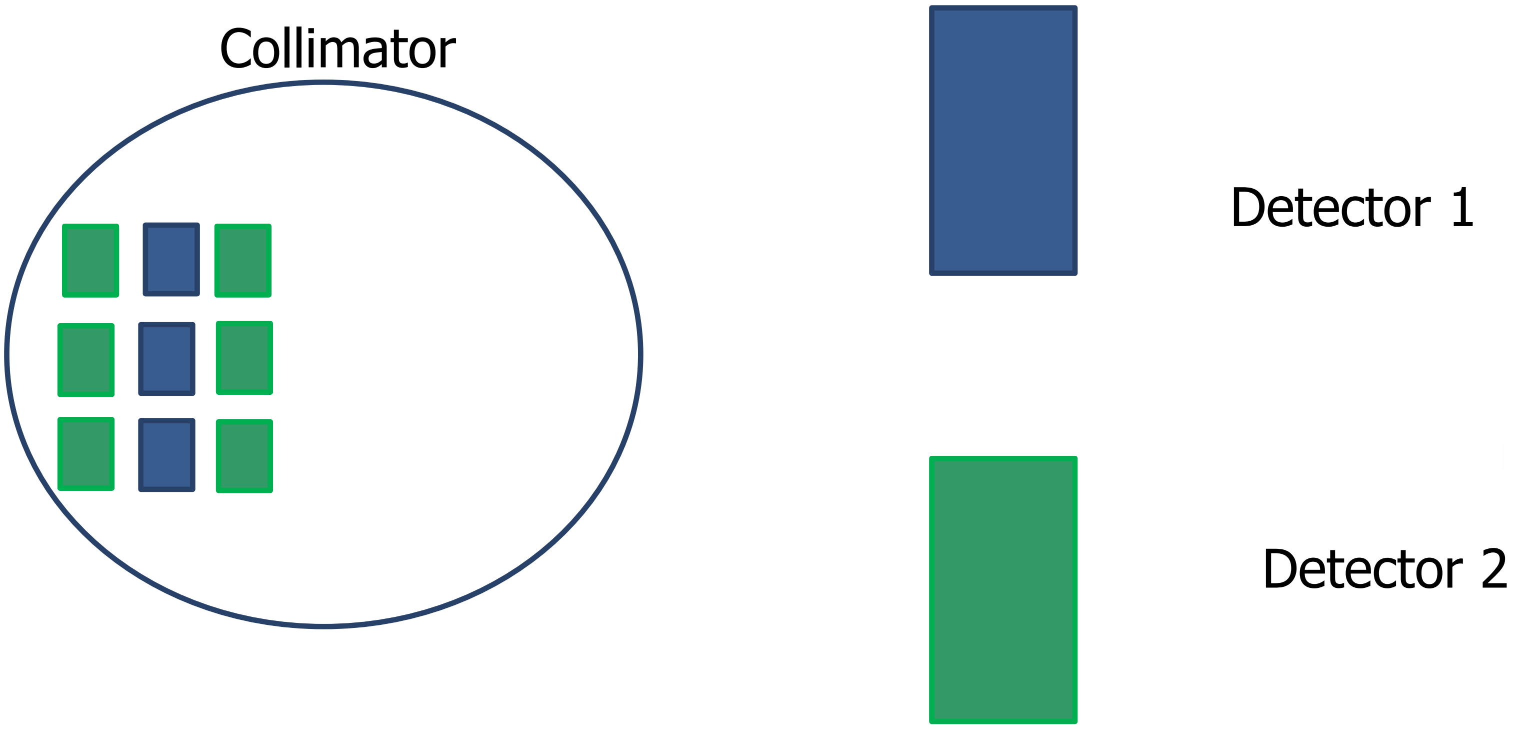
- To coalesce global memory reads and writes
- To maximize 32-bit GFLOPS
- To maximize occupancy of GPU cores

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

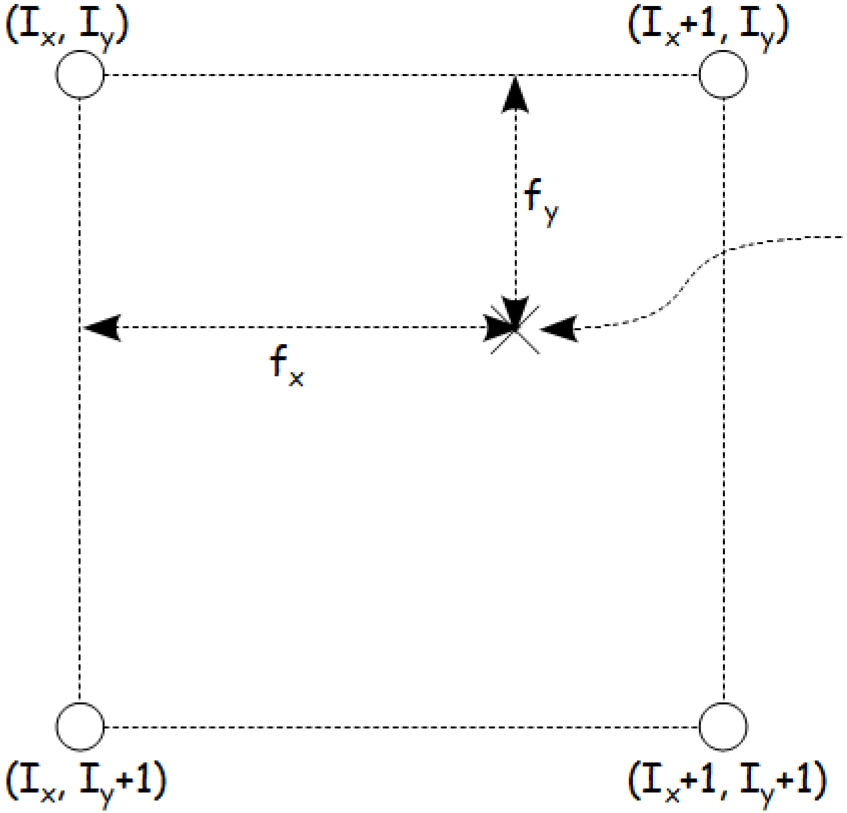


Combine Dual-Detector Data into Single Image

```
1 $row \leftarrow blockIdx.y,$
   $col \leftarrow blockIdx.z,$
  $Collimator\_index \leftarrow threadIdx.x + blockIdx.x * blockDim.x$

2 if $threadIdx.x == 0$ then /* fill in __shared__ memory for broadcast */

3 if $blockIdx.y\&1$ then
            image offset to account for left detector
else
            image offset to account for right detector
```



Point of intersection of the back-projected ray

Weights (Bilinear coefficients)

(I_x, I_y)	$(1-f_x) * (1-f_y)$
(I_x+1, I_y)	$f_x * (1-f_y)$
(I_x, I_y+1)	$(1-f_x) * f_y$
(I_x+1, I_y+1)	$f_x * f_y$

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:

```
row_index = floor(x_det[k]*z_value) - 1 + collimatorRow*m + offset;
col_index  = floor(y_det[k]*z_value) - 1 + collimatorCol*m    + offset;
```

Then write weighted values to the square region in pattern:

```
(row_index, col_index), (row_index, col_index+1),
(row_index+1, col_index), (row_index+1, 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

val= v*(1.0f – dx); *//two floating point operations*

val= fmaf(-dx, v, v); *//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

Vectorized Global Memory Loads

```
//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*100*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.

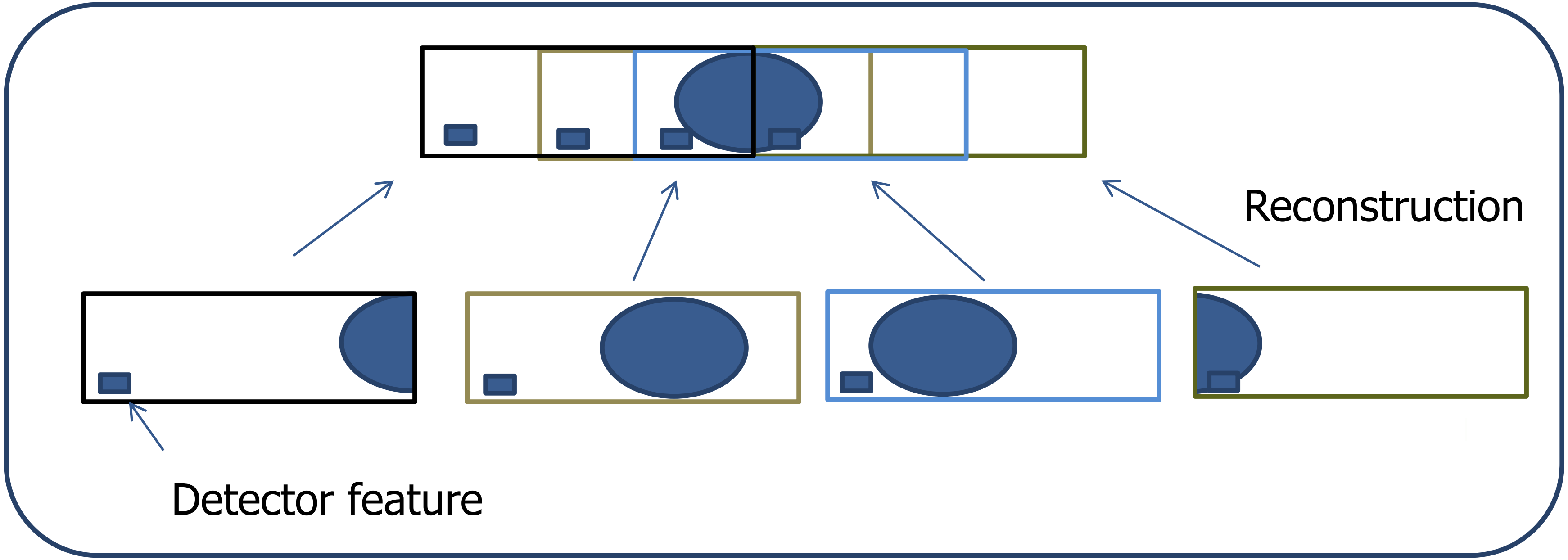
Per each 32 llhdata loads there are 128 global memory updates

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

Total memory operations done per reconstruction= 4,000,000*800 = 3.2GBs,

it takes 19-23 ms which is **139-168 GBs** of utilized memory bandwidth

Alpha Correction

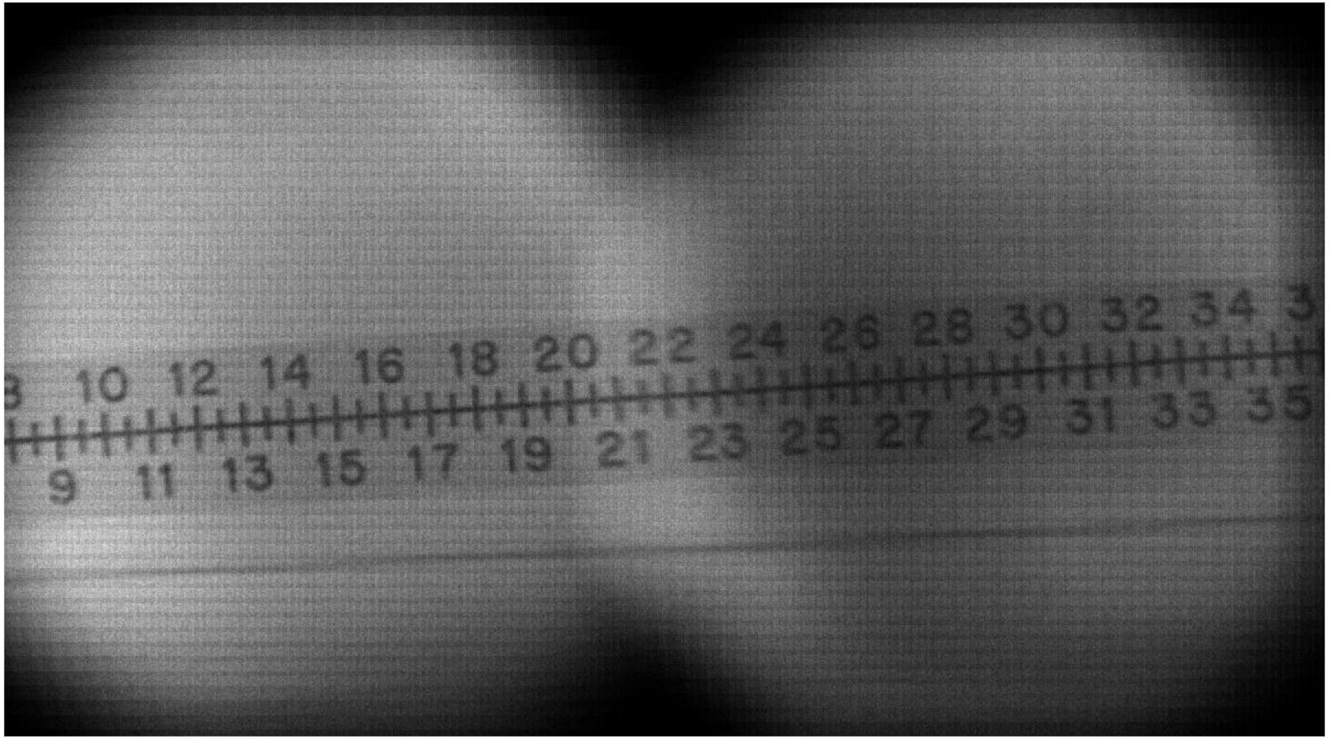


- Detector features become patterns with regular spacing

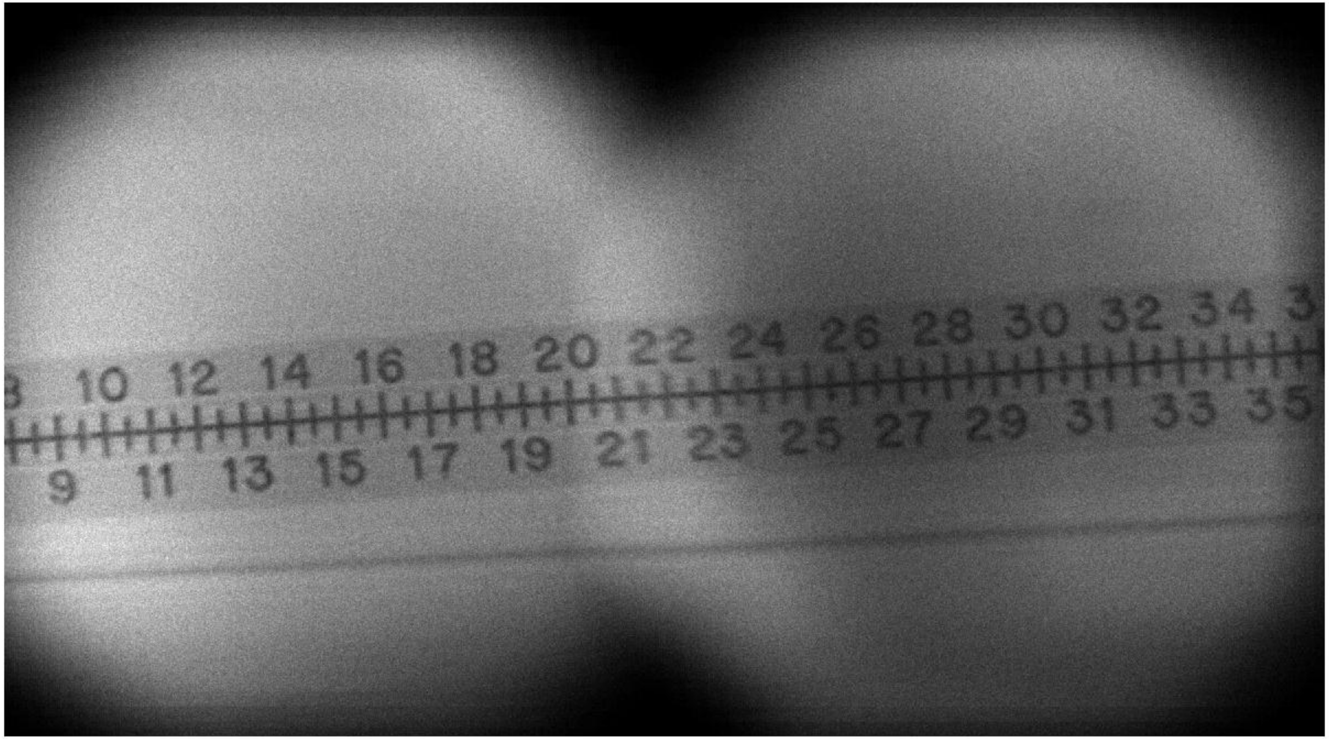
Post Processing of Reconstructed Image

- 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



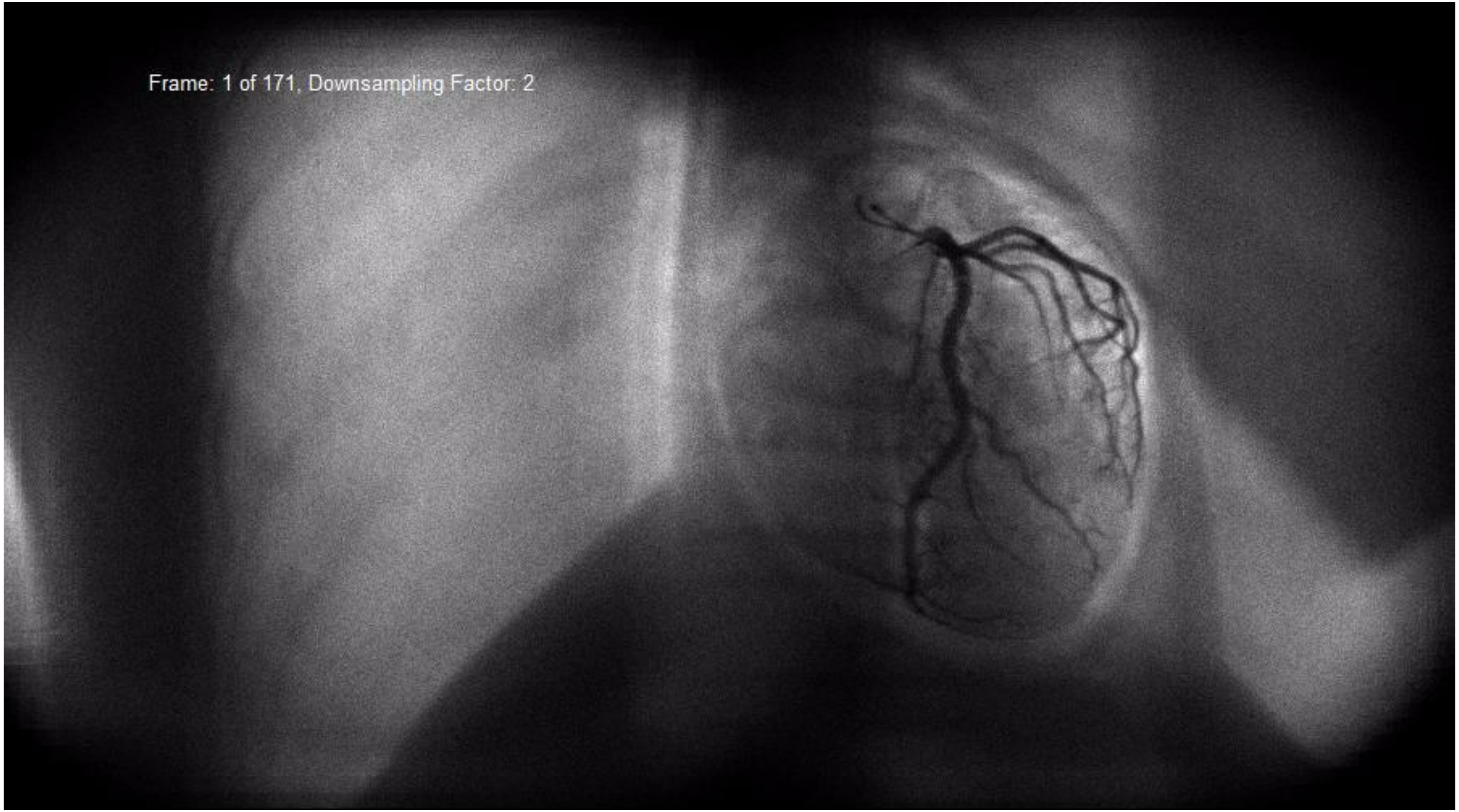
Reconstruction Output after Alpha Correction



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

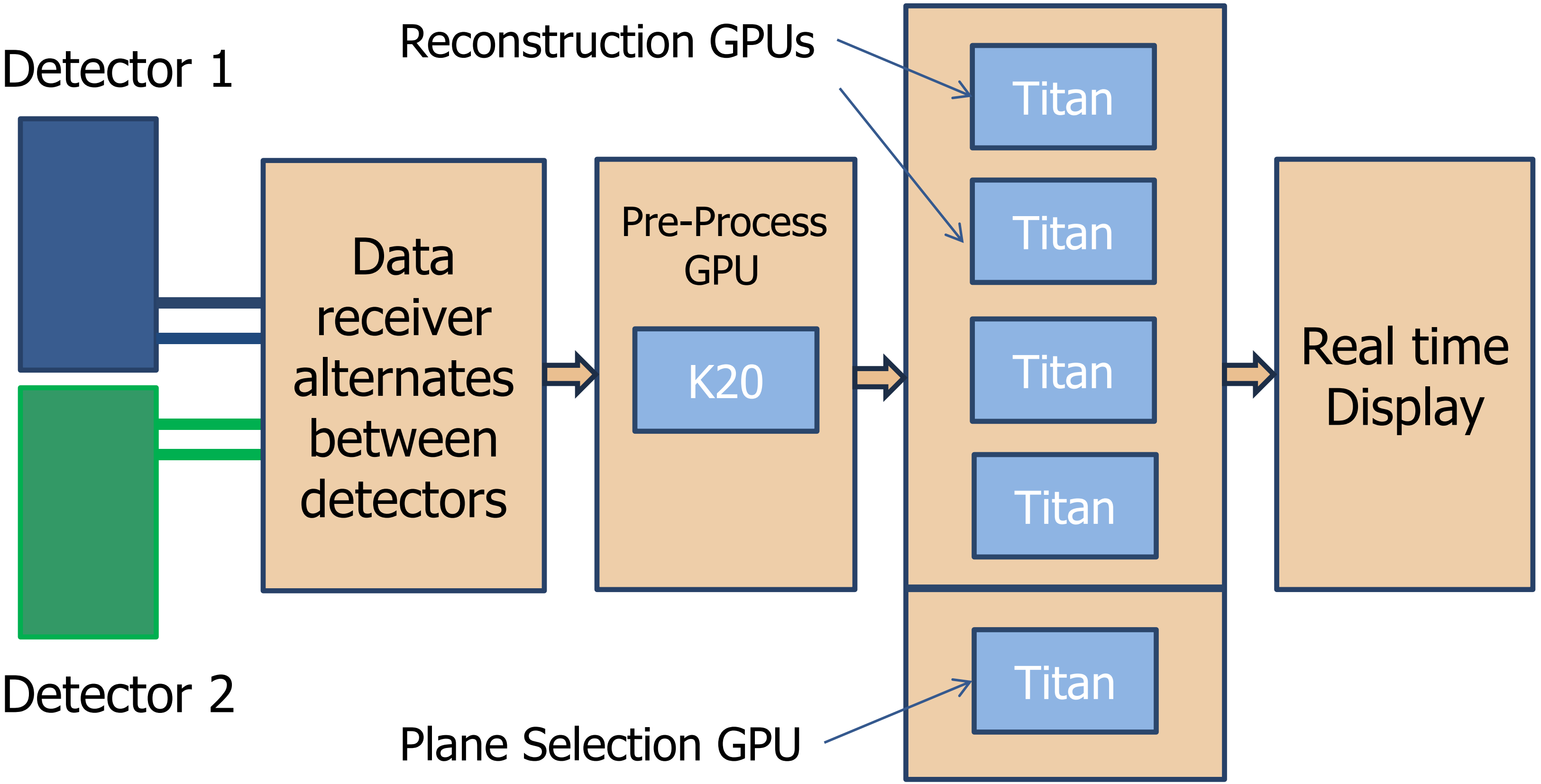
Real-time Imaging



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



Acknowledgements

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

- NIH SBIR grant: 5 R44 EB015910-03