Status report su algoritmi per CT

Giovanni Di Domenico

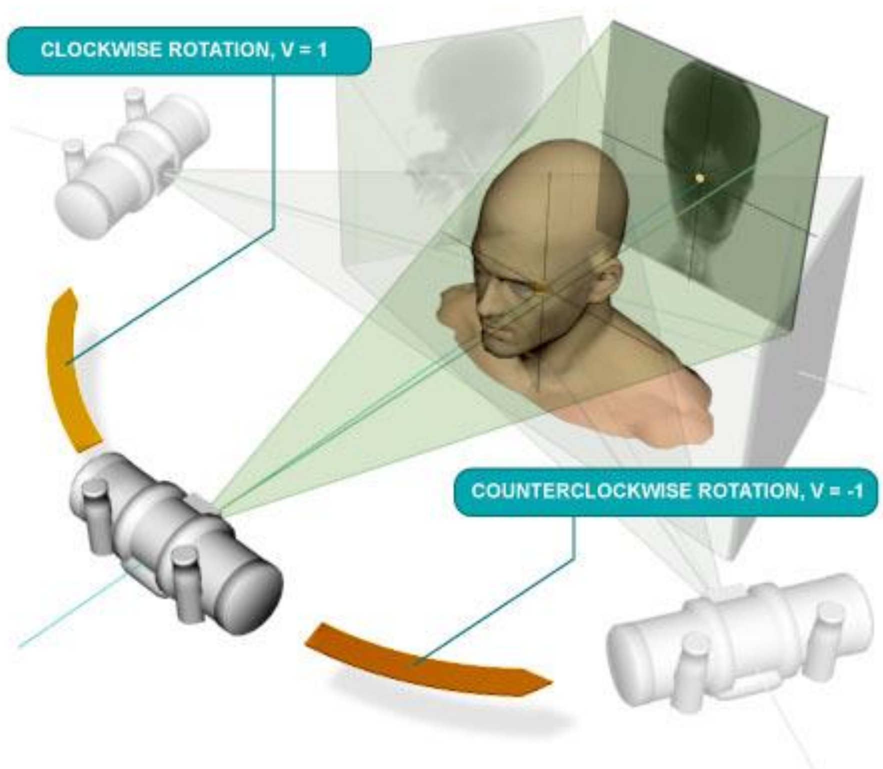
Università degli Studi di Ferrara & INFN Sezione di Ferrara

23 March 2016 – Ferrara

Outline

- Cone-beam CT systems
- Feldkamp Davis Kress (FDK) algorithm
- GPU implementations
- Optimization steps
- Iterative Reconstruction
- DBT and OpenRTK

Cone-beam CT system

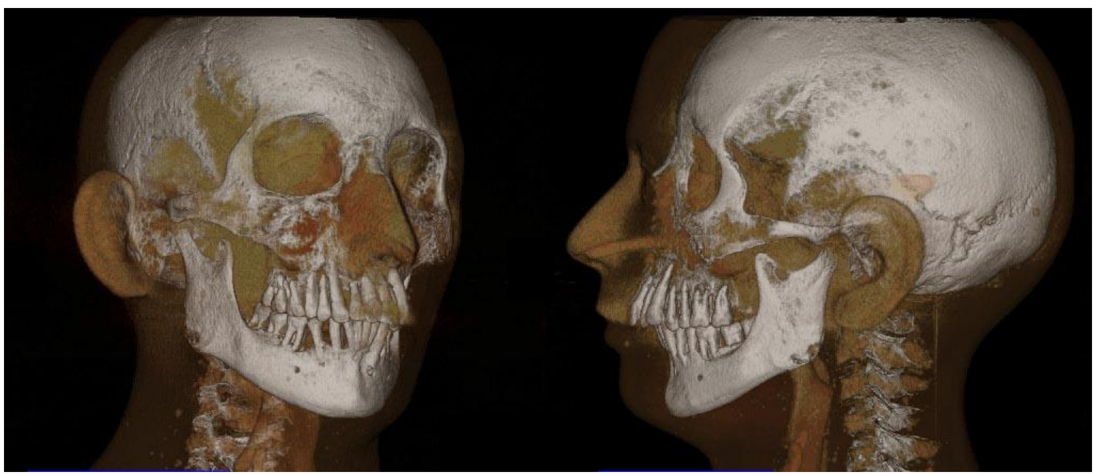


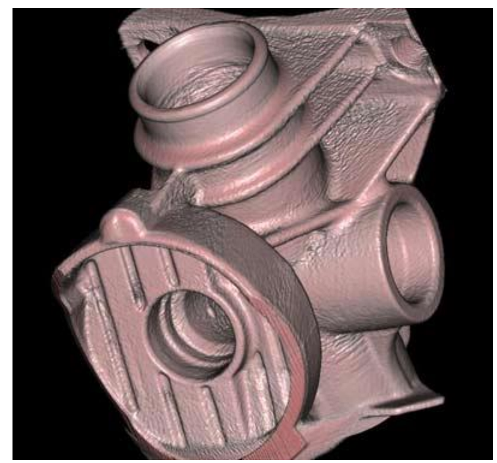
Tipically, a CBCT system tipically has:

- X-ray tube (30kVp – 120 kVp),
- flat panel detector (2048 x 2048),
- rotating gantry (200-1000 angular step),
- reconstructed volume size ~ 512^3

Cone-beam CT applications

- Non-destructive testing
- Dental imaging
- Microtomography
- Image-Guided Interventions





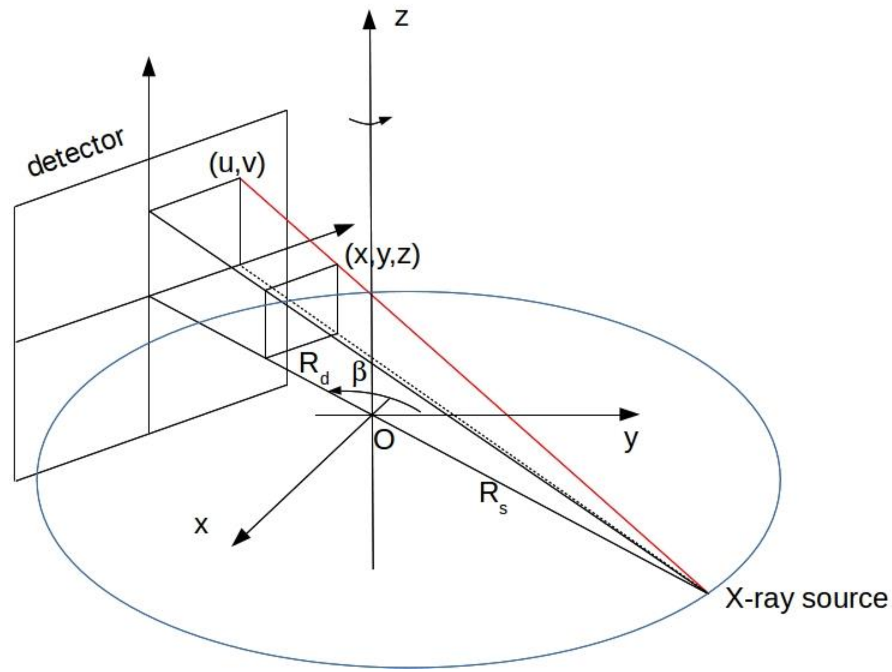
Circular Cone-beam Projection Geometry

In cone-beam CT, the detector acquires the line integral of attenuation coefficient $\mu(x,y,z)$ as function of rotation angle β:

$$g(u,v,\beta) = \int \mu(\vec{r}_O(\beta) + \alpha\hat{\theta})d\alpha$$

where:

- $\vec{r}_O(\beta)$ is the source position as function of rotation angle,
- $\hat{\theta}$ is an x-ray line beam



FDK algorithm - 1

- The Feldkamp-Davis-Kress algorithm is used to obtain an approximate solution of 3D tomographic problem.

$$\hat{\mu}(x,y,z) = \frac{1}{2}\int_0^{2\pi} d\beta \frac{1}{U^2}\int_{-u_m}^{u_m} du \frac{D}{\sqrt{D^2+u^2+v^2}} \cdot g(u,v,\beta)\cdot h(u-u')$$

where

- $\frac{D}{\sqrt{D^2+u^2+v^2}}$ is the cosine weighting function,
- $U = \frac{R_S - x\sin\beta + y\cos\beta}{R_S}$ is the backprojection weighting factor

FDK algorithm - 2

The FDK has three steps:

$$\hat{\mu}(x,y,z)=\frac{1}{2}\int_{0}^{2\pi}d\beta\,\frac{1}{U^2}\int_{-u_m}^{u_m}du\,\frac{D}{\sqrt{D^2+u^2+v^2}}\cdot g(u,v,\beta)\cdot h(u'-u)$$

1 - normalization step

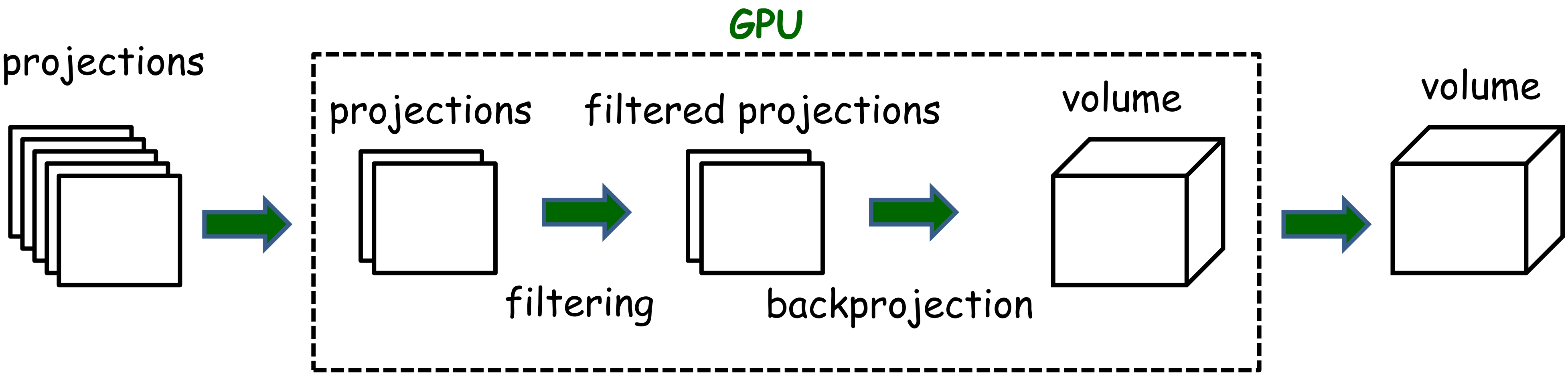
2 - convolution step with ramp filter h(u)

3 - backprojection step

The reconstructed volume size is tipically 128^3, 256^3, 512^3, 1024^3.

Data distribution and parallelization -1

- A 512^3 voxel volume requires at least 512 MB of memory space, it is no easy to store the entire volume and the projections on GPU device memory.
- We have decided to store the entire volume (or a 512^3 portion if the size > 512^3) in device memory and to load the projections into device memory when needed and remove it after its backprojection.



Data distribution and parallelization -2

In the first implementation (naive code) we transfer the first projection P_1 to the device global memory and perform:

1. weighting step by using 1 Cuda thread per pixel,
2. 1-D filtering step by using CUDA - FFT library applied to each projection with stride Nv,
3. backprojection step by using 1 Cuda thread for K voxels along z-direction.

The 3 steps are repeated for the remaining projections.

Setup for tests

Two datasets are used for testing the naive inplementation in CUDA:

- **Dataset 1** consists of 200 projections acquired on 360° circular scan trajectory. The size of each projection is 1024x512.
- **Dataset 2** consists of 642 projections acquired on 360° circular scan trajectory. The size of each projection is 256x192.

The GPU devices used are a GTX-680 and a GTX-Titan .

GPU	GTX-680	GTX-Titan
Architecture	Kepler GK104	Kepler GK110
Performance [GFlops]	3090.4	4500
Texture fillrate [GT/s]	128.8	187.5
Bandwidth [GB/s]	192.2	288.4

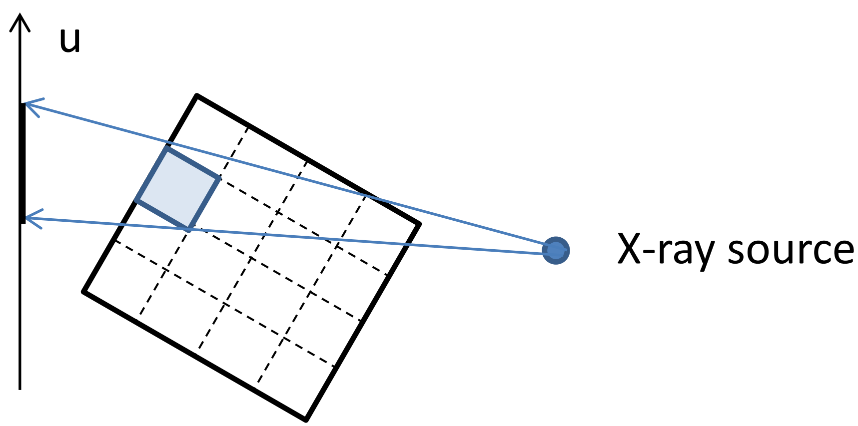
Results: naive code vs OMP code

GTX680	Step0 [s]	Step1_2 [s]	Step3 [s]	Total [s]
dataset1	0.4	6.99	12.41	19.8
dataset2	0.18	2.21	4.37	6.76
Titan	**Step0[s]**	**Step1_2 [s]**	**Step3 [s]**	**Total [s]**
dataset1	0.57	5.69	8.05	14.31
dataset2	0.45	1.81	3.1	5.36
OpenMP	**Step0 [s]**	**Step1_2 [s]**	**Step3 [s]**	**Total [s]**
dataset1	0.67	5.09	206.95	212.71
dataset2	0.21	1.26	80.96	82.43

- step 0: normalization
- step 1: cosine weighting
- step 2: filtering
- step 3: backprojection

Optimizing backprojection step

1. Memory coalescing
   - Organize the threads inside a block to access contiguous memory location: the volume is LxLxL is processed by a grid (L/Bx)x(L/By)x(L/K),
2. Global memory access reduction
   - Use a kind of memory that has a cache mechanism (texture and constant memory)
   - Increase the number of projections processed by a single kernel to reduce the number of access to volume global memory,



Cuda FDK results

Backproiection step comparison

GTX680	V0 [s]	V1 [s]	V2 [s]	OMP[s]
dataset1	12.41	3.75	2.27	296.6
Titan	**V0 [s]**	**V1 [s]**	**V2 [s]**	**OMP [s]**
dataset1	8.05	2.24	1.42	206.95

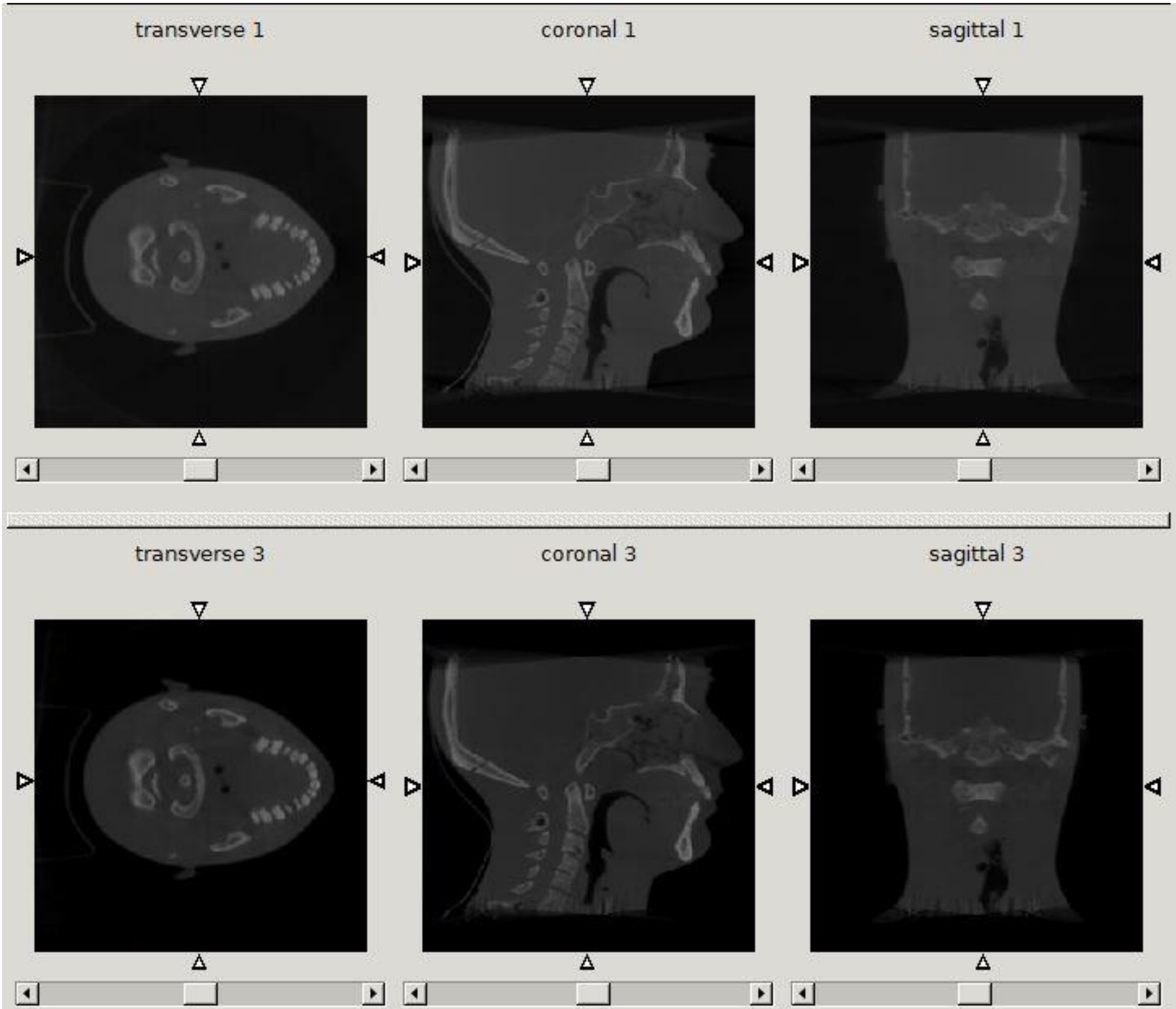
All steps comparison

GTX680	V0 [s]	V1 [s]	V2 [s]	OMP[s]
dataset1	19.8	11.15	9.66	302.3
Titan	**V0 [s]**	**V1 [s]**	**V2 [s]**	**OMP [s]**
dataset1	14.31	8.49	7.68	212.7

Dataset_1 reconstructed images

CUDA

OMP



$\langle \Delta I \rangle / \langle I \rangle = 4.0 \times 10^{-3}$

$\Delta I_{max} / Ipix = 4.0 \times 10^{-2}$

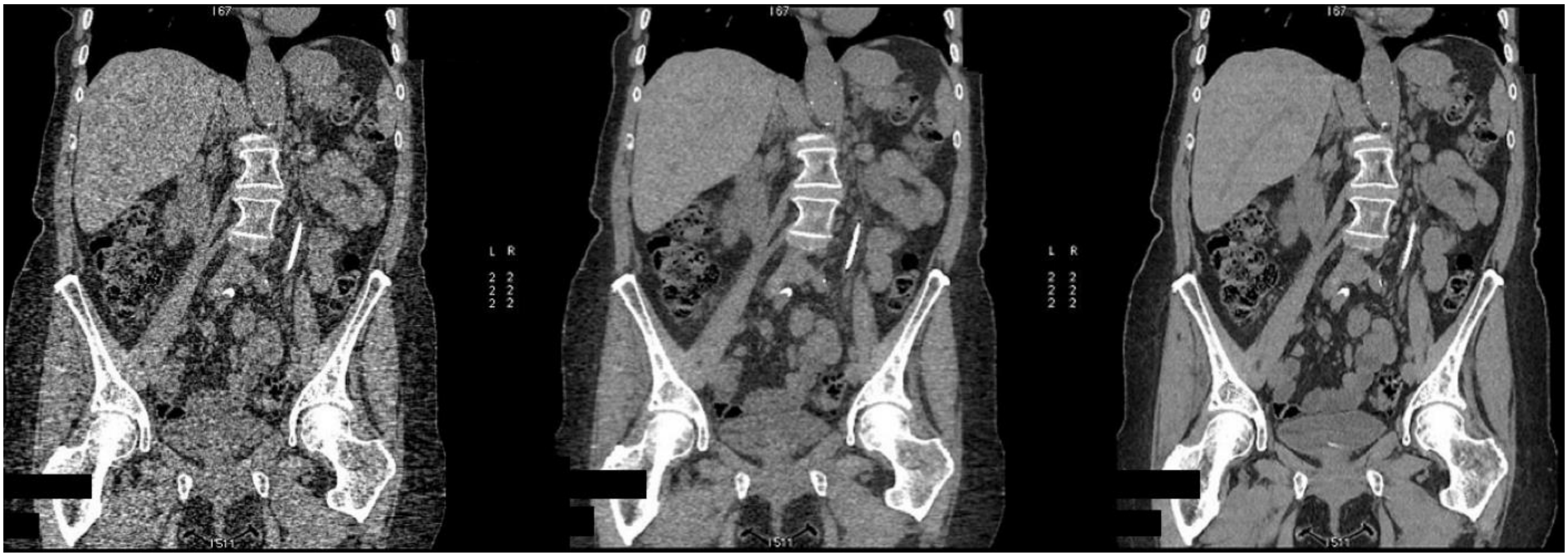
CT Iterative Reconstruction

- Accurate physics models:
  - We can use information like X-ray spectrum, beam-hardening, scatter,..
  - We can model the detector PSF, the focal spot size,...

- It is possible to reconstruct data from nonstandard geometries.

- We can use the right model of measurement statistics.

Why ?



FBP - fast　　　　ASIR – a bit longer　　　　Statistical - longer

Iterative methods

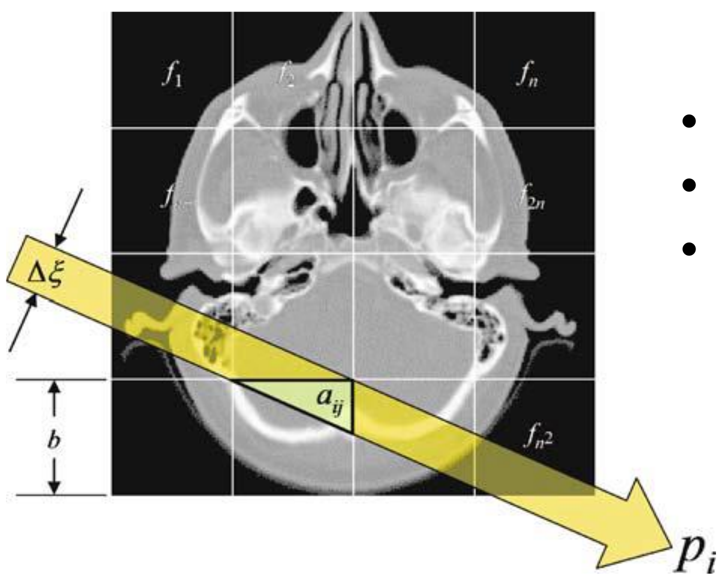
- Algebraic reconstruction methods:
  - Given a sinogram data y and a system model A, we want reconstruct the object x solving y = Ax,
  - ART, SIRT, SART,…
  - iterative FBP:

$$x^{(k+1)} = x^{(k)} + \beta \cdot FBP\left[y - Ax^{(k)}\right]$$

- Statistical reconstruction methods:
  - EM-ML, MAP, OSEM, CG,…
  - $\hat{x} = argmin\ \frac{1}{2}\|y - Ax\|_2^2 + \beta \cdot \|Cx\|_1$ (TV)
  - $\hat{x} = argmin\ \frac{1}{2}\|y - Ax\|_2^2 + \mu \cdot \|z - Cx\|_2^2$ (CG)

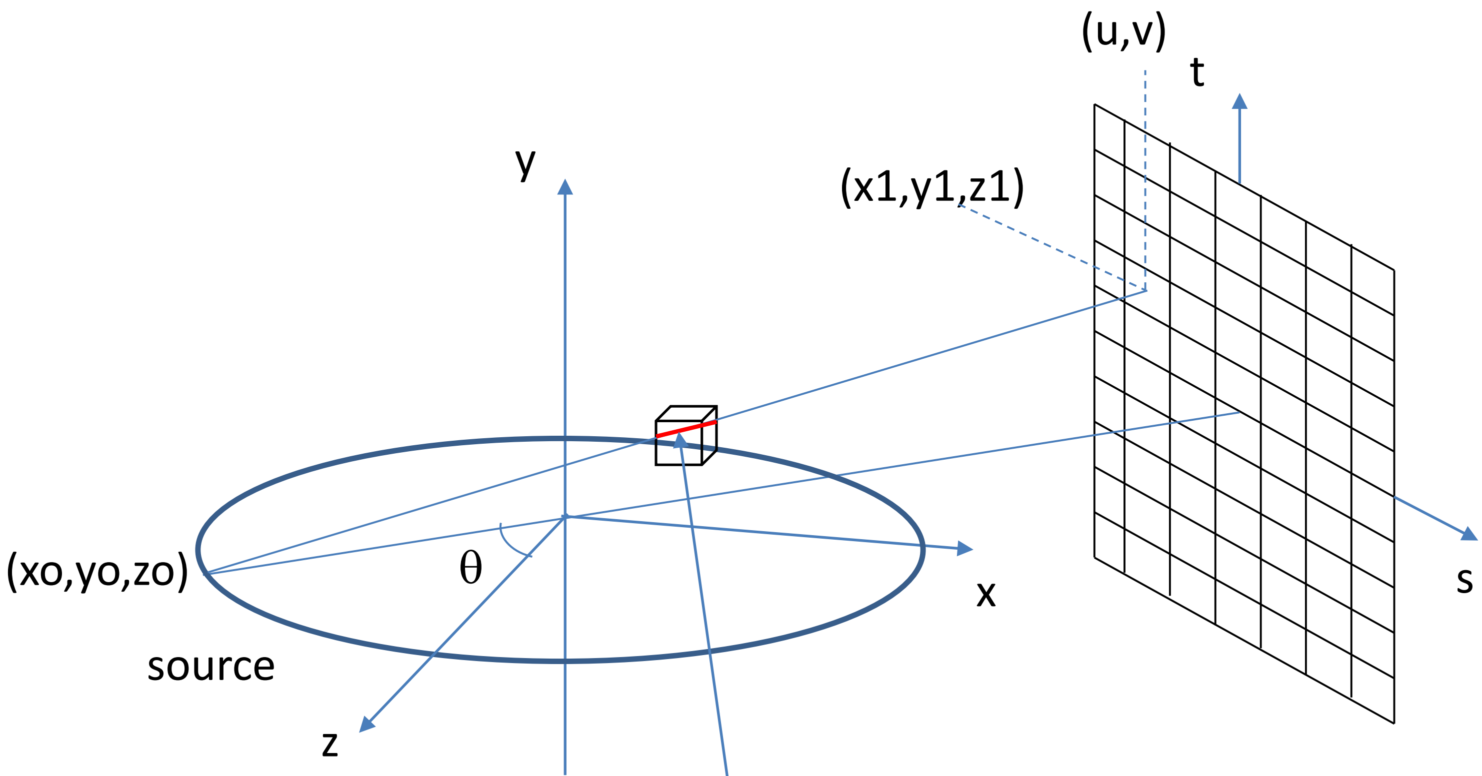
System Matrix Evaluation

- The calculation of *A* (system matrix) in CT is not possible:
  ex. Volume size 512^3, Sinogram Size 1024^2*512,
  rows~10^8, colums=5x10^8
- The calculation of *A* must be done «on the fly», we need fast projection (*A*) e fast backprojection (A^+) operators.



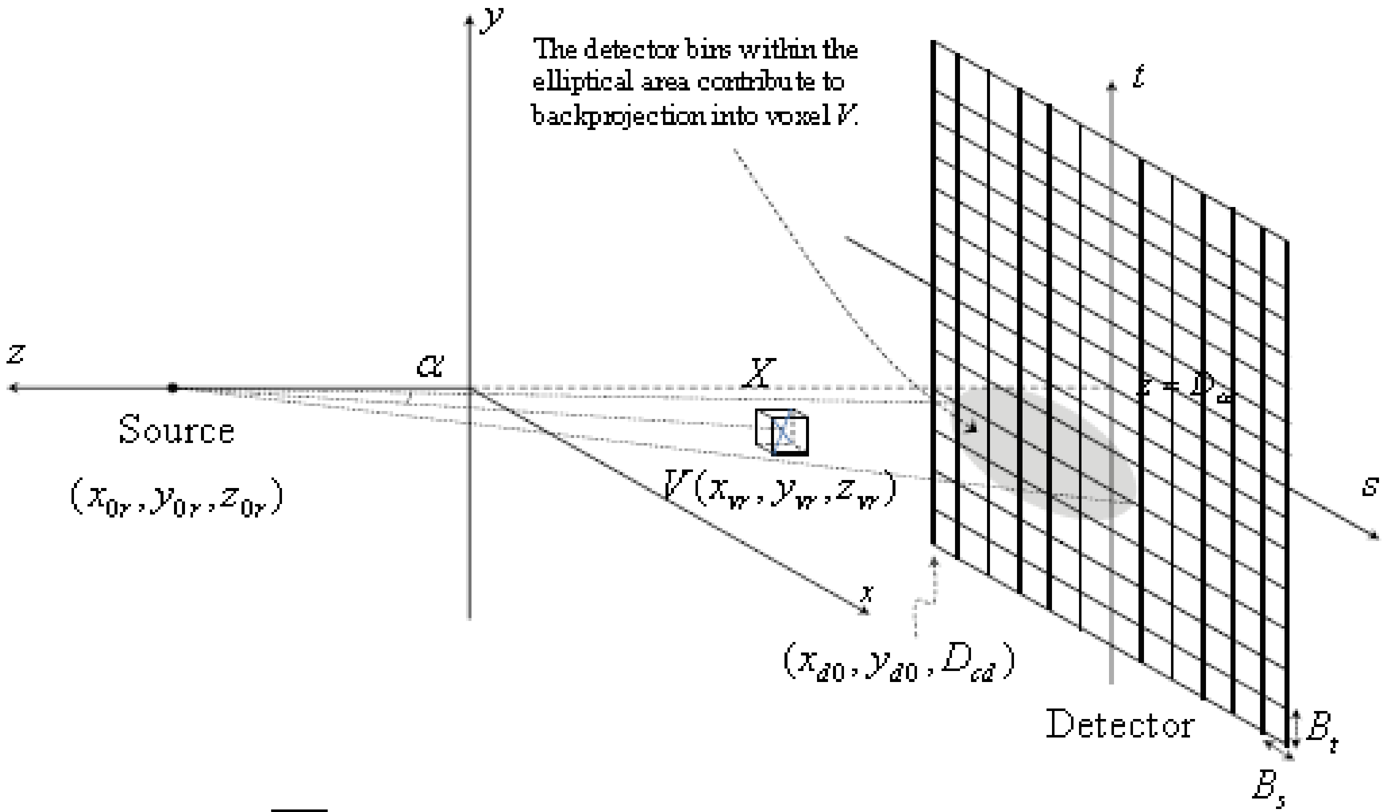
- Ray Tracing Method: Siddon (1985)
- Distance Driven Method: De Man (2004)
- Separable Footprint Method: Long (2010)

Ray Tracing Method-projection



$$g(u,v,\theta) = \sum_{i,j,k} a(u,v,\theta,i,j,k)\mu(i,j,k)$$

Ray Tracing Method-backprojection



$$v(i,j,k) = \sum_{u,v,\theta} \boxed{a(u,v,\theta,i,j,k)}\, g(u,v,\theta)$$

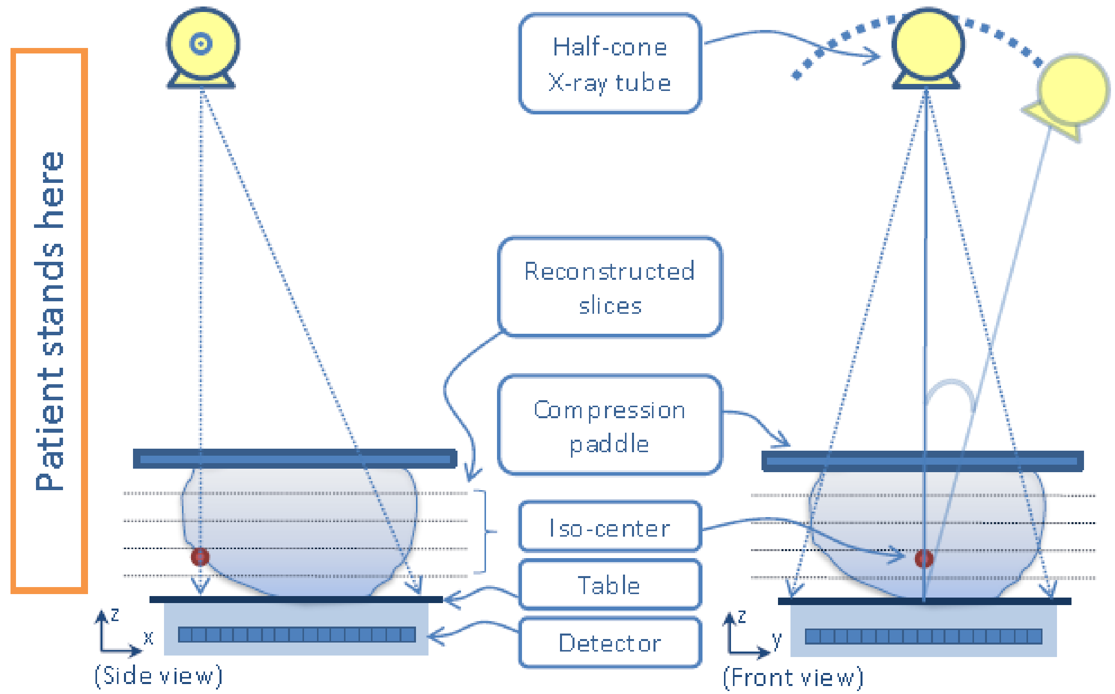
GPU Implementation

- We have implemented a version 0.1 of RT projector-backprojector on GPU (CUDA).
- The preliminary results shown a gain of factor 70x in execution time for GPU code respect to CPU.

GTX680	GPU Prj [s]	GPU Bck-prj[s]	CPU Prj [s]	CPU Bck-prj [s]
phantom	6.7	40.3	414.5	2850.6

The next step is to optimize the CUDA implementation for RT and to start the implementation of Distance Driven Method.

Digital Breast Tomosynthesis

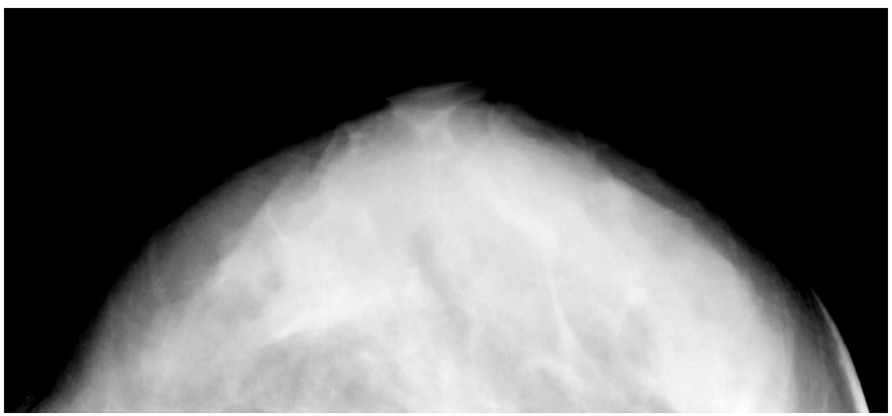


- Angular Range +- 25°
- Angular Step 2°
- Acquisition time 30 s
- Detector Size 3584 x 2816
- Detector pitch 0.085 mm
- Voxel size 0.085x0.085x1 mm

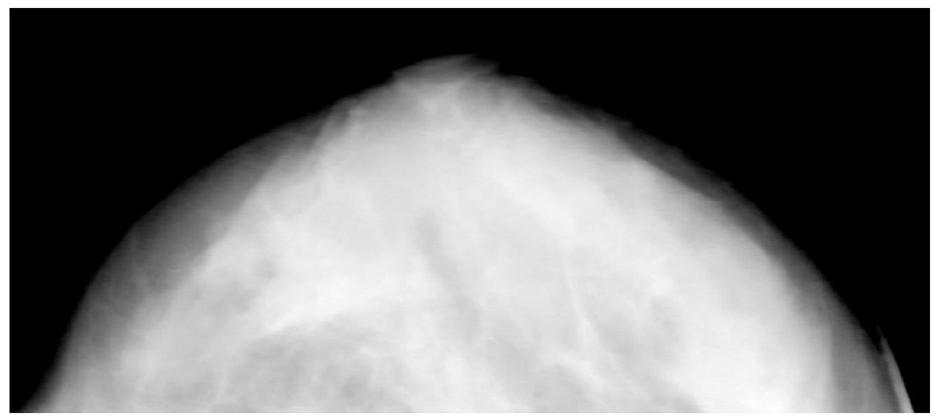
Synthetized 2D Image in DBT

- The aim is to obtain from the DBT reconstructed volume a synthetic 2D image comparable to the conventional 2D mammography.
- An optimized projector has been implemented both on GPU and on CPU.

CPU



GPU



OpenRTK

- The reconstruction toolkit (RTK) is an open-source cross-platform software for fast circular cone-beam CT reconstruction based on the Insight Toolkit (ITK).
- RTK provides o will provide:
  - Basic operators for reconstruction, e.g., filtering, forward, projection and backprojection
  - Multithreaded CPU and GPU versions
  - Tools for respiratory motion correction
  - I/O for several scanners
  - Preprocessing of raw data for scatter correction
- We have started a project to implement our reconstruction algorithm in this platform.