Status report on PET IMAGE RECONSTRUCTION

Carmela Luongo
carmela.luongo@pi.infn.it
Niccolò Camarlinghi
GAP Collaboration meeting
Ferrara, 23 march 2016

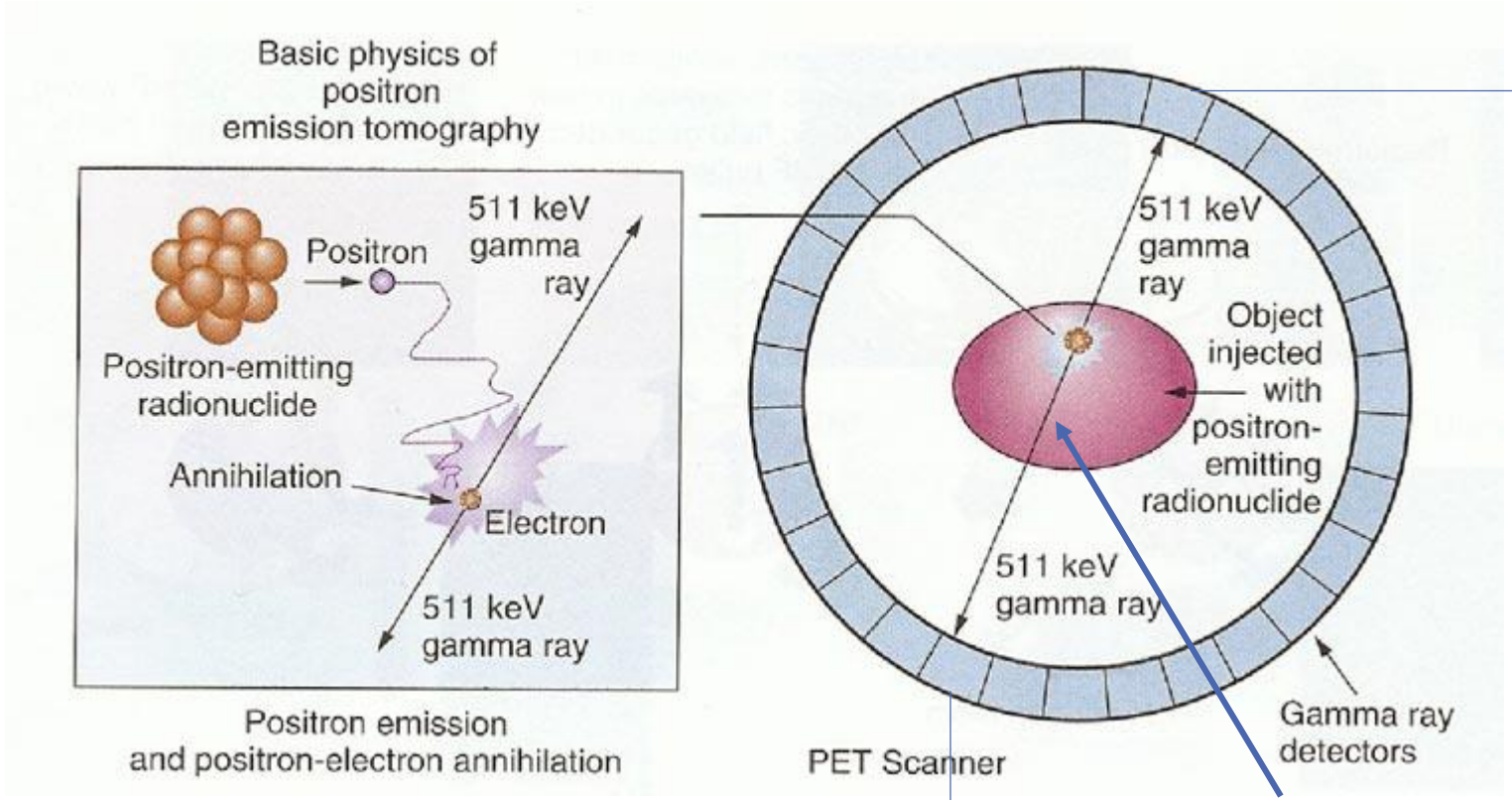
Outline

- PET system
- Image Reconstruction
- Implementation
- CPU vs GPU approach
- Conclusions

Positron Emission Tomography

- PET is a molecular imaging technique that uses radiolabeled molecules to image molecular interactions of biological processes in vivo
- PET imaging can measure the spatial distribution of active functional processes, such as glucose metabolism, in living tissue
- Emission imaging
- Functional imaging

Physics in PET

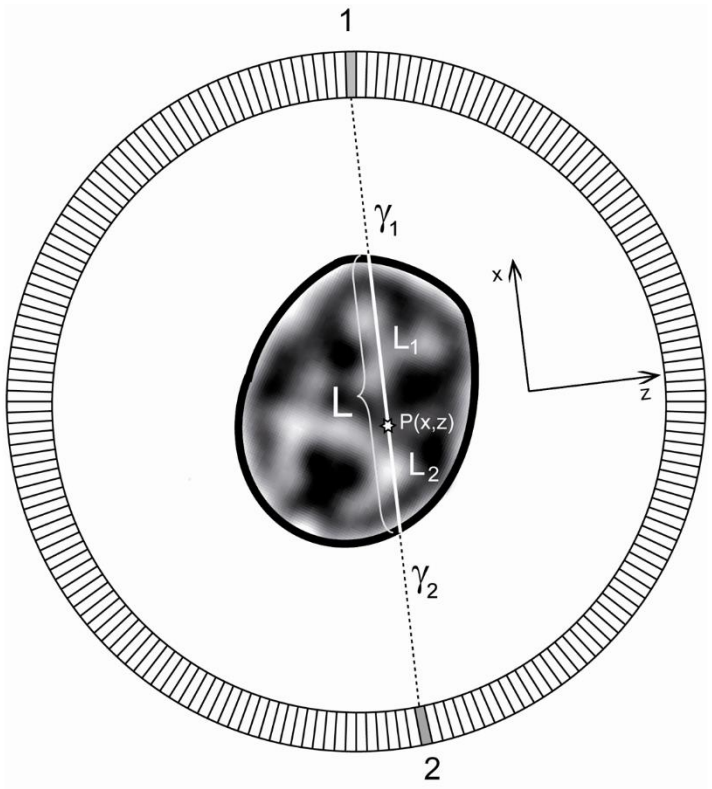


Coincident events

Line of Response (LOR)

IMAGE RECONSTRUCTION

The line integral model



$$P_{\gamma-\gamma} \propto \int_L \rho(x,y,z)dl$$

- The activity distribution $\rho(x,y,z)$ is measured in terms of projections along lines L (LOR)
- Each projection is obtained from the activity distribution with the line integral operator
- This is an ideal model

Iterative Reconstruction

- The image is discretized in pixels or voxels
- Image reconstruction can be obtained by solving a system of linear equations

$$\boldsymbol{P = AX} \Rightarrow \boldsymbol{X = A^{-1}P}$$

If the inverse matrix of A exists

P: projections
X: reconstructed image
A: coefficient matrix of the system

System Response Matrix

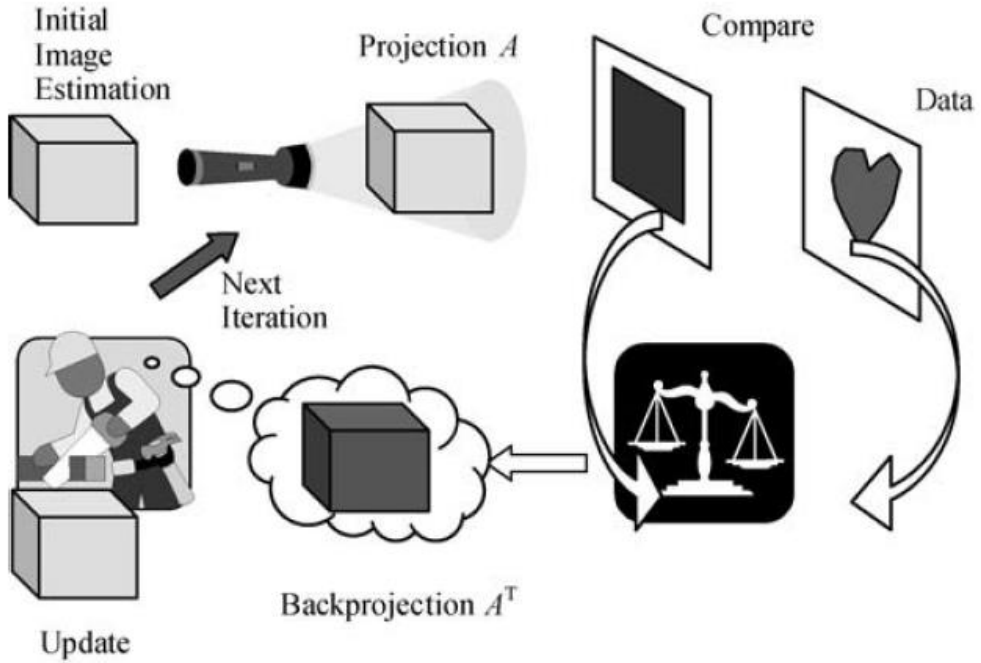
- A: $N_{LOR} \times N_{Image\ pixels}$
- E.g. 10^7 LORs x 10^6 Pixels
- Storing all the coefficients of A as 4 bytes elements would require 4×10^{13} bytes ≈ 40000 GB
- A is huge and cannot be inverted!

→ Solve inverse problem iteratively

Iterative Algorithms

- The reconstruction problem is solved iteratively

- Objective function
- Optimization algorithm
- System Response Matrix (SRM)



Maximum Likelihood Expectation Maximization (ML-EM)

- Algorithm hypothesis: detected photon pairs are Poisson distributed
- Maximum likelihood: maximizes the likelihood function
  - Maximum = the image generating the measured data (LOR)
- Expectaction Maximization : iterative algorithm to find the ML estimate
- ML-EM relies on a discrete representation of both the data and the reconstructed image
- The field of view (FOV) is represented as digital 3D image
- Needs a lot of iterations in order to obtain good images (30-100)

ML-EM

$$\boldsymbol{X^{k+1} = X^{k} \cdot \frac{1}{A^{T}1} \cdot \left(A^{T} \cdot \frac{p}{AX^{k}} \right)}$$

- N : field of view dimension (in pixels)
- M: number of LOR
- $X^k \in \mathcal{R}^N$: image estimate at iteration k
- $p \in \mathcal{R}^M$: LOR data
- $A \in \mathcal{R}^N \times \mathcal{R}^M$: System Response Matrix (SRM)
  - A_{ij}: probability that a photon pair emitted in the voxel j is detected in the LOR i

ML-EM steps

$$X^{k+1} = X^k \cdot \frac{1}{A^T 1} \cdot \left(A^T \cdot \frac{p}{AX^k} \right)$$

Backprojection

Sensitivity

Forward projection

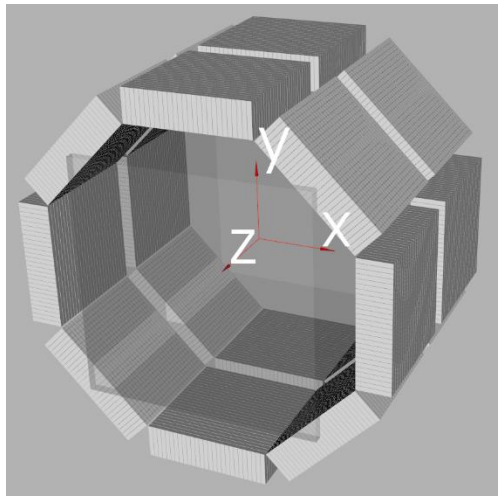
1. Initial guess for the image (uniform)
2. Forward projection: For each LOR gets the number of events expected if X was the true image
3. Backprojection: Comparison between the measured and estimated projections is backprojected to the image
4. Divide by a sensitivity image
5. Update the image

System Response Matrix evaluation

- Experimentally obtained model → very accurate results but very difficult in practice
- Monte-Carlo simulation obtained model → very accurate but still difficult to be evaluated
- Semi-analytical models → quality of the images is similar to the other methods but it is easier to handle

IMPLEMENTATION

SCANNER IRIS PET



- PET-CT tomography
- 16 detector arranged on 2 octagonal rings
  - PMT optically coupled with a segmented LYSO of 27x26 (702) crystals
- Coincidence scheme 1 vs 6 detectors → 48 detector pairs
- Number of LORs = $702^2 \cdot 48 \sim 24\ million$
- Number of FOV pixels = $101 \cdot 101 \cdot 120 \sim 10^6$
- Crystal size = 1.6x1.6x12 mm
- Crystal pitch = 1.7 mm
- Axial FOV = 95 mm
- Ring diameter = 110 mm

SRM implementation

- Ray-tracing → Siddon algorithm

Approaches:

- **CPU**: SRM pre-calculated
  - Limit in RAM
  - Symmetries
  - Slow reconstruction

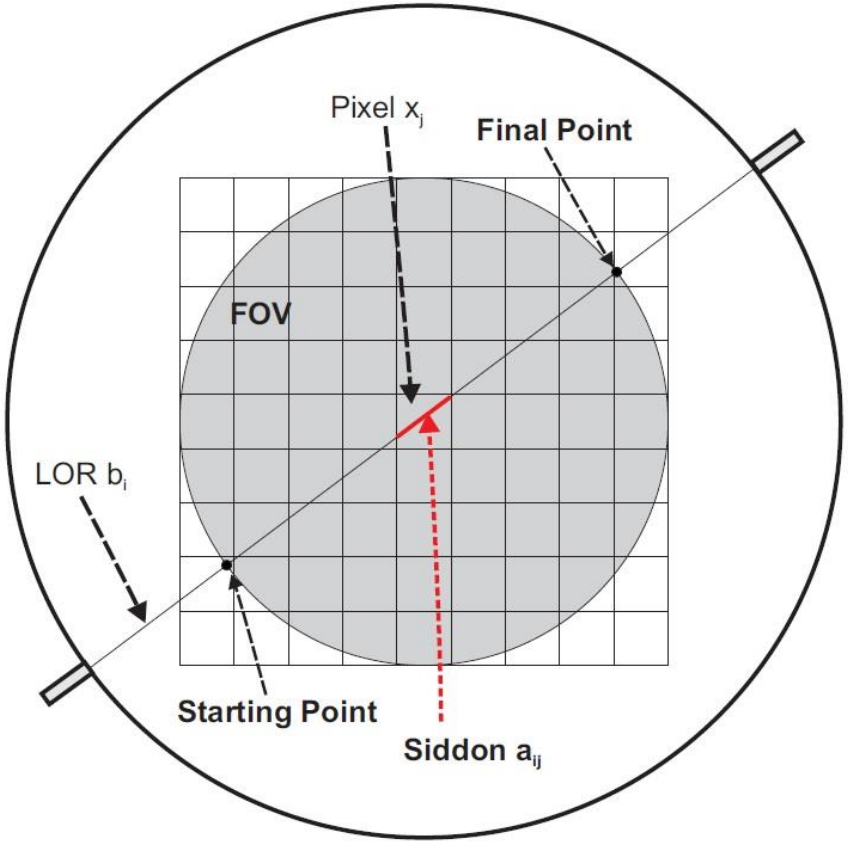
Number of LORs · Number of FOV pixels
$\sim 24 \cdot 10^6 million$

- **Graphics Processing Units (GPU)**: SRM calculated on-the-fly
  - The reconstruction problem is suitable for massively parallel architectures
  - NVIDIA CUDA architecture
  - CUDA-C

Siddon algorithm

- Ray tracing of a LOR
  - s_{ij} : **Siddon coefficient**

    → the length of the segment generated by the intersection between the LOR and the pixels of the image



Hardware

- **CPU Intel i7 3770 - 3.4 GHz – 8 cores**

- **GeForce GTX TITAN (Kepler)**
  - **Compute Capability 3.5**
  - 14 Multiprocessors
- **GeForce GTX980 Ti (Maxwell)**
  - **Compute Capability 5.2**
  - 22 Multiprocessors

- Max dimension size of a thread block (x, y, z): (1024, 1024, 64)
- Max number of threads per block: 1024
- Total amount of shared memory per block: 49152 bytes
- Total number of registers per block: 65536

GPU Implementation (1)

- Parallelization of each stage of the EM iteration
  - **Forward projection**: each thread computes the sum of all activity along one projection path
  - **Backprojection**: each thread re-distributes the activity back to its original path
- Different pattern of access to memory
  - **Forward projection** only reads texture memory
  - **Backprojection** reads and writes global memory

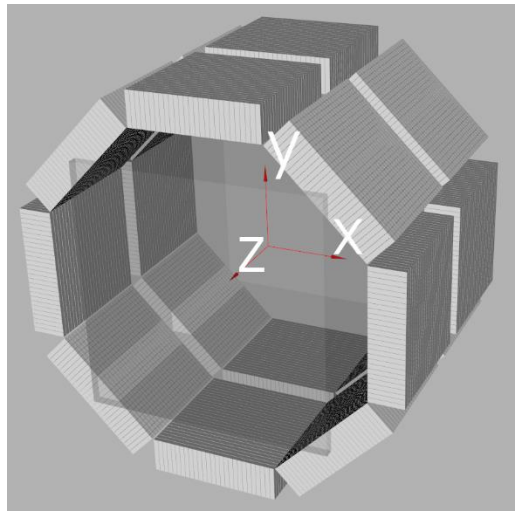
GPU Implementation (1)

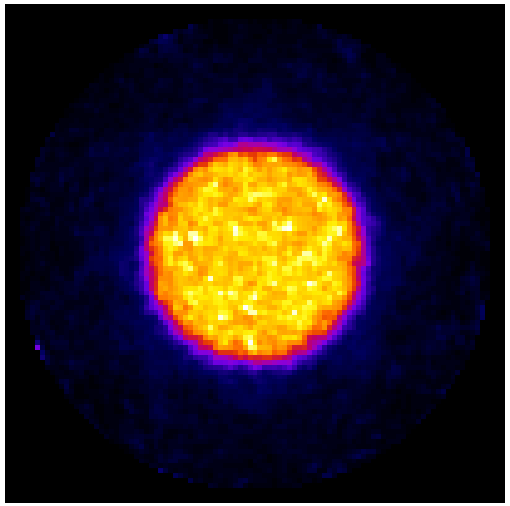
- Profiling → memory access optimization
  - Constant Memory ← kernel parameters
  - Texture Memory ← image at previous iteration

- Problem: race conditions in backprojection
- Solution: atomic operations

Results

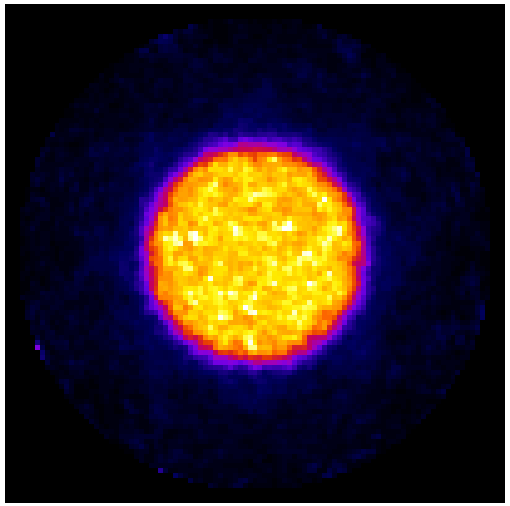
- Monte Carlo Gate ($\sim 10^6 coincidences \; - \; \sim 10^6 LOR$)
- Cylinder filled with uniform FDG solution
- Transaxial view





CPU reconstruction
Intel i7 3770 - 3.4 GHz

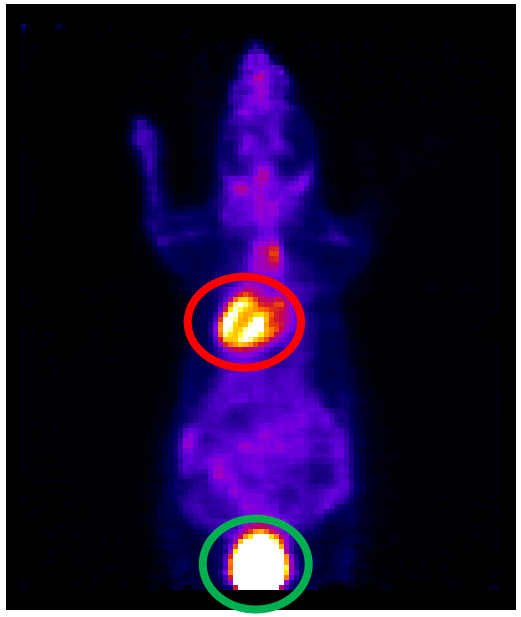
% difference between the two images is 10^{-4}



GPU reconstruction
GeForce GTX TITAN

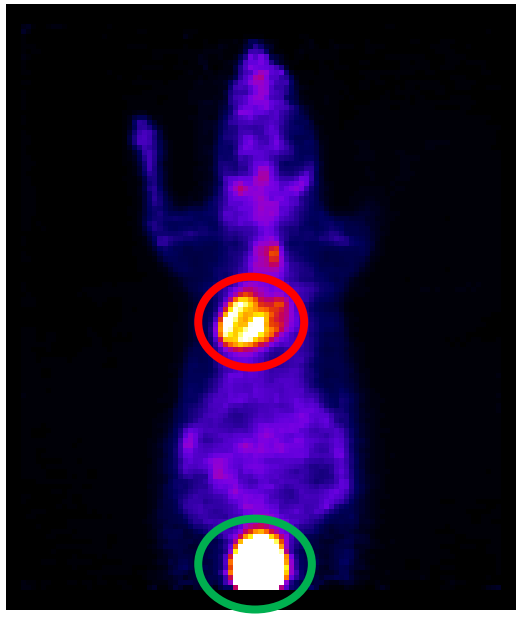
Small animal imaging

- PET imaged mouse 60-70 min after FDG injection



Heart

Bladder



MLEM 30 iterations

CPU reconstruction

GPU reconstruction

Courtesy of Dr. Piero Salvadori and Daniele Panetta, Istituto di Fisiologia Clinica (IFC) Pisa

Benchmark cylinder

- Time per iteration (Forward Projection and BackProjection)
- 2 integration points – 4 depth integration points
- For each iteration number of rays $\propto 5 * 10^6 * 2 * (2 * 2 * 4)^2 \sim 3 * 10^9$

TIME (sec)	CPU	GPU Kepler	GPU Maxwell
Sensitivity	235	94	49
ForwardProjection	67	9	9
BackProjection		29	13

GPU achieves speed up factor of **5** with respect to CPU

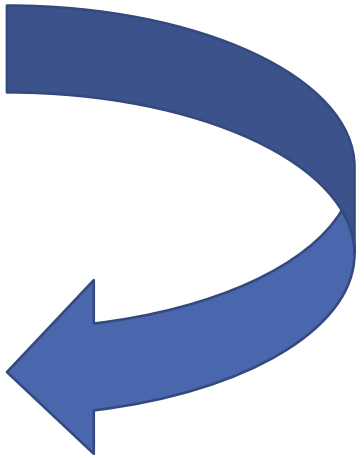
Benchmark mouse

- Time per iteration (Forward Projection and BackProjection)
- 2 integration points – 4 depth integration points
- For each iteration number of rays $\propto 13 * 10^6 * 2 * (2 * 2 * 4)^2 \sim 7 * 10^9$

TIME (sec)	CPU	GPU Kepler	GPU Maxwell
Sensitivity	235	94	49
ForwardProjection	133	30	23
BackProjection		85	28

GPU Implementation (2)

- Backprojection bottleneck of the algorithm
  - Random accesses (reading and writing) to global memory



- **<u>Tiling strategy</u>**
  - The field of view is divided in tiles
  - Backprojection is executed on each tile
  - The tiles are stored in the Shared Memory
- Total sensitivity evaluation time (2 integration pts) ~ 23 seconds

→ **TOO SLOW!**

GPU Implementation (3)

- **Tiling strategy with line partitioning**
  - Categorize the LORs into 3 classes according to their predominant direction
  - For each class, the tiles are selected to be orthogonal to the predominant LOR direction
  - Each tile is stored in Shared Memory

- Total sensitivity evaluation time (2 integration pts) ~ 18 seconds

→ **STILL SLOW!**

- Profiling

Conclusions

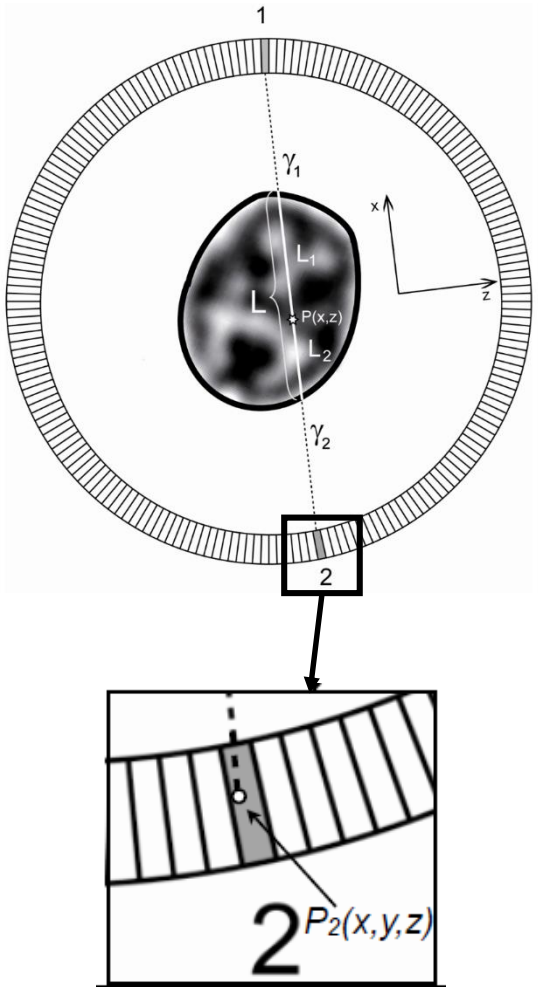
- We implemented an iterative algorithm for PET image reconstruction on GPU
- This computing application fits the capabilities of massively parallel architectures like GPUs
- We made a comparison with an existing CPU implementation with respect to image quality and processing time
  - The reconstructed images are "identical"
  - GPU implementation shows in the sensitivity evaluation speed up factor of 5 with respect to CPU implementation
- We found the bottleneck of the algorithm and implemented new versions
  - The result is slower than the standard version but it seems to be promising

Next steps

- Profiling of the new implementation
  - Inefficient alignment in Global Memory
    - **Structure of arrays** instead of Array of structures
  - Kernel performance is limited by computation
    - What can we pre-calculate?
  - GPU utilization is limited by register usage
    - Launch_bounds or code refactoring?
- What is the real contribution of the architecture?
- Can we achieve speed up factor of 2 with respect to new implementation?
- MultiGPU

BACKUP

PET: image reconstruction

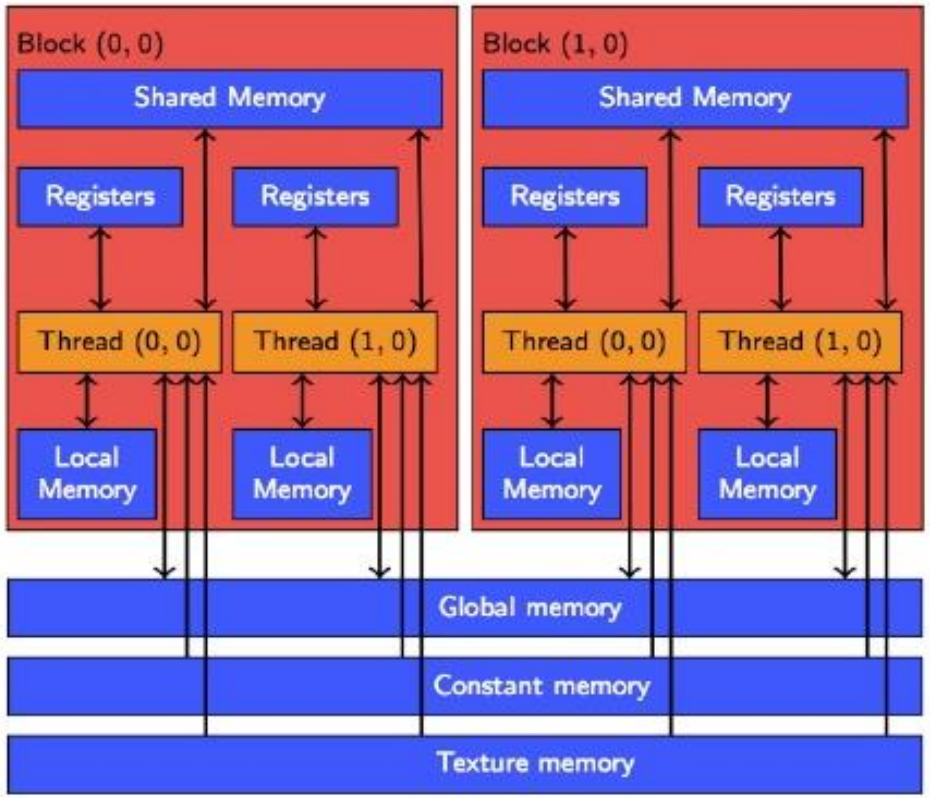


- Providing cross-sectional images of the radiotracer distribution in an object, using the coincidence events detected by a scanner
  - Obtaining the spatial coordinates of the line L (LOR) where the count is detected
  - Indirect measure of the original activity

CUDA

- NVIDIA's parallel computing architecture and programming model
- Software environment that allows developers to use C as a high-level programming language
- **CUDA-C → kernels → threads**
- **Host**: CPU and system's memory
- **Device**: GPU and its memory

Device Memory Hierarchy



- Registers on chip
- Shared Memory on chip
- Constant Memory resides in device memory and is cached on chip
- Texture Memory resides in device memory and is cached on chip