CT Iterative Reconstruction within the CASToR Platform using GPU Architecture

D. Benoit[1], T. Merlin[1], J. Bert[1], T. Carlier[2], F. Lamare[3] and D. Visvikis[1]

[1] *Laboratoire de Traitement de l'Information Médicale (LaTIM), INSERM U1101, Brest, FRANCE*
[2] *Centre de Recherche en Cancérologie et Immunologie Nantes – Angers (CRCINA), INSERM 892 – CNRS 6299, Nantes, FRANCE*
[3] *Institut de Neurosciences Cognitives et Intégratives d'Aquitaine (INCIA), CNRS 5287, Bordeaux, FRANCE*

Laboratoire de Traitement de l'Information Médicale

I. INTRODUCTION

CASToR [1] is a generic open-source platform dedicated to tomographic reconstruction already implementing emission tomography (PET and SPECT). In this study we present:

- **Extension** of **CASToR** to transmission (**CT**) iterative reconstruction. Additions include:
  - → New optimizer: Maximum-Likehood for Transmission (MLTR) [2]
  - → New classes for CT geometry description and input datafiles
  - → "**Distance Driven**" [6] projector adapted to CT reconstruction

- First attempt to adapt the CASToR iterative loop for **GPU** using **OpenCL** [3]

II. SCANNER, DATA FILE and EVENT for CT

In CASToR, the **scanner configuration** file and the information about the acquisition (**data file**) are included in 2 different ascii files, summarized in the following table:

CT SCANNER FILE (*.geom)	CT DATA FILE (*.Cdh)
- Scanner name - Modality - Transaxial/Axial number of pixels - Transaxial/Axial pixel size - Detector depth - Detector radius - Source radius - Transaxial/Axial spot size	- Data filename (*.Cdf) - Number of events - Data mode - Data type - Scanner name - Number of projections - Blank correction - Scatter correction - Projection angles

The geometry file allows to describe any plane detector. Curved detectors are described through a look-up-table providing the positions and orientations of the detection pixels.

In CASToR, input data (*.Cdf) has to be organized as a list of **events**, even for histogrammed data. As for PET and SPECT systems, a new event format for CT has been created. Each CT event includes the following parameters:
- Time
- Number of events in bin
- Projection ID
- Pixel ID (in detector)
- Blank value (acquisition without volume to reconstruct)

III. GPU STRATEGY using OpenCL

The CASToR software is already a multithreaded code using both OpenMP (for multi-core usage) and OpenMPI (for multi-computer usage). For the **GPU** implementation, the **OpenCL** library

A currently implemented, the iterative reconstruction algorithm in CASToR is not designed for GPU computation. Here, a first and simple approach is proposed: the graphic card is only used to compute the system matrix (SM) elements using the incremental Siddon projector [4]. Differences between the standard code and the GPU code are presented below:

Standard iterative core algorithm	GPU iterative core algorithm
→ Loop over events inside a subset (openMP) → Get the event → Apply deformations → Compute the SM elements from a projector → Update correction image using the SM	→ Loop over batch containing events inside a subset → Loop over events inside a batch → Get the event and load it into a buffer → **Load the event buffer on GPU** → Compute the SM elements on GPU → **Load the computed lines on CPU** → Loop over events inside a batch (openMP) → Get the event → Apply deformation → Update correction image using the SM

IV. ACQUISITION & IMAGE RECONSTRUCTION

→ Acquisition:
- → Data from the Varian TrueBeam CBCT
- → 211 projections over 360°
- → 1024x768 pixels a projection
- → (0.388x0.388) mm² pixel size
- → Source – Detector distance : 1500 mm
- → Source – Isocenter distance : 1000 mm

$$\hat{\mu}_{j+1}^{k} = \hat{\mu}_j^k + \frac{\alpha_j \sum_i l_{ij}(\bar{y}_i - y_i)}{\sum_i l_{ij} \left[\sum_h l_{ih}\alpha_h\right] \bar{y}_i}$$

$$\bar{y}_i = b_i \times exp\left[-\sum_h l_{ih}\hat{\mu}_h\right]$$

→ Image reconstruction:
- → MLTR algorithm
- → 20 iterations 28 subsets
- → 512 x 512 x 93 image size
- → (0.51 x 0.51 x 1.99) mm³ voxel size
- → Used projectors: incremental Siddon [4], Joseph [5] and Distance Driven [6]

→ Analysis:

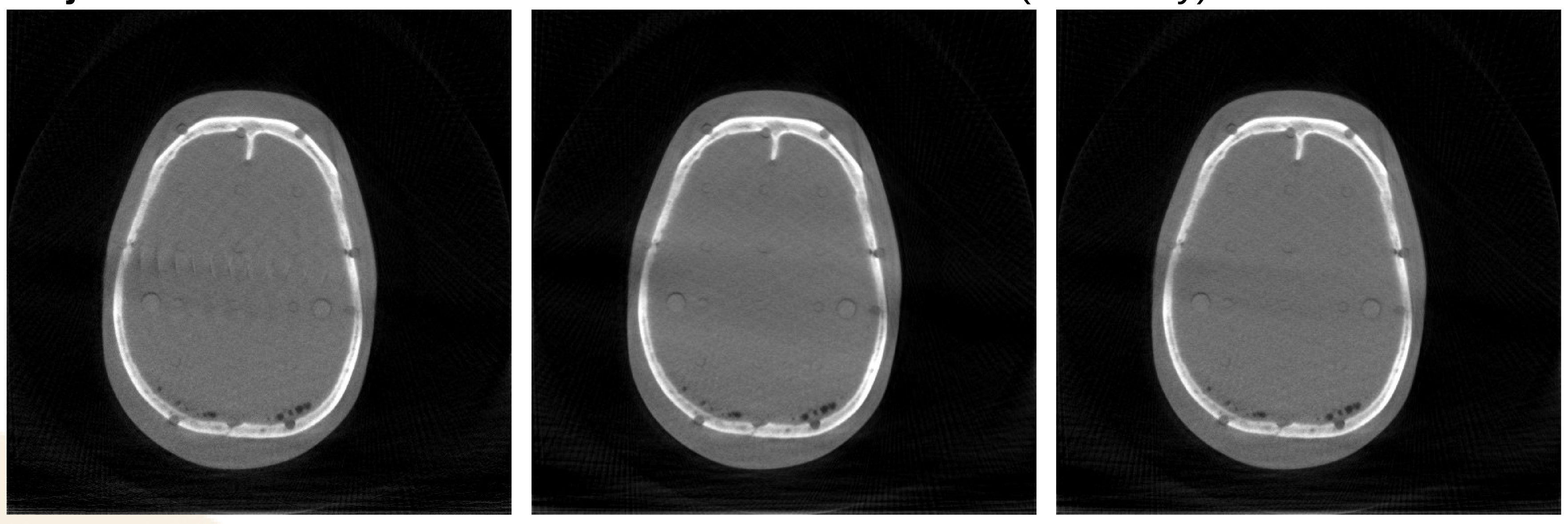
→ Distance Driven, as more adapted for use with CT, is compared to incremental Siddon and Joseph previously introduced for PET and SPECT.

→ Current OpenCL implementation performance is compared to CASToR using only CPU and to an optimised, for GPU using OpenCL, CT reconstruction implementation

V. RESULTS

Hardware Configuration:
→ CPU: Intel Core i7-4790K @ 4.0 GHz, 4 cores 8 threads
→ GPU: NVIDIA Geforce GTX 1050 Ti

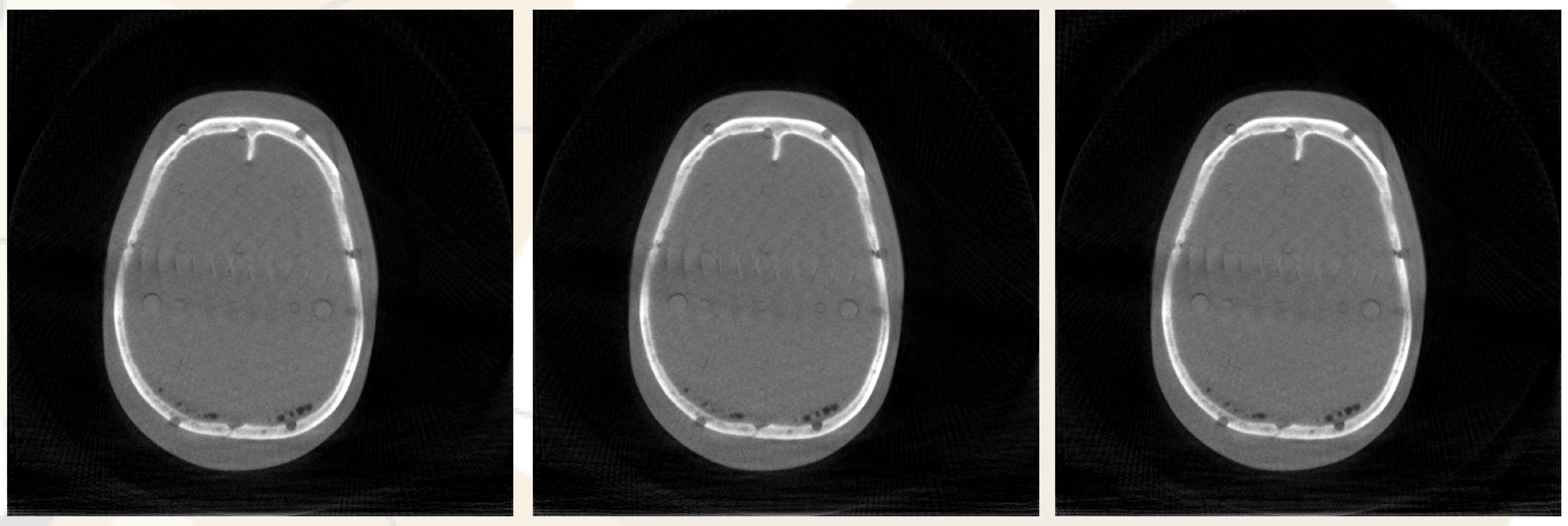
Projector Results: Distance Driven validation in CASToR (CPU only)



Incremental Siddon — Joseph — Distance Driven

	Incremental Siddon	Joseph	Distance Driven
Computation Time per iteration (s)	1173	2380	1428
Factor vs. Incr Siddon	1	2	1.2

OpenCL Results: Using incremental Siddon projector



CASToR CPU — CASToR OpenCL — Optimized CT GPU OpenCL

	CASToR CPU	CASToR OpenCL	Optimized CT GPU OpenCL
Computation Time per iteration (s)	1173	3800	148
Factor vs. Opt. CT GPU	8	26	1

VI. DISCUSSION & CONCLUSION

→ The CASToR platform is now capable to reconstruct data from CT scanners. In the current implementation:

(i) CT reconstruction in CASToR is generic. Plane detectors can be added using a simple ascii description while other types of scanners can be added providing a look-up-table of detection pixels positions and orientations.

(ii) The MLTR [7] algorithm for transmission data is implemented.

(iii) The Distance Driven projector is implemented.

→ The results show that the first attempt of using OpenCL within CASToR, while maintaining the whole genericity of the code, does not lead to any speed up. The amount of data to load from the GPU to the RAM (corresponding to the non-null SM elements returned by the projector) is too large so the transfer burden dominates.

→ Using OpenCL in an efficient way consists in implementing the data update step on GPU too, avoiding as much as possible any data transfert on the PCI express.

→ An alternative way to speed up the computation time would be to cleverly use SIMD (SSE/AVX) programming instructions which will be investigated in the future.

VII. REFERENCES

[1] http://www.castor-project.org
[2] Nuyts J. et al, PMB 43(4), 729
[3] https://www.khronos.org/opencl
[4] Jacobs F. et al, JCIT 6(1), 89
[5] Joseph PM, IEEE TMI 1(3), 192
[6] De Man B. et al, PMB 49(11), 2463
[7] Van Slambrouck K. et al, TNS 61(1), 173

VII. ACKNOWLEDGMENTS

This work was supported by grants from France Life Imaging (FLI) ANR-11-INBS-0006.
We thank ***Simon Stute*** for his helpful discussions and his advice for the implementation of the CT reconstruction in the CASToR software.

Contact: *didier.benoit@inserm.fr*