Application du GPGPU

avec CUDA en tomographie RX

Christophe Meessen

meessen@cppm.in2p3.fr

Webinaire CCRI 19 janvier 2010

Sommaire

Pixscan : Scanneur à Rayon X pour petits animaux

La reconstruction tomographique

Cartes graphiques et GPGPU

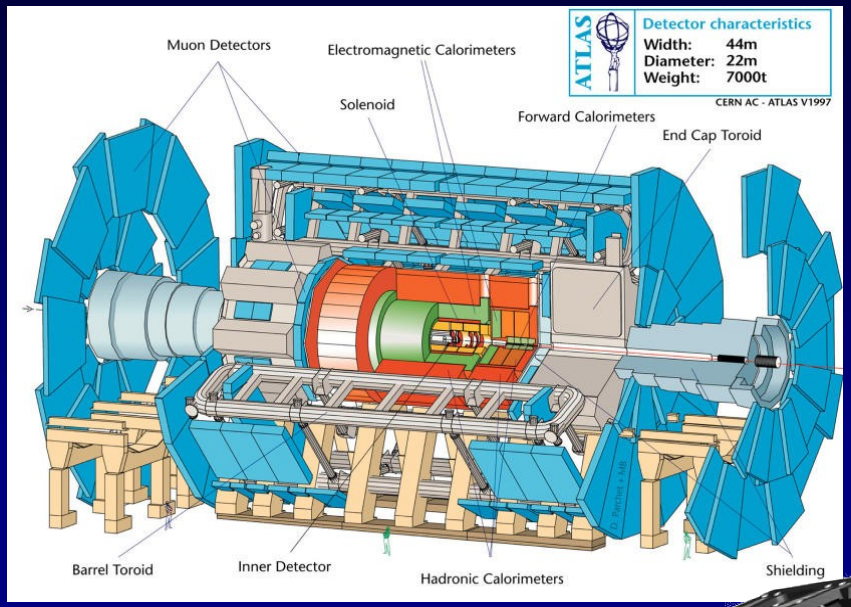
Exemple de programme simple en CUDA

Portage de l'algorithme FDK sur carte graphique et résultats

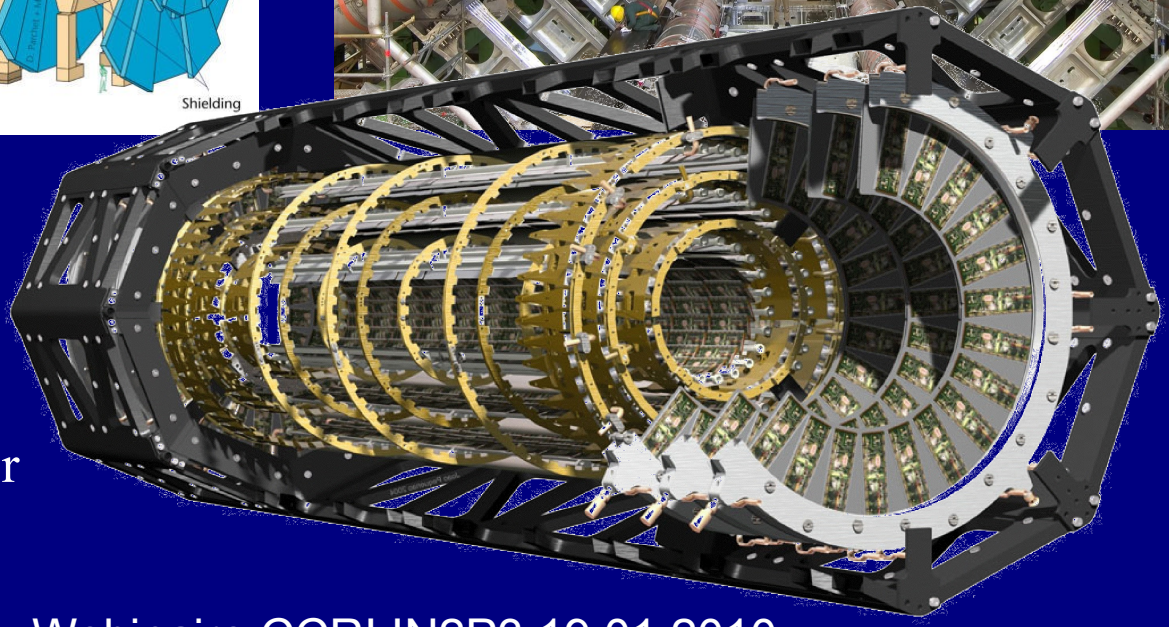
Conclusions

Valorisation des détecteurs hybrides (Pixel)

Détecteur de trajectoire de particule de l'expérience Atlas

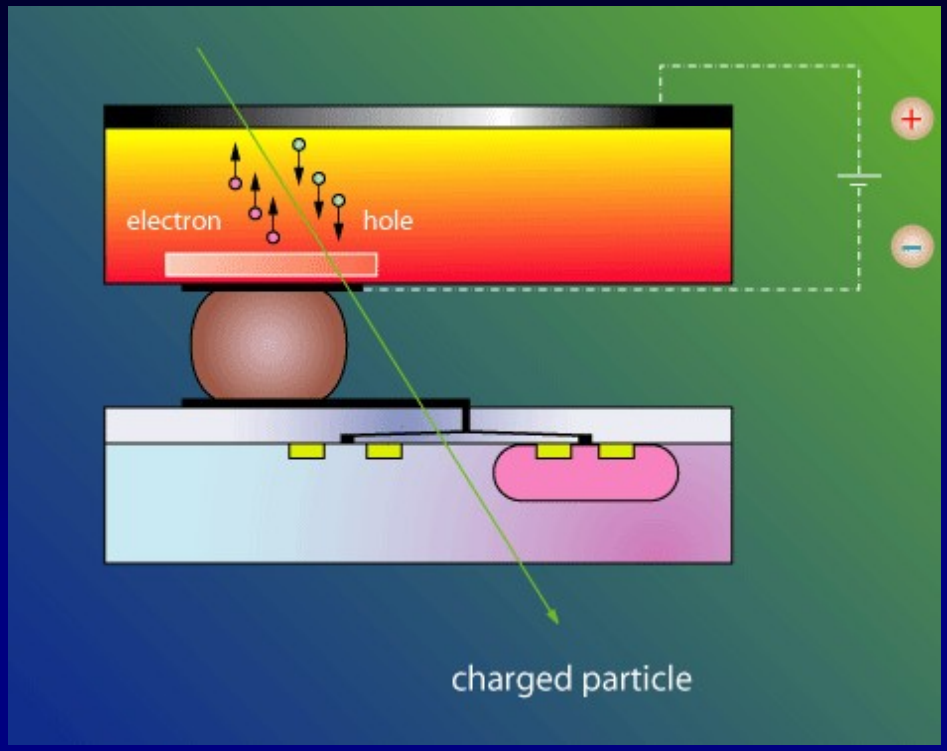


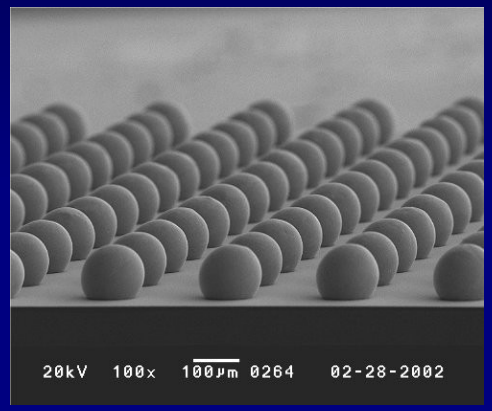
Atlas detector (LHC)

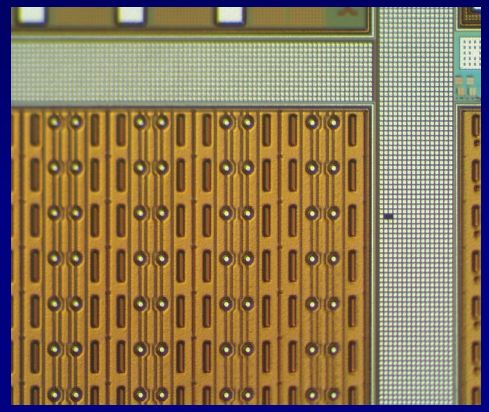


Pixel Inner detector

Architecture et principe de fonctionnement du détecteur hybride





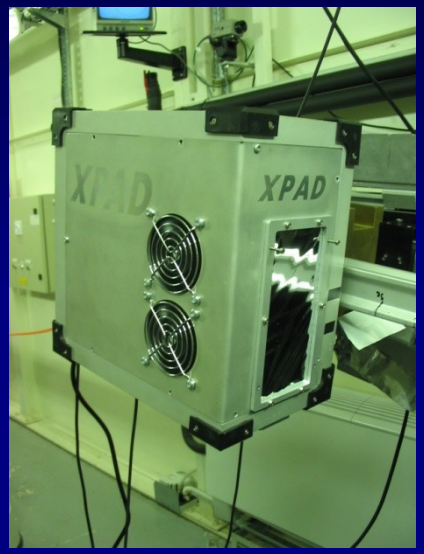


Détecteur Si ou CdTe fixé par *bump bounding* sur circuit électronique

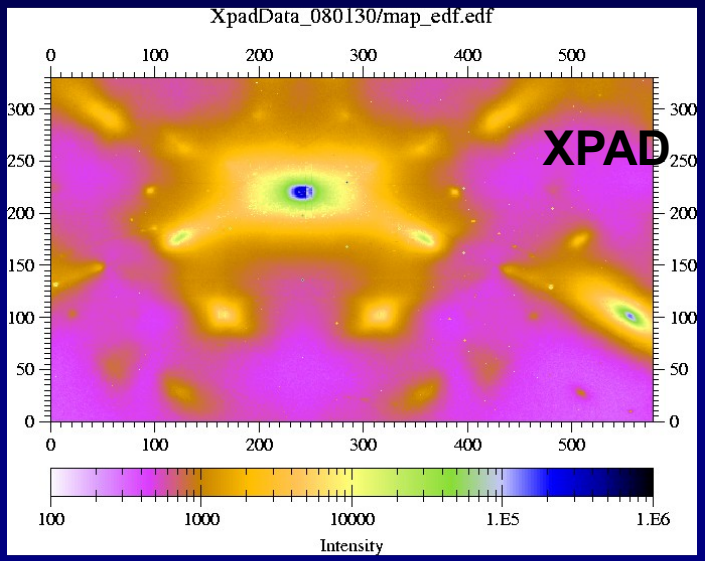
Circuit électronique mixte analogique et numérique (2 millions de transistors)

Application en cristallographie et en imagerie biomédicale

Startup imXPAD

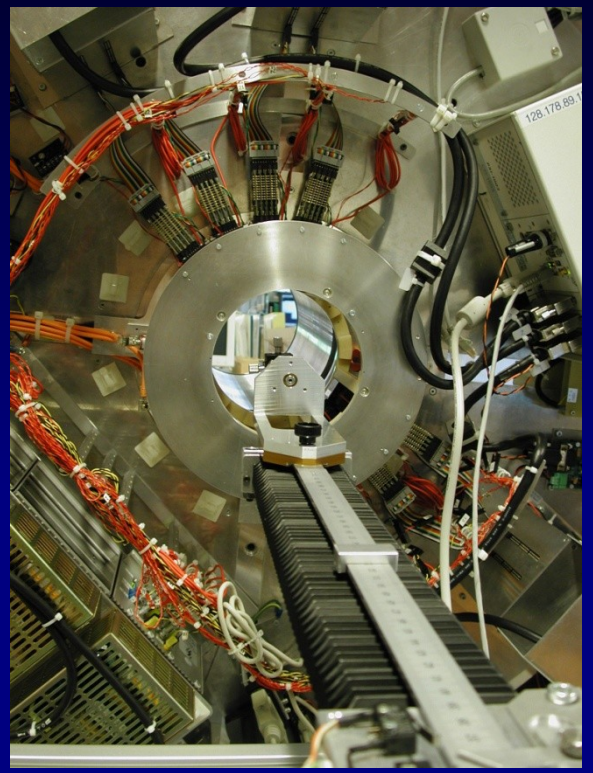


Détecteur XPAD3



Cristallographie

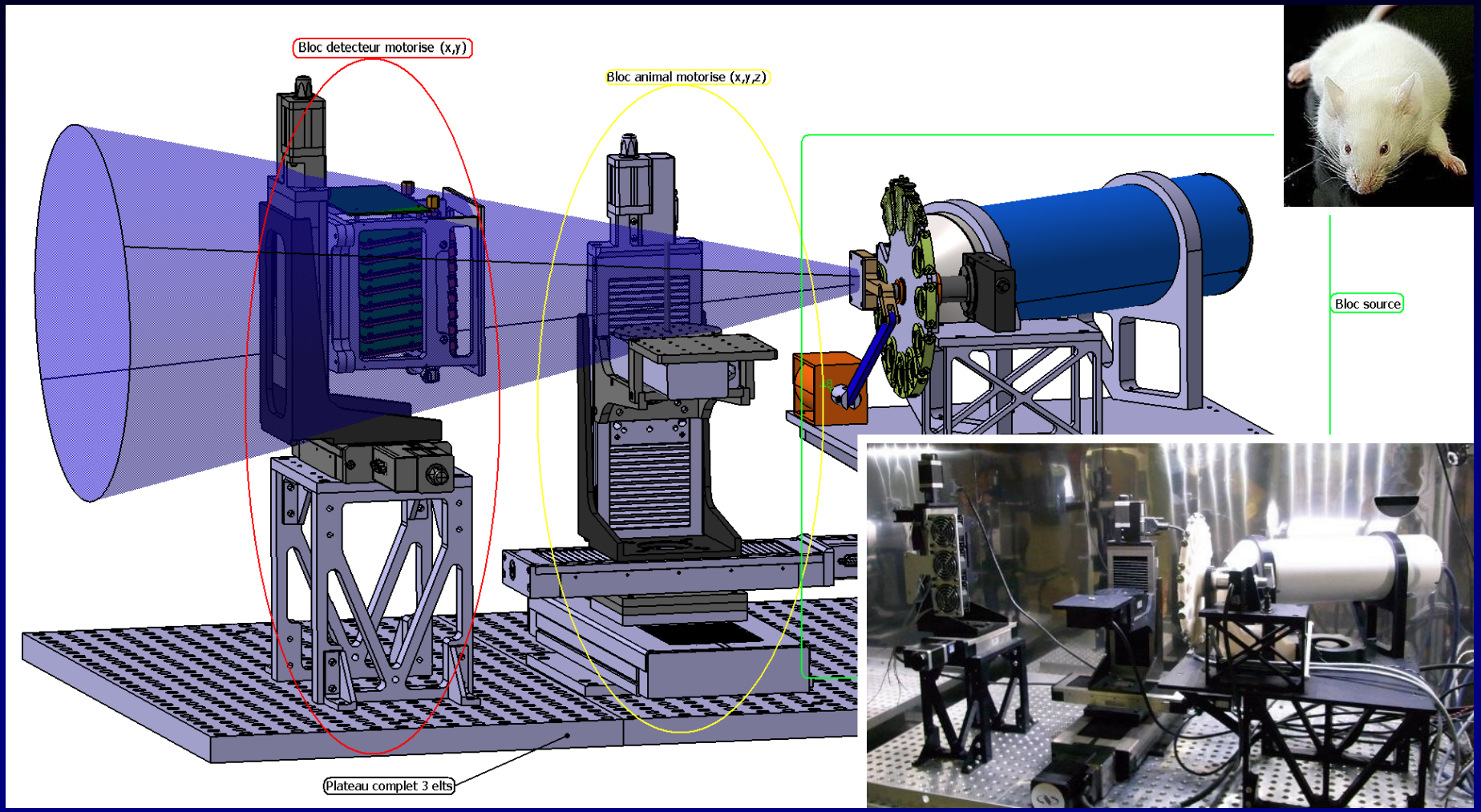
ESRF, SOLEIL



Imagerie biomédicale

imXgam

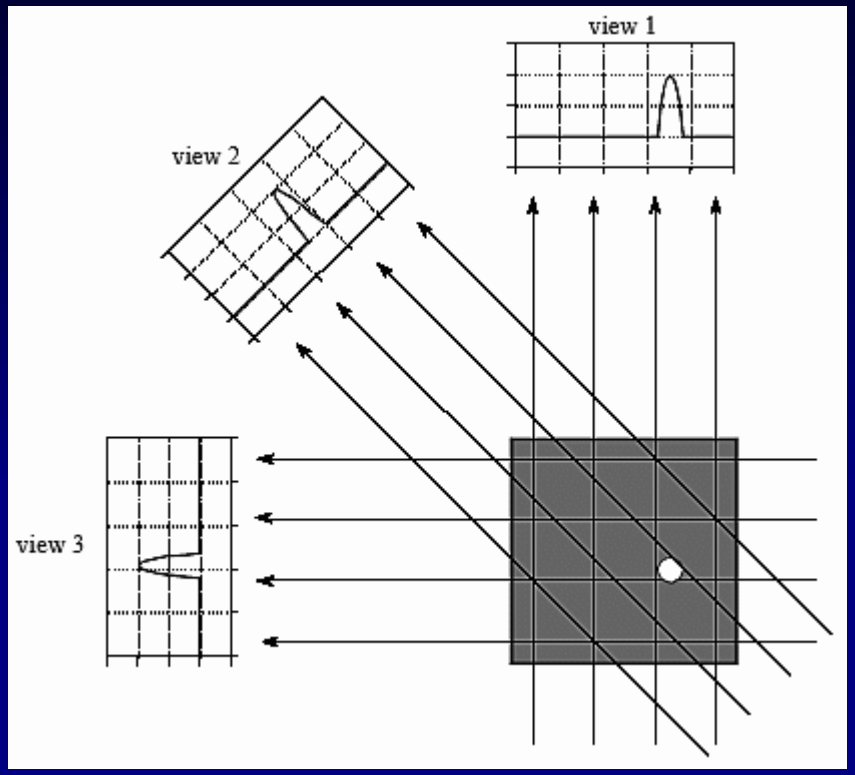
Le scanneur Pixscan II pour petits animaux



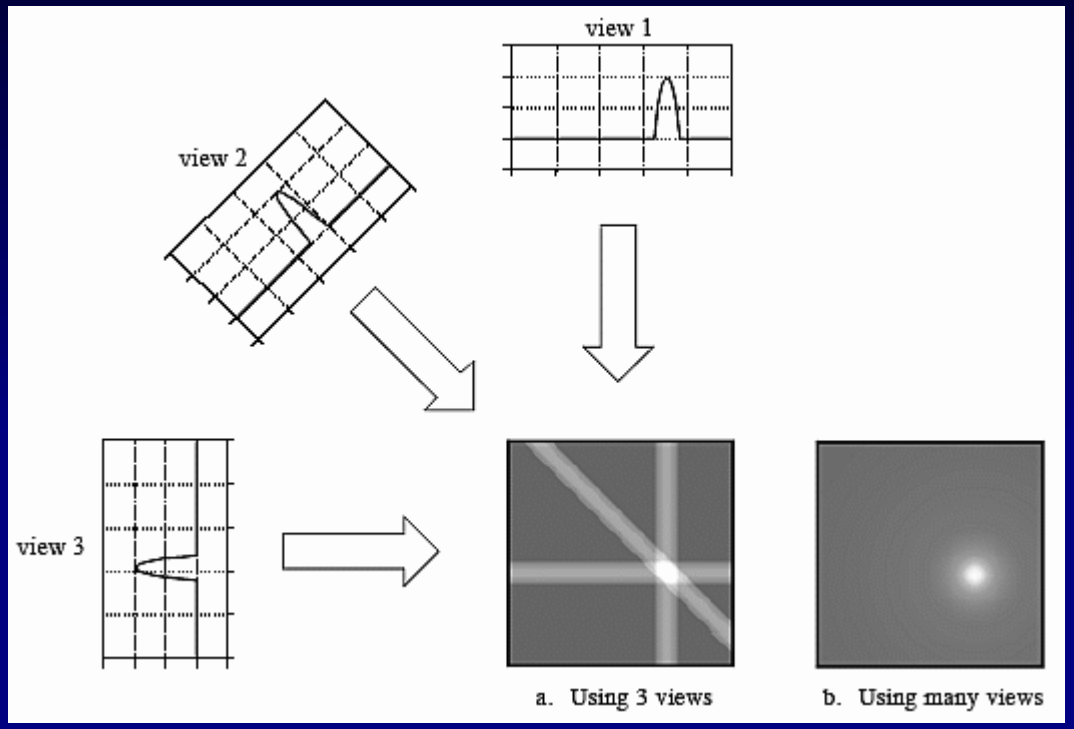
Visualisation 3D du plan CAO du scanneur Pixscan II

La reconstruction tomographique

Principe de fonctionnement ...



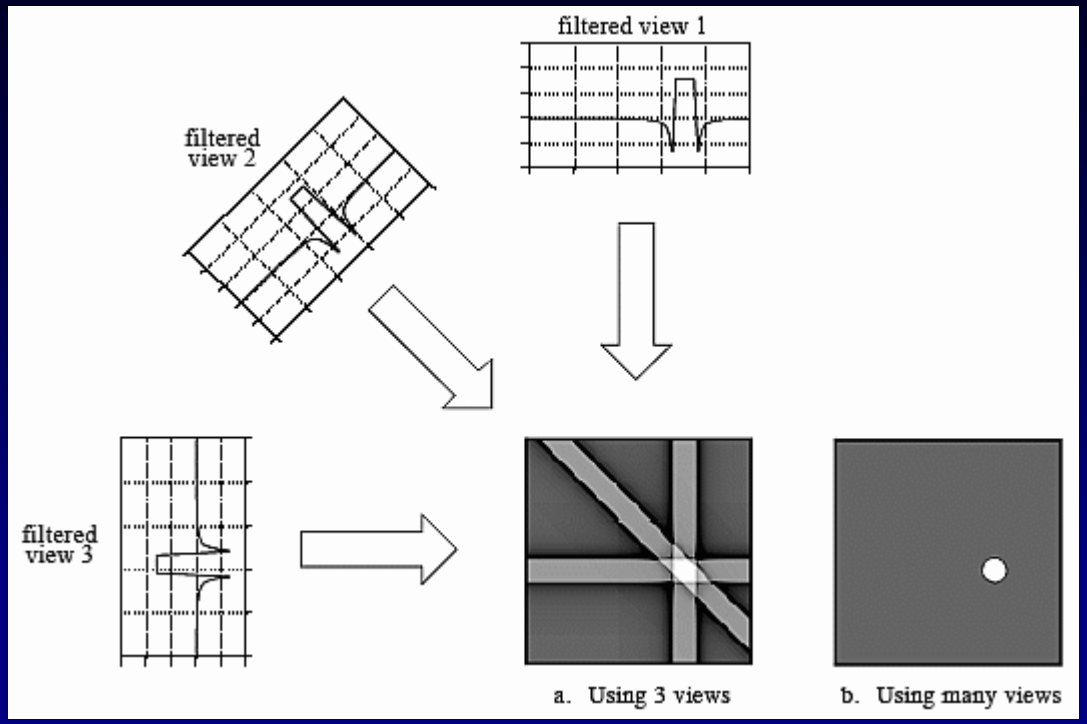
Réalisation du scan



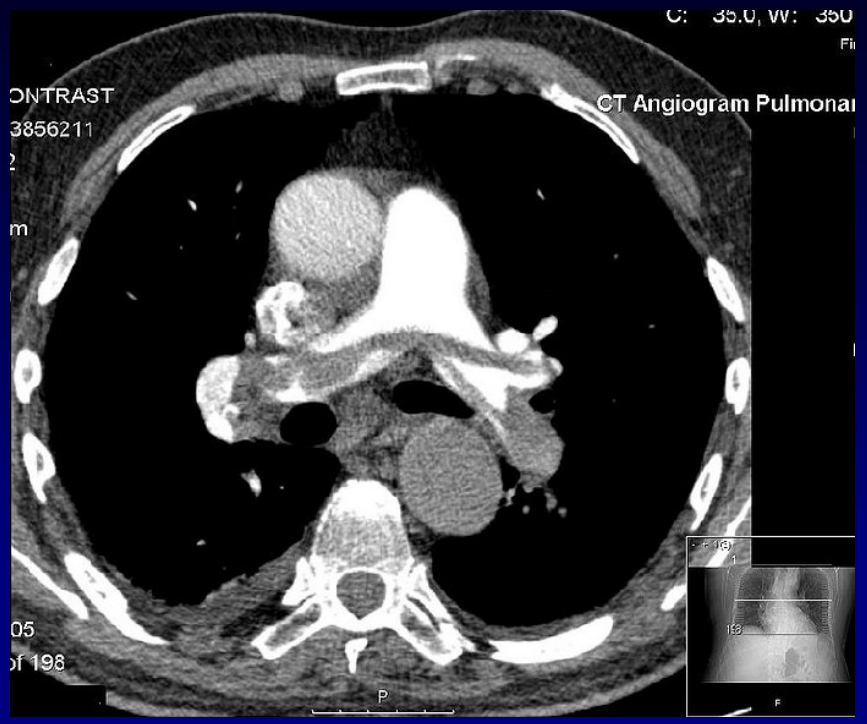
Rétroprojection simple

"The scientist and Engineer's Guide to Digital Signal Processing", S. W. Smith, http://www.dspguide.com

Reconstruction par rétroprojection filtrée



Rétroprojection filtrée

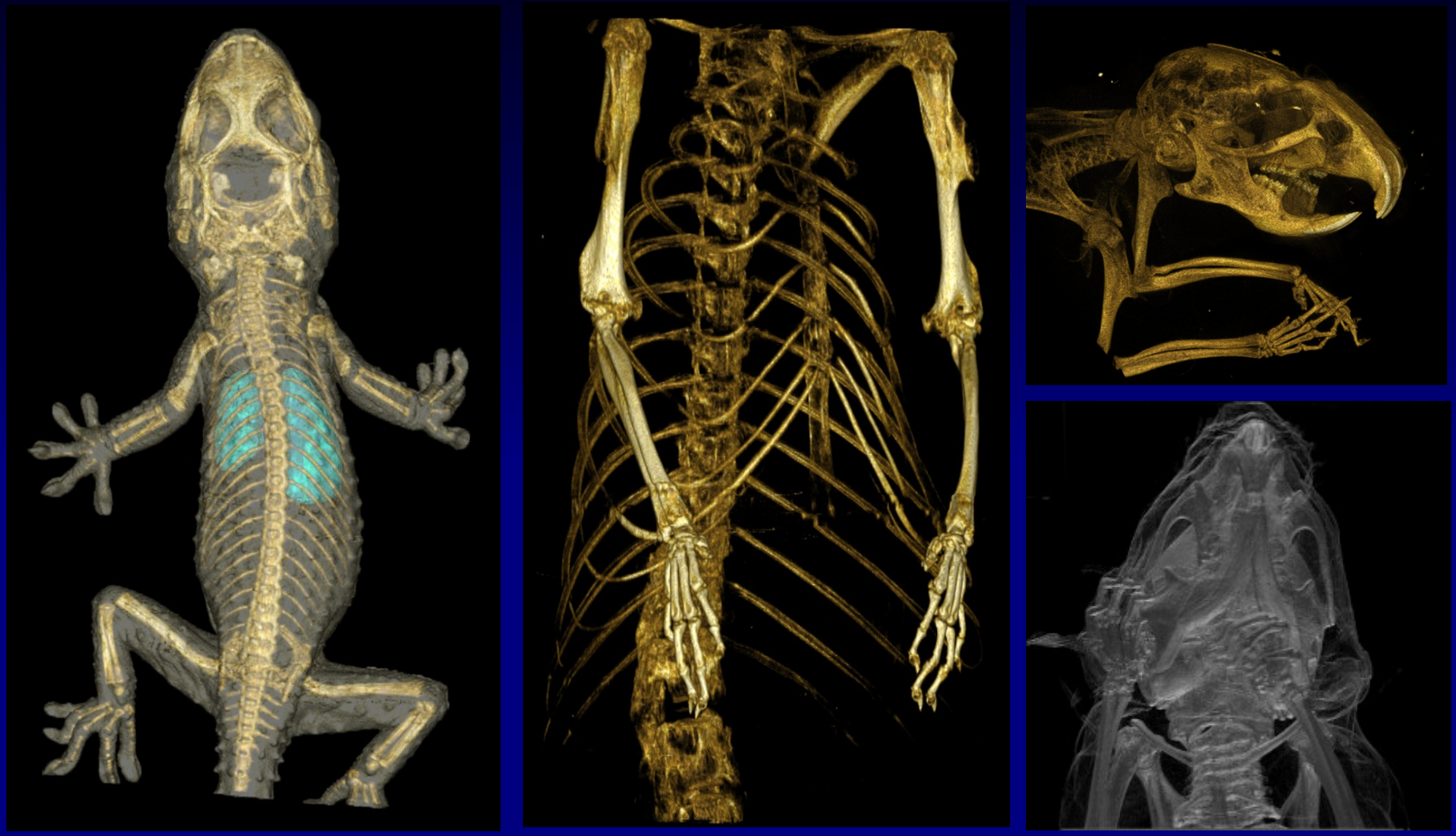


Reconstruction tomographique

http://en.wikipedia.org/wiki/File:SADDLE_PE.JPG

Algorithme de reconstruction en projection conique 3D : FDK

Post-traitements : segmentation, rendu surfacique et projection ...



Gecko et rendu surfacique du squelette de souris réalisé avec le logiciel OSIRYX et des scans XPAD3

Animation : *http://en.wikipedia.org/wiki/File:VolRenderShearWarp.gif*

Cartes graphiques et GPGPU

Calculs pour FDK

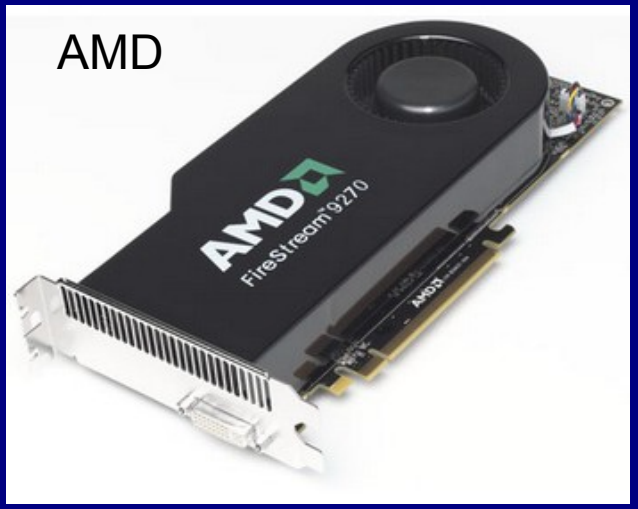
- ✓ simples mais très nombreux
- ✓ plusieurs heures de calcul

Parallélisation massive des calculs

- ✓ avec des processeurs graphiques (GPU)
- ✓ Pionnier : Klaus Mueller avec stations Silicon Graphics (thèse 1998)

Aujourd'hui : cartes graphiques de jeux

AMD



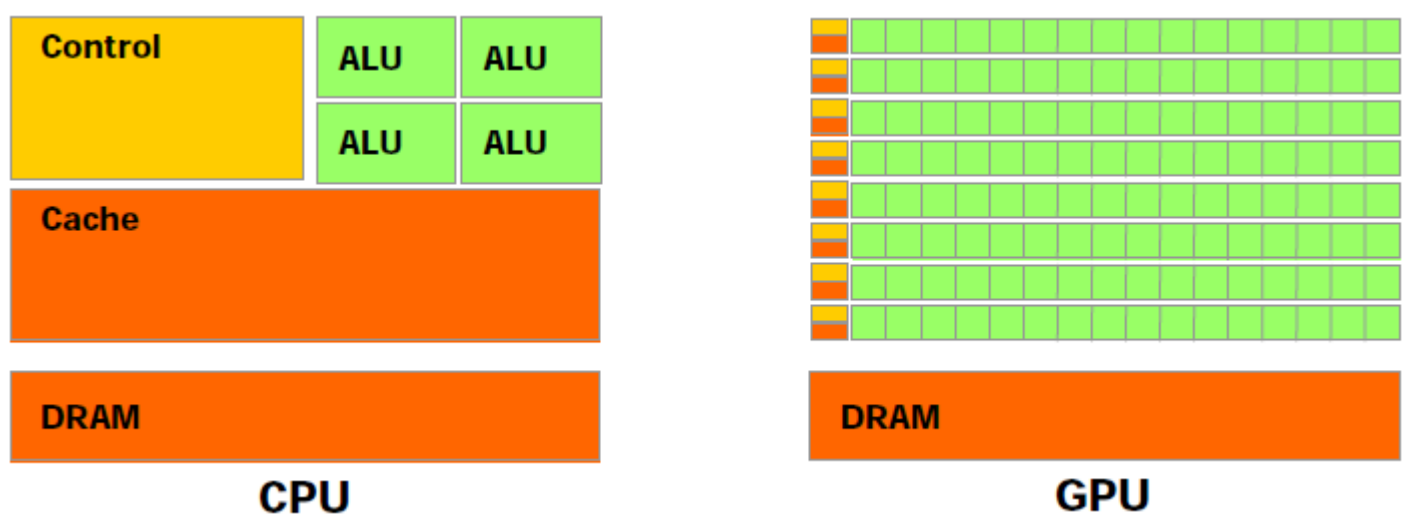
NVIDIA

AMD-ATI

- ✓ GPU conçu pour traitements d'images en flux spécialisés (stream)
- ✓ Processeur vectoriel

NVIDIA

- ✓ GPU organisé en multi-coeurs génériques
- ✓ Parallélisation de threads (pas du pur vectoriel)



NVIDIA_CUDA_Programming_Guide_2.3.pdf
http://www.nvidia.com/object/cuda_develop.html

Programme CUDA

- ✓ Compilateur compatible C++ (nvcc)
- ✓ Définir une fonction *kernel* exécutée en parallèle
- ✓ Exemple d'une définition et d'invocation d'un *kernel*

  Le transfert des données n'est pas présenté dans l'exemple

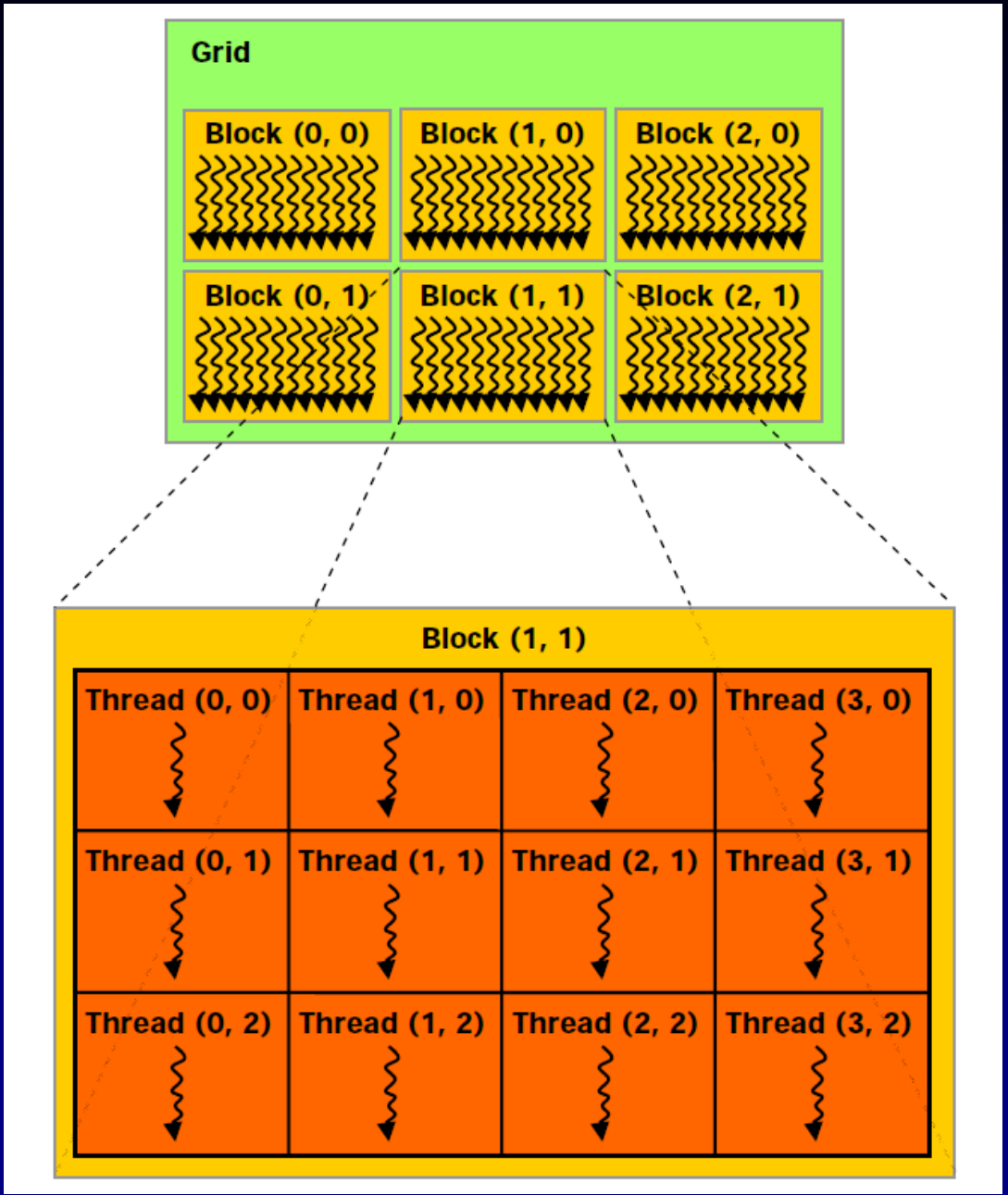
```
// Kernel definition
__global__ void MatAdd(float A[N][N], float B[N][N],
                       float C[N][N])
{
    int i = blockIdx.x * blockDim.x + threadIdx.x;
    int j = blockIdx.y * blockDim.y + threadIdx.y;
    if (i < N && j < N)
        C[i][j] = A[i][j] + B[i][j];
}

int main()
{
    ...
    // Kernel invocation
    dim3 dimBlock(16, 16);
    dim3 dimGrid((N + dimBlock.x - 1) / dimBlock.x,
                 (N + dimBlock.y - 1) / dimBlock.y);
    MatAdd<<<dimGrid, dimBlock>>>(A, B, C);
}
```

Threads lancés par grille

- ✓ Matrice 3D de blocs
- ✓ Matrice 3D de threads

Détermine la répartition dans les coeurs du GPU



Le transfert des données

- ✓ Espace mémoire distinct entre CPU et GPU
- ✓ Le transfert doit être explicitement décrit dans le code

Plusieurs zones mémoires différentes (NIVIDIA)

- ✓ Plusieurs zones mémoires différentes avec GPU
- ✓ Mémoire globale, mémoire locale, mémoire textures,...

Bibliothèque CUDA Templates

- ✓ Simplifie considérablement la programmation
- ✓ Projet open source http://cudatemplates.sourceforge.net/
- ✓ Documentation insuffisante
- ✓ Utiliser les programmes de tests comme exemples

Exemple de programme CUDA fonctionnel

✓ Calcule le carré des éléments d'un vecteur

Prérequis :

✓ Pilotes supportant CUDA et SDK installés : *nvcc* fonctionnel

✓ CUDA Templates : installer cudatemplates dans /usr/local/include

Compilation : > nvcc main.cu -o testSquare

Programme :

```
#include <iostream>
#include <cudatemplates/copy.hpp>
#include <cudatemplates/devicememorylinear.hpp>
#include <cudatemplates/hostmemoryheap.hpp>

typedef Cuda::HostMemoryHeap1D<float> memhost_t;
typedef Cuda::DeviceMemoryLinear1D<float> memdev_t;

// GPU kernel computing square of vector elements
__global__ void sqr( memdev_t::KernelData arg )
    { arg.data[threadIdx.x] *= arg.data[threadIdx.x]; }
```

```
// main program
int main( int argc, char *argv[] )
{
  const size_t SIZE = 256;  // define vector size
  memhost_t h_arg(SIZE);    // declare and instantiate vector in host memory
  memdev_t d_arg(SIZE);     // declare and instantiate vector in device memory

  for( int i = 0; i < SIZE; ++i ) h_arg[i] = i; // initialize host vector

  copy( d_arg, h_arg ); // copy data from host memory to device memory

  // execute kernel:
  dim3 dimGrid(1, 1, 1);
  dim3 dimBlock(SIZE, 1, 1);
  sqr<<<dimGrid, dimBlock>>>( d_arg );

  copy( h_arg, d_arg ); // copy result from device memory to host memory

  for( int i = 0; i < SIZE; ++i ) // display result to standard output
     std::cout << i << "\t" << h_arg[i] << std::endl;

  return 0;
}
```

Portage du code FDK et résultats

Code FDK

- ✓ Fonctionnel et validé sur CPU
- ✓ Isoler le kernel dans une fonction et ajouter transfert de données
- ✓ Premier programme CUDA avec CUDA Templates (1 week-end)
- ✓ Modification FDK CPU → FDK CUDA (1 week-end)

Résultats

- ✓ FDK sur 720 images 560x600 → volume 512x512x600
- ✓ VolRec (Solène Valton CREATIS) : 300 minutes
- ✓ RayFDK (Christophe Meessen CPPM) : 150 minutes
- ✓ CudaFDK sur GTX 280 (Christophe Meessen CPPM) : 15 secondes
- ✓ CudaFDK **600x** plus rapide que RayFDK !
- ✓ CudaFDK **1200x** plus rapide que VolRec !

Conclusions

Performances époustouflantes

- ✓ Problème adapté à la parallélisation par kernels
- ✓ Calculs en simple précision (float)

Prise en main

- ✓ Simple et rapide pour une première approche
- ✓ CUDA Templates simplifie considérablement le code
- ✓ Grille de threads et textures sont non intuitive

Optimisation

- ✓ L'optimisation en CUDA est un art (beaucoup de paramètres,...)
- ✓ CUDA Templates simplifie considérablement le code

Avenir ?

- ✓ Langage OpenCL développé par Apple et inspiré de CUDA