Optimization of navigation in regular geometries

John Apostolakis (CERN), **Pedro Arce** (CIEMAT), Gabriele Cosmo (CERN)

12th GEANT4 WORKSHOP

Hebden Bridge (UK), 13-19 September 2007

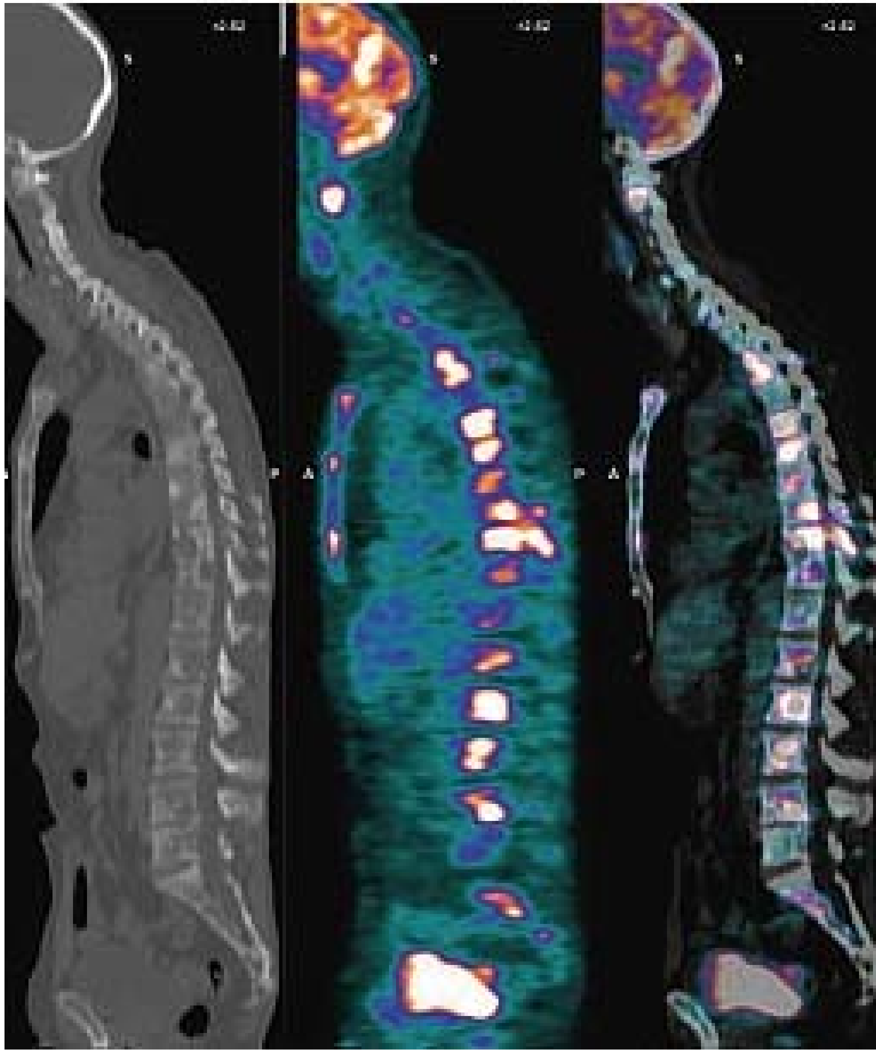
Outline

- Description of the problem
- History of solutions
- Proposed solution
  - G4RegularNavigation
  - Container volume
  - G4RegularParameterisation
  - Skipping voxels with same material
  - Stress tests
  - Performance results
    - 4.5 M phantom
    - 13.6 M phantom

Description of the problem

In medical physics simulation it is frequent the need to simulate a DICOM image of a body

- It consists on a big number of voxels (millions – tens of millions)
- All voxels have the same dimensions
- All voxels are touching each other
- They form a prism with no holes
- Each voxel may have a different material
  - Usually there are a few materials (<10/20) with many different densities (100´s, 1000´s)



☹ **This takes too long time in normal navigation**

☹ **Or needs too much memory and too long initialisation in voxel navigation**

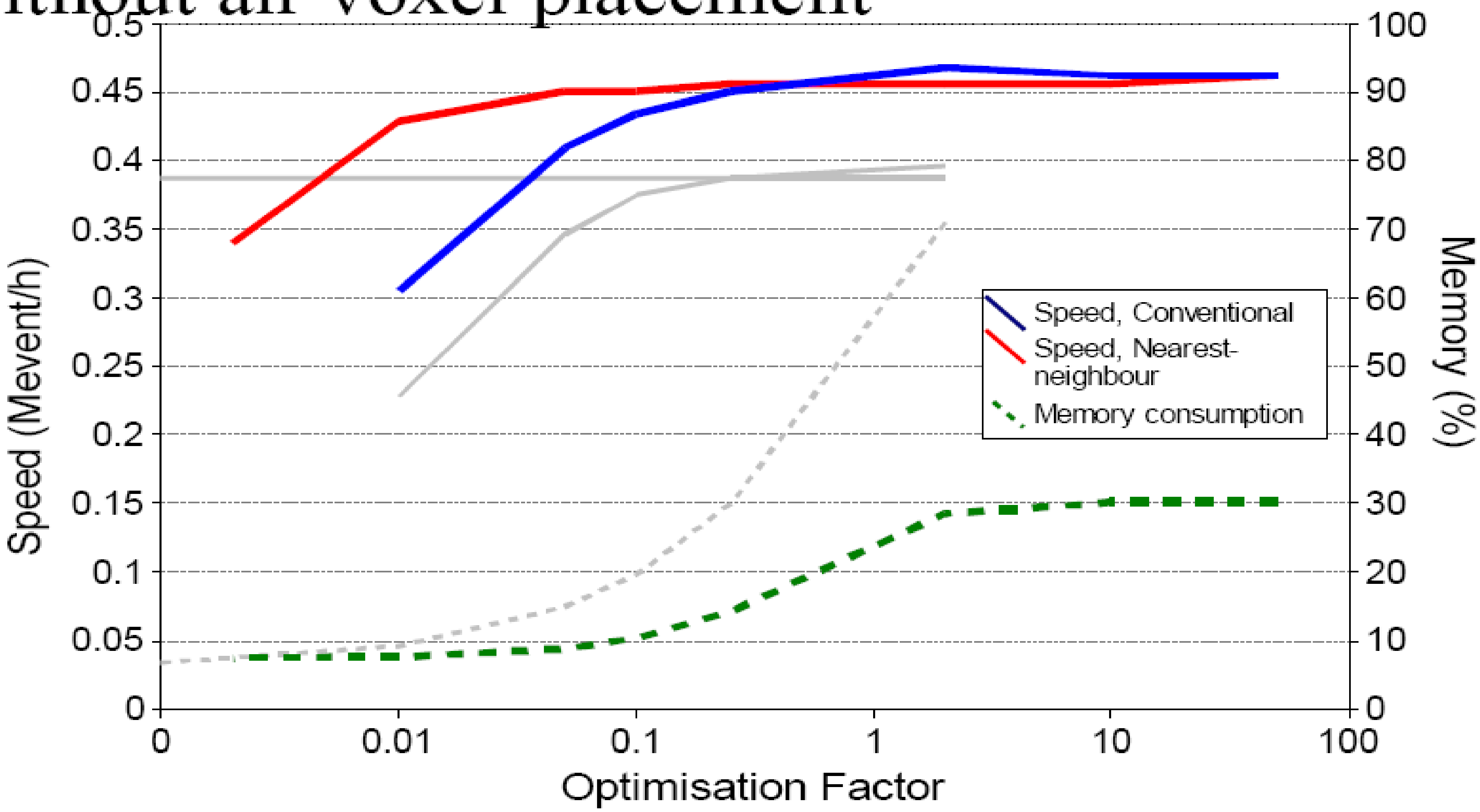
History

Several people have tried to solve this problem

- H.Jiang, H.Paganetti (A. Raijmaakers)
  - Fast navigation by looking at the six nearest voxels only
  - Dynamic assignment of mass density
- K. Sutherland
  - Rewrite G4ParameterisedNavigation using a fast voxel locate algorithm
- V. Hubert-Tremblay, L.Archambault, D.Tubic, R.Roy, L.Beaulieu
  - Reducing the number of voxels by the octree method
- L. Guigues, D. Sarut
  - Implemented in the THIS framework several methods to reduce number of voxels
  - Speed up ComputeSafety to take into account new voxel organisation
- Proposed solution to be included in GEANT4
  - Implementation of a fast navigation for regular geometries (as foreseen in the official GEANT4 plan for 2007)
  - Dynamic assignment of mass density to be discussed during this Workshop

Calculation speed and memory consumption

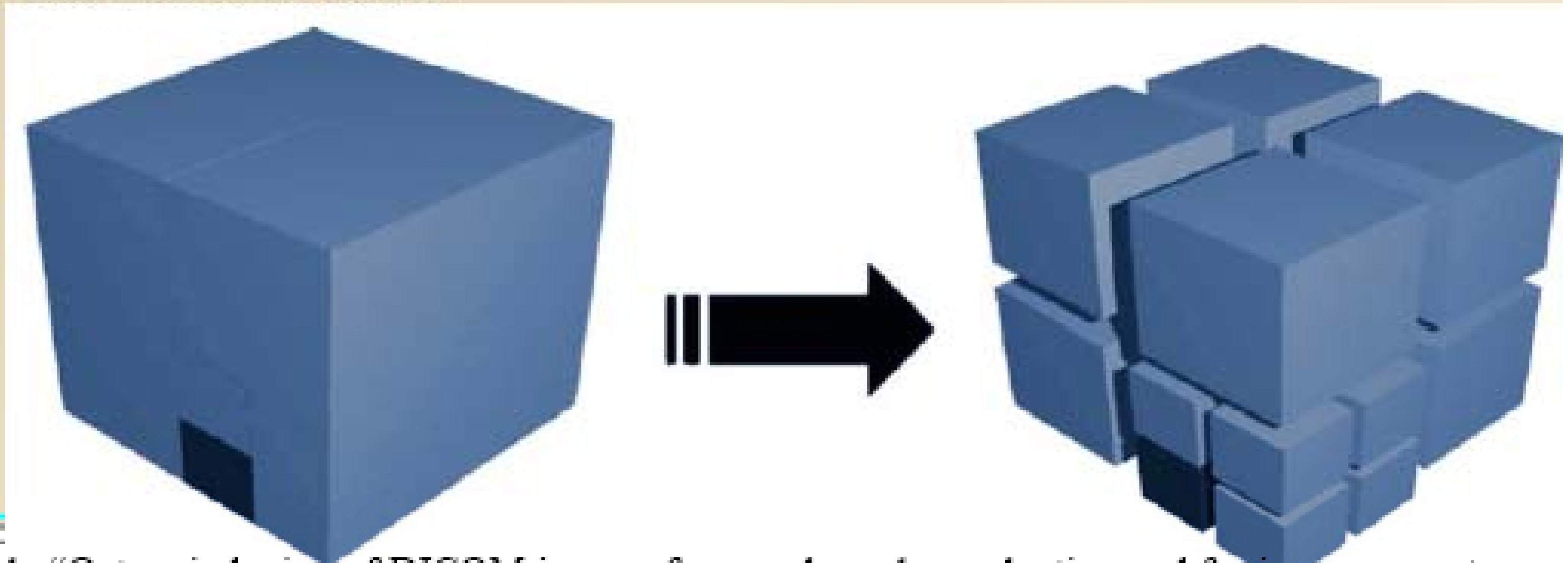
without air voxel placement



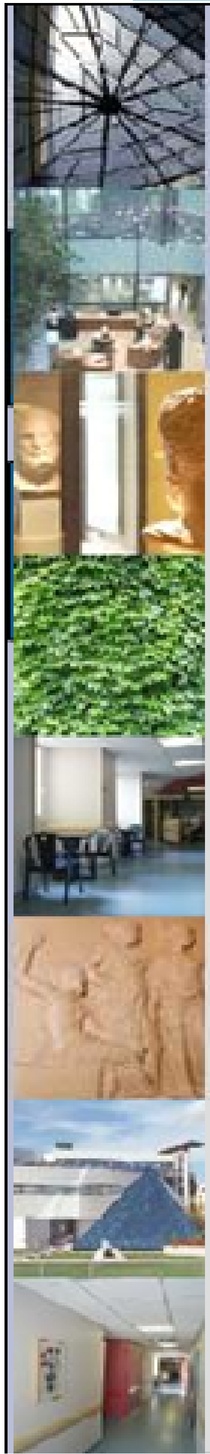
V. Hubert-Tremblay et al.

Voxels Compression

- Dicom Octree compression[1] to lower CPU and memory consumption.
- Algorithm :
  - Take 8 neighbors
  - If the density is almost the same, replace the 8 voxels by the equivalent bigger voxel.
  - Continue on each scales.



1: Hubert Tremblay V, et al., "Octree indexing of DICOM images for voxel number reduction and for improvement of Monte Carlo simulation computing efficiency", Med Phys. 2006 Aug;33(8):2819-31.



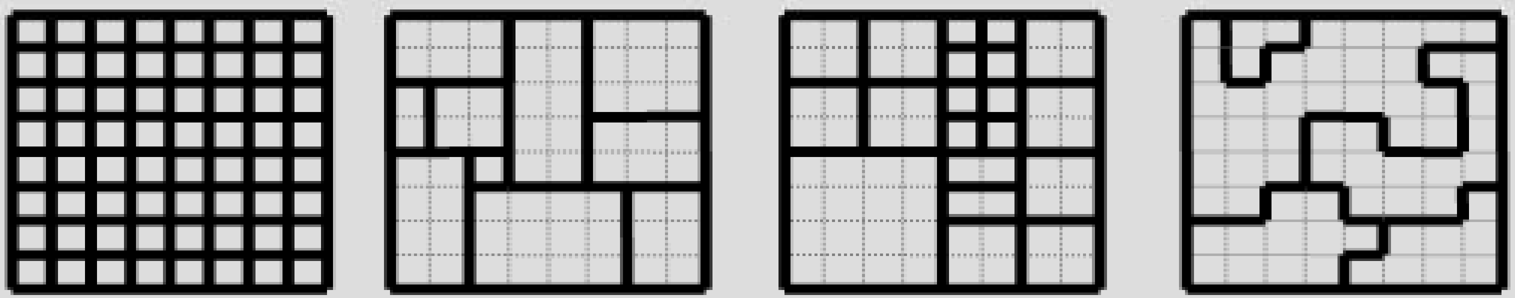
Scene module

- Insert images in simulation

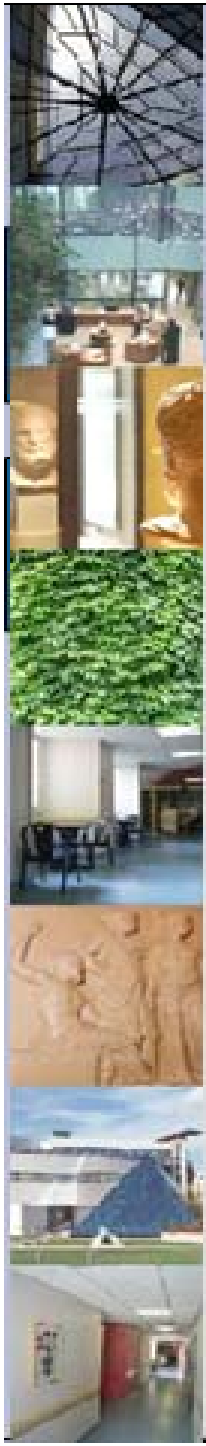
  1) Reduce image complexity

  Available image representations in ThIS :

  - **ImageBoxes** (multiple G4Box)
  - **ImageParameterised** (G4VParameterisedVolume)
  - **ImageNestedParameterised** (G4VNestedParam...Volume)
  - **ImageIsothetic**
  - **ImageRegionalised**

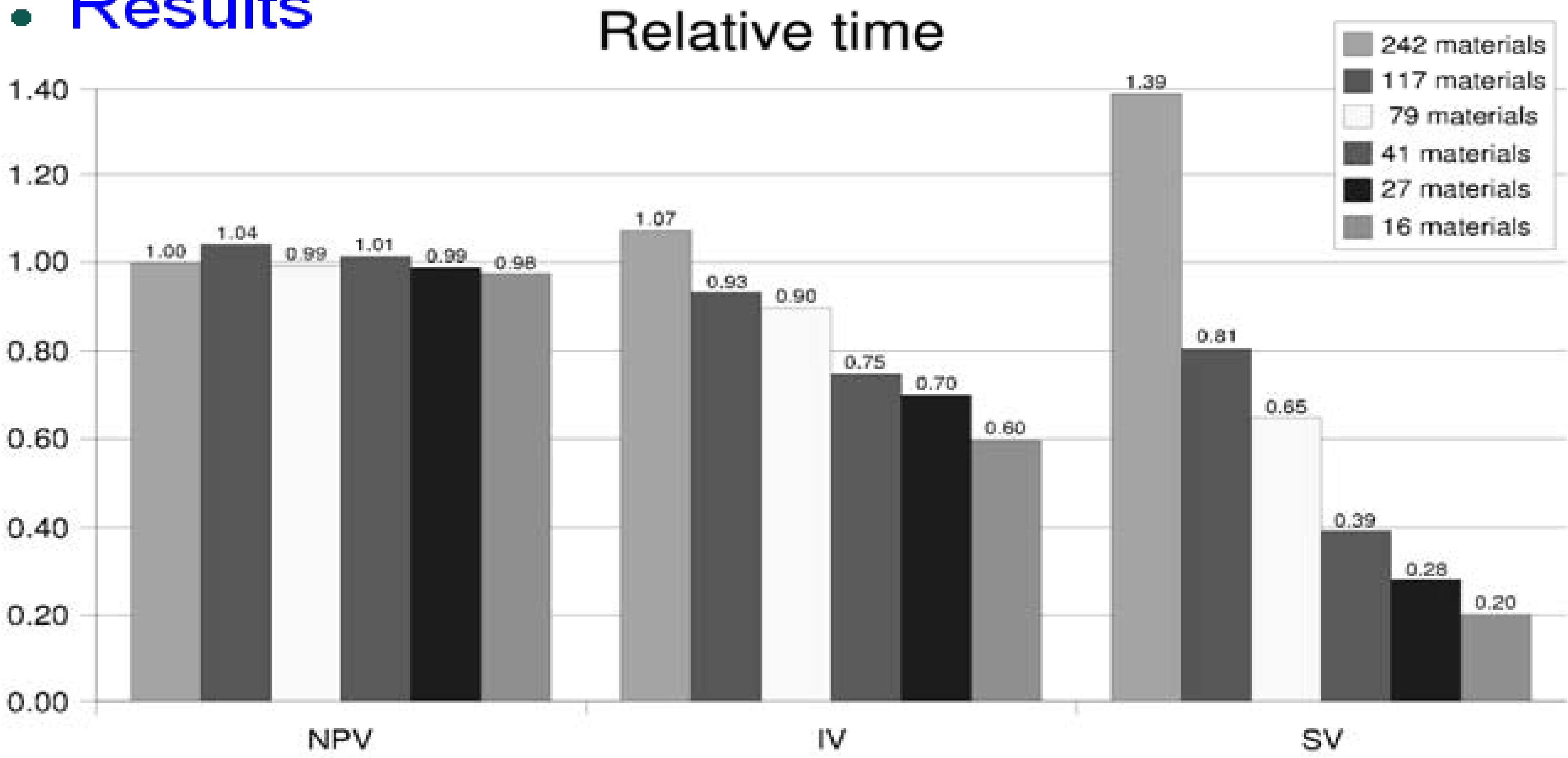


Voxels Isothetic Octree Regions



Scene module

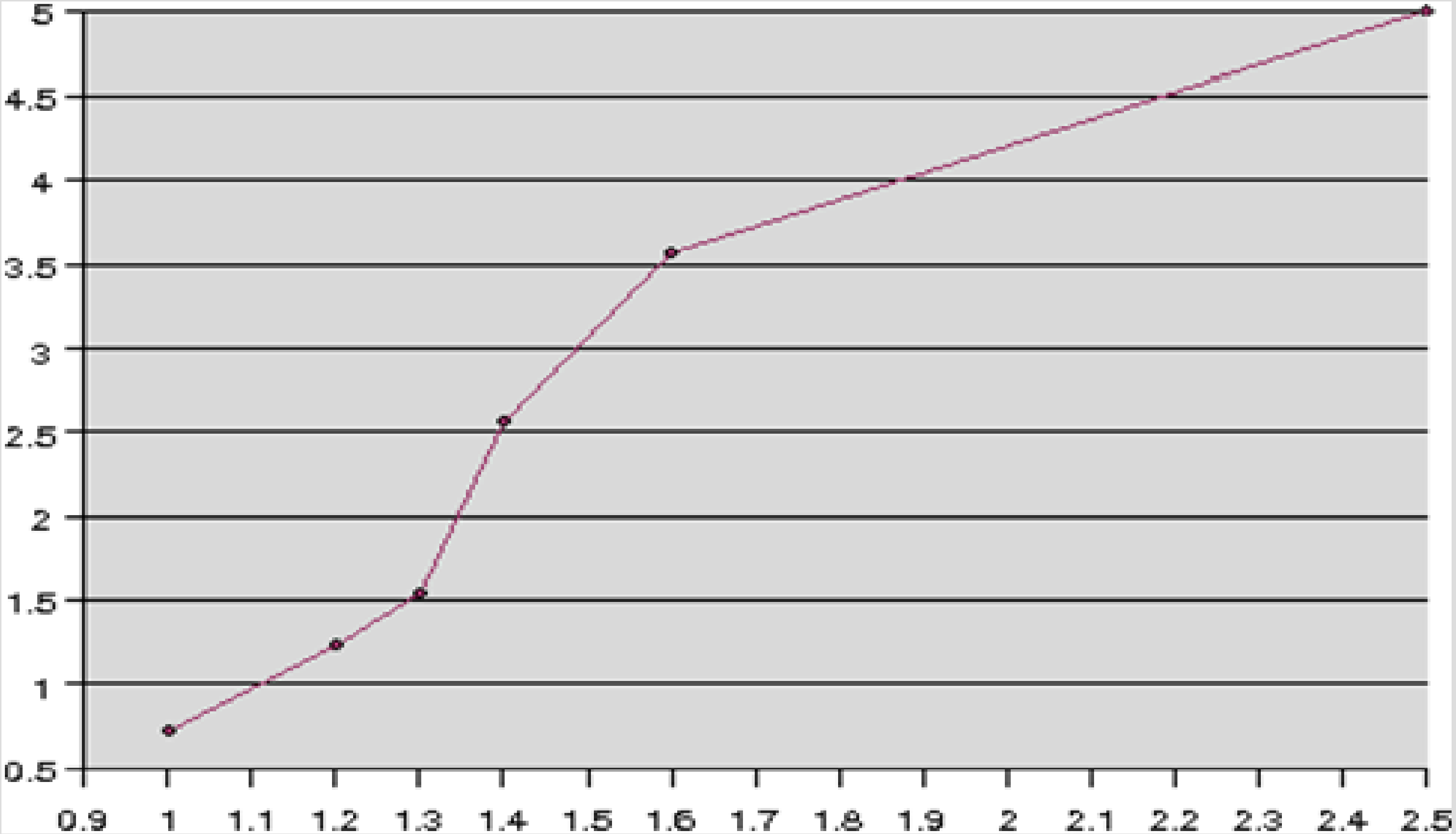
- **Results**



Scene module

- Results

Speed up (time NPV / time RV)



Dose difference RV-NPV (%)

Results

- 3.2 GHz Pentium 4, 2 GB memory
- Water phantom, 20 cm deep target
- 120 MeV proton beam
- Before:
  - 7 events/minute
- After:
  - 1000 events/minute
- Nearly 150 times faster with no noticeable difference in result.

Where is the time spent?

If **voxelised navigation (G4VoxelNavigation)** is chosen:

- Initialization time and memory spent for building the navigation voxels are very big

If **normal navigation (G4ParameterisedNavigation)** is chosen:

- **ComputeStep/ComputeSafety**
  - When moving in the volume mother of the voxels it has to be checked the distance to each voxel
- **LevelLocate**
  - When the track enters a voxel it needs to make a loop to all the voxels to know in which one it is

G4RegularNavigation

G4RegularNavigation::LevelLocate

- Calls G4RegularParameterisation::GetReplicaNo to locate the copy number corresponding to a position
- It computes the voxel number in X, Y & Z axis by a simple calculation (plus precision corrections)

```
G4double fx = (localPoint.x()+fContainerWallX+kCarTolerance)/fVoxelHalfX/2.;
G4int nx = G4int(fx);
```

Voxel container volume

The '**localPoint**' of G4RegularNavigation::LevelLocate has to be in the reference frame of the voxels, the **voxel container volume**

- This volume is the mother volume of the voxels and has as dimension the number_of_voxels X voxel_dimension
- It has to be created by the user, placing the voxels in it so that they fill it completely (with a precision equal to the cartesian tolerance = 1.e-9 mm)

- ✓ It facilitates the computation of the copy number
- ✓ It also makes G4RegularNavigation::ComputeStep and G4RegularNavigation::ComputeSafety unnecessary
  - When a track is inside the container it necessarily is inside one of the voxels, so it never happens that a track is in the voxel parent and needs to loop to get the distance of safety to each voxel

How regular navigation is called

- Your G4VPVParameterisation has to be of type **G4RegularParameterisation**

- Your G4PVParameterised needs to have the variable **fRegularStructureCode equal to 1**

```
patient_phys->SetRegularStructureId(1);
```

- G4Navigator is modified to invoke G4RegularNavigation

G4RegularParameterisation

- Your paramaterisation has to be of type G4RegularParameterisation
- Your G4PVParameterised volume has to be placed on a container volume made of the sum of the voxels

```
G4Box* cont_solid = new G4Box("PhantomContainer",
                    nVoxelX*dimX/2.,nVoxelY*dimY/2.,nVoxelZ*dimZ/2.);
```

<u>G4RegularParameterisation</u>

- Define the list of materials in your phantom:

```
G4RegularParameterisation::SetMaterials(std::vector<G4Material*>& mates);
```

- Set the index of materials
  - You have to create a vector of indexes (size_t for optimal speed)

    ```
    size_t* mateIDs = new size_t[nVoxelX*nVoxelY*nVoxelZ];
    ```

    - For each voxel it contains the index of its material in the list of defined materials

```
G4RegularParameterisation::SetMaterialIndices( size_t* matInd );
```

G4RegularParameterisation

- Set the voxel dimensions

G4RegularParameterisation:: SetVoxelDimensions(G4double halfx, G4double halfy, G4double halfz);

- Set the number of voxels

G4RegularParameterisation:: SetNoVoxel(size_t nx, size_t ny, size_t nz);

- Store the container dimensions in the parameterisation

G4RegularParameterisation:: BuildContainerSolid(G4VPhysicalVolume *pMotherPhysical ;

pMotherPhysical is the container volume

- If you want you can check that the voxels fill completely the container (GEANT4 will give an exception at run time if it is not)

G4RegularParameterisation::CheckVoxelsFillContainer(G4double contX, G4double contY, G4double contZ)

contX/Y/Z are the container solid dimensions

Skipping equal material-voxels

- When the track traverses **two contiguous voxels with same materials**, there is no real need to stop on the frontier

✓ We have implemented the **skipping of equal-material frontiers** as default option in G4RegularNavigation

➢ **It can be set off**

  - If there are many different materials in the voxels, there will be very few steps where equal materials are found and the time checking it may not compensate it

Performance: CPU time

Intel Core2 @ 2.0 GHz, 2Gb RAM

256X256X68 = 4.5 Mvoxels

- Tracks starting at a point in the container volume towards (0.,0.,0.) (a difficult point)

CPU user time (sec per 1k events)

geantinos

Regular (4 materials)	Regular (57 materials)*1	Regular (500 mate)*2	Regular (no skip material)	Normal (57 materials)	Voxelised (57 materials)
0.41	0.52	0.80	1.36	1695	0.58

*1 diff in density < 0.1 g/cm^3 *2 diff in density < 0.01 g/cm^3

6 MeV gammas

Regular (57 materials) (no skip mat.)	Normal (57 materials) (no skip mat.)	Voxelised (57 materials) (no skip mat.)
0.93	1942	0.55

120 MeV protons

Regular (57 materials) (no skip mat.)	Normal (57 materials) (no skip mat.)	Voxelised (57 materials) (no skip mat.)
6.2	850	3.3

Frequent warnings and some crash at Normal and Voxelised navigation at point (0.,0.,0.)

Performance: CPU time

Intel Core2 @ 2.0 GHz, 2Gb RAM

512X512X52 = 13.6 Mvoxels

- Tracks starting at a point in the container volume towards (0.,0.,0.) (a difficult point)

CPU user time (sec per 1k events)

geantinos

Regular (4 materials)	Regular (69 materials) *1	Regular (579 mate) *2	Regular (no skip material)	Normal (69 materials)	Voxelised (69 materials)
0.55	0.83	1.38	2.06	4683	*

*1 diff in density < 0.1 g/cm^3 *2 diff in density < 0.01 g/cm^3

6 MeV gammas

Regular (69 materials) (no skip mat.)	Normal (69 materials) (no skip mat.)	Voxelised (69 materials) (no skip mat.)
1.53	5126	*

120 MeV protons

Regular (69 materials) (no skip mat.)	Normal (69 materials) (no skip mat.)	Voxelised (69 materials) (no skip mat.)
8.3	2235	*

*** Memory is exhausted**

Performance: Init time & memory

Intel Core2 @ 2.0 GHz, 2Gb RAM

256X256X68 = 4.5 Mvoxels

- Tracks starting at a point in the container volume towards (0.,0.,0.) (a difficult point)

Initialisation time (seconds)

Regular (4 materials)	Regular (57 materials)	Regular (500 materials)	Voxelised (4 materials)	Voxelised (57 materials)	Voxelised (500 materials)
27	100	662	169	240	805

Memory (Mb)

Regular (4 materials)	Regular (57 materials)	Regular (500 materials)	Voxelised (4 materials)	Voxelised (57 materials)	Voxelised (500 materials)
195	279	955	2061	2142	2821

Performance: Init time & memory

Intel Core2 @ 2.0 GHz, 2Gb RAM

256X256X68 = 13.6 Mvoxels

- Tracks starting at a point in the container volume towards (0.,0.,0.) (a difficult point)

Initialisation time (seconds)

Regular (4 materials)	Regular (69 materials)	Regular (579 materials)	Voxelised (4 materials)	Voxelised (69 materials)	Voxelised (579 materials)
70	162	916	*	*	*

Memory (Mb)

Regular (4 materials)	Regular (69 materials)	Regular (579 materials)	Voxelised (4 materials)	Voxelised (69 materials)	Voxelised (579 materials)
318	418	1226	*	*	*

*** Memory is exhausted**

Stress tests

✓ We have included several checks in the code to avoid precision problems

❖ We have simulated

- ✓ 1.2 milliard events on 1-million voxels phantom
- ✓ 20 million events on a 4.5-million voxels phantom
    - - Tracks passing by corners, tracks along walls

No crash!

Conclusions

❖ Simulation in DICOM files is usually too slow and/or requires too much memory

✓ We have developed a fast navigation algorithm for regular geometries

- ✓ Plus the option to skip voxel frontiers when both materials are the same

✓ Dynamic material denstiy assignment is under study

✓ Quite robust: > 1.2 milliard events with no crash

✓ Similar speed as voxelized navigation (3D optimisation) but much faster initialization time and smaller memory consumption

✓ Same initialisation time and memory consumption as for normal navigation (non optimised), but far more performant

Conclusions

- ✓ For middle/big size phantoms, we get **improvements in navigation** (geantinos) of a factor **>1000** (bigger as number of voxels increases)
  - ✓ Plus **a factor 2/4 if equal-materials voxel frontier is skipped**

- ✓ For middle/big size phantoms, we get **improvements in physics events of >1000 for 6 MeV gammas and ~200 for 120 MeV protons**

- ➢ Before us other people had interesting ideas to solve this problem
  - ➢ We will be happy to hear their feedback and discuss their suggestions before releasing in GEANT4