STATUS OF THE UNIFIED SOLIDS LIBRARY

Gabriele Cosmo/CERN

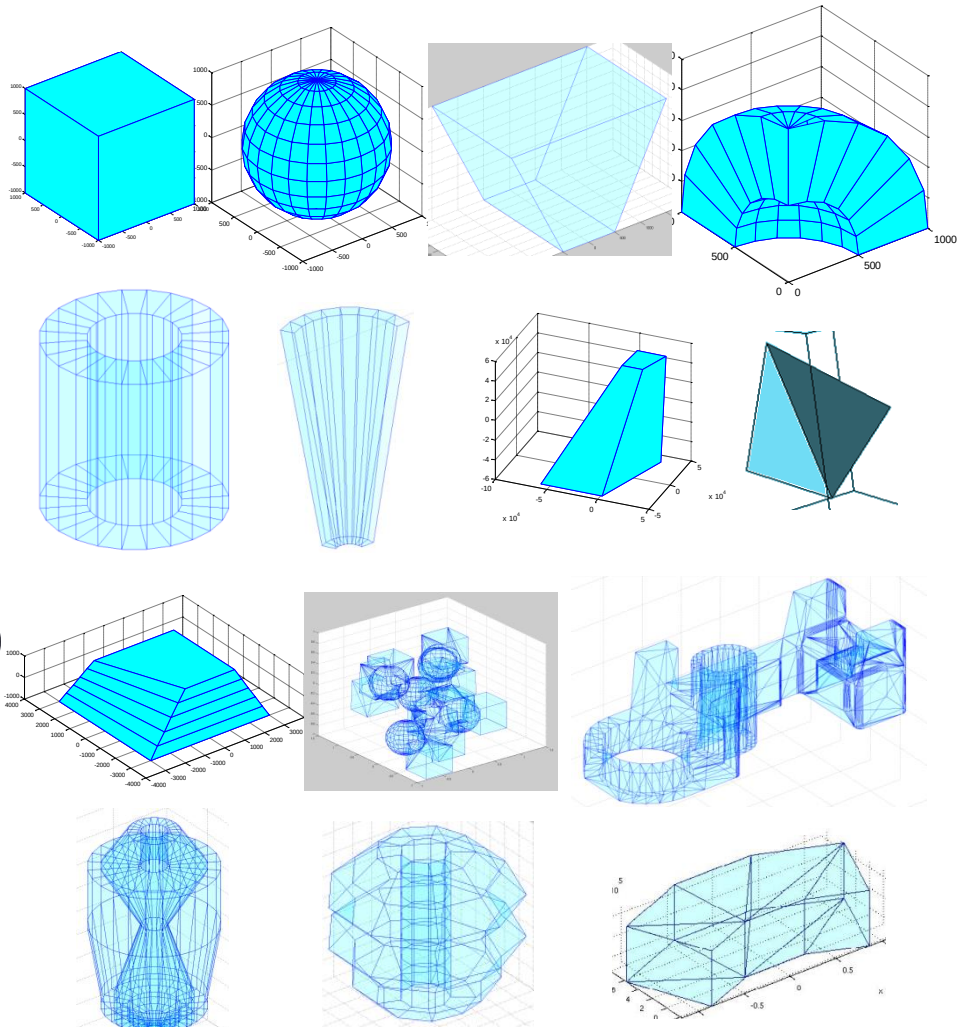
Tatiana Nikitina/CERN

Motivations for a common solids library

- Optimize and guarantee better long-term maintenance of ROOT and Geant4 solids libraries
- Create a single high quality library to replace solid libraries in Geant4 and ROOT
  - Starting from what exists today in Geant4 and ROOT
  - Adopt a single type for each shape
  - Significantly optimize (Multi-Union, Tessellated Solid, Polyhedra, Polycone)
  - Reach complete conformance to GDML solids schema
- Create extensive testing suite

Solids implemented so far

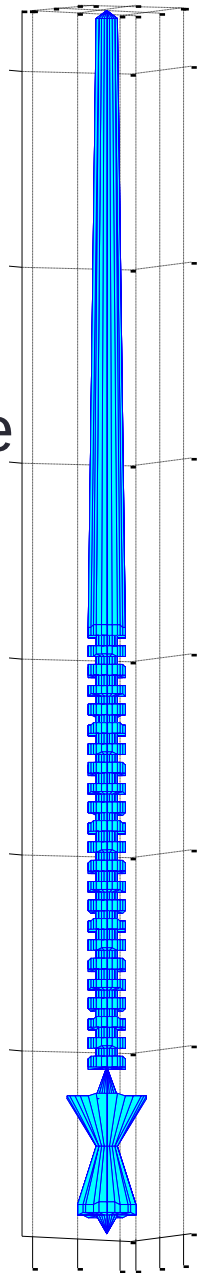
- Box
- Orb
- Trapezoid
- Sphere (+ sphere section)
- Tube (+ cylindrical section)
- Cone (+ conical section)
- Generic trapezoid
- Tetrahedron
- **Arbitrary Trapezoid** (ongoing)
- Multi-Union
- Tessellated Solid
- **Polycone**
- **Polyhedra**
- **Extruded solid** (ongoing)



Polycone

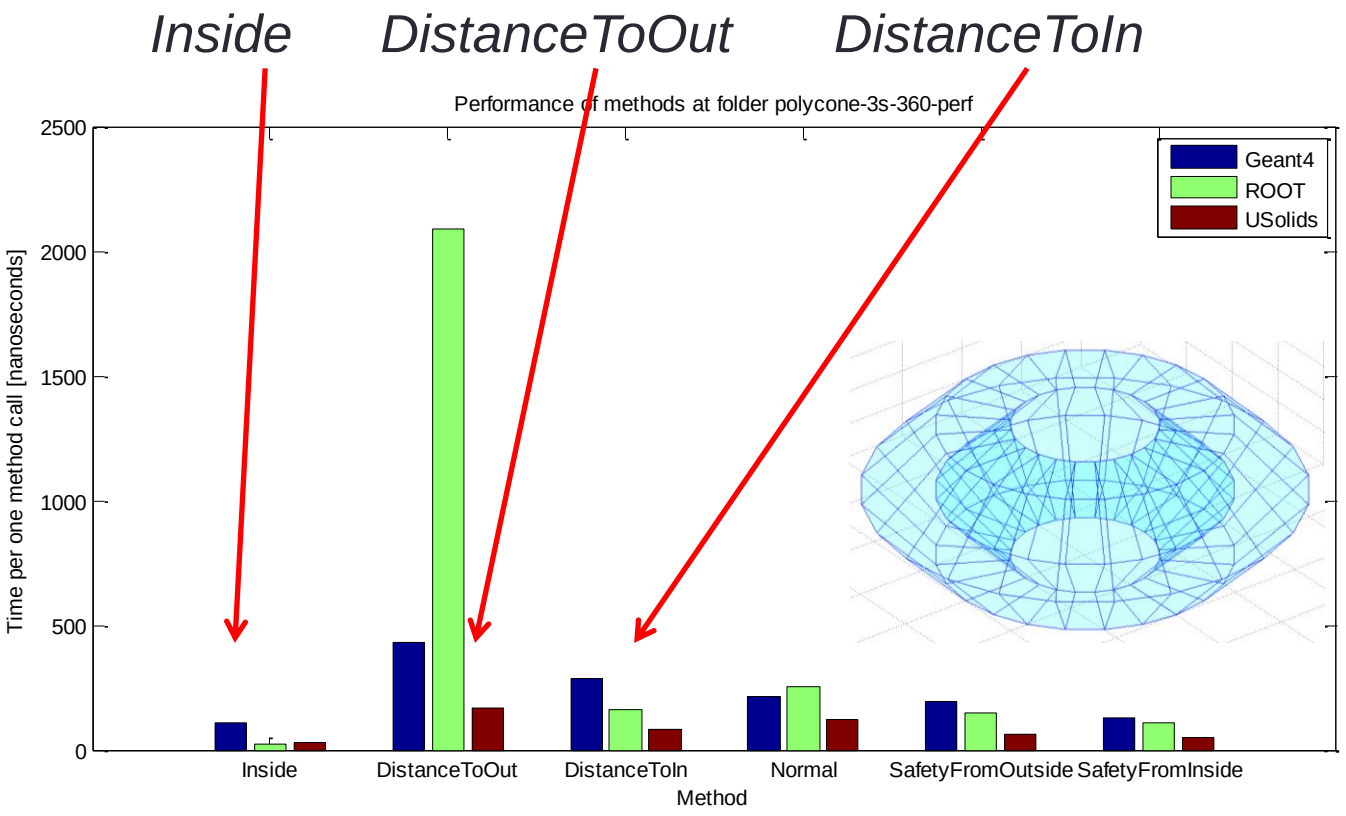
New Ordinary Polycone

- Special optimization for common cases, where Z sections can only increase (big majority of real cases)
- New implementation based on composition of separate instances of cones, tubes (or their sections)
  - With spatial optimisation over Z
  - Helping for more readable, more compact and faster code
  - Excellent scalability over the number of Z sections
  - Significant performance improvement



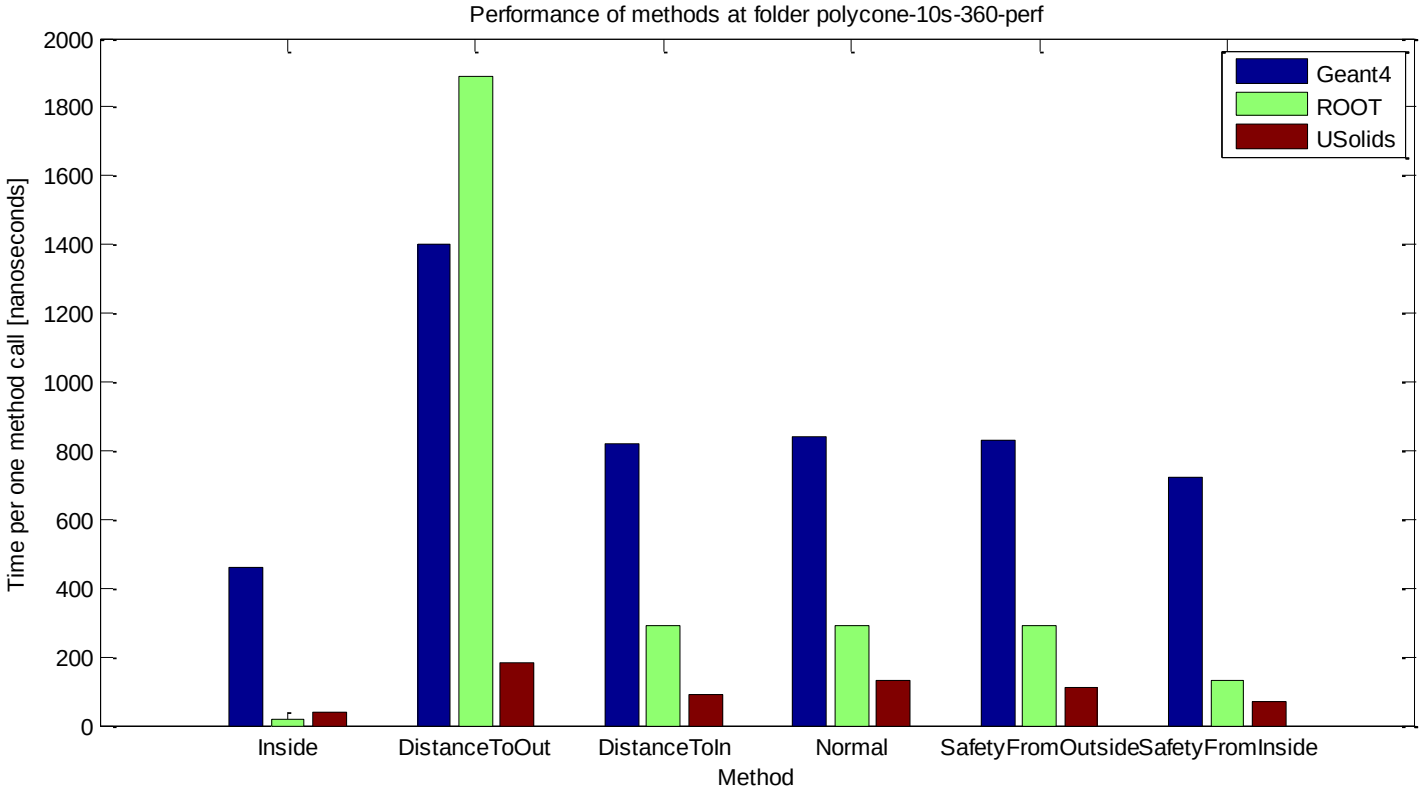
Ordinary Polycone performance
3 Z-sections

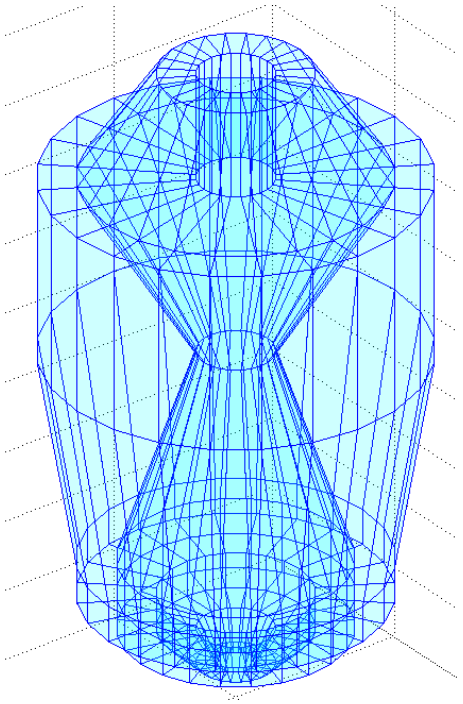
- Speedup factor **3.3x** vs. Geant4, **7.6x** vs. ROOT for most performance critical methods, i.e.:



Ordinary Polycone performance
10 z sections

- Speedup factor **8.3x** vs. Geant4, **7.9x** vs. ROOT for most performance critical methods

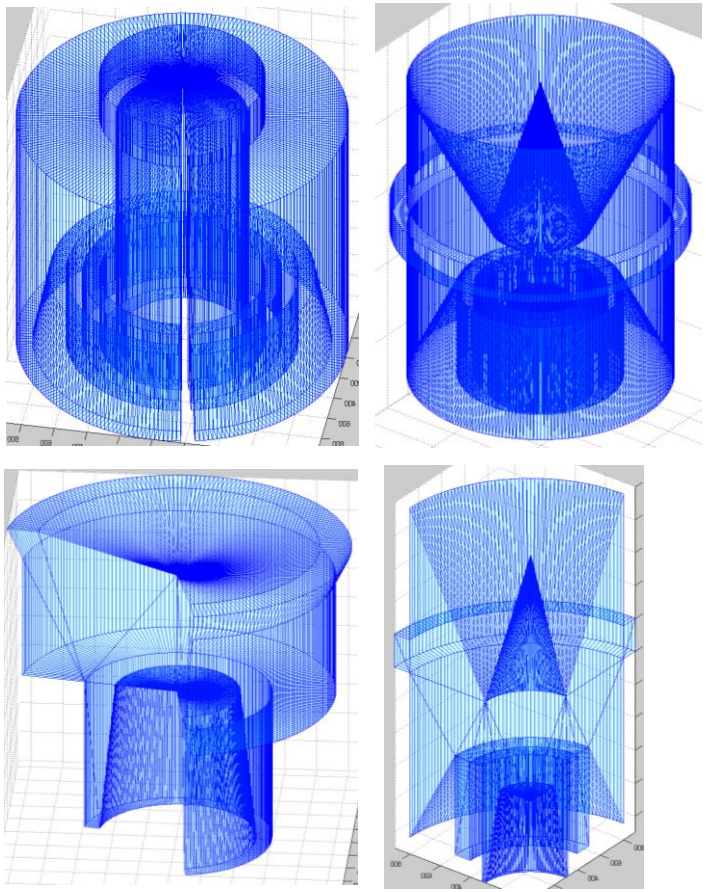




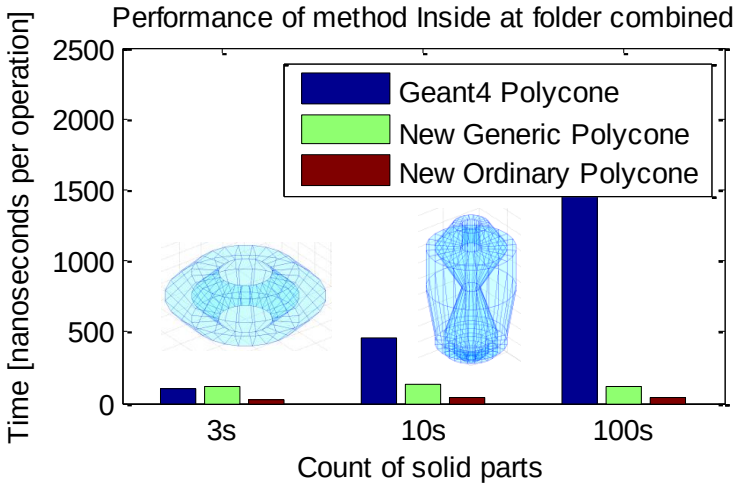
NOTE: *Geant4 poor performance scalability (high number of Z sections) due to absence of spatial optimization*

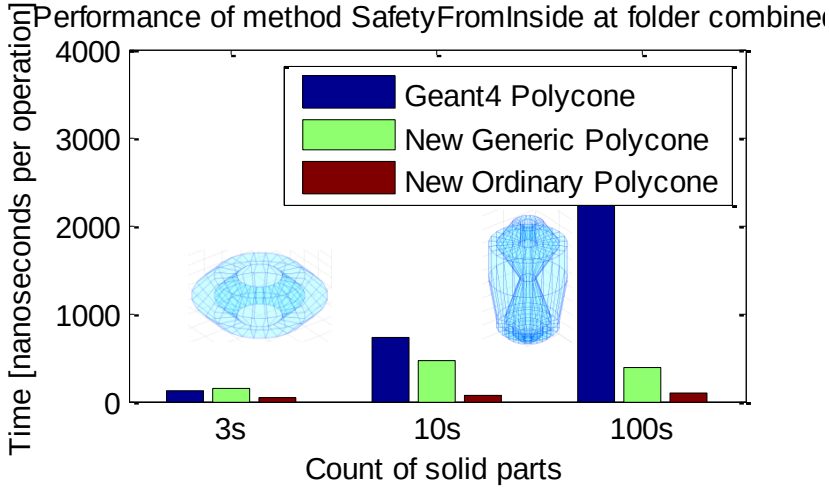
Generic Polycone

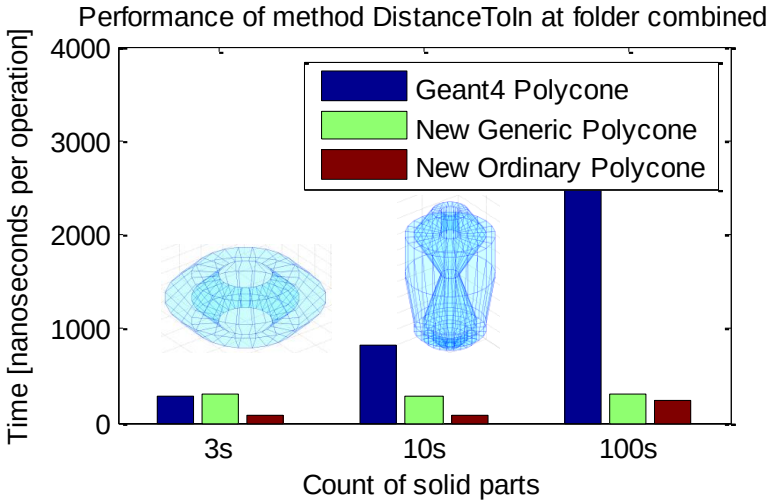
- Case in which either the outer or the inner surface has more than one cone or tube section over a finite interval of Z values
  - **Using voxelization** on generalized surface facet model
  - Performance improvement over Geant4 (generic polycone exists only in Geant4)
  - Performance improvement depends on the number of sections and is slightly worse than in case of ordinary polycone
  - Scalability is as excellent as for the ordinary polycone case
  - Tested and measured on original Geant4 test cases of solids
  - Values are 100% numerically coincident with Geant4

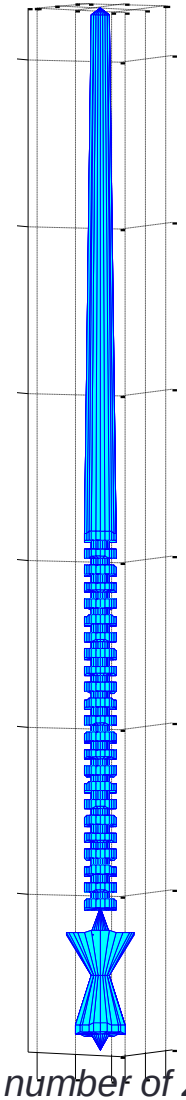


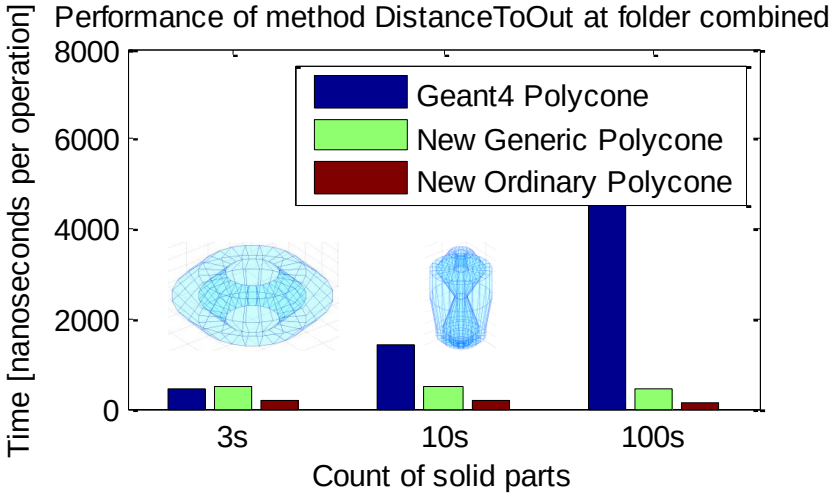
Ordinary vs. Generic Polycone









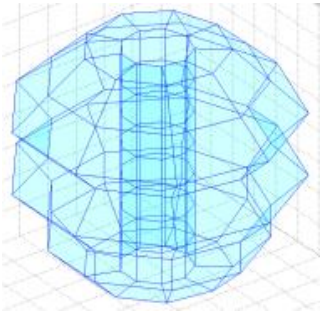


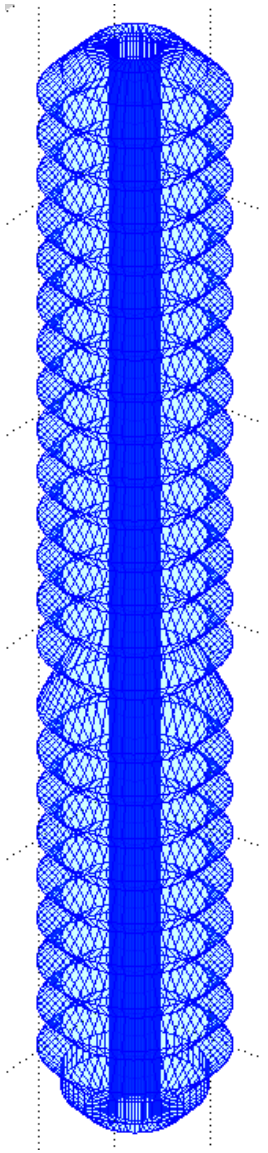
NOTE: *Geant4 poor performance scalability (high number of Z sections) due to absence of spatial optimization*

Polyhedra

New Polyhedra implementation

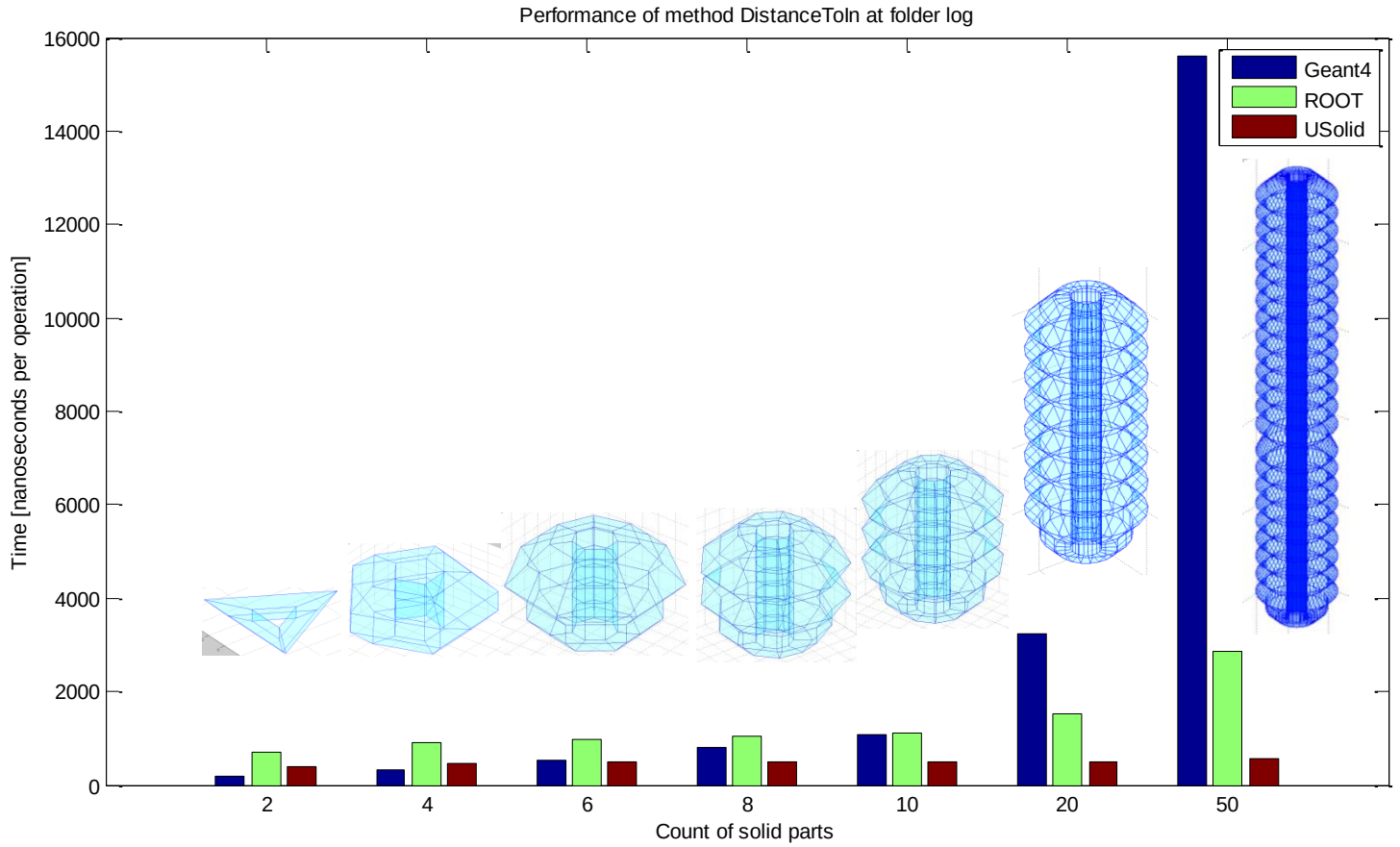
- Similar methodology adopted as for the Generic Polycone
  - Spatial optimisation for Z sections
  - Works for both ordinary and generic polyhedra
  - Provides improvement of performance and scalability
- Values are 100% numerical coincident with Geant4





Example: DistanceToIn() scalability

- Similar figures for the other key functions

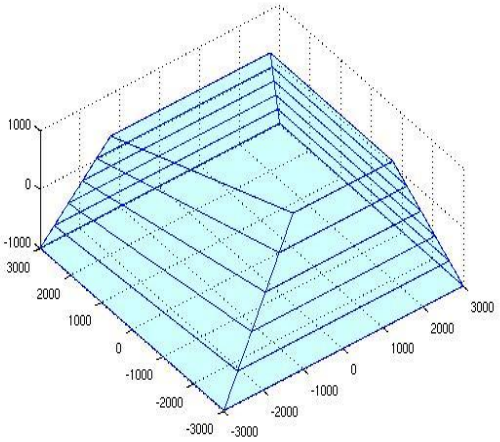


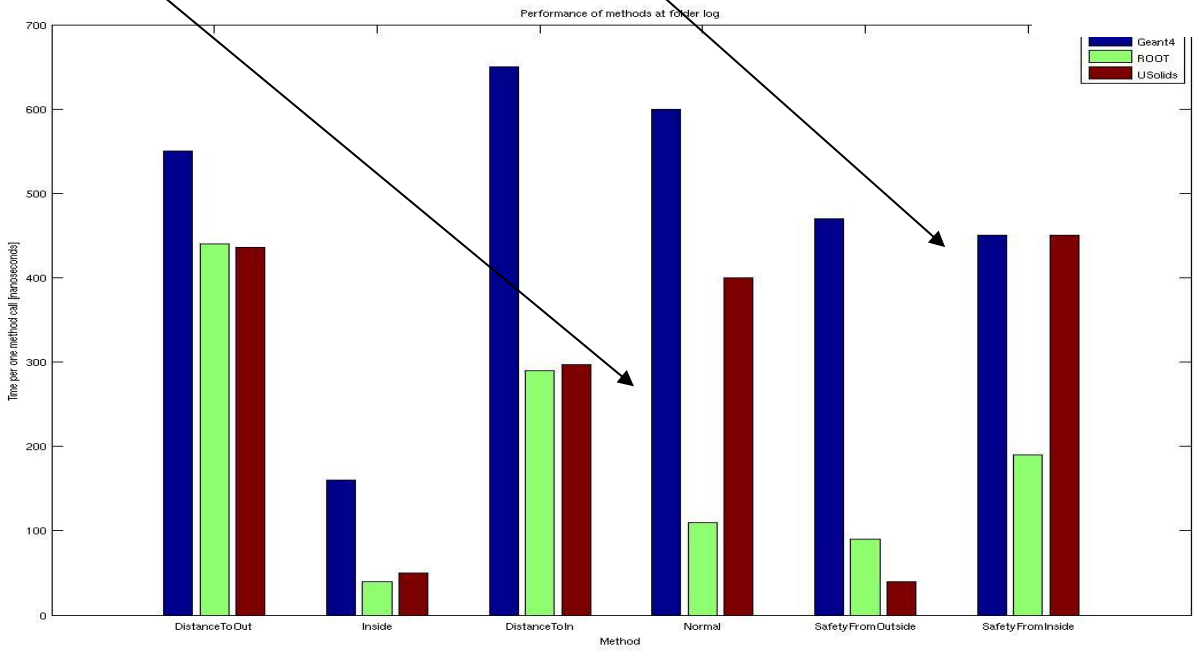
NOTE: *Geant4 poor performance scalability (high number of Z sections) due to absence of spatial optimization*

Arbitrary Trap

- Implemented first version of Arbitrary Trap
- Optimization is done for main methods
  - Still some methods to optimise:

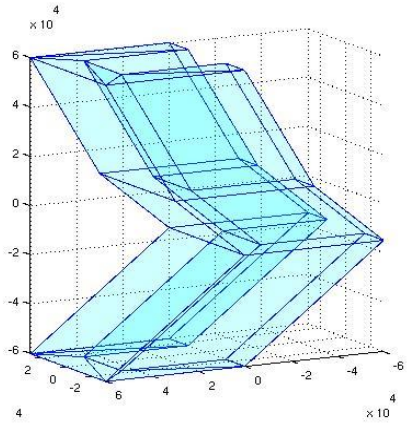
    SurfaceNormal(), SafetyFromInside()



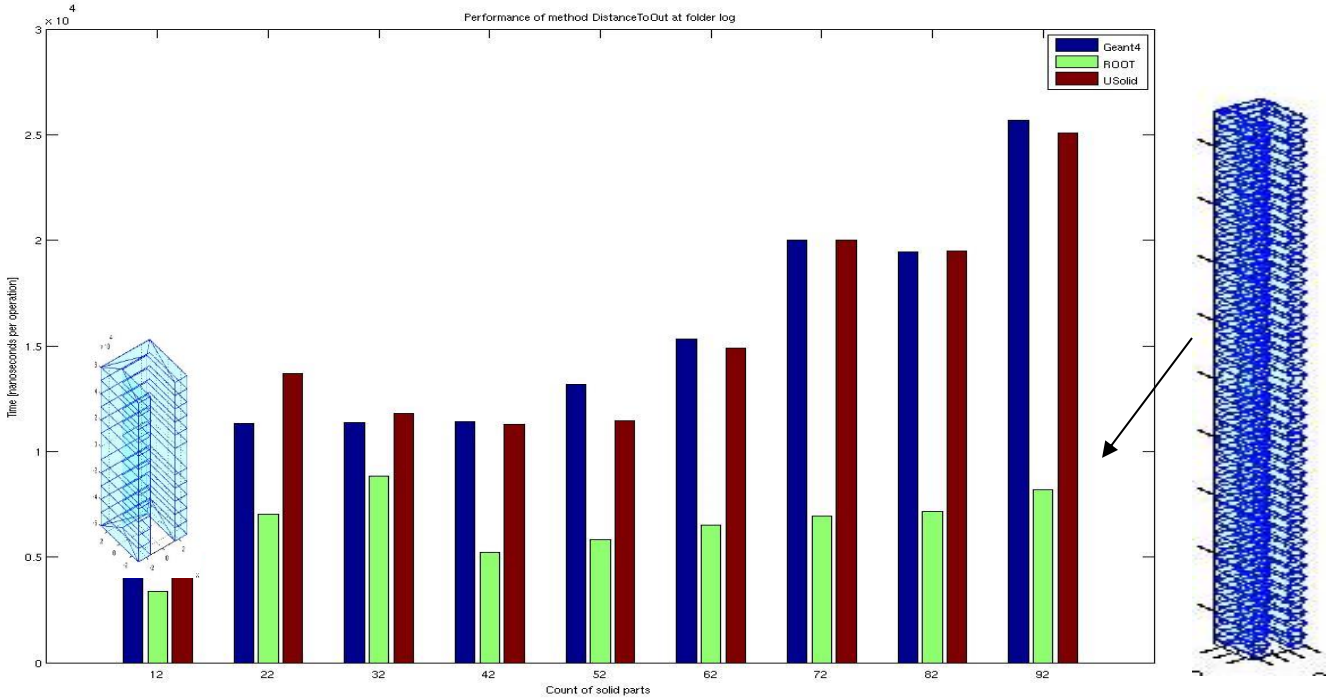


Performance for UGenericTrap with twist

Extruded Solid



- Implemented first version of ExtrudedSolid based on TessellatedSolid
- Scalability studies show that this shape can be more optimized. Work in progress…



Scalability for DistanceToOut() in UExtrudedSolid

Current Status & Plans

Status of USolids library

- Review of secondary functions and tools for all implemented primitives has been completed
- Library in the current form (excluding few primitives) distributed as optional module in latest **Geant4 release 10.0**
  - Expecting a lot of useful feedback !
  - Validation of available shapes on realistic detector geometries is ongoing
- Testing suite further extended for performance measurements and exact comparisons to Geant4 and ROOT
- **Code available in the AIDA SVN repository**
  - Using standard AIDA CMake setup for build/installation
- **Documentation available from AIDA WP2 web**
  - http://aidasoft.web.cern.ch/USolids

USolids plan of work

- Short term (by end 2014)
  - Complete implementation for missing shapes
    - Cut Tube, Torus, Ellipsoid, Hypeboloid, Paraboloid, Elliptical Tube, Elliptical Cone
  - Provide ability to perform Boolean operations
    - Simple Union, Subtraction, Intersection
- Medium/Long term (AIDA2 proposal)
  - Extend signatures of classes to enable use of vectorisation
    - SIMD instructions
  - Review algorithms on all developed shapes to efficiently apply vectorisation and strong code specialization
    - Define proper interfaces for use in Geant4, ROOT and Vector prototype

Thanks