Fast Tessellated Solid Navigation in GEANT4

Christopher M Poole, Iwan Cornelius, Jamie V Trapp, Christian M Langton

***Abstract*—Navigation through tessellated solids in GEANT4 can degrade computational performance, especially if the tessellated solid is large and is comprised of many facets. Redefining a tessellated solid as a mesh of tetrahedra is common in other computational techniques such as finite element analysis as computations need only consider local tetrahedrons rather than the tessellated solid as a whole. Here within we describe a technique that allows for automatic tetrahedral meshing of tessellated solids in GEANT4 and the subsequent loading of these meshes as assembly volumes; loading nested tessellated solids and tetrahedral meshes is also examined. As the technique makes the geometry suitable for automatic optimisation using smartvoxels, navigation through a simple tessellated volume has been found to be more than two orders of magnitude faster than that through the equivalent tessellated solid. Speed increases of more than two orders of magnitude were also observed for a more complex tessellated solid with voids and concavities. The technique was benchmarked for geometry load time, simulation run time and memory usage. Source code enabling the described functionality in GEANT4 has been made freely available on the Internet.**

***Index Terms*—navigation, tessellated solid, tetrahedron, GEANT4**

I. Introduction

Tessellated solids in Geometry and Tracking (GEANT4) [1], [2] offer a simple method for defining complex and irregular geometry, such as those created using computer aided design (CAD) [3], [4]. Navigation through these tessellated solids however, can be a performance bottleneck, especially if the tessellated solid is defined by many facets and a large number of steps take place within its boundary, see section III-A. Specifically, the bottle neck occurs when the GEANT4 navigator determines if the current step is inside or outside a tessellated solid.

Every step, at a boundary crossing or forced step governed by the setting of a maximum step length, the GEANT4 navigator determines the current step position relative to the user geometry [1], [2] - if the step is on the surface of a volume, or within a volume for example. For primitives such as cubes, spheres, tetrahedra and others, the determination of position within the volume is $O(1)$, that is to say the computation time required is constant and independent of the size of the volume described by the primitive. Being *primitive* is what gives rise to this property, the volume is well defined and has a specific and expected arrangement of facets. Consider a rectangular prism made up of exactly six faces and eight vertexes, where opposite faces are parallel and adjacent faces are perpendicular. Verifying if an arbitrary point is inside or outside of this prism is a trivial application of minimum and maximum bounds in the x, y, and z directions (assuming of course the prism has not been rotated). For a sphere, a similar operation may be applied, however the bound is applied in the radial direction.

Complex geometries may be too difficult to define with primitives alone; the tessellated solid was introduced to facilitate the definition of CAD derived surface meshes. Results presented in section III-A show that the time required to navigate a tessellated solid however is $O(n)$, linearly proportional to the number of facets that define the tessellated solid. Inspection of the GEANT4 source code implementing the process of inside determination for a tessellated solid shows that it is a multi-step process requiring iteration over each facet defining the solid. Firstly, a simple bounding box check is performed, if the step is outside of the bounding box, there is certainty the step is not within the tessellated solid and the function exits. Next, each facet is tested and compared to the current step position, if the distance to the facet is within a threshold, the step is considered to be on the volumes surface and the function exits. Finally, if the previous checks fail, 20 random rays are projected from the current step point. Each facet is again iterated over to test if a ray intersects it. The direction of the intersection for the first facet crossed by a ray is determined by examining the normal vector of the facet, for a normal vector of $0 < \theta < \pi/2$ the ray is exiting the solid or if the normal vector is $\pi/2 < \theta < \pi$ the ray is entering the solid. Note that each facet for the tessellated solid must be defined with its vertexes ordered in an anti-clockwise direction such that the normal for the face is pointing to the inside of the solid. This algorithm is described in more detail elsewhere [5], and it is noted that the algorithm is very robust when considering tessellated solids with voids and concavities compared to other techniques which are generally optimised to consider tessellated solids without concavities only.

Geometry optimisation techniques already present within GEANT4 such as smart-voxelisation and geometry parameterisation tend not to be effective when applied to tessellated solids. Smart-voxelisation is an automatic partitioning of the geometry where groupings of nearby and smaller volumes are assigned to a common local mother volume, referred to as a smartvoxel [6]. Geometry parameterisation is a technique whereby regular and repeating geometries such as voxelised data can be represented as multiple copies or replicas of the same initial volume. Properties assigned to each copy, such as material and position can be described as a function of copy or replica number without the need to initialise a new volume [7], [8]. As an individual instance of a tessellated solid is not a collection of smaller individual or repeating solids,

This work was supported in part by the Queensland Cancer Physics Collaborative, and Cancer Australia (Department of Health and Ageing) Research Grant 614217.

C. M. Poole is with Cancer Care Services, Royal Brisbane and Womens Hospital, Herston, QLD 4029, Australia.

C. M. Poole, I. Cornelius, J. V. Trapp and C. M. Langton are with the Discipline of Physics, Faculty of Science and Technology, and the Institute of Health and Biomedical Innovation, Queensland University of Technology, Brisbane, QLD 4000, Australia. (email: christopher.poole@qut.edu.au)

the navigator cannot take advantage of these optimisation techniques. An alternative description of a tessellated solid that made available these techniques for geometry optimisation would provide an immediate performance improvement.

Computational techniques such as finite element analysis (FEA) redefine complex closed tessellated surfaces as tetrahedral meshes where each mesh element is a simple tetrahedron [9]. Redefining a tessellated surface in this way allows for computations to be performed considering local tetrahedra only, instead of the entire tessellated volume [9]. As is the case with other computational techniques, navigation through a tetrahedron in GEANT4 is very fast as the geometric properties of the solid are well defined. Herewith we do not attempt to optimise the aforementioned G4TessellatedSolid methods that result in arrested navigation; on the contrary, we offer an alternative geometry definition that allows for discrete tetrahedrons to define the same tessellated solid thereby making available smart-voxelisation for automatic geometry optimisation. The technique is evaluated for individual and nested tessellated solids represented as tetrahedral meshes in GEANT4.

II. Methods

A. Tetrahedral Meshing

For ease of integration with GEANT4, the freely available C++ quality tetrahedral meshing generator, TETGEN was chosen for this work The generator may be compiled as a standalone application or in this case, as a shared object library for linking with a C++ user application [10]. Many configuration parameters are available to the user when creating a tetrahedral mesh using TETGEN; for example, refinement of pre-existing meshes can be performed, the quality of the generated meshes may be finely controlled and tetrahedron volume constraints may be applied - a comprehensive description of functionality provided by TETGEN is described by others [10]. Using the TETGEN library, two `tetgenio` (the TETGEN input/output object) meshes, `input` and `output`, were initialised. The input mesh was populated with a tessellated triangular facet surface mesh described in a stereo lithography (STL) file using the `tetgenio::load_stl` method with the mesh file name as an argument. The output mesh was then populated with a constrained Delaunay tetrahedralisation of the input mesh using the TETGEN `tetrahedralize` function with a configuration string and the input and output `tetgenio` objects parsed by reference as arguments. Configuration for the constrained Delaunay tetrahedralisation (enabled with the `p` flag) was such that boundary facet splitting was suppressed (by setting the `Y` flag), thereby preserving the mesh surface described by the input mesh. No mesh quality was specified (ordinarily set with the `q`n flag, where n is an arbitrary value specifying mesh quality), resulting in an output mesh with a minimum of tetrahedra.

Access to individual tetrahedra in the output mesh was available via the `output.tetrahedronlist` vector where every four elements indicated the vertex index numbers of the current tetrahedron. The coordinates of each vertex could then be retrieved from the `out.pointlist` vector. Iterating over the `output.tetrahedronlist` vector, a `G4Tet` solid was initialised along with a `G4LogicalVolume` for each tetrahedron, at which point material properties were also assigned. Subsequently each logical volume was added to a `G4Assembly Volume` allowing for all tetrahedra to be positioned (including translation, rotation and reflection operations) within the user detector geometry as a single entity using the `G4AssemblyVolume::MakeImprint` class method. Code listing 1 shows example usage for loading the same object described in a CAD file as a tessellated solid or as a tetrahedral mesh using the `cadmesh` CAD interface for GEANT4.

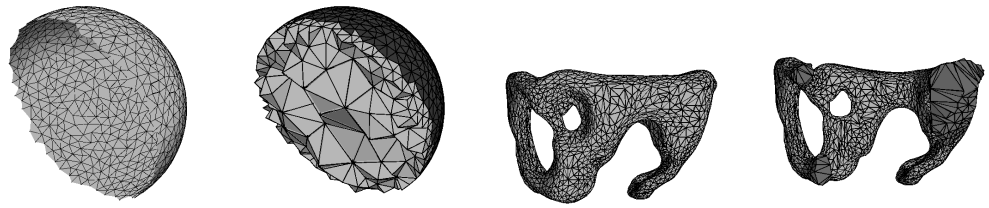
Listing 1: Basic usage of tessellated solid tetrahedralisation in a user detector constructor

```
#include "CADMesh.hh"
...
CADMesh * mesh = new CADMesh("sphere.stl",
                             "STL");

// Tessellated Mesh
G4TessellatedSolid * solid =
    mesh->TessellatedMesh();

// OR Tetrahedral Mesh
G4Material * material;
G4AssemblyVolume * assembly =
    mesh->TetrahedralMesh(material);

// Tetrahedral Mesh Placement
G4Transform3D transform;
assembly->MakeImprint(mother_logical,
                      transform, 0, 0);
```

B. Simulation Set-up & Geometry Definition

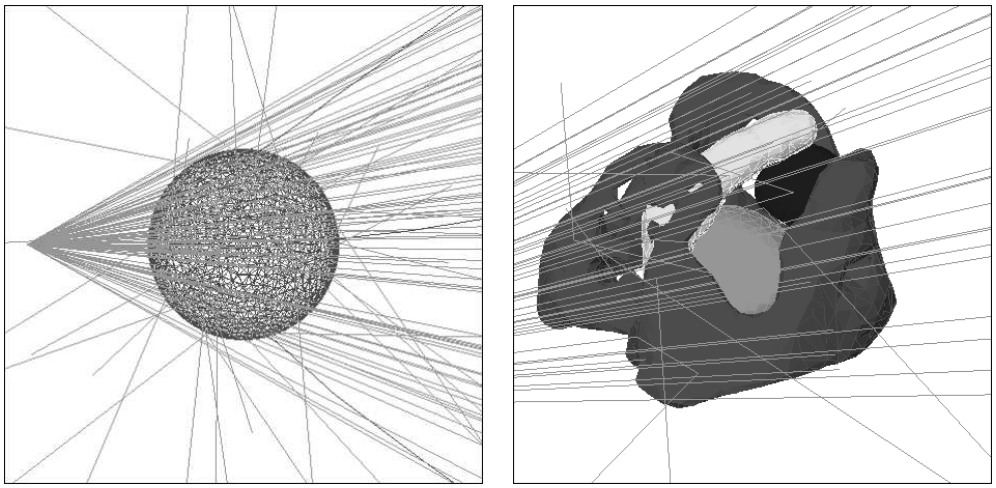
Using the freely available mesh manipulation program MeshLab [11], a sphere of $400\ mm$ diameter was created with its surface defined by 45,000 tessellated triangular facets. Marching cube mesh refinement in MeshLab was then used to reduce the number of facets defining the sphere in steps of 2,500 to a lower facet count of 2,500, yielding a set of meshes all defining the same sphere over a range of facet counts, see figures 1(a) and 1(b) for an example surface mesh with corresponding tetrahedralisation. A more complex mesh, that of a model pelvis (CIRS Multi-modality Pelvic Phantom, model 048) was also created, again using marching cube mesh refinement to create a range of meshes with facet counts between 5,000 and 35,000 facets, see figures 1(c) and 1(d).

A simple GEANT4 detector geometry was created with an air-filled (G4_AIR) cube of $1\ m$ edge length initialised as the world volume; the test geometry material was set to water (G4_WATER) and positioned at the center of the world volume. Standard electromagnetic physics were used throughout by registering the `G4EmStandardPhysics` module in the user physics list; range cuts were set to $1\ mm$. A general particle source was positioned $5\ cm$ inside the world volume and aimed at the test geometry, see figure 2(a) and (b); the beam divergence angle was set to 30° thereby ensuring full beam coverage of the test geometry.



(a) surface mesh (b) tetrahedral mesh (c) surface (d) cut-away

Fig. 1: Tetrahedral meshing; (a) shows a cut-away of the triangular tessellated surface of a sphere and (b) shows its corresponding tetrahedralisation. The tessellated surface of a model pelvis is shown in (c) with its corresponding tetrahedralisation cut-away in (d).



(a) (b)

Fig. 2: The general simulation set-up where the outer box defines the edge of the mother world volume with the test geometry in the center. Test particles are shown fired from the general particle source from left to right in (a).

C. Computational Performance

Each of the above test geometries were first loaded as native GEANT4 tessellated solids using the `cadmesh` GEANT4 CAD interface, also developed by the authors of the present work [12]. These tessellated solid geometry simulations were used as controls for the equivalent tetrahedral mesh simulations. Smart-voxelisation optimisation was automatically disabled by default in GEANT4, as the world mother volume contained only one daughter volume - the `G4TessellatedSolid` test geometry. Subsequently, the test geometries were loaded as tetrahedral meshes using the method described in section II-A and optimised for a range of smart-voxelisation values between 0.2 and 2.0. For all simulation configurations, the geometry was bombarded with 10^5 `G4Gamma` particles and repeated 10 times; a number of parameters were recorded in order to measure computational performance including; geometry load time, simulation beam on time and smart voxel memory usage. These parameters were then evaluated as a function of source tessellated volume facet count.

D. Geometry Equivalence

Testing was performed so as to ensure the tetrahedral meshes defined the same geometry as the original tessellated solids. Firstly, qualitative visual inspection of the geometry was carried out using the GEANT4 OpenGL viewer, checking that tetrahedral geometry was correctly located within its mother volume. Geometry overlap was tested using the GEANT4 `/geometry/test/` user interface commands, ensuring adjacent tetrahedra were in fact adjacent and not overlapping as this could result in navigation error. The maximum extents of the tetrahedral geometry were also calculated and compared to that of the source tessellated solid. Further, the coordinates of all tetrahedra vertices, as loaded in GEANT4, were acquired using the `G4Tet::DumpInfo` class method and compared to the coordinates of the vertices in the source tessellated solid where missing vertices indicated an invalid geometry definition. Finally, a test of missing tetrahedra was devised; the tetrahedral mesh was loaded and geantinos (the GEANT4 debugging pseudo-particle) were fired into the geometry from three orthogonal directions; tracking verbosity was set to one so as to output the unique name of each solid navigated through as the geantinos traversed the user detector geometry. A search through this output was performed so as to

verify that all tetrahedra created during the tetrahedralisation of the source tessellated solid were navigable and included in the GEANT4 user detector geometry as `G4Tet` volumes.

E. Nesting Tessellated Solids

For an assembly of nested tessellated volumes it is possible to generate an equivalent tetrahedral mesh using TETGEN, where individual tetrahedra are labelled depending on which nested volume they belong to (by setting the `A` flag in the tetrahedralisation configuration). Regions of interest defining several geometric features found in the CIRS pelvic phantom described in section II-B, namely the pelvis, prostate, bladder, rectum and body were extracted from a DICOM CT dataset and the constituent vertexes and facets appended to `.node` and `.ele` mesh files respectively. Unique region labels were assigned to each volume thereby ensuring volume boundaries were preserved during tetrahedralisation. Average material properties, calculated statistically from the source CT dataset were assigned to each volume. Iteratively adding the tetrahedra defining the combined mesh as in the same manner described in II-A and assigning material properties based on the unique volume label, the equivalent tetrahedral mesh of a collection of nested tessellated volumes was loaded into GEANT4.

The geometric equivalence of the tessellated and tetrahedral geometries was evaluated in the same manner as described previously and computation time and memory consumption was measured for two parallel and opposed $5 \times 5\ cm$ fields aimed towards the prostate from either side of the phantom using a phasespace file from a pre-existing clinical linear accelerator geometry. Using a `G4SensitiveDetector` configured to score radiation dose in both the tessellated and tetrahedral geometries, the dose distribution of the two parallel and opposed fields was calculated and compared using gamma evaluation [13] with a pass/fail criterion of $1\ \%$ and $2\ mm$ scored on a $2\ mm$ dose grid. Gamma evaluation is a statistical technique used to quantitatively compare two datasets. The current point in the sample dataset is directly compared to the corresponding point in the reference dataset and points that surround it. Distance penalties are applied such that a matching point close to the current point is more heavily weighted compared to one that is further away. The resulting gamma value reports the agreement between the datasets according to the specified pass/fail or distance to agreement criterion with a gamma value $\gamma \leq 1$ indicating a pass and a gamma value $\gamma > 1$ indicating a fail.

III. Results

A. Computational Performance

Figure 3(a) show the geometry load time for both tessellated and tetrahedral mesh geometries. Specifically, the tessellated geometry load time represents the time required to add all facets to the `G4TessellatedSolid` and position it within the user detector geometry. For the tetrahedral mesh geometry, the load time represents the time required to initialise one `G4Tet` for every tetrahedron in the mesh, add it to a `G4AssemblyVolume` and make an imprint of this same assembly volume in the user detector geometry. Note that the time required to load a tetrahedral mesh as geometry is more than two times that of the time required to load the equivalent tessellated surface mesh; in both cases, load time increases with the square of the number of facets in the source tessellated volume as the operation requires the initialisation of a TETGEN object and a GEANT4 tessellated solid or assembly volume, operations all of which are $O(n)$. Additionally, figure 3(b) shows tetrahedron count increasing linearly with increasing source tessellated solid facet count at a rate of 1.60 ± 0.3 tetrahedra per facet. Measured data points in all cases represent the mean value for ten samples and error bars indicate two standard deviations about this mean.

Figures 4(a) and 4(b) show that the smartvoxel memory consumption increases as a higher smart-voxelisation parameter is specified, as does the time required to perform the voxelisation. Further, as the facet count of the source tessellated solid increases, hence an increasing tetrahedron count, again the smartvoxel memory consumption increases along with the time required to perform the smart-voxelisation. Smart voxel memory consumption was found to be proportional to the smartvoxel count with smartvoxel memory consumption and smart-voxelisation time both proportional to the square root of tessellated solid facet count. Data points at 7,500 and 15,000 facets were excluded from analysis due to the exceptionally large number of smart voxels generated during the optimisation of these geometries; these artefacts are examined later.

For increasing facet count, figures 5(a) and 5(b) show that simulation run time increases linearly with $R^2 \geq 0.95$ for both a tessellated solid and its equivalent tetrahedral geometry as a function of source tessellated solid facet count. Spheres described as a tessellated solid with more than 10,000 facets showed speed-ups of more than two orders of magnitude. As with previous analysis, data points at 7,500 and 15,000 facets were excluded.

Figures 4(a), 4(b) and 5(b) display artefacts due to higher numbers of smartvoxels generated at 7,500 and 15,000 facets when compared to the underlying trend, noting that the number of tetrahedral generated for these meshes follows the expected linear trend in figure 3(b). Inspection of the these tetrahedral meshes in figures 6(a) and 6(b) show a characteristic tetrahedron arrangement; specifically the majority of tetrahedra all share a common vertex, this is particularly evident when compared to the typical tetrahedron arrangement shown in 6(c) for a source tessellated solid of 17,500 facets.

B. Complex Geometry

For the complex mesh geometry show in figures 1(c) and 1(d) the tetrahedra generated per facet in the source tessellated solid was found to be 1.77 ± 0.05 tetrahedra per facet, higher than that found for the spherical test geometry. Tetrahedral geometry load time was again found be to more than twice that of the native `G4TessellatedSolid` load time. Simulation time for the tetrahedral geometry was over two orders of magnitude faster than the equivalent tessellated solid geometry for meshes with over 20,000 facets. As with the spherical geometry, smartvoxel memory usage increased with increasing facet count and smart-voxelisation value.

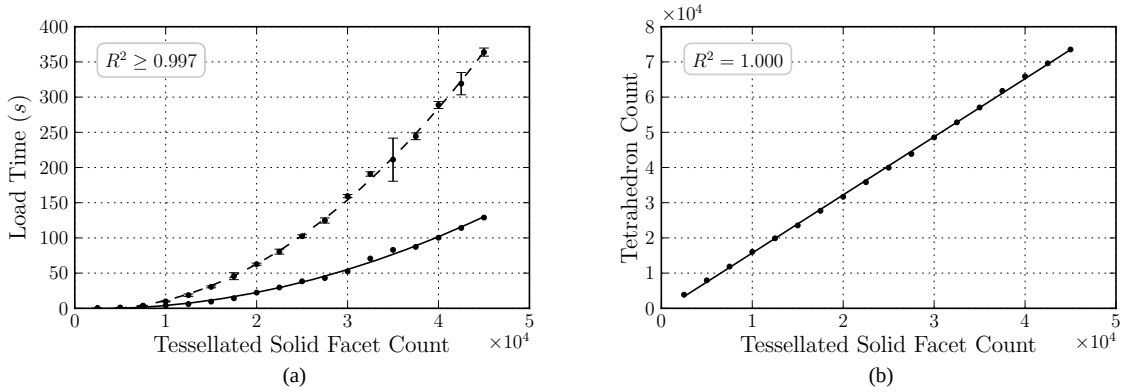


Fig. 3: Geometry load time is shown in (a) where dashes indicate a second order fit to tetrahedral geometry load time with corresponding measured data and the solid line indicates a second order fit to tessellated geometry load time. Plot (b) shows tetrahedron count versus tessellated solid facet count with a linear fit.

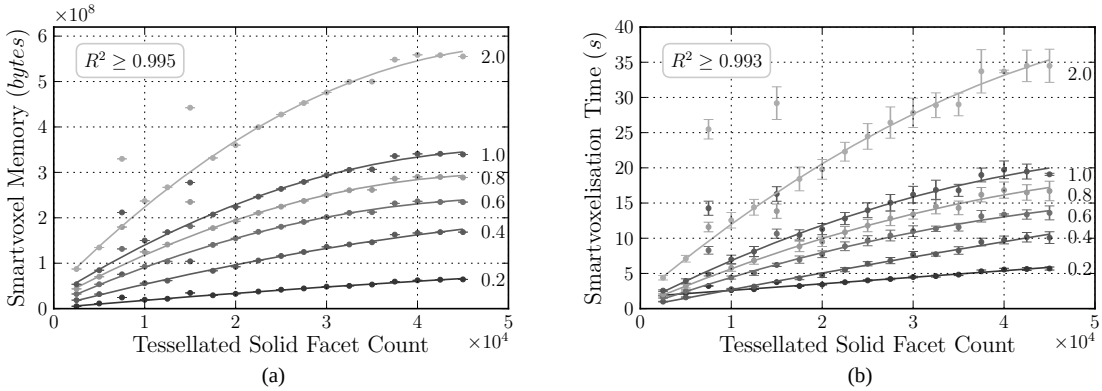


Fig. 4: Smartvoxel memory consumption is shown in (a) where solid lines indicate a second order fit to the measured data. The time required to perform the smartvoxel geometry optimisation is shown in (b). Data labels indicate the smartvoxel value for the dataset.

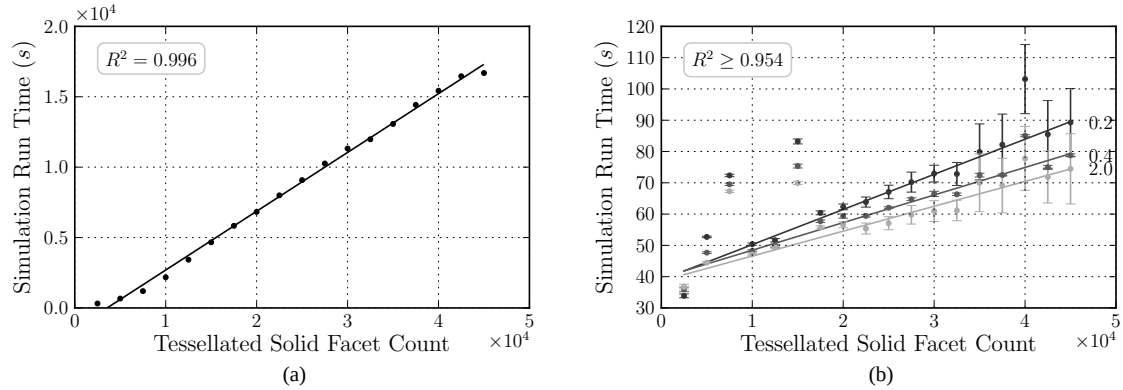
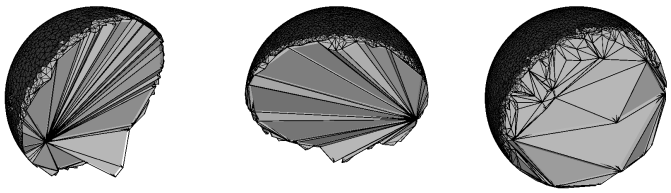


Fig. 5: Simulation run time for 100,000 initial histories and a tessellated geometry is shown in (a), and (b) shows the simulation time for the equivalent tetrahedral geometry for selected smart-voxelisation value (others omitted for clarity). Data labels in (b) indicate the smartvoxel value used.



(a) 7,500 facets (b) 15,000 facets (c) 17,500 facets

Fig. 6: Meshing artefacts in (a) and (b) that result in suboptimal smart-voxelisation compared to a typical tetrahedralisation in (c).

C. Geometry Equivalence

All tetrahedral meshes were found to be geometrically equivalent in GEANT4 to the source tessellated mesh from which they were derived. Using the inbuilt geometry overlap testing in GEANT4, no overlapping tetrahedra were found in any geometry configuration. Direct comparison of the tetrahedra vertex coordinates to the coordinates of the facet vertexes in the source tessellated meshes showed no difference in the boundary definition of the meshes. Further, firing geantinos into the test geometry from three orthogonal directions showed that all tetrahedra defined in the tetrahedralisation of the source tessellated mesh were correctly loaded as native `G4Tet` tetrahedrons in GEANT4.

D. Nesting Tessellated Solids

Figures 7(a), (b) and (c), show the progression of nested surfaces to a nested tetrahedral mesh. Using the same methods as described previously, it was found that the volume boundaries defining the surface models were exactly reproduced in the tetrahedral mesh. Isodose contours are shown in figure 8(a) for the tessellated geometry representation of the CIRS pelvic phantom. Using gamma evaluation, the comparison between these dose distributions is shown in figure 8(b). For a $2\ mm$ dose scoring grid and a pass fail criterion of 1% and $2\ mm$, 100% of all points within the phantom pass the gamma evaluation when comparing the tessellated and tetrahedral geometries. That is to say no voxel in one grid when compared to the corresponding voxel in the other grid had a dose that was different by greater than 1%. The tetrahedral geometry representation required one fifth the simulation time required for the equivalent tessellated geometry.

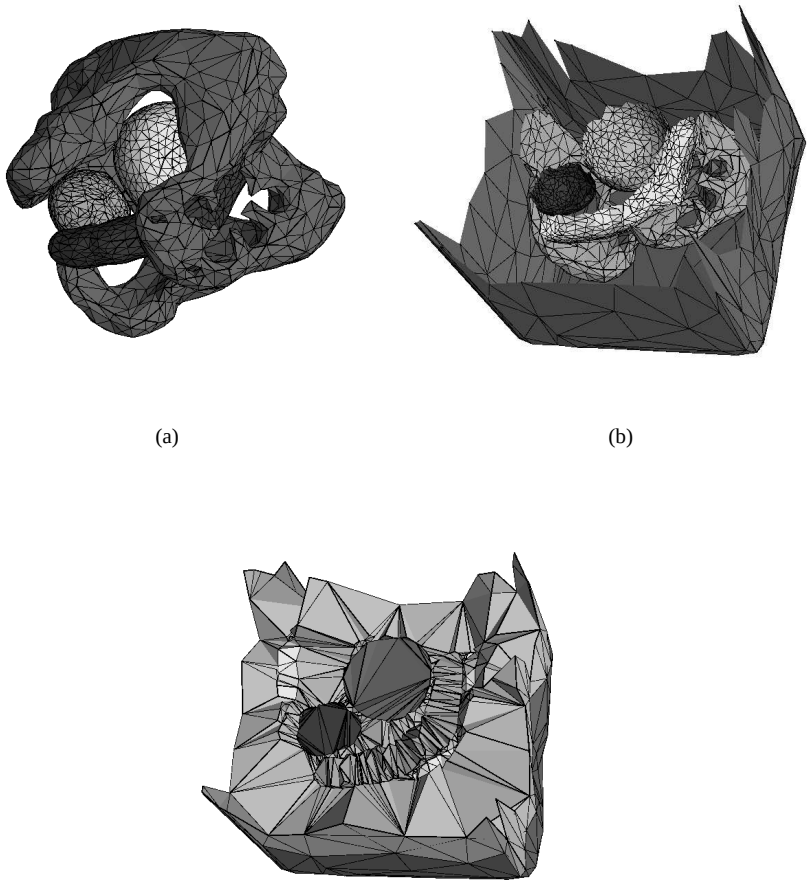
IV. Discussion & Conclusion

We have demonstrated fast navigation of tessellated volumes using tetrahedral meshes as an alternative to native `G4TessellatedSolid`'s in GEANT4. Using the C++ library TETGEN, automatic tetrahedral meshing of the input tessellated solid can be performed and the resultant tetrahedral mesh placed within the user geometry as an assembly volume. For simple tessellated solids with more than 10,000 facets, navigation has been found to be over two orders of magnitude faster when the solid is loaded as a tetrahedral mesh. Navigation through the complex pelvis tessellated solids, with more than 20,000 facets was also found to be over two orders of magnitude faster when loaded as a tetrahedral mesh. Speedup when considering nested geometries is diminished due to a higher tetrahedron to tessellated facet ratio, however there is still a significant speedup when using the tetrahedral equivalent geometry compared to the source tessellated geometry. Whilst the load time for a tetrahedral mesh can be over twice the load time for the equivalent tessellated solid, this can be considered insignificant when the load time may be on the order of minutes, and simulation time on the order of hours, as is the case in this study.

Low quality tetrahedral meshes where many tetrahedra share a common vertex may result in sub-optimal smart-voxelisation. Using TETGEN, higher quality meshes may be generated in order to avoid this vertex sharing artefact, however quality meshing will result in more tetrahedra being created, which in turn will result in higher smartvoxel memory consumption. Specific tetrahedra arrangement is very much dependant on the volume defined by the source tessellated solid, though limited user supervision should be sufficient to capture poor quality meshing and evaluate its significance on the performance of a simulation.

Memory consumption for geometries exploiting smart-voxelisation must be closely managed, especially if the user geometry contains many volumes, as is the case for tetrahedral meshes which may contain tens of thousands of tetrahedra. Tuning the degree of smart-voxelisation allows for basic control of geometry memory consumption and simulation run time. Evaluating the results shown in figures 4(a) and 5(b) however, shows that increasing the aggressiveness of the smartvoxel optimisation results in small simulation time reductions whilst increasing memory usage from tens of megabytes to hundreds of megabytes. This increase in memory usage may result in suboptimal computation performance in an environment where multiple simulation instances are running on the same computer and all of the available memory is consumed by the simulation. In this case, reducing the smart-voxelisation parameter will reduce memory consumption substantially whist effecting only a small increase on simulation run time.

By remeshing tessellated solids as tetrahedral meshes, smart-voxelisation is made available for geometry optimisation. This allows for the GEANT4 navigator to consider only local tetrahedra rather than the tessellated solid as a whole at



(a) (b)

(c)

Fig. 7: For the nested tessellated solids in (a) for a phantom pelvis (medium gray), bladder (light gray), rectum (dark gray) and prostate (gray), the equivalent tetrahedral mesh is shown in (c). A cut-away for all source tessellated solids including the mother body volume is shown in (b). Colours are arbitrary and simply indicate separate solids.

each step. Consequently, the computationally intensive inside determination for tessellated solids need not be performed during navigation. Using this technique, a significant reduction in simulation run time is observed for user geometry containing tessellated solids defined by many facets.

REFERENCES

[1] S. Agostinelli, J. Allison, K. Amako, J. Apostolakis, H. Araujo, P. Arce, M. Asai, D. Axen, S. Banerjee, G. Barrand *et al.*, "Geant4 simulation toolkit," *Nuclear Instruments and Methods in Physics Research-Section A Only*, vol. 506, no. 3, pp. 250–303, 2003.

[2] J. Allison, K. Amako, J. Apostolakis, H. Araujo, P. Dubois, M. Asai, G. Barrand, R. Capra, S. Chauvie, R. Chytracek *et al.*, "Geant4 developments and applications," *IEEE Trans. Nucl. Sci.*, vol. 53, no. 1, pp. 270–278, 2006.

[3] M. Constantin, D. E. Constantin, P. J. Keall, A. Narula, M. Svatos, and J. Perl, "Linking computer-aided design (CAD) to Geant4-based Monte Carlo simulations for precise implementation of complex treatment head geometries," *Physics in Medicine and Biology*, vol. 55, p. N211, 2010.

[4] R. Chytracek, J. McCormick, W. Pokorski, and G. Santin, "Geometry description markup language for physics simulation and analysis applications," *Nuclear Science, IEEE Transactions on*, vol. 53, no. 5, pp. 2892–2896, 2006.

[5] P. Schneider and D. Eberly, *Geometric tools for computer graphics*. Morgan Kaufmann Pub, 2003.

[6] G. Cosmo, "The geant4 geometry modeler," in *Nuclear Science Symposium Conference Record, 2004 IEEE*, vol. 4. IEEE, 2004, pp. 2196–2198.

[7] T. Aso, A. Kimura, T. Yamashita, and T. Sasaki, "Optimization of patient geometry based on ct data in geant4 for medical application," in *Nuclear Science Symposium Conference Record, 2007. NSS'07. IEEE*, vol. 4. IEEE, 2007, pp. 2576–2580.

[8] P. Arce, J. Apostolakis, and G. Cosmo, "A technique for optimised navigation in regular geometries," in *Nuclear Science Symposium Conference Record, 2008. NSS'08. IEEE*. IEEE, pp. 857–859.

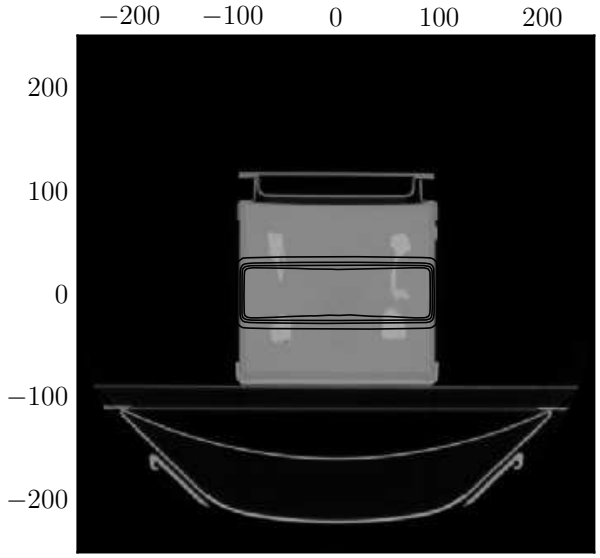
[9] O. Zienkiewicz and R. Taylor, *The finite element method for solid and structural mechanics*. Butterworth-Heinemann, 2005, vol. 2.

[10] "Tetgen: A Quality Tetrahedral Mesh Generator and a 3D Delaunay Triangulator," Computer software. http://tetgen.berlios.de/, Retrieved April 1, 2012.

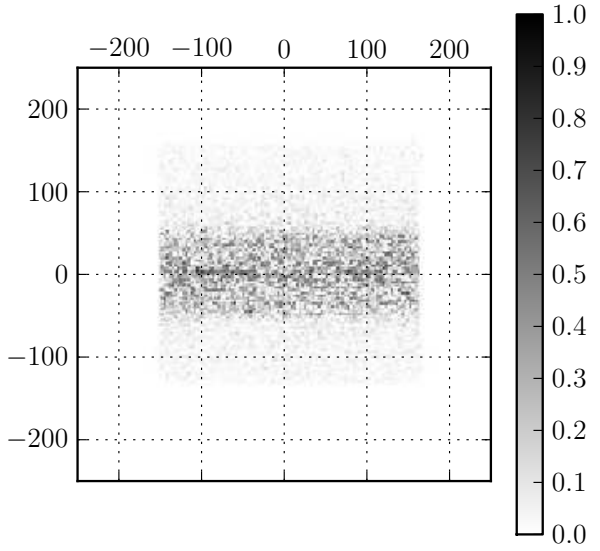
[11] "MeshLab: An open source, portable, and extensible system for the processing and editing of unstructured 3D triangular meshes," Computer software. http://meshlab.sourceforge.net/, Retrieved April 1, 2012.

[12] "CADMESH: A CAD interface for GEANT4," Computer software. http://code.google.com/p/cadmesh/, Retrieved April 1, 2012.

[13] M. Wendling, L. Zijp, L. McDermott, E. Smit, J. Sonke, B. Mijnheer, and M. van Herk, "A fast algorithm for gamma evaluation in 3D," *Medical physics*, vol. 34, p. 1647, 2007.



(a)



(b)

Fig. 8: Isodose contours for the tessellated geometry are shown in (a) with the contours representing 20, 60, 80, 95% of the mean dose at the iso-centre. The gamma evaluation comparing the tessellated and tetrahedral geometries in shown in (b) with a pass fail criterion of 1% and $2\ mm$.