The Monte Carlo Method: Versatility Unbounded In A Dynamic Computing World
Chattanooga, Tennessee, April 17–21, 2005, on CD-ROM, American Nuclear Society, LaGrange Park, IL (2005)

METHODS OF INTEGRATING COMPUTER AIDED DESIGN FOR MONTE CARLO SIMULAIONS

Peter F. Caracappa, X. George Xu
Nuclear Engineering and Engineering Physics
Rensselaer Polytechnic Institute
Troy, NY 12180, USA
caracp3@rpi.edu

ABSTRACT

Monte Carlo codes such as MCNP, EGS or GEANT4 are frequently used as a simulation tool for a wide variety of applications, including reactor design, shielding analysis, and radiation treatment. A significant obstacle to its use is the construction and visualization of the target geometry within the code. Many of these applications could benefit from the ability to share geometry data between computer aided design (CAD) and Monte Carlo radiation transport codes. We have analyzed three alternate strategies for translating CAD data into Monte Carlo geometry. An algorithm has been developed to convert STEP format CAD files into MCNP input, but is currently limited to solid geometry constructs. Other available CAD output formats can approximate object surfaces with planar segments, which can be incorporated into Monte Carlo input. Voxels can be generated from a series of planar intersections with a three-dimensional object. These two strategies can be applied to objects of arbitrary shape, but require a tradeoff between geometric precision and computing time. Because of limitations in computing power, there is not a single optimal solution to data sharing. Although choice of data handling strategy is likely to remain application-dependent for some time to come, the tools described can help in maintaining harmony between Monte Carlo simulations and other design groups.

Key Words: Computer aided design, geometry definition, STEP standard

1 INTRODUCTION

The leading method for studying radiation transport and interaction continues to be the use of Monte Carlo simulations. The applications include validation of nuclear power plant operations, shielding design, accelerators, radiation treatment planning, dosimetry, and for many others. Several well-validated codes are commonly used for these purposes. Each one of these codes, however, requires the geometry be provided in a unique manner, without providing robust or user-friendly tools.

Currently, the geometry design of a target and the specification in a Monte Carlo code are separate. One often has to "manually" translate the shape of a target into the Monte Carlo code. Simple geometry definition routines are acceptable for transport problems consisting of a handful of basic shapes. But as the model description becomes increasingly complex, the geometry input process becomes significantly more tedious. Further, the increasing complexity of the geometry results in increased opportunities for geometric errors, such as voids, collisions, over-definition and under-definition of entities. Several tools have been developed to provide for the visualization and validation of geometry inputs for different Monte Carlo transport codes [1-4].

However, the definitions of shapes in these methods are not standardized and are often difficult to use.

An added difficulty occurs when radiation transport simulation is desired for a target or facility design that has not yet reached a final form. Minor design modifications can lead to significant changes in the input to radiation transport codes. A system to allow the sharing of design information between designers utilizing Computer Aided Design (CAD) packages, and those performing radiation transport calculations, utilizing Monte Carlo particle transport codes, could have significant advantages. This paper reviews the current methods in combining the CAD with the Monte Carlo methods and proposes a new method that is potentially useful.

2 METHOD AND MATERIALS

2.1 Monte Carlo Transport Codes

In relation to the discussion of geometry, the most important part of the transport process is when the code checks to see if a particle step has intersected a boundary. The transport code accepts the geometry in such a way that it can find these intersections. Calculating the intersection between the particle vector and the boundary may use a great deal of computational time.

A large number of computer codes exist to perform Monte Carlo transport modeling. Often, a specialized code will be developed for a limited set of situations, and the required geometry constructs can be designed accordingly. However, three general-purpose codes are addressed, which are used across many different applications for the purpose of modeling radiation transport and effects.

MCNP [1] and MCNPX [2] are broadly used, general-purpose Monte Carlo codes. MCNP can model the transport of neutrons, photons, and electrons, while MCNPX is used for charged particle transport. Of the common general-purpose codes, the geometry input for MCNP is the most restrictive. Geometry input routines, or "cards," are available for constructs such as planes, prisms, conics, and torroids. Boolean combinations of these constructs can be used to create complex objects, but there are no user-definable geometry routines.

EGS4 [3], or the Electron-Gamma-Shower code, is limited to the modeling of electron and photon transport, but is often used in applications where other types of interactions are irrelevant, such as most medical applications. In EGS, a user code is written in FORTRAN or MORTRAN (macro-enhanced FORTRAN) in which the geometry is specified. In particular, the geometry definition is part of the subroutine HOWFAR, which informs the transport kernel as to the distance to the nearest boundary for the given conditions. Since the user creates this code, there is greater flexibility than in the MCNP cards. However, the FORTRAN encoding presents a significant obstacle to the geometry input.

The GEANT4 [4] code is largely used for Monte Carlo simulation problems involving high-energy particle physics. GEANT4 is written in C++ with an object-oriented architecture, which takes advantage of the extensibility of the language. Sophisticated programming knowledge is necessary to take advantage of the possibilities of the code, but GEANT4 offers greatest opportunities in defining geometries.

Most geometric objects, particularly the simpler shapes, can be represented in a variety of different ways. Three major classifications of geometry descriptions are discussed below.

2.2 Constructive Solid Geometry (CSG)

Constructive solid geometry, or CSG (sometimes also referred to as combinatorial geometry, or CG) is generally the simplest method of defining a geometric shape [8]. A CSG-defined object consists of regions defined as the intersection of two or more algebraic half-spaces, and may include Boolean combinations of those regions. Codes that are used to generate CSG objects generally have developed "shortcuts" for the input of commonly used solid objects, although they are defined most basically as half-space intersections. For instance, a box is the intersection of six planar-bounded half-spaces, and a right circular cylinder of a given length is an infinite cylinder bounded by two planes.

Figure 1 shows the construction of a cubic box, centered at the origin with an edge length of $2d$, with a circular hole through the y-axis of radius r. The cube is the intersection of the six half-spaces defined by the equations:

$$x \leq d, \tag{1}$$

$$x \geq -d, \tag{2}$$

$$y \leq d, \tag{3}$$

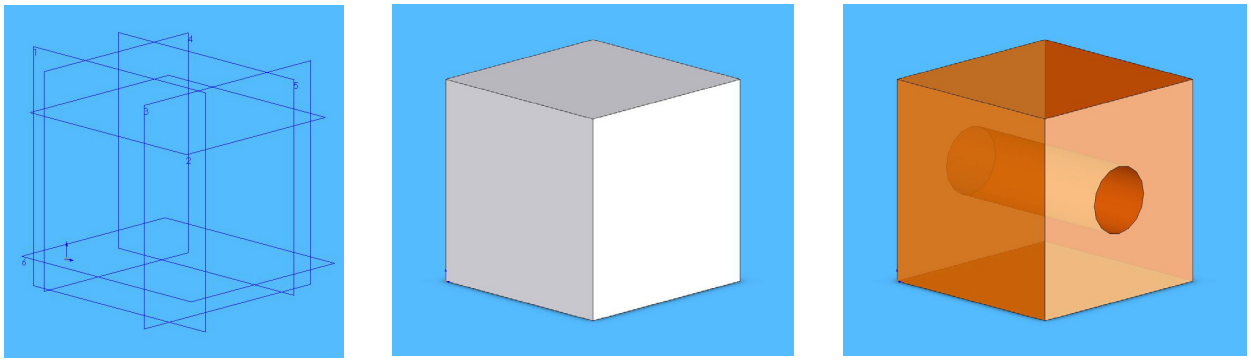
$$y \geq -d, \tag{4}$$

$$z \leq d, \tag{5}$$

$$z \geq -d. \tag{6}$$

Subtracted from the cubic construct defined by those planes is the infinite right circular cylinder defined by:

$$x^2 + z^2 \leq r^2. \tag{7}$$



(a) (b) (c)

Figure 1 – Construction of a solid object. (a) Planar bounded half-spaces defined by equations 1 through 6, (b) the resultant box, and (c) the box with a Boolean subtraction of the cylinder defined by equation 7.[1]

Because the construction of CSG objects is computationally simple, this tends to be the preferred method of geometry definition for Monte Carlo simulation. Boundary crossings are easily computed, although there is a penalty for an excessive number of Boolean operations.

2.3 Boundary Representation Geometry (BREP)

Boundary representation geometry, or BREPS, is a more complex and robust method of defining geometry [8]. A BREP object is defined by specifying the constituent boundaries of the object and tying them together. These boundaries are constructed from the constituent parts, as seen in the schematic in Figure 2. Vertices are connected into edges, which form edge loops, which can define a surface.

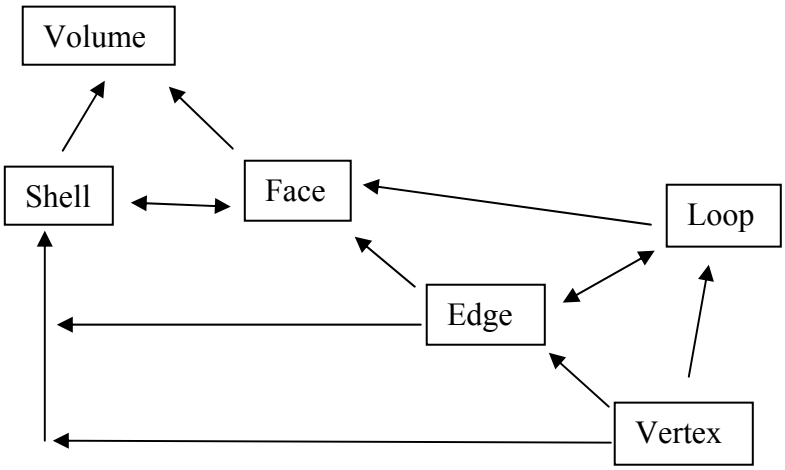


Figure 2 – Schematic formation of a BREP object (adapted from [9])

[1] All CAD images were generated using SolidWorks 2004 SP03.1. http://www.solidworks.com

Elementary BREPS consist of surfaces that are either planar or second-order constructs. The resulting solids are either polygonal or polyconcical. Advanced BREPS can consist of higher-order surface definitions, such as Bezier functions, B-splines, or NURBS (non-uniform rational b-splines). The use of higher-order functions allows for precise definition of complex shapes that cannot be constructed from algebraic elements.

The calculation of boundary crossings with BREPS can be computationally costly, especially for advanced BREPS (see [7]). As such, Monte Carlo transport codes generally do not include the use of BREPS solids in their geometry definition tools. However, this is the dominant mode of geometry definition in computer aided design software.

2.4 Voxels

A third method of geometry representation ignores the definition of the boundaries altogether. The entire object or set of objects is represented by a series of identically shaped volume elements, or voxels. Voxels are arranged in a regular lattice, and typically consist of right rectangular pyramids.

Voxel geometry is most often used when generating input from medical images, such as those derived from the use of tomographic images which constructs the data in that manner. It is very difficult to find the boundaries between objects in a medical image, which usually requires at least some degree of manual intervention. Figure 3 shows an example of how a tomographic medical image is constructed from constituent voxels.

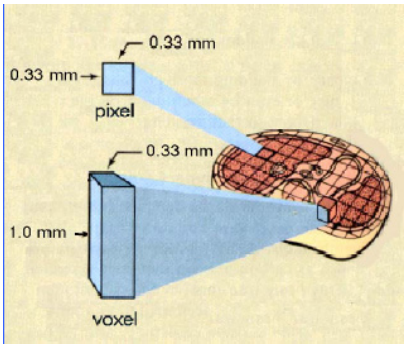


Figure 3 – Voxels from medical images (dimensions shown are examples)

As a modeled particle traverses the voxel lattice, the code computes and verifies boundary crossings at each and every voxel boundary. Since the voxel boundaries are planar surfaces, this is a simple calculation, but the sheer number of required computations makes this a costly requirement. Greater voxel size decreases the computational penalty for the use of voxel geometry.

Voxel geometry is an inherently imprecise definition of objects. An object's true boundary is likely to bisect a given voxel, but that voxel must be assigned to either one object or the other (or assigned to inside the object or outside the object). Smaller voxel size increases the precision in the representation of the object shape and volume. There is therefore a tradeoff between the precision of the object model and the computational speed. The voxel size may be effectively

limited by the memory requirements for handling a large number of voxels. Still, for highly complex objects or objects with a great deal of fine structure, there may be no other reasonable method of creating a computational model.

Another requirement for using voxels in organ dose calculations is the segmentation and label process to identify the organs that are made of a group of voxels. This is a time-consuming process and there are currently 20 voxel models with segmented organs including the VIP-Man model [6].

3 RESULTS

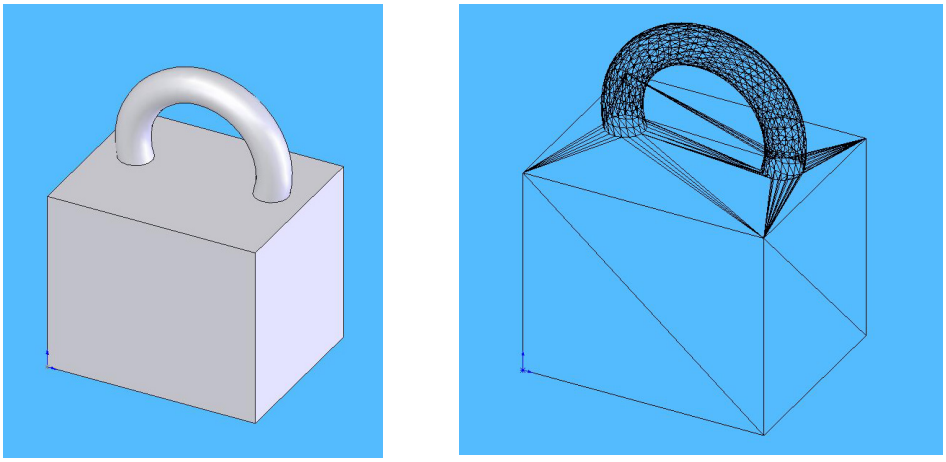
3.1 Data Sharing between CAD and Monte Carlo Codes

To share data between CAD applications and Monte Carlo codes, it is necessary to have the geometry data in a format that is either readable by both systems, or that can be easily translated between systems. Without access to the native format in use by a CAD program, access is limited to data file translation. Most CAD programs can read and write multiple file formats, so different sharing strategies may be available.

3.2 Surface approximations

One approach to the transfer of CAD data to Monte Carlo is to replace the exact surface descriptions with surfaces approximated by planar segments, usually in the shape of triangles or squares. Most CAD packages are capable of generating an output file in this format by creating a VRML (virtual reality markup language) or STL (stereo lithography) file. These are both text files that contain the descriptions of the constituent surfaces.

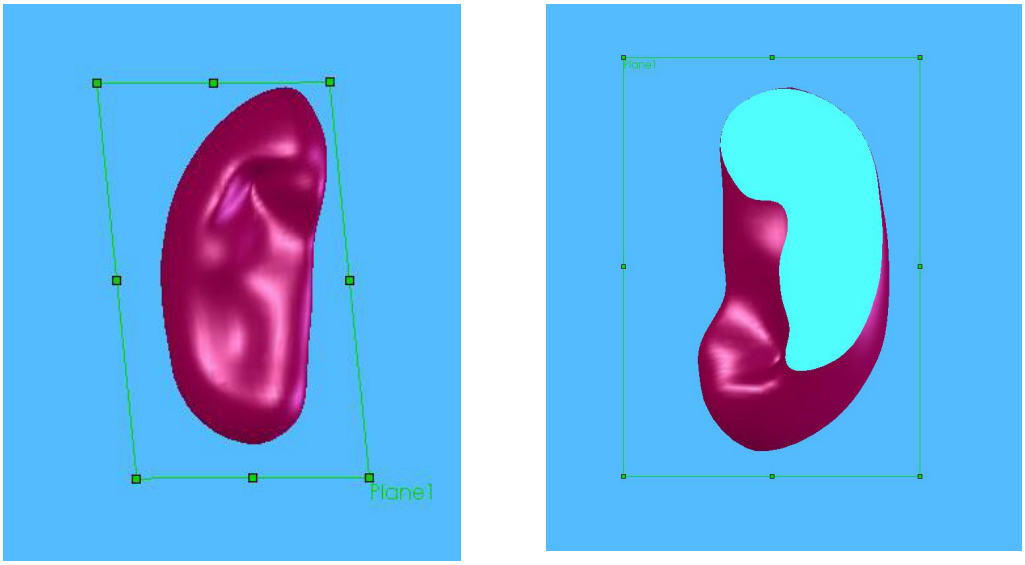
The approach presents a tradeoff between geometric precision and computability. Figure 4 demonstrates an example of the problem in generating a tessellated surface. Large and relatively flat surfaces can be accurately modeled with just a few planar segments. Highly curved surfaces or surfaces with fine detail require significantly more segments to maintain relatively accurate representations of the original.



(a) (b)

Figure 4 – A solid object and its definition with triangular planar segments

3.3 Voxel Constructs



(a) (b)

Figure 5 – Generating voxels from a three-dimensional image. (a) A CAD rendering of a kidney with an intersection plane shown. (b) The resultant two-dimensional slice (the remainder of the 3D image is visible beyond the sectioning plane)

If the CAD assembly contains a significant amount of complexity or fine structure, it might be appropriate for the solid objects to be devolved into voxels for input into a Monte Carlo transport code. The three-dimensional model in the CAD program is intersected with a plane, and an image output of that intersection is generated. The plane can then be stepped through the model at the desired vertical resolution. Figure 5 shows an example of a three-dimensional

model and the resultant planar image. The image is of the left kidney, derived from the VIP-Man data [6].

The process of generating voxel geometry is much more dependant upon the CAD program in use, since the data is generated internal to the program and not from an output file. Many packages may be capable of creating routines to perform the process automatically. It is likely that the output images will require a post-processing step to translate the image information (in the form of color information) into the material and region information necessary for Monte Carlo input.

As described above, the use of voxel geometry also requires a tradeoff between object precision and computation time.

3.4 Solid Geometry Constructs

If a CAD file consists of only of objects that can be described in CSG constructs, then geometry can be translated into a set of cards that is readable by MCNP, or re-written into the code segments required by EGS4 or GEANT4. An example of a fairly simple CAD image that was designed to be incorporated into MCNP is shown in Figure 6 [10].

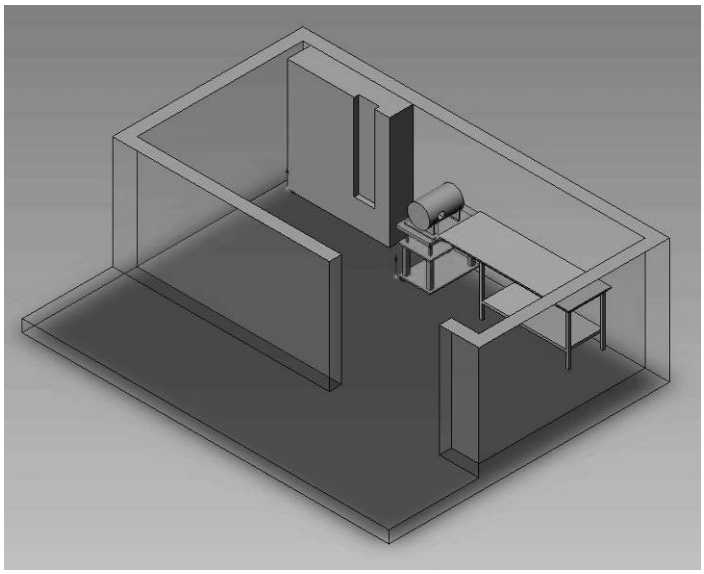


Figure 6 – CAD construction of a calibration lab room suitable for conversion to MCNP input

Information exchange can be best preserved with the use of a standard data format. Every major CAD program can read and write files in the STEP (STandard for the Exchange of Product model data) format, which is an international standard for product data exchange [5]. The STEP file is a text-only file containing a non-linear relationship network. The STEP file standard is a BREP format, so the individual objects are built up from the constituent parts. The file is "non-linear" in the sense that the logical relationships between the elements are not dependent upon or related to the physical position of the descriptions in the STEP file. Figure 7 is a segment of a STEP file, illustrating the relational nature of the listing.

```
#4 = VERTEX_POINT ( 'NONE', #219 ) ;
#5 = ADVANCED_FACE ( 'NONE', ( #220 ), #221, .T. ) ;
#6 = EDGE_LOOP ( 'NONE', ( #1461, #1397, #1174, #629 ) ) ;
#7 = ORIENTED_EDGE ( 'NONE', *, *, #92, .T. ) ;
#8 = EDGE_CURVE ( 'NONE', #1512, #491, #283, .T. ) ;
#9 = EDGE_CURVE ( 'NONE', #61, #111, #287, .T. ) ;
#10 = ORIENTED_EDGE ( 'NONE', *, *, #1424, .T. ) ;
```

Figure 7 – Example STEP code

An algorithm has been developed to translate a STEP file into an input for Monte Carlo simulation (see Figure 8). MCNP was chosen as the input format, although the process could be adapted for other codes. The process begins with iterative parsing of the STEP file, which identifies the key words that correspond to object boundary descriptions. The file parsing identifies the number of objects in the model. Algebraic descriptions of the object surfaces are generated through an iterative drilling down of the BREP elements. Once each object has been defined, MCNP cards are generated and output. Algorithm improvements are still needed to perform error handling in the STEP parsing, and manual checks for region ambiguity in the MCNP output are still necessary.

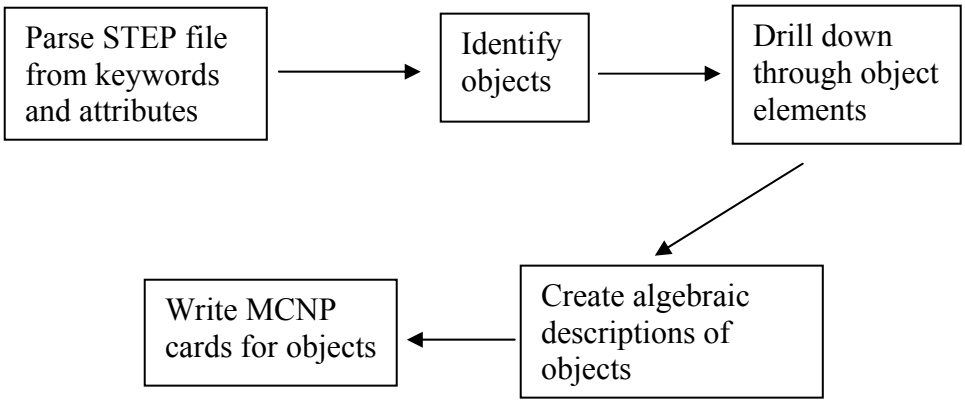


Figure 8 – STEP to MCNP conversion flowchart

4 CONCLUSIONS

At present, there is no single optimum strategy for sharing data between CAD systems and Monte Carlo radiation transport codes. The most exact solution is to incorporate precise geometry definitions, such as NURBS surfaces [11], but the computational penalty remains far too great. We have reviewed and described three strategies for the translation of CAD objects into Monte Carlo geometry elements. We have also shown some preliminary results using these methods. Although choice of data handling strategy is likely to remain application-dependent for some time to come, the tools described are a step forward in maintaining harmony between Monte Carlo simulations and other design groups.

5 ACKNOWLEDGEMENTS

Mark Furler and Mathew Oakley performed undergraduate research on the CAD simulations using SolidWorks during the summer of 2004.

6 REFERENCES

1. X-5 MONTE CARLO TEAM, "MCNP – A General Monte Carlo N-Particle Transport Code, Version 5, Volume I: Overview and Theory," LA-UR-03-1987, Los Alamos National Laboratory (2003).
2. The MCNPX Team, "MCNPX, Version 2.4.0", Los Alamos National Laboratory, LA-UR-02-5253 (2002).
3. W. R. Nelson, H. Hirayama, D. W. O. Rogers, "The EGS4 Code System," Stanford Linear Accelerator Center, SLAC-265 (1985).
4. S. Giani, et al., "Geant4: An object-oriented toolkit for simulation in HEP," CERN/LHCC 98-44 (1998). http://cern.ch/geant4
5. ISO, "Industrial automation systems - Product data representation and exchange Part 1: Overview and Fundamental Principles," ISO/IEC, ISO 10303-1 (1994)
6. X. G. Xu, T. C. Chao, A. Bozkurt, "VIP-MAN: An Image-Based Whole-body Adult Male Model Constructed from Color Photographs of the Visible Human Project for Multi-Particle Monte Carlo Calculations," *Heath Physics*, **78**, pp. 476-486 (2000)
7. T. J. Tautges, M.-K. Wang, D. L. Henderson, "CAD-Based Monte Carlo Transport Using MCNPX and CGM," *Transactions of the American Nuclear Society,* Washington, DC, November 14-18, 2004, Vol. 91, pp.185-186 (2004).
8. S. F. Buchele, W. A. Ellingson, "Reverse Engineering: Algebaric Boundary Representations to Constructive Solid Geometry," Argonne National Laboratory, ANL/ET/CP-93260 (1997).
9. W. Sun. "CT/MRI Based 3D Reconstruction, Part 1 – Basic," http://www.pages.drexel.edu/faculty/sunwei/MEM-CATE/CATE-L5-3DR1-Basic.pdf (2004)
10. J. Leone, M. Furler, M. Oakley, P. F. Caracappa, B. Wang, X. G. Xu, "Dose Mapping for a Cs-137 calibration source using MCNP5 Mesh Tallies," Accepted for publication by *Operational Radiation Safety* (2005).
11. X.G. Xu, C. Shi. "Preliminary Development of a 4d Anatomical Model for Monte Carlo Simulations," *Proceedings of the Monte Carlo 2005 Topical Meeting. The Monte Carlo Method: Versatility Unbounded In A Dynamic Computing World*, Chattanooga, TN, April 17-21, 2005.