Grid voxelization with partial volume effects in VTK

Release 1.00

Cory Quammen[1] and Russell M. Taylor II[1]

March 24, 2011

[1]Computer Integrated Systems for Microscopy and Manipulation
University of North Carolina at Chapel Hill, NC, USA

Abstract

We present a filter that voxelizes the volume of a 3D structured, unstructured, or rectilinear grid into a `vtkImageData` with partial volume effects. The partial grid volume occupying each voxel is computed exactly from the intersection of the grid volume and the voxel volume.

Latest version available at the Insight Journal [http://hdl.handle.net/10380/3254]
Distributed under Creative Commons Attribution License

Contents

Some applications require the conversion of a 3D surface mesh or volumetric grid to a 3D raster image. This conversion is called *voxelization*. This contribution is restricted to the voxelization of volumetric grids and does not discuss resampling of scalar values associated with nodes or cells in the grid.

VTK currently has two filters that can be used to perform voxelization of volumetric grids. The `vtkVoxelModeller` class converts geometry to a binary occupancy image represented by a `vtkImageData` object. Voxels whose centers are within the geometry are assigned the value 1 and voxels outside are assigned the value 0. The `vtkImplicitModeller` can also be used for voxelization of grids. It computes the distance from an arbitrary geometry or grid to image voxel centers. When applied to grids, voxelization may be obtained by setting the lower threshold to 0.

Neither vtkVoxelModeller or vtkImplicitModeller calculates partial volume effects that arise from the intersection of the grid and a voxel. We introduce a new class, vtkPartialVolumeEffectsModeller, that computes a scalar occupancy image where the scalar value at each voxel represents the fraction of that voxel occupied by cells inside the grid.

1 Algorithm

The algorithm executed by the filter is straightforward. For each voxel in the output image, do the following:

1. Compute a tetrahedral grid that represents the intersection of the grid cells and a volumetric box representing the boundaries of the voxel.
2. Sum the volumes of the tetrahedra in the intersection grid.
3. Divide the volume of the intersection by the volume of the entire voxel.
4. Store the result.

The most challenging portion of the algorithm involves computing the intersection of the grid and each voxel. Fortunately, the vtkBoxClipDataSet filter in VTK can be used for this computation by setting the clipping box to be the bounding box of a voxel to be processed.

2 Performance Optimizations

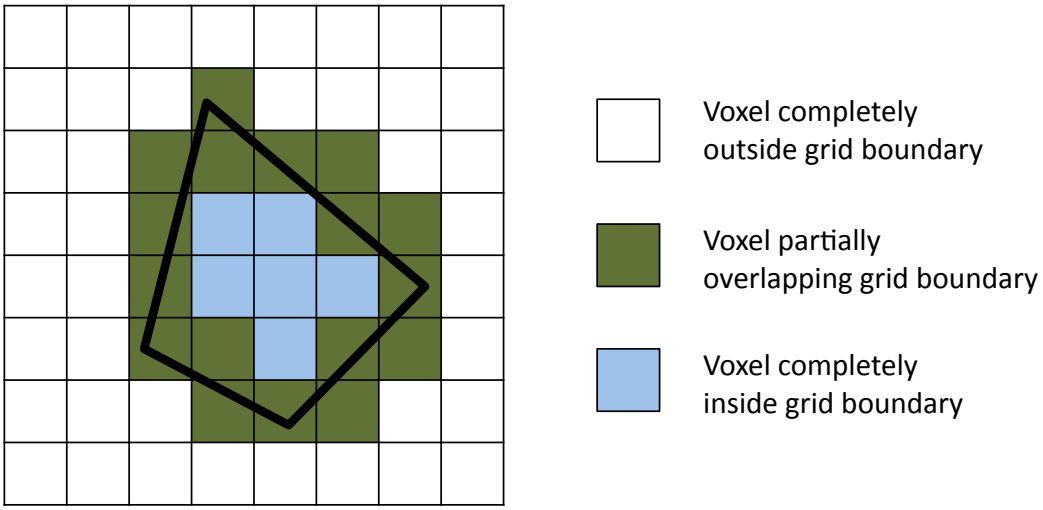


Figure 1: Partial volumes need to be calculated only for voxels containing portions of the grid boundary.

Computing the intersection of the grid and each voxel is expensive. Great efficiency gains can be made by computing the intersection only where voxels partially overlap the grid. This occurs whenever a voxel contains part of the grid boundary (see Figure 1). We extract the grid boundary using the vtkDataSetSurfaceFilter prior to voxelization. To determine whether a partial volume needs to be calculated for a voxel, we find the point on the grid boundary closest to the voxel center using a vtkCellLocator

operating on the boundary geometry. If that point is within the voxel bounds, then the `vtkBoxClipDataSet` is applied to cut the grid, and the volume of the intersection is computed. If the point is outside the voxel, then it is either entirely within the grid, in which case the volume of the intersection is set to 1, or outside the grid, in which case the volume of the intersection is 0. A second test using the `FindCell` method of the input grid determines whether the voxel is inside or outside grid.

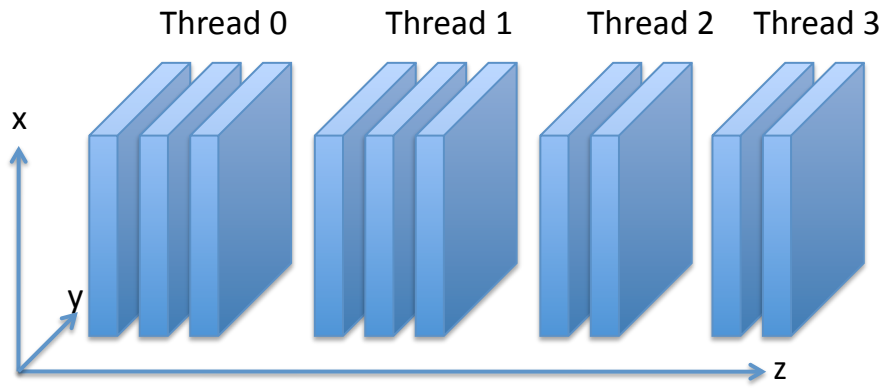
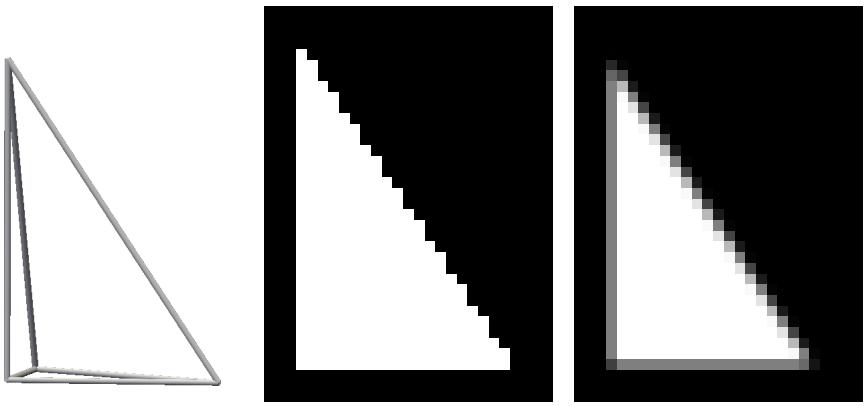


Figure 2: Partitioning scheme for an image with ten z-planes and four threads.

The `vtkPartialVolumeEffectsModeller` filter is multi-threaded to exploit multi-core architectures. Work is partitioned among threads by splitting the output image along the z-dimension, resulting in each thread being assigned a slab of z-planes. If the image has n z-planes and there are t threads, each thread is assigned $\lfloor n/t \rfloor$ planes. The first $mod(n,t)$ threads process an extra z-plane. This partitioning scheme is shown in Figure 2. For thread safety, one copy of the grid is created for each thread.

3 Results

Voxelization results using `vtkVoxelModeller` and `vtkPartialVolumeEffectsModeller` are shown in Figure 3.



(a) Input grid with one tetrahedron. (b) vtkVoxelModeller output. (c) vtkPartialVolumeEffectsModeller output.

4 Conclusions and Future Work

We have presented a filter that voxelizes the volume of a grid data set with partial volume effects. This filter may be useful for generating realistic ground truth test data with sensor integration effects for evaluating medical image segmentation algorithms, for example, or for generating synthetic blurred images of geometry for evaluating deconvolution algorithms.

One potential problem with the `vtkPartialVolumeModeller` is excess memory use from duplicating the input mesh for each thread. One way to alleviate this problem is to copy only the cells that partially intersect the image slab assigned to a thread. This could be accomplished by using the `FindCellsWithinBounds` method in the `vtkCellLocator` class to identify the cells on which a thread should operate and using the `vtkExtractCells` filter to create a grid structure with only these cells.