Computers and Fluids 121 (2015) 114–132

Contents lists available at ScienceDirect

Computers and Fluids

journal homepage: www.elsevier.com/locate/compfluid

Volumetric geometry for DSMC and the Voldipar code

Craig Turansky*, Brian Argrow

University of Colorado, Boulder, Colorado 80309, USA

ARTICLE INFO

Article history:
Received 26 February 2014
Revised 18 April 2015
Accepted 11 August 2015
Available online 18 August 2015

Keywords:
Rarefied
DSMC
Volumetric
Rasterization
Voxelization
Hypersonic
Verification

ABSTRACT

A quantized, volume-based approach to representing geometry in direct simulation Monte Carlo (DSMC) simulations is presented, and its implementation in a reference code is assessed. Volumetric or discrete/rasterized geometry is shown to be an acceptable way to perform, and possibly accelerate, the particle movement step of the DSMC algorithm. Canonical flow problems are solved that demonstrate the effectiveness of this method. These simulations capture expected results to within a few percent of results obtained from established codes in two-dimensions, and to within engineering tolerances for three-dimensions. The excellent scaling behavior of this approach may allow for high-resolution simulations in significantly less time than more classical geometry representations in DSMC for a given cell gridding method.

1. Introduction

Computational rarefied gas dynamics simulations are now most often performed with the direct simulation Monte Carlo (DSMC) method [1,2]. While many investigators may create their own DSMC codes, most are not described in great detail; nor are they codes that are large-scale and general-purpose. The DSMC codes that might be considered complete research tools include G. Bird's DS2V/DS3V family of tools [2–4], NASA's DAC [5,6] by G. LeBeau, and the University of Michigan's MONACO [7] which is maintained by I. Boyd.

These three codes can simulate common rarefied, hypersonic flow problems such as low-orbiting and re-entering spacecraft, and do so without restrictions on object geometry. The MONACO code utilizes a classical model of object geometry that has long been common in computational fluid dynamics (CFD). In this model, a computational mesh is formed that conforms in some meaningful way to an object over which flow is to be simulated. This mesh is composed of often disparately-sized polygons or polyhedra that form the collision cells in DSMC. Such a mesh is often constructed (with non-trivial effort on the part of the operator and/or software) to provide a favorable environment in which to allow the core algorithm of DSMC to evolve. The DAC code uses a Cartesian-cut cell model in which only the cells near to the solid geometry of any internal boundaries are modified to align with the planar facets of the body. An example of simple meshes used by DAC and MONACO can be found in a comparison report by Padilla [8].

The cost (both in human and computer time) of creating this type of classical CFD mesh can become taxing. And while good accuracy may be achieved, some elemental algorithms of DSMC may not perform well under certain types of mesh, resulting in undesirably costly computations. Particle tracking, for example, is time-consuming when cells are irregularly shaped and positioned. Tracking involves propagating simulation particles through the domain, colliding them with surfaces, and placing them within appropriate cells so they may contribute to the macroscopic interpretation of the flow. All of these operations can be difficult and costly when classical CFD geometry models are employed in DSMC.

An alternative to the "continuous" approach to geometry modeling is presented in this work that has certain advantages in simplicity, speed and scalability. Instead of representing solid objects as consisting of line and surface segments, the objects are discretized into their raster format that is called *voxelization* [9–11]. This creates a "digital" version of what is (originally) a continuous surface in the domain. All space in the domain is then quantized. An approach of this type was used in Bird's original three-dimensional (3D) code F3 [3], which was later extended to become the X2 code by Rault [12]. Though Bird's geometry model was briefly described as being voxelized, many remaining details were omitted and he makes no description of the discrete topology that is used. In this work, voxelization details are explained in depth with the aim of clarifying exactly how a volumetric geometry model is implemented and why it can be advantageous in DSMC.

* Corresponding author. Tel.: +1 303 2608 236.
E-mail addresses: turansky@colorado.edu (C. Turansky), brian.argrow@colorado.edu (B. Argrow).

http://dx.doi.org/10.1016/j.compfluid.2015.08.009
0045-7930/© 2015 Elsevier Ltd. All rights reserved.

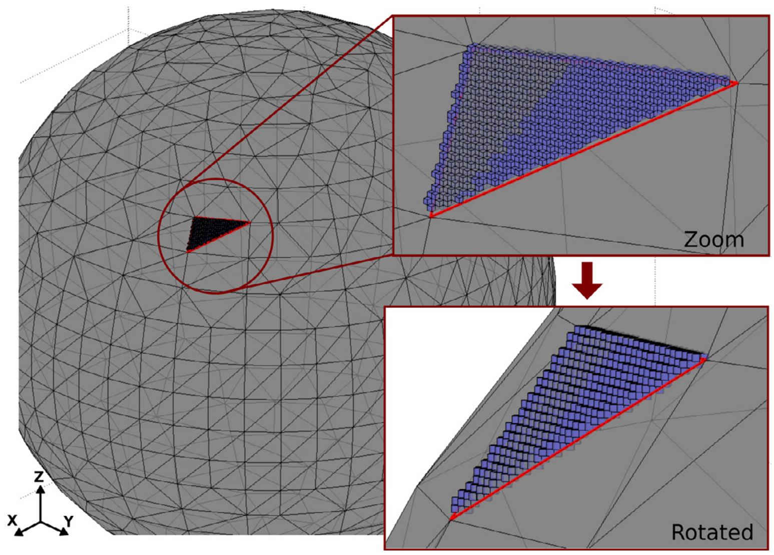

Fig. 1. Example of 3D voxelization of one triangle of a sphere.

All space in the domain is considered to be composed of (usually small) irreducible elements of space called pixels in two dimensions (2D) or voxels in three dimensions. This concept appears mainly in the fields of computer graphics and digital imaging. It has been considered for particle-based simulations, but mainly for visualization only [13]. A voxelized domain is shown here to function as an acceptable spatial model, and one that has no adverse affects on the accuracy of the DSMC algorithm. Section 2 provides a more detailed background on discrete geometry models and provides motivation for its use in DSMC. Fig. 1 shows an example of a segment of rasterized 3D geometry (a single triangle of an entire sphere). The source triangle is embedded within its voxelized discretization, causing some voxels to be behind/beneath the triangle.

The Voldipar (**Vol**umetric **Di**screte **Par**ticle) code has been developed as a reference implementation of the volume-based geometry method in full, as a complete program, by the first author. It is a generalized code with many common features that uses voxelized geometry. This code was previously applied to compute rarefied aerodynamic inputs used to simulate detailed rigid-body dynamics in continuum-rarefied transition flow by the authors [14,15]. In the remainder of this document the terms “Voldipar” or “the code” are used to refer to this code. This work aims to demonstrate the effectiveness of voxelized geometry and to partially verify the current state of the Voldipar code itself.

The document begins with a description of volume-based geometric methods, which includes some specific treatment of the most salient concerns involved, and algorithms to address them. Topics covered in Section 2 include generation of discrete geometry, discrete ray-casting for collision-detection, volume-filling, and a set of other minor algorithmic discussions that provide a more comprehensive treatment of volume-based methods than has yet been seen in DSMC literature. Section 3 compares the Voldipar code’s results to some of the other DSMC codes’ results available to the community via computation of common benchmark problems. Finally, some code performance evaluations are performed for 3D problems.

2. Volume-based geometry

In spaces of dimension two or greater, geometry may be discretized into a volume-based representation of itself that may be understood as a kind of analog-to-digital conversion. The core of the volume-based geometry method is the conversion of original or *source* geometry to working, *discrete* geometry. Note that the source geometry itself is not literally “analog”, but rather a discrete representation of a potentially curved surface. Interaction with this discrete geometry by simulator particles then occurs via a *discrete* ray-cast operation. This section defines the basic concepts behind discretized space models and discusses how the DSMC method may be implemented using such a model.

2.1. Voxelization basics

The source geometry is converted from continuous data in the form of points, line segments or polygons to a set of voxels. The term *voxel* (**Vo**lume + **x** + **El**ement) is used to refer to the smallest element of space recognizable by the code’s geometry algorithms. In 2D, a voxel reduces to the more-familiar term *pixel*, but for the sake of conciseness, the term *voxel* will be used to describe both 3D and 2D elements unless specifically denoted otherwise (e.g. “a 2D voxel”).

In general treatments, voxels may be of any size or shape, but the overwhelming majority of applications restrict them to being *d*-cubes (where *d* is the dimension of the space) that are all of uniform size everywhere in the domain and whose edges are parallel to the domain’s coordinate directions. It then follows that voxels can be said to have only a single dimension that defines their scale, termed their “size”. The voxel size L_v is the length from one corner to any other adjacent corner of a voxel.

Conversion of geometry is performed by a discretization algorithm known as *voxelization*. A number of *pixelization* algorithms have existed for decades, usually termed *rasterization*. The most famous of these algorithms is, perhaps, the Bresenham Line Algorithm [16], which is ubiquitous in the field of computer graphics. Voxelization is the analogous method for 3D geometry. Voxelization has recently begun to appear as a valid alternative to classical methods of representing and rendering computer graphics; mostly for video games and medical imaging [17]. In these contexts, however, the topological properties of the discrete, vox elized geometry is usually of little concern. Instead, the overall visual appeal of the resulting imagery is prioritized as both games and medical imaging are concerned with providing the best visually appealing or informative experience to a human viewer. This means that voxelizations in these applications may often be incomplete or inaccurate in some sense since they represent the final goal of whatever software is creating them, rather than an intermediate construction upon which further computations rely.