Computer-Aided Design 77 (2016) 98–106

Contents lists available at ScienceDirect

Computer-Aided Design

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

Out-of-core real-time haptic interaction on very large models

A. Aguilera [a], F.J. Melero [b,*], F.R. Feito [a]

[a] *Dpt. Informática, Univ. Jaén, Spain*

[b] *Dpt. Lenguajes y Sistemas Informáticos, Univ. Granada, Spain*

ARTICLE INFO

Article history:
Received 28 September 2015
Accepted 4 April 2016

Keywords:
Haptic rendering
Collision detection
Hierarchical bounding volumes

ABSTRACT

In this paper we address the problem of fast inclusion tests and distance calculation in very large models, an important issue in the context of environments involving haptic interaction or collision detection. Unfortunately, existing haptic rendering or collision detection toolkits cannot handle polygonal models obtained from 3D digitized point clouds unless the models are simplified up to a few thousand polygons, which leads to an important lack of detail for the scanned pieces. We propose a data structure that is able to manage very large polygonal models (over 25M polygons), and we explain how this can be used in order to compute the inclusion of a point into the solid surface very efficiently, performing several thousand point-in-solid tests per second. Our method uses a data structure called EBP-Octree (Extended Bounding-Planes Octree), which is a very tight hierarchy of convex bounding volumes. Based on a spatial decomposition of the model using an octree, at each node it defines a bounding volume using a subset of the planes of the portion of the polygonal model contained at that node. We use the EBP-Octree in a haptic interaction environment, where distance tests and the orientation of collided triangles must be accurate and fast. We also demonstrate that the proposed algorithm largely meets the interactive query rate demanded by a haptic interaction (1 kHz), despite being executed in a single CPU thread on a commonly available computer.

1. Introduction

Haptic interaction in virtual environments has certain time restrictions that require much more efficient data structures and algorithms than other collision detection techniques. When a user handles a 6 degrees-of-freedom (DOF) haptic device, he expects to perceive the *real* contact with the surface, and not just an approximation of it. It is generally accepted that a minimum update rate of 1 kHz is required for the collision detection thread to provide continuous (i.e. realistic) feedback [1]. Hence, the algorithm must be able to dispatch every collision or distance query in less than a millisecond. It is not straightforward to achieve this for very large polygonal models obtained from 3D scanners. In the context of art curators, for example, it is not conceivable to use simplified models of a few thousand polygons (like those usually used in traditional haptic applications) to virtually plan or test the restoration process of a sculpture.

We address the problem of fast inclusion tests and distance calculation in very large models by developing a system that performs several thousand point-in-solid and distance tests per second on models over 25M polygons. Our method uses a data structure termed EBP-Octree (Extended Bounding-Planes Octree). This is a very tight hierarchy of convex bounding volumes that, supported by a spatial decomposition of the model using an octree, defines a bounding volume at each node using a subset of the planes of the portion of the polygonal model contained at that node. To summarise, our main contributions are:

- A data structure, the EPB-Octree, which can virtually host models of unlimited-size. Our tests only used level 11 of 20 using models of 30 million polygons.
- A point-in-solid test that runs comfortably at a haptic rendering frame rate when applied to very large polygonal models, achieving from 2.5 kHz (purely random point test) to 32 kHz (haptic-like paths).
- A distance and contact normal function that allows us to use the EBP-Octree in haptic applications designed for training or planning for virtual curators that runs at around 1 MHz in classic haptic interaction over the surface of models over 25M polygons.

We tested the EBP-Octree in a simulated haptic interaction environment, where distance tests and the orientation of collided triangles must be accurate and fast. In addition, we demonstrate

* Corresponding author.
E-mail addresses: angel@ujaen.es (A. Aguilera), fjmelero@ugr.es (F.J. Melero), ffeito@ujaen.es (F.R. Feito).

http://dx.doi.org/10.1016/j.cad.2016.04.002
0010-4485/© 2016 Elsevier Ltd. All rights reserved.

that the proposed algorithm largely meets the interactive query rate demanded by a haptic interaction (1 kHz), despite being executed in a single CPU thread on a home-featured computer. We tried to test our models using other state-of-the-art approaches, but none of the packages and libraries we know were able to handle such large models.

Section 2 reviews the most relevant prior work, whereas Section 3 presents the data structure and its related construction algorithms. Section 4 specifies the out-of-core caching mechanism that allows us to perform point-in-solid and distance tests, the description and experimental results of which are presented in Sections 5.1 and 5.2 respectively.

2. Related work

Haptic Rendering is a term used to describe the process of calculating a reaction force for a given position of the haptic feedback device. It is closely related to collision detection algorithms, and this area of research has been extensively investigated by the graphics community. Lin et al. [2] introduce collision detection concepts and offer an overview of existing algorithms and data structures. However, collision detection algorithms are not directly applicable to haptic rendering, since it is not only necessary to detect collisions, but also to compute distances and normals at a high frequency rate (around 1 kHz). The simplest way of approaching haptic rendering is to consider a point-based device, as described in [3] when explaining the *contact levels of detail* (CLODs) model. With this approach, the main task is to compute distances from the haptic 3D cursor to the model's surface. When the distance is below a preset threshold, the haptic rendering thread presents a given force, calculated to avoid penetrating into the model.

The well known PQP package developed by Larsen et al. [4] uses swept sphere volumes as a bounding hierarchy for triangle clouds to detect collision between models. However, there is no possibility of testing its performance on very dense meshes. In [5], Gregory et al. present the basis for the HCollide method, using quite small models. A solution proposed by Otaduy in [6] handles models below 50k polygons.

Distance fields have been used extensively for collision detection and rapid distance computation. There are three main approaches to build distance fields: those based on Voronoi diagrams [7,8], on distance propagation methods [9] or techniques that use trees and grids as supporting data structures. Among the latter, a three-dimensional grid is used to store the distance field in [10], while an Adaptive Distance Field using an octree was developed in [11]. A quite original approach is provided by the *haptic textures* of Theoktisto et al. [12]. This proposes a texture-based normal mapping of the surface in a similar manner as bump mapping does for rendering in order to obtain a fast distance query rate.

McNeely [13] implements a voxelized model of the surface, the Voxmap, to give approximate distance values in a 6-DOF haptic environment, testing it in a 3-DOF haptic environment with a model of 593k polygons. In [14], Barbic et al. presented a CPU-based method that uses the Voxmap point shell to create distance maps for deformable models. In this work, the authors state that "only small point shells fit into the computational budget of one haptic cycle". They then propose using a hierarchy that handles models of up to 256k points. The work by Gueziec [15] generates a multiresolution hierarchy of bounding volumes via geometric simplification of the polygonal model in order to dynamically compute the distance from a point to an arbitrary polygonal mesh. However, the largest model tested is composed of 60k polygons. Similarly sized models can be found in recent papers based on GPU and parallel programming, as in the case of Lauterbach's work [16] where a parallel implementation of OBB (*Object Oriented Bounding Box*) and rectangular swept spheres runs over models of up to 75k triangles, and the CPU/GPU-based approach of Pabst et al. [17], where the largest model contains 146k polygons and the continuous collision detection computation time is 184 ms for this model in a single thread. Morvan et al. [18], also using a GPU approach, handle models of up to 1.7M polygons, offering proximity query rates of around 5 ms.

In [19], Walker et al. describe how to perform haptic interaction on huge terrain models (100M triangles, 2.5D), but their technique is not usable on 3D models, as they use parallel computer vision algorithms to detect collision by projecting the proxy onto the terrain image. The proposal by Yoon [20] runs on models similar to those presented in our paper, but their goal was to achieve interactive rendering frame rates (12–30 frames per second, 18 ms per collision query), not haptic rendering frame rates.

3. EBP-Octree data structure

Our proposal was inspired by the BP-Octree data structure [21], originally conceived for progressive visualization, which guided our work to create a data structure suitable for collision detection. The most characteristic feature of this data structure is that each node stores a set of planes that define a convex bounding volume of the part of the model contained in that node. These planes are restricted to be either face planes or planes parallel to faces, so it is possible to maintain as much of the original surface orientation as possible. This data structure leads to a tighter bounding volume than other BVHs, e.g. KDOPs or AABBs, as the plane's orientation is unrestricted and the number of planes at each node is not predefined.

Our EBP-Octree building process, described in the following subsections, is based on the steps described by Melero et al. in [21]. In addition, several new features and algorithms have been developed in order to achieve our goal of handling huge polygonal models at interactive haptic query rates. Noteworthy among these improvements are: the extension to a 64-bit octcode, the detection of special configurations and white/black nodes, the management of the temporary file system that handles the huge amount of geometric data computed during the EBP-Octree construction, and the cache-like out-of-core management of the tree. This means it can be used in haptic interaction environments, computing not only point-in-solid tests but also the distance and orientation of collisions.

3.1. Computing bounding volumes at leaves

Following the BP-Octree bottom-up construction algorithm, we define the deepest level of the tree as the level whose cell size is at least five times the average triangle edge length. Then, using that three dimensional grid, we apply an exhaustive 3DDDA (3-Dimensional Digital Differential Analyzer) algorithm to each polygon to detect any traversed cell, using *Morton codes* [22] to locate every cell (i.e. leaf node) in the octree space. In our new proposal, the *octcode* is 64 bits long, which allows us to handle octrees of up to 19 levels: 57 bits for the code itself and 5 bits for the level that the code belongs to. This makes the EBP-Octree capable of holding polygonal models of almost any size, limited only by the computer's available disk space, while the BP-Octrees 32 bit octcode limits the size of the input models to about 10M polygons (9 levels).

Once the polygons are distributed among the leaf nodes, we compute the bounding volume (illustrated in two dimensions in Fig. 2) in the same manner as in [21]. In order to recall briefly how this works, we describe the process visually in Fig. 2:

- At each leaf node n_l we define a set of *candidate planes*. These *candidate planes* are the set of supporting planes of each polygon of the node n_l.