COMMUNICATIONS IN COMPUTATIONAL PHYSICS
Vol. **2**, No. 3, pp. 545-576

Commun. Comput. Phys.
June 2007

Use of the Spatial *k*D-Tree in Computational Physics Applications

A. Khamayseh[1,*] and G. Hansen[2]

[1] *Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA.*
[2] *Multiphysics Methods Group, Idaho National Laboratory, Idaho Falls, ID, USA.*

Received 13 July 2006; Accepted (in revised version) 8 November 2006

Available online 4 December 2006

Abstract. The need to perform spatial queries and searches is commonly encountered within the field of computational physics. The development of applications ranging from scientific visualization to finite element analysis requires efficient methods of locating domain objects relative to general locations in space. Much of the time, it is possible to form and maintain spatial relationships between objects either explicitly or by using relative motion constraints as the application evolves in time. Occasionally, either due to unpredictable relative motion or the lack of state information, an application must perform a general search (or ordering) of geometric objects without any explicit spatial relationship information as a basis. If previous state information involving domain geometric objects is not available, it is typically an involved and time consuming process to create object adjacency information or to order the objects in space. Further, as the number of objects and the spatial dimension of the problem domain is increased, the time required to search increases greatly. This paper proposes an implementation of a spatial k-d tree (*sk*D-tree) for use by various applications when a general domain search is required. The *sk*D-tree proposed in this paper is a spatial access method where successive tree levels are split along different dimensions. Objects are indexed by their centroid, and the minimum bounding box of objects in a node are stored in the tree node. The paper focuses on a discussion of efficient and practical algorithms for multidimensional spatial data structures for fast spatial query processing. These functions include the construction of a *sk*D-tree of geometric objects, intersection query, containment query, and nearest neighbor query operations.

AMS subject classifications: 52B10, 65D18, 68U05, 68U07

Key words: Geometric query, bounding volume hierarchy, *sk*D-tree, containment query, mesh generation, *h*-refinement, remapping.

*Corresponding author. *Email addresses:* `khamaysehak@ornl.gov` (A. Khamayseh), `Glen.Hansen@inl.gov` (G. Hansen)

1 Introduction

Computational physics applications are rapidly increasing in complexity to address evolving requirements to include more realistic models and more detailed domain representations. Requirements often include the incorporation of more complex geometric forms of the parts and components within the model along with a larger number of parts and components being used to form the computational domain. Indeed, the simple two-dimensional models of the recent past that typically employed structured mesh discretizations in which geometric objects were represented by line segments, have been replaced by complex three-dimensional unstructured meshes containing general objects defined by compositions of parametric curves and surfaces.

Common across a wide variety of applications is the need to perform spatial queries involving the geometric objects contained within the domain with respect to computational abstractions employed within the simulation application. For example, it is necessary to track the movement of objects with respect to the elements in a transient Eulerian finite element analysis application. Many other applications, including biological population modeling, molecular dynamics, multiphase fluid dynamics, scientific visualization, and solid modeling also require spatial query capabilities. Typically, there is a spectrum of application requirements for spatial query functionality, which range from a very general geometry query to the need to perform highly localized searches of nearby objects. In the general problem, the state and relationship of the geometric objects are unknown. This query problem involves determining the relationship between the domain objects and simulation abstractions in the most general case. In the latter localized search, the state of the objects and their relationships to each other are typically known to some degree. Perhaps the spatial location of the objects were known at a previous time step, for example. For this application, it is typically more efficient to make use of this known state information to economize the spatial query processing; a general search each time step is usually too costly. However, a general search is usually needed when the application initializes to construct the local information.

It is always preferable to use spatial adjacency information if it is available, instead of general spatial searching. For example, the use of an *Eulerian Walk* [1] for the transfer of data from one distinct mesh to another in a multiphysics application has a complexity of $\mathcal{O}(m+n)$, where m is the number of elements in the *source* mesh, and n is the number of elements in the *target* mesh. Generalized spatial searching, such as the algorithm proposed here, can determine element intersection in $\mathcal{O}(m\log n)$ time. Clearly, as the number of objects in the source and target meshes increase, the time required to perform the general search increases at a faster rate when the general approach is used. This paper proposes an implementation of a general search method; a spatial k-d tree (*sk*D-tree) for use by various applications when a general domain search is required.

In the parlance of geometric modeling, objects are usually described by their associated spatial attributes (*e.g.* location). Within a given modeling configuration, objects may "intersect" each other, be "adjacent" to one-another, and may "contain" other objects.