The TR*-tree: A new Representation of Polygonal Objects Supporting Spatial Queries and Operations

Ralf Schneider and Hans-Peter Kriegel

Institut für Informatik, Universität München, Leopoldstr. 11B, D-8000 München 40, Germany

Abstract

In application areas such as graphics and image processing, computer aided design (CAD) as well as geography and cartography complex and time consuming spatial queries and operations have to be performed on polygonal objects. In the area of computational geometry different specialized data structures and techniques, such as plane sweep or divide and conquer, are used to design efficient algorithms for the different queries and operations. In this paper, we propose a new representation of polygonal objects, called the TR*-tree that efficiently supports various types of spatial queries and operations. The TR*-tree is a dynamic data structure that represents single objects as well as scenes of objects. The TR*-tree representations of polygonal objects are persistently stored on secondary storage and they are completely loaded into main memory for efficient query processing. In an experimental performance analysis, we demonstrate the fruitfulness of using TR*-trees investigating the 'Point in Polygon'-query and the 'Intersection of two Polygons'-operation.

1 Introduction

In application areas such as graphics and image processing, computer aided design (CAD) as well as geography and cartography complex spatial objects have to be managed. An important class of such objects are two dimensional polygonal objects. The management of polygonal objects imposes stringent new requirements on spatial database systems. One of the most challenging requirements is efficient processing of spatial queries and operations, e.g., 'Point in Object'-test or 'Intersection of Objects'.

In this paper, we present a new representation of polygonal objects, called the TR*-tree that efficiently supports various types of spatial queries and operations. The TR*-tree is designed to represent single polygonal objects as well as sets of objects, called scenes. These representations are persistently stored on secondary storage. In order to perform the query processing they are completely loaded into main memory. Additionally, a TR*-tree is completely dynamic, updates, insertions and deletions within a scene can be intermixed with queries and no periodic global reorganization is required.

The basic idea of our approach is to handle complex polygonal objects more efficiently by decomposing them into a set of simple components. On these simple components the geometric queries and operations mentioned above can be solved relatively simple. The success of such processing depends on the ability to narrow down quickly the set of components that are affected by the queries and operations. In order to design an efficient 'Point in Object'-test the line of argumentation in the area of computational geometry is as follows (see [PS 88]): The
fastest known search methods are based on bisection, or binary search. We easily realize, after a little reflection, that from the viewpoint of query time the ability to use binary search is more important than the minimization of the size of the set to be searched, due to the logarithmic dependence of search time upon the latter. Therefore, the fundamental idea is for the theoretical analysis to create new geometric objects in a preprocessing step to permit binary searching.

In our approach, we want to efficiently support the average case of various types of queries and operations in real applications where only one single representation of the objects is used. Therefore, we decompose in a preprocessing step the polygonal objects into a minimum set of disjoint trapezoids using the plane sweep algorithm, proposed by Asano & Asano [AA 83]. We cannot define a complete spatial order on the set of trapezoids that are generated by this decomposition process. Thus binary search on these trapezoids is not possible. Therefore, we propose to use spatial access methods for the spatial search. The R*-tree, an optimized variant of the well known R-tree [Gut 84], exhibits efficient query processing on spatial objects (see [BKSS 90]). Due to its tree structure the R*-tree also permits logarithmic searching in the average case but due to the overlap within its directory the search is not restricted to one path and thus logarithmic search time cannot be guaranteed in the worst case. The R*-tree was designed as a spatial access method for secondary storage. In order to speed up the queries and operations mentioned above we developed the TR*-tree which is a variant of the R*-tree minimizing the main memory operations and explicitly storing the trapezoids of the decomposed objects. We want to emphasize that in a TR*-tree the trapezoids of the decomposed objects are explicitly stored and not approximated by minimum bounding boxes.

The performance of the TR*-tree cannot be analytically proven because the TR*-tree is a data structure that uses heuristic optimization strategies. Therefore, we want to perform an experimental performance analysis investigating synthetically generated data as well as real cartography data in a systematic framework.

In the next chapter we define the spatial queries and operations that we want to support. Chapter 3 explains the decomposition approach of polygonal objects. In chapter 4 the structure as well as the performance of the TR*-tree is presented. Finally, in chapter 5 we describe how the spatial queries and operations defined in chapter 2 are algorithmically performed using the TR*-tree representation of polygonal objects. An experimental performance analysis that indicates the efficiency and performance of our approach is presented in section 6. The paper concludes with a summary that points out the main contributions of our proposal and gives an outlook to future activities.

2 Which set of spatial queries and operations should be supported?

First, we have to define on which types of objects the operations and queries should be performed. Our base-objects are simple polygons with holes where simple polygonal holes may be cut out from the enclosure polygon (see figure 1). A polygon is called simple if there is no pair of nonconsecutive edges sharing a point. From our experience, the class of simple polygons with holes is adequate for GIS applications (see [Bur 86]) and most 2D CAD/CAM applications.