Available online at www.sciencedirect.com
ScienceDirect
Information Systems 32 (2007) 1073–1100
www.elsevier.com/locate/infosys

Fast computation of spatial selections and joins using graphics hardware ☆

Nagender Bandi[a,*], Chengyu Sun[b], Divyakant Agrawal[a], Amr El Abbadi[a]

[a] *University of California, Santa Barbara, USA*
[b] *California State University, Los Angeles, USA*

Received 11 April 2006; accepted 5 December 2006
Recommended by Dr. K.A. Ross

Abstract

Spatial database operations are typically performed in two steps. In the *filtering* step, indexes and the minimum bounding rectangles (MBRs) of the objects are used to quickly determine a set of candidate objects. In the *refinement* step, the actual geometries of the objects are retrieved and compared to the query geometry or each other. Because of the complexity of the computational geometry algorithms involved, the CPU cost of the refinement step is usually the dominant cost of the operation for complex geometries such as polygons. Although many run-time and pre-processing-based heuristics have been proposed to alleviate this problem, the CPU cost still remains the bottleneck. In this paper, we propose a novel approach to address this problem using the efficient rendering and searching capabilities of modern graphics hardware. This approach does not require expensive pre-processing of the data or changes to existing storage and index structures, and is applicable to both intersection and distance predicates. We evaluate this approach by comparing the performance with leading software solutions. The results show that by combining hardware and software methods, the overall computational cost can be reduced substantially for both spatial selections and joins. We integrated this hardware/software co-processing technique into a popular database to evaluate its performance in the presence of indexes, pre-processing and other proprietary optimizations. Extensive experimentation with real-world data sets show that the hardware-accelerated technique not only outperforms the run-time software solutions but also performs as well if not better than pre-processing-assisted techniques.

Keywords: Databases; Geographic information systems; Query optimization

1. Introduction

Spatial databases are commonly used in applications such as geographical information systems (GIS) and computer-aided design (CAD) systems. The data stored in spatial databases are *spatial objects* such as locations, road segments, and geographical regions, which can be abstracted as geometries such as points, polylines, and polygons

☆ Earlier versions of this work appeared as "Hardware Acceleration for Spatial Selections and Joins" in the 29th International Conference of Management of Data, SIGMOD 2003 and "Hardware Acceleration in Commercial Databases: A Case Study of Spatial Operations" in the 30th International Conference on Very Large Databases, VLDB 2004.

* Corresponding author. Tel.: +1 8054235042.
E-mail address: nagender_iitg@yahoo.com (N. Bandi).

0306-4379/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.is.2006.12.001

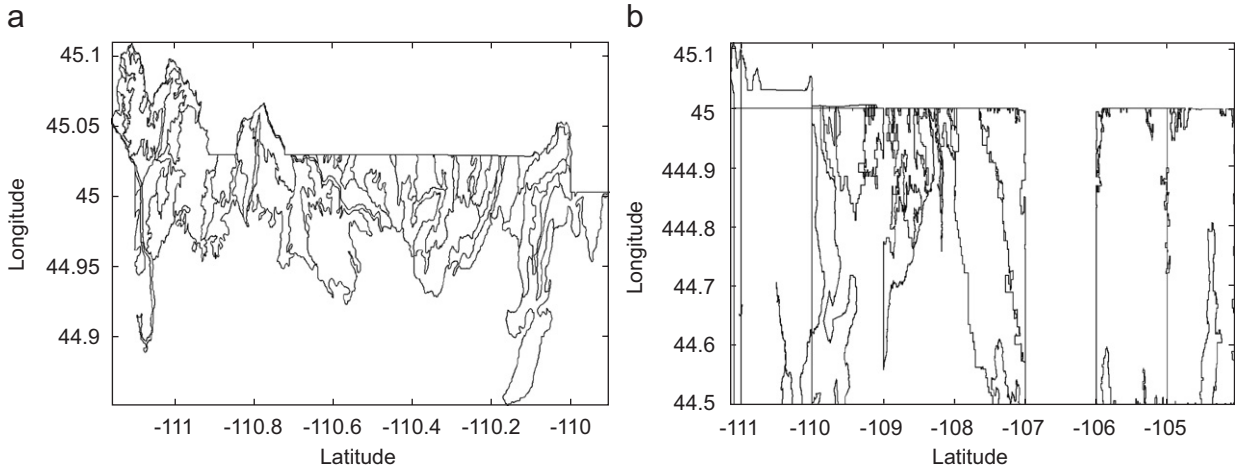

Fig. 1. Sample objects from two data sets: (a) LANDC and (b) LANDO.

in a 2D or 3D coordinate system. Some of the most commonly asked spatial queries are the polygon intersection and the polygon within-distance queries. In a polygon intersection query, given a data set of 2D or 3D polygons and a query polygon, we are interested in finding all the data polygons which intersect with the given query polygon. Similarly, in the case of within-distance queries, we are interested in all the data set polygons which lie within a certain distance from the given polygon.

Spatial database queries are typically evaluated in two steps: the *filtering step* and the *refinement step*. In the filtering step, the minimal bounding rectangles (MBRs) of the objects and spatial indexes such as R-tree [1] are used to quickly determine a set of candidate results. In the refinement step, the final results are determined by retrieving the actual geometries of the candidates from the database, and comparing them to either a query geometry or to each other. For complex geometries such as polygons, the cost of the refinement step usually dominates the query cost due to the complexity of the underlying computational geometry algorithms.

The cost of the refinement step consists of two factors: the *I/O cost* of loading the geometries from disk to main memory, and the *computational cost* of geometry–geometry comparison. The ratio of the computational cost over the I/O cost varies significantly depending on the types of spatial queries and the complexity of the geometries, which can be roughly characterized by the number of vertexes of a geometry. Generally speaking, the more complex the data, the more significant the computational cost. For instance, a study on spatial selections [2] shows that for point geometries, the I/O cost is the dominant factor, but for polygon geometries, both costs are significant. In the case of a spatial join, the computational cost could be orders of magnitude higher than the I/O cost, because once a geometry is loaded, it is buffered in the main memory and compared to many other geometries.

The high computational cost of the spatial operations comes from the complexity of the data in the real world, where it is not uncommon that a polygon has tens of thousands of vertexes. Furthermore, the shapes of the polygons can be arbitrarily complex, as can be seen from Fig. 1, which shows the first 100 polygons in the Wyoming land cover data set (LANDC) and the Wyoming land ownership data set (LANDO). In many cases, the polygons are concave, and sometimes, even non-simple.[1] Processing these types of polygons is very expensive. For instance, assuming the number of vertexes of two polygons are n and m, the complexity of the commonly used intersection test algorithm is $(O((n+m)\log(n+m)))$ [3], and the distance calculation algorithm is even more expensive with $O(n \times m)$ worst case complexity [4]. Since the I/O cost remains linear, the computational cost quickly outweighs the I/O cost as the data complexity increases.

Over the last decade, much effort has been directed to improving spatial query processing for complex geometries. For intersection queries,

[1]Non-simple polygons are polygons with self-intersecting edges or with vertexes that have degrees greater than 2 (more than 2 edges incident to a vertex).