

Using vTree Indices for Queries over Objects with Complex Motions

Sandeep Gupta Chinya Ravishankar
Department Of Computer Science and Engineering
University of California, Riverside
Riverside, CA 92521
{sandeep,ravi}@cs.ucr.edu

Abstract

We introduce the *vTree*, an index structure for efficient processing of spatiotemporal queries over sets of objects moving along complex trajectories. The *vTree* is a tiered structure, and partitions space at different granularities at different tiers. It uses two novel strategies to enhance the performance of spatiotemporal queries. First, it groups objects by velocity, and indexes objects from each group at an appropriate tier in the *vTree*, to localize the loss of precision induced by fast objects. Second, it accommodates complex trajectories by controlled replication of object descriptors at each tier. These features permit *vTree* indices to remain useful for longer time durations, and to support very efficient query processing. Our algorithms for *vTree* joins are designed to limit the portions of index and data space explored, as well as to maximize locality within the portion of space explored.

1. The vTree Index

We present a new index structure, called the *vTree*, as well as associated algorithms for range and join queries over complex trajectories. Our work considers both range and join queries over mobile objects with complex trajectories. Despite their obvious importance in applications, less attention has been paid toward the efficient processing of spatio-temporal joins. We present an accurate cost model for our index, and show how to construct efficient tree structures for given datasets. We also evaluate the efficiency of our method over different data sets and demonstrate substantial performance gains.

The *vTree* is a hierarchical organization that partitions the plane coarsely at its higher tiers and finely at its lower tiers. It also separates moving objects by velocity, placing faster objects at higher tier and slower entities at lower tier.

Stratification and replication are two concepts that are central to the structure of a *vTree* (see Figure 1). Stratifi-

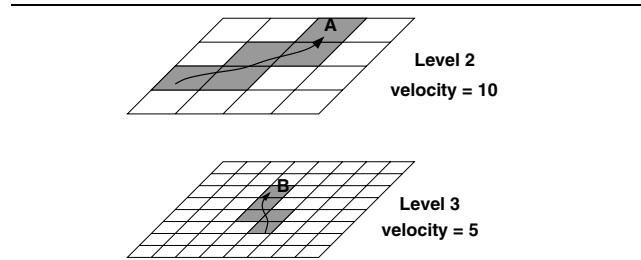


Figure 1. Stratification and object replication

cation separates objects by velocity. Thus objects with velocities falling into different ranges are placed into different tiers of the index.

Replication allows us to handle objects moving along arbitrary trajectories. We replicate each object into all the grid cells of its associated tier, (see Figure 1) which it enters during the lifetime T of the index. To minimize replication, we place faster objects at higher tiers, where space is partitioned more coarsely. Thus a faster object enters fewer grid cells during the index lifetime, although they might travel a greater distance.

Figure 1 illustrates this process for two objects A and B . Object A has higher velocity, and is placed at a higher tier. Object B has relatively a much lower velocity. Placing it at a lower tier allows us to index it at a higher resolution for the same degree of replication. At any given time, the location of B can be predicted with far better accuracy than that of A .

We need to design a *vTree* that minimizes the response time of the query operation, yet having the low overhead of the index size. To minimize the response time we need to minimize the disk access performed and simultaneously reduce the cpu cost. In this work we discuss design choices that attempt to achieve better response time by either reducing the disk access or by lowering the refinement cost.