Visualizing Query Execution and Performance Characteristics of Novel Techniques for Interactive Spatial Data Exploration

Overview: The goal of this project is to create a demo application to showcase recent research prototypes developed in the DIAS lab in the spatial data management domain. We have already developed in the lab both a desktop and an iPad application that have been presented in major database conferences and events [1] and we aim to integrate our two most recent research directions in a similar framework. These two research directions are briefly described below:

Incremental Spatial Indexing: Given today’s massive and rapidly growing amounts of spatial data, algorithms to query them efficiently are crucial. The database community has proposed many indexing techniques for fast and scalable querying of massive spatial datasets [2, 3]. There are, however, two major problems with such static approaches. First, they can significantly delay scientists from the actual querying: fully indexing a brain model in the Human Brain Project, for example, can take several hours [2]. Second, very often scientists only explore or analyze a small fraction of the entire spatial dataset and as a result the computational and storage resources spent on indexing the entire dataset are not fully amortized. We have developed incremental indexing approaches for spatial data in main memory. More specifically, we propose incremental alternatives for three representative spatial indexing categories: 1) a one-dimensional approach that incrementally sorts multi-dimensional objects that are mapped to 1D using a space-filling curve, 2) a multi-dimensional hashing approach that builds grid partitions incrementally, and 3) a hierarchical multi-dimensional partitioning approach. Our approaches reduce data-to-query time without introducing performance overheads, achieving comparable performance with an upfront-built index once a sufficient number of queries are executed. The visual interface should provide the user the ability to define a range query on a (small) spatial dataset, visualize the index structure after every query (see example figure), and visualize the query results as a 3D model that can be panned, rotated and zoomed. Finally the visual interface should present to the user the query execution statistics of the incremental approaches, comparing them to existing static approaches.

Bundled Spatial Indexing: In several real-life applications the data are naturally divided into distinct categories and users are often interested in some of the categories but not all of them. Additionally, users issue queries that explore different combinations of categories as they rarely know a priori which categories need to be combined to answer a particular question or to test a particular hypothesis. A bundled spatial range is conceptually equivalent to running a spatial range query on multiple spatial datasets (where all the objects of one dataset belong to a single category) and returning the query results for each dataset (or category). There are two straightforward ways to apply spatial indexes to answer bundled spatial range queries. The first strategy, 1-for-each, builds a separate spatial index (e.g. R-Tree [4] or FLAT [2]) for each category. Given a bundled spatial range query, we simply search each one of the spatial indexes corresponding to the queried categories. The second strategy, all-in-1, builds a single index structure containing all the categories. Given a bundled spatial range
query, we query the index using the spatial predicate and then we filter out the irrelevant objects that do not belong to any of the queried categories. We have developed STITCH, a disk-based index that combines the benefits of the aforementioned straightforward approaches and provides efficient query execution while scaling with an increasing number of categories.

The visual interface should provide the user the ability to define both a spatial range and a set of categories to be queried (each category will be mapped to a different color). It should then show the partitioning strategy of STITCH and compare it with the partitioning strategy of the 1-for-Each and the all-in-1 approaches. Finally, the 3D query results will be visualized and query execution stats will be provided (total execution time for each strategy, performed I/O etc.).

**Technical Skills:** All the approaches that have to be integrated in the framework (our own techniques and the existing techniques used for the comparisons) are implemented in C++. VTK OpenGL ES Rendering Toolkit will be used to create the visualizations.
References:


