Spatial searching

Johan W.H. Tangelder, Remco C. Veltkamp
Institute of Information and Computing Sciences, Utrecht University
Andreas Fabri,
Geometry Factory

Abstract

The **spatial searching** package implements exact and approximate distance browsing by providing implementations of algorithms supporting

- both nearest and furthest neighbor searching
- both exact and approximate searching
- (approximate) range searching
- (approximate) k-nearest and k-furthest neighbor searching
- (approximate) incremental nearest and incremental furthest neighbor searching
- query items representing points and spatial objects.

In these searching problems a set P of data points in d-dimensional space is given. The points can be represented by Cartesian coordinates or homogeneous coordinates. These points are preprocessed into a k-d tree data structure, so that given any query item q the points of P can be browsed efficiently. The approximate spatial searching package is designed for data sets that are small enough to store the search structure in main memory (in contrast to approaches from databases that assume that the data reside in secondary storage).

Spatial searching supports browsing through a collection of d-dimensional spatial objects stored in a spatial data structure on the basis of their distances to a query object. The query object may be a point or an arbitrary spatial object, e.g a d-dimensional sphere. The objects in the spatial data structure are d-dimensional points.

Often the number of the neighbors to be computed is not know beforehand, e.g. because the number may depend on some properties of the neighbors (e.g. when querying for the nearest city to Paris with population greater than a million) or the distance to the query point. The convential approach is k-**nearest neighbor searching** that makes use of a k-nearest neighbor algorithm, where k is known prior to the invocation of the algorithm. Hence, the number of nearest neighbors has to be guessed. If the guess is too large redundant computations are performed. If the number is too small the computation has to be reinvoked for a larger number of neighbors, thereby performing redundant computations. Therefore, Hjaltason and Samet introduced **incremental nearest neighbor searching** in the sense that having obtained the k nearest neighbors, the $k + 1^{st}$ neighbor can be obtained without having to calculate the $k + 1$ nearest neighbor from scratch.

Spatial searching typically consists of a preprocessing phase and a searching phase. In the preprocessing phase one builds a search structure and in the searching phase one makes the queries. In the preprocessing phase the user builds a k-d tree data structure storing the spatial data. In the searching phase the user invokes a searching method to browse the spatial data.

With relatively minor modifications, nearest neighbor searching algorithms can be used to find the furthest object from the query object. Therefore, **furthest neighbor searching** is also supported by the spatial searching package.

The execution time for exact neighbor searching can be reduced by relaxing the requirement that the neighbors should be computed exactly. If the distances of two objects to the query object are approximately the same, instead of computing the nearest/furthest neighbor exactly, one of these objects may be returned as the approximate nearest/furthest neighbor. I.e., given some non-negative constant ϵ the distance of an object returned as an approximate k-nearest neighbor must not be larger than $(1+\epsilon)r$, where r denotes the distance to the real k^{th} nearest neighbor. Similar the distance of an approximate k-furthest neighbor must not be smaller than $r/(1+\epsilon)$. Obviously, for $\epsilon = 0$ we get the exact result, and the larger ϵ is, the less exact the result.

Exact range searching and **approximate range searching** is supported using exact or fuzzy d-dimensional objects enclosing a region. The fuzziness of the query object is specified by a parameter ϵ denoting a maximal allowed distance to the boundary of a query object. If the distance to the the boundary is at least ϵ, points inside the object are always reported and points outside the object are never reported. Points within distance ϵ to the boundary may be or may be not reported. For exact range searching the fuzziness parameter ϵ is set to zero.

Our presentation at the user workshop will illustrate using the spatial searching package by several examples.