Astronomical Data Analysis Software and Systems XIII
ASP Conference Series, Vol. 314, 2004
F. Ochsenbein, M. Allen, and D. Egret, eds.

HTM2: Spatial Toolkit for the Virtual Observatory

György Fekete[1], Alex Szalay[1], Jim Gray[2]

[1] *Dept. of Physics & Astronomy, Johns Hopkins University, Baltimore, MD 21218, USA*

[2] *Microsoft Bay Area Research Center, San Francisco, CA 94105, USA*

Abstract. The hierarchical tringular mesh (HTM) is a discrete foundation for describing location, size and shape on the celestial sphere. Indices derived from HTM descriptors are used in a relational database for managing spatial information. Some of the new features available in the current implementation support operations such as the ability to perform searches based on arbitrary polygons, convex hulls of polygons or any region bounded by great or small circles. These functions are reached through a language that is implemented as an extension to MSQL Server 2000 relational database engine. The heart of the HTM tools can also be used through various interfaces in several langugages, like C++, C# and Java. An extensive XML specification for describing spatial structures to support spatial queries is also under development.

1. Motivation

One of the major functions of a spatial data management system is the effective exploitation and manipulation of the information inherent in points and regions. It does this to make searching through a vast collection of objects as efficient as possible. When this power is coupled to a state-of-the-art database management system, it provides a substanatial speedup for queries that have a spatial component.

Current astronomical surveys, like the Sloan Digital Sky Survey are expected to describe on the order of hundreds of millions of objects. The metadata associated with each object is managed in a database. Astronomers use databases of this kind to find objects not only by their physical attributes, but also by their location. In fact the spatial component of the query can take complex forms, such as *"find pairs of objects that are separated by θ degrees, and have other physical attributes"*. Furthermore, spatial correlation of pairs, triples, etc. of objects and regions of interest require complex (therefore expensive, time consuming) geometric computation. The challenge facing the astronomical database designer is that databases handle information that is organized into rows and columns (tables) very well, but the nature of spherical topology is such, that mapping (flattening) a sphere into a rectangle using *any* cartographic projection invariably introduces distortions, and worse, unacceptable discontinuities. But relational database management systems (RDBMS) are extremely good at or-

ganizing information contained in tables by manipulating *indices*. By mapping the sphere into a table, or onto a line, we harness the power of the database engine maximally.

2. Constraints, Convexes, Regions

These are the building blocks of regions on the sphere. A *constraint* is a region of the sphere that looks like a cap. Its boundary is described by the intersection of a plane with the sphere. This circle divides the sphere into two regions, so we make precise which region is the constraint by specifying the unit normal vector $\vec{v} = (x, y, z)$ so that $|\vec{v}| = 1$, that points to the center of the cap, and a signed distance D, that controls the distance of the of the perpendicular cutting plane from the origin in the direction of the unit vector. For $D < 0$, we get large caps, for $D > 0$ smaller ones. The two extreme cases are a degenerate constraint that collapses into a point when $D = 1$, and the whole sphere when $D = -1$. In fact, in our language for building shapes, the simplest is the constraint, and is represented by a 4-tuple (x, y, z, D)

3. Trixels

Trixels form the web of spherical triangles that tesselate the sphere. The mesh is obtained by recursively subdividing the spherical triangles created by projecting the edges of an octahedron onto the unit sphere. We can get an arbitrarily fine triangular mesh by increasing the number of succsessive subdivisions for each triangle. Because the method is so systematic, there is a natural way to encode each trixel at each level with a number. We call these numbers HTM ID's, or *hids* for short. For practical considerations, we impose an upper limit of 64 bits on the number of bits.

4. HTM Fundamentals

The *Hierarchical Triangluar Mesh* (HTM) is used for managing spatial information without the problems of flattening the sphere. It provides an arbitrary resolution partitioning of the sphere, so that sets of *hids* encode a region. This effectively collapses two dimensions into one, so that the index management capabilities of relational databases are maximally exploited.

The 64-bit *hids* that represent unique trixels are used as indices by the RDBMS. As a result, queries that involve examining regions merely manipulate sets of *hids*. These functions are made accessible to the database engine by defining wraper functions particular to the databse in use. In our testbed and production system, Microsoft SQL Server 2000 is extended by providing an interface to our function through so-called *extended stored procedures*. These functions provide adequate encapsulation of the HTM methods, so that users need not know the details of HTM algorithms, but can formulate their requests at a substantially high level.

Familiar shapes, like rectangles, circles, bands, are transformed into an internal normal form based on the union of *convexes*, which, in turn are inter-

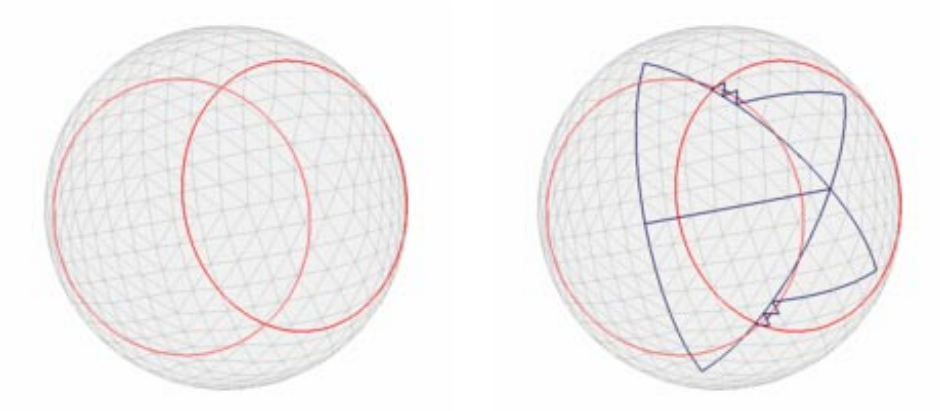

Figure 1. Two intersecting constraints and a trixel covering.

sections of *constraints*. In a computer program, the region is an object that contains the *hids* of the trixels that represent the region. These are generated by the library from descriptions in terms of familiar shapes, such as circles, rectangles, arbitrary polygons. If a user needs to know whether an observation is outside of a region of interest, a simple call to the HTM toolkit with the coordinates of the observation provides the answer.

It is important to note, that *any* single shape (or a convex) is an intersection of a finite number of constraints, and any region consisting of disconnected shapes is nothing more that a union of convexes.

5. SQL Interface

Typical queries with spatial components would sound like:

- Find all objects within a cone
- Find all objects within these Ra/Dec limits
- Find all objects within this spherical polygon

With SQL and the extra functions that implement the HTM algorithms, we can express these easily. The details of using SQL to express a general query is beyond the scope of the limited space in this article, but as an illustration, we show one simple example.

The most powerul funtion is `HTMCover()`. The argument given to it is a description of the region expressed as a union of convexes, but the interface also supports more intuitive shape descriptors made up of rectangles, polygons and circles. Figure 1 shows two search cones. In this example, our region of interest is their intersection, which is represented a *convex* with two constraints. The language which allows coordinates to be expressed in either as RA DEC pairs, or as Cartesian coordinates.

```
CONVEX CARTESIAN 1 1 0 0.8  1 0 0 0.8
```

This convex specification is a intersection of two constraints, both of whose $D = 0.8$. The two centers are $(1, 0, 0)$ and $(1, 1, 0)/||(1, 1, 0)||$, respectively. The toolkit automatically normalizes the vectors. The hids that cover the specified convex is the list:

```
   32     62     141     253  9120   9121
16144  16146  36494   36608  64589 65408
```

The smaller numbers represent the larger spherical triangles in Figure 1. In some queries, the list of hids may become huge. Fortunately, the hid values have a local coherence property, that is to say, clumps of trixels in a neighborhood have large runs of consecutive hids. Therefore we can consider a list of *runs* of hids instead of a very large number of individual hids. Consequently, the actual value returned by `HtmCover()` is an SQL table where each row is a pair hid values. This is an effective run-length encoding of the hid list.

It is possible, that a particular request for a cover would produce a very fragmented hid range list. Therefore our software ensures that the list of hid range values is always bound by a specific, but tunable parameter (between 15 - 100). This is accomplished by merging ranges with small gaps first, then those with greater gaps until the number of ranges is within the specified parameter. This introduces "false positives" into the cover, but that is preferable to ignoring an hid value that is actually inside (or intersecting) the convex.

For an extensive discussion and references the reader is directed to the link at the end of this page.

6. Concluding Remarks

Version 1 of the HTM has already been in operation for a while, but as the size of the databases grow, so does our need for improved performance. The current version (HTM2) of the libraries has had a significant improvement in performance and stability. Many bottlenecks in the code were identified, and recoded for speed. The bitlist feature was eliminated to give way to hid ranges. Internal data structures were redesigned where appropriate for better overall performance. Ports and a major restructuring of the code is underway to provide better support for multiple platforms without sacrificing performance.

Links

`http:/www.sdss.jhu.edu/htm`