The Hierarchical Triangular Mesh

Peter Z. Kunszt, Alexander S. Szalay, and Aniruddha R. Thakar

Johns Hopkins University, Baltimore MD 21218, USA

Abstract. The Hierarchical Triangular Mesh (HTM) is a method to subdivide the spherical surface into triangles of nearly equal shape and size. The HTM gives us a very efficient indexing method for objects localized on the sphere. We also developed a simple geometrical method to define an arbitrary area on the sphere which can be intersected by the HTM, returning the triangles covered by the area. Thus we have a powerful querying tool to find objects on the sphere by location.
In this contribution we define the Hierarchical Triangular Mesh, the geometric querying and we give concrete numbers of the performance of the current implementation.

1 Introduction

There is a great interest in indexing and geometrical querying of spherical surfaces not just in Earth Sciences but also in Astronomy and Astrophysics where the ancient 'index' of stellar constellations is still in common use. Up and until the recent all-sky surveys observations were localized to a well-defined area in the sky, making a planar projection straightforward so objects could be localized by their x and y coordinates in the respective projection (usually right ascension and declination). But with ever larger survey areas, the fact that the sky is a sphere starts to kick in and one has to deal either with singularities at the pole or with complicated cartographic projections.

The Hierarchical Triangular Mesh presented here conveniently maps triangular regions of the sphere to unique identifying names that can be used in itself as a reference. The technique to subdivide the sphere into spherical triangles is a recursive process. At each level of recursion the area of the triangles is roughly the same, which is a major advantage over the usual spherical coordinate system subdivisions where we have to deal with the singularities around the poles. Also, in areas with high data density, the recursion process can be applied to a higher level than in areas where data points are rare. This enables us to structure uneven data distributions into equal-sized bins. The data structure of the HTM is a quad-tree.

This scheme for subdividing the sphere has been advocated earlier [1]. The idea of the HTM has been described in [2]. Quad trees and geometrical indexing are discussed in [3], its applications in [4]. We have a more detailed work in preparation [5].

2 Definition of the HTM

The starting point is the octahedron on the sphere which gives us 8 spherical triangles with equal size. A "spherical triangle" is given by three points on the unit sphere connected by great circle segments. The octahedron has six vertices, given by the intersection points of the x, y, z axes with the unit sphere, which we enumerate $\boldsymbol{v}_o$ through $\boldsymbol{v}_5$:

$$\begin{array}{lr} (\ \ 0,\ \ \ 0,\ \ \ 1\ ) & \boldsymbol{v}_o \\ (\ \ 1,\ \ \ 0,\ \ \ 0\ ) & \boldsymbol{v}_1 \\ (\ \ 0,\ \ \ 1,\ \ \ 0\ ) & \boldsymbol{v}_2 \\ (-1,\ \ \ 0,\ \ \ 0\ ) & \boldsymbol{v}_3 \\ (\ \ 0,-1,\ \ \ 0\ ) & \boldsymbol{v}_4 \\ (\ \ 0,\ \ \ 0,-1\ ) & \boldsymbol{v}_5 \end{array} \tag{1}$$

The first 8 nodes of the HTM Index are defined as

$$\begin{array}{llll} \boldsymbol{v}_1, \boldsymbol{v}_5, \boldsymbol{v}_2 & S0 & \boldsymbol{v}_1, \boldsymbol{v}_o, \boldsymbol{v}_4 & N0 \\ \boldsymbol{v}_2, \boldsymbol{v}_5, \boldsymbol{v}_3 & S1 & \boldsymbol{v}_4, \boldsymbol{v}_o, \boldsymbol{v}_3 & N1 \\ \boldsymbol{v}_3, \boldsymbol{v}_5, \boldsymbol{v}_4 & S2 & \boldsymbol{v}_3, \boldsymbol{v}_o, \boldsymbol{v}_2 & N2 \\ \boldsymbol{v}_4, \boldsymbol{v}_5, \boldsymbol{v}_1 & S3 & \boldsymbol{v}_2, \boldsymbol{v}_o, \boldsymbol{v}_1 & N3 \end{array} \tag{2}$$

The triangles are all given with the vertices traversed counterclockwise. The construction of the next level makes use of this ordering. To get the second level, each triangle is divided into four new ones. We rename the vertices of the initial triangles (2) to $(\boldsymbol{v}_o, \boldsymbol{v}_1, \boldsymbol{v}_2)$ by keeping their counterclockwise order. (For example, $S0 : (\boldsymbol{v}_1, \boldsymbol{v}_5, \boldsymbol{v}_2) \rightarrow (\boldsymbol{v}_o, \boldsymbol{v}_1, \boldsymbol{v}_2)$)

So a given spherical triangle $(\boldsymbol{v}_o, \boldsymbol{v}_1, \boldsymbol{v}_2)$ with the vertices ordered counterclockwise is subdivided into four spherical triangles by connecting the side-midpoints $(\boldsymbol{w}_o, \boldsymbol{w}_1, \boldsymbol{w}_2)$

$$\begin{aligned} \boldsymbol{w}_o &= \frac{\boldsymbol{v}_1 + \boldsymbol{v}_2}{|\boldsymbol{v}_1 + \boldsymbol{v}_2|} \\ \boldsymbol{w}_1 &= \frac{\boldsymbol{v}_o + \boldsymbol{v}_2}{|\boldsymbol{v}_o + \boldsymbol{v}_2|} \\ \boldsymbol{w}_2 &= \frac{\boldsymbol{v}_o + \boldsymbol{v}_1}{|\boldsymbol{v}_o + \boldsymbol{v}_1|} \end{aligned} \tag{3}$$

The four new triangles are given by

$$\begin{array}{ll} \text{Triangle } 0 & : \boldsymbol{v}_o, \boldsymbol{w}_2, \boldsymbol{w}_1 \\ \text{Triangle } 1 & : \boldsymbol{v}_1, \boldsymbol{w}_o, \boldsymbol{w}_2 \\ \text{Triangle } 2 & : \boldsymbol{v}_2, \boldsymbol{w}_1, \boldsymbol{w}_o \\ \text{Triangle } 3 & : \boldsymbol{w}_o, \boldsymbol{w}_1, \boldsymbol{w}_2 \end{array} \tag{4}$$

The node name of the new triangles is the name of the original triangle with the triangle number appended. If the original node name was $S0$, the new node names are $S00$, $S01$, $S02$ and $S03$ (see figure 1). To repeat the iteration, we rename the vertices of the four new triangles in (4) to $\boldsymbol{v}_o$, $\boldsymbol{v}_1$, $\boldsymbol{v}_2$ again – keeping their order counterclockwise. Now we can use each of these new triangles as starting points for the next iteration, generating a quad-tree. The current implementation can handle a depth up to level 25. After that

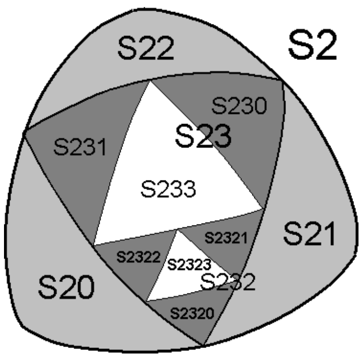

Fig. 1. The quad tree is obtained by subdividing spherical triangles into four smaller ones. The procedure is repetitive, and the naming scheme is just to add the number of the triangle to the parent's name.

level the 64-bit double precision arithmetics of any standard compiler breaks down; the difference of the vectors defining the corners of a HTM node are too small to handle. Level 25 corresponds to a resolution of 9.6 milliarcseconds.

The HTM is a mesh of triangles that do **not** have the same size and shape. Figure 2 shows the distribution of the triangle areas for any level (after a certain depth, say 4). There are some triangles whose area is less than average and some whose area is larger than average but the distribution of these and their deviation is constant. The mean area is of course always 1/4th of the previous one, starting with the area of the root-level triangles, which is exactly 1/8th of the sphere surface.

3 Lookup by Coordinate

Now that we have defined the HTM nodes we want to be able to look up which triangle a certain object belongs to. We assume we have the object's cartesian coordinates $\boldsymbol{v} = (x, y, z)$ because any coordinate system can be converted into cartesian coordinates. The lookup procedure is a simple walk down the HTM quad-tree, starting from the 8 root nodes. At any given level we can test for a point being inside or outside a spherical triangle by

$$\begin{aligned}(\boldsymbol{v}_o \times \boldsymbol{v}_1) \cdot \boldsymbol{v} > 0 \;\&\& \\ (\boldsymbol{v}_1 \times \boldsymbol{v}_2) \cdot \boldsymbol{v} > 0 \;\&\& \\ (\boldsymbol{v}_2 \times \boldsymbol{v}_o) \cdot \boldsymbol{v} > 0\end{aligned} \tag{5}$$

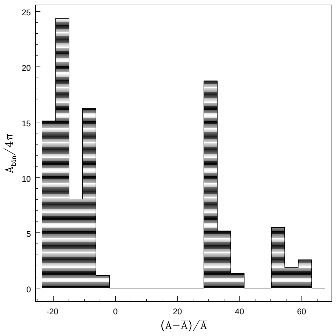

Fig. 2. The area of the triangles deviating from the mean.

If the condition (5) is satisfied for a node $(\boldsymbol{v}_0, \boldsymbol{v}_1, \boldsymbol{v}_2)$ we go down one level and test for each of its 4 child nodes. The condition will only be true for one of them, and so we can repeat the procedure to any level. The name of the triangle node that corresponds to the point $\boldsymbol{v}$ is assembled on the way.

The current implementation has been tested on a 600MHz Pentium III laptop running RedHat Linux 6.1. On that system we can do over 50,000 lookups per second for the HTM level-5 ID for an object. If we want to get the level-25 HTM-node ID we still can do over 11,000 lookups per second.

4 Definition of the Geometrical Querying

We want to have a generic possibility to intersect the HTM with any given surface on the sphere. In this section, we give the geometry primitives that can be used to define any area on the sphere. These primitives can easily be intersected with the HTM nodes to decide whether a given triangle is inside or outside an area.

We have the following hierarchy of geometry primitives:

- **Constraint** – a circular area, given by the plane slicing it off the sphere.
- **Convex** – a combination of constraints (logical AND of constraints)
- **Domain** – several convexes (logical OR of convexes)

A *constraint* can be characterized by a vector $\boldsymbol{p}$ and a distance d from the origin:

$$c := \{\boldsymbol{p}; d\} \qquad |\boldsymbol{p}| = 1 \quad ; \quad -1 < d < 1 \tag{6}$$

We allow for 'negative' constraints $d < 0$, which cover more than half of the spherical surface. The sign of the constraint is defined to be the sign of d. Example: $\{\boldsymbol{p} = (0,0,1); d = 0.5\}$ defines the cap north of 30^o latitude north.

A *convex* is the logical AND of constraints, i.e. the intersection of caps on the sphere:

$$x := c_1 \;\&\&\; c_2 \;\&\&\; \cdots \;\&\&\; c_n \qquad n \in \mathcal{N}^+ \tag{7}$$

Such a geometry is a convex in 3D but not necessarily a convex area on the spherical surface. As an example, the convex

$$\{(0,0,1);-0.01\} \qquad \&\& \qquad \{(0,0,-1);-0.01\}$$

defines a narrow strip around the equator. It is not even true that convexes are always a contiguous area on the sphere. Imagine the convex given by the intersection of two great-circle strips that are not in the same plane – the resulting convex is given by two disjoint areas where the strips meet. We define a sign for the convexes too:

$$\mathbf{sign}(x) := \begin{array}{ll} \text{negative} & c_i \text{ all negative or zero} \\ \text{zero} & c_i \text{ all zero} \\ \text{positive} & c_i \text{ all positive or zero} \\ \text{mixed} & \text{at least one positive and one negative constraint} \end{array} \tag{8}$$

The sign of the convex is an important indicator how the intersection with the HTM should be carried out.

Finally, define a *domain* to be logical OR of convexes:

$$d := x_1 \mid\mid x_2 \mid\mid \;\cdots\; \mid\mid x_n \qquad\qquad n \in \mathcal{N}^+ \tag{9}$$

Thus a single domain can represent any complicated area on the sphere. By intersecting domains with the HTM triangles we have a powerful indexing tool at hand.

5 Intersecting with the HTM

For a certain area, we want to get all HTM nodes of a given depth that are either completely or partially covered by it. We would like to know for each triangle which category it falls into – completely contained in the area, partially contained or not at all. Figure 3 shows a constraint on the sphere and the corresponding triangles. To test the intersection of a constraint with an HTM node, we check how many of the 3 corners of the HTM triangle are inside or outside the constraint $(\boldsymbol{p}, d)$:

$$\boldsymbol{p} \cdot \boldsymbol{v}_i > d \tag{10}$$

If the condition (10) is true for all 3 corners, the triangle is completely covered by the constraint. If just one or two of the corners $\boldsymbol{v}_i$ meet the condition the triangle is flagged as partially inside the constraint. If none of the corners $\boldsymbol{v}_i$ meet the condition, we need to carry out further tests to determine whether the triangle is completely outside the area or just partially inside.

It can happen that the constraint is small enough to intersect with one of the triangle's sides twice, not including any of the corners. The quadratic equation giving the intersection of the plane defining the constraint with the

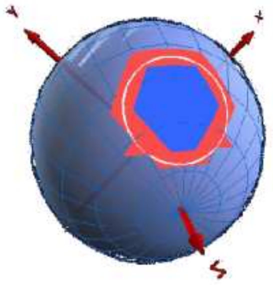

Fig. 3. A circular area is intersected with a level 5 HTM.

great circle of an HTM triangle side can easily be solved, giving us the answer to this case. Still, it can be that it is small enough to be completely inside the triangle so we even need to make one more test. We look whether the center of the constraint $\boldsymbol{p}$ is inside the triangle by applying the lookup condition (5). This tells us the final answer.

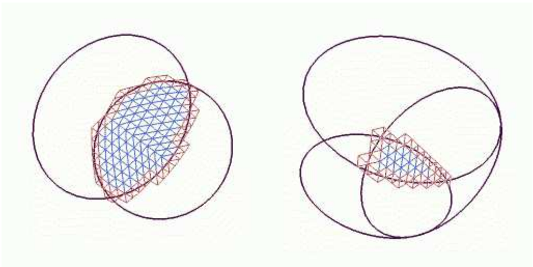

Fig. 4. The HTM nodes for a convex composed of two or three constraints.

To test for inclusion in convexes, the same tests have to be applied to all constraints of the convex. Only triangles that survive the conditions of all constraints will be included in the list of triangles covered by the convex. Figure 4 shows the results from intersectinos with convexes.

The decision for areas defined by domains is much more difficult and the decision trees are much more complicated, given in [5]. However, complicated cases are very rare and one needs to think hard to come up with exceptions to the simple solutions where the domain is just given by the sum of intersections with all convexes.

The performance of the current implementation regarding intersections depends on the size and complexity of the area. For a convex with 4 constraints we can do 5000 intersections per second down to level 20.

6 Discussion and Outlook

The HTM is a very efficient indexing mechanism for objects having spherical coordinates. Object catalogs can use the HTM as a means for object clustering and object containment. In Astronomy, there are several projects that have endorsed HTM as part of their system. Among those are for example the Sloan Digital Sky Survey (SDSS), the 2 Micron All Sky Survey (2MASS), the Second Guide Star Catalog (GSC*II*). The discussion of its advantages and disadvantages over another partitioning system, HEALPIX, can be found in this volume [6].

The high resolution capability of the HTM triangles offer a fast means to crossmatch objects across catalogs. Matching HTM levels and names can even be achieved without having a stored HTM id just by knowing the object coordinate and its error. The hierarchy helps to establish a common resolution and the uniqueness can be exploited as a second naming scheme. A prototype code of a crossmatching algorithm shows that with the current implementation one can do at least 2000 crossmatches per second. This is usually fast enough because object retrieval from catalogs especially across networks is much slower.

The current implementation with all the documentation and many versatile interfaces can be freely downloaded from `http://www.sdss.jhu.edu/htm/`

Planned enhancements and additions include the capability to have an onion layer of HTM indices to have 3D queries and to include the possibility to have balanced trees in the implementation (as opposed to level trees).

References

1. P. Barrett, Application of the Linear Quadtree to Astronomical Databases, Poster at ADASS 1994.
2. Kunszt, P. Z., Szalay, A. S., Csabai, I., & Thakar, A. R. 2000, in ASP Conf. Ser., Vol. 216, Astronomical Data Analysis Software and Systems IX, eds. N. Manset, C. Veillet, D. Crabtree (San Francisco:ASP), 141
3. H. Samet, The Design And Analysis of Spatial Data Structure, 1989, Addison Wesley.
4. H. Samet, Application of Spatial Data Structuers, 1990, Addison Wesley
5. P. Z. Kunszt, et al. (2000) in preparation.
6. W. O'Mullane, this volume.