Indexing a Large-Scale Database of Astronomical Objects

Bin Fu (Student), Eugene Fink, Garth Gibson and Jaime Carbonell
{binf, e.fink, garth, jgc}@cs.cmu.edu
Computer Science
Carnegie Mellon University

ABSTRACT

When astronomers analyze sky images, they need to identify the newly observed celestial objects in the catalog of known objects. We have developed a technique for indexing of astronomical catalogs, which supports fast retrieval of matching catalog objects for every object in new images. It allows processing of a sky image in less than a second, and it scales to catalogs with billions of objects.

1. PROBLEM

When astronomers analyze telescope images, they check whether the newly observed objects are listed in catalogs of known objects. Due to atmospheric and optical distortions, the positions of celestial objects in a telescope image may change slightly from observation to observation. Thus, astronomers need to retrieve not only exact catalog matches but also close approximate matches.

We assume that we have a catalog of known celestial objects, and a new image. The position of each object is represented by two values, called *right ascension* and *declination*, which define its equatorial coordinates (Figure 1).

We also assume that the edges of the image are parallel to the directions of right ascension and declination (Figure 1). We need to find the *matches* for all image objects. Specifically, for each object p in the image, we are looking for an object q in the catalog such that:

- Among all objects in the image, p is the nearest to q.
- Among all objects in the catalog, q is the nearest to p.
- The distance between p and q is at most 1 arc second, which is 1/3600 of a degree.

If the catalog contains an object q that satisfies all these constraints, we call it the *match* for p.

2. APPROACH

The sizes of modern astronomical catalogs exceed the memory of desktop computers. For example, the Guide Star Catalog II (GSC-II) contains almost one billion objects [3]. If we represent each object by a ten-byte record, this catalog takes ten gigabytes. We therefore store a catalog on disk and load only parts relevant to processing a given image.

We first describe the organization of the catalog on disk. We then present the retrieval procedure that identifies the relevant part of the catalog and loads it into memory. Finally, we explain the in-memory matching.

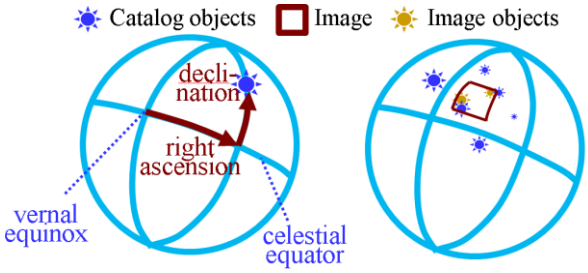

Figure 1. Left: The representation of a celestial object in spherical coordinates. Right: An example of the matching problem. There are two objects in the image, both of which have catalog matches.

Indexing: The indexing procedure arranges the catalog objects on disk, with the purpose to minimize the number of disk accesses during the retrieval.

We split the celestial sphere into multiple longitudinal strips, which are parallel to the direction of right ascension; each strip is exactly one degree wide (Figure 2). The objects within a strip are stored as a separate file, where the elements in the file are in sorted order by their right ascension.

Retrieval: Given an image, we need to retrieve the catalog objects that may potentially match the image objects. Since a matching catalog object must be within 1 arc second from a newly observed object, possible matches may be located at most 1 arc second away from the image area. We determine the *axis-aligned minimum bounding box* of the image, which is the smallest rectangle that covers all image objects, with sides parallel to the directions of right ascension and declination. We then extend the bounding box by one arc second on all sides.

We retrieve all catalog objects inside the extended bounding box from the catalog files, which is done in three steps. The first step is to locate the strips that overlap the extended bounding box; the second is binary search within each respective file, which identifies all catalog objects whose right ascension value falls inside the extended bounding box; the third is to load all the related objects into memory.

Matching: The last main step is to identify matches among the objects loaded into memory. To perform this step, we further subdivide the retrieved strips into thinner *substrips*. These substrips are also parallel to the direction of right ascension, and each substrip is exactly one arc second wide. The objects within each substrip are sorted by their right ascensions.

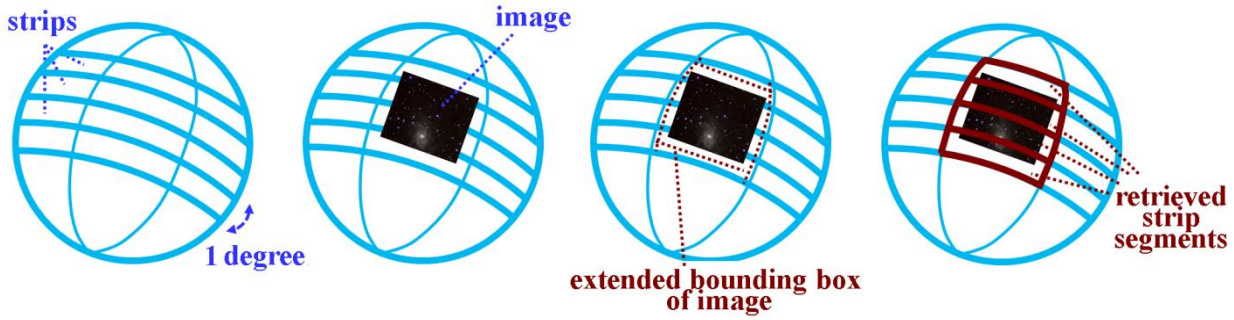

Figure 2. Indexing and Retrieval procedure. From left to right: The celestial sphere is divided into one-degree-wide strips; An image is given to find matches; The axis-aligned minimum bounding box of the image is calculated and extended by 1 arc second on all sides; Then, the part of the catalog that covers the extended bounding box of the image (that is, the retrieved strip segments) is loaded into memory.

For each image object, since its match can be at most one arc second away, we only consider the catalog objects in its three nearby substrips, that is, its own sub-strip and the two adjacent substrips. We use binary search in each nearby substrip to identify the close catalog objects for the image object, and calculate the distances between the image object and each of its close catalog objects. We illustrated the matching procedure in Figure 3.

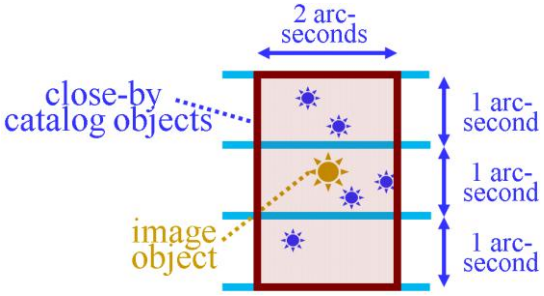

Figure 3. Matching procedure. For each image object, we extract all catalog objects that are at most one arc second away, and then calculate their distances.

Finally, after we have processed the above procedure for all the image objects, their matches can be determined: For each image object *p*, assume its closest catalog object is *q*. If *q*'s closest image object is also *p*, then *q* is the match for *p*.

3. EXPERIMENTS

We have run the experiments on a desktop computer with 2.66 GHz dual core.

Indexing: The time complexity of the indexing procedure is $O(N \cdot \lg N)$, where N is the number of catalog objects. For a catalog of two billion objects, it takes about 1.7 hours.

Retrieval: To find matches for an image, we retrieve related catalog objects (Section 2.2) and then apply the matching procedure (Section 2.3). We show the running time of these two procedures in Figure 4. Both procedures are fast, and it takes less than 0.8 seconds to find the matches for all image objects.

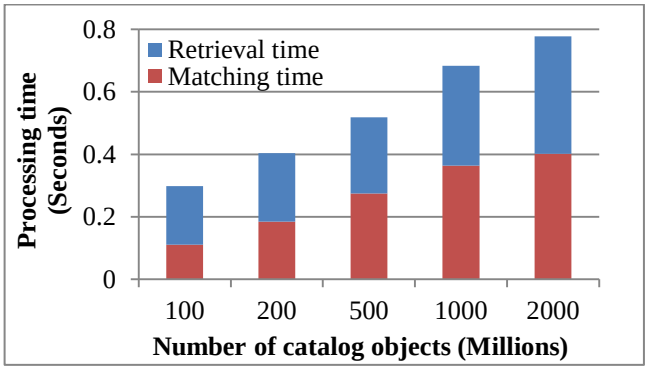

Figure 4. Processing time to match a 2.5 × 2.5 degree image with 100 thousands objects. We show the dependency of the running time on the catalog size.

4. RELATED WORK

Scientists from the database community have developed several tools for organizing astronomical data [1]. For example, MySQL and PostgreSQL both support R-tree [2] spatial indices. However, the scalability of current off-the-shelf solutions is limited. Experiments show that it takes over two thousand seconds to index 50 million objects. Furthermore, the database approach requires 5 GB to store those 50 million objects, while our solution takes only 500 MB on the same dataset.

5. REFERENCES

[1] Andrea Baruffolo. R-trees for astronomical data indexing. Astronomical Data Analysis Software and Systems VIII, ASP Conference Series, 172, 1999.

[2] Antonin Guttman. R-Trees: A dynamic index structure for spatial searching. In Proceedings of the ACM SIGMOD International Conference on Management of Data, pages 47–57, 1984.

[3] Barry Lasker *et al*. The second-generation Guide Star Catalog: Description and properties. The Astrophysical Journal, 136, pages 735–766, 2008.