A Spatial Data Handling System for Retrieval of Images by Unrestricted Regions of User Interest

Erik Dorfman
Hughes STX
NASA/Goddard Space Flight Center
Greenbelt, MD
dorfman@nails.gsfc.nasa.gov

Robert F. Cromp
Code 934
NASA/Goddard Space Flight Center
Greenbelt, MD
cromp@nails.gsfc.nasa.gov

Abstract

The Intelligent Data Management (IDM) project at NASA/GSFC has prototyped an Intelligent Information Fusion System (IIFS), which automatically ingests metadata from remote sensor observations into a large catalog which is directly queryable by end-users. The greatest challenge in the implementation of this catalog has been supporting spatially-driven searches, where the user has a possibly complex region of interest and wishes to recover those images that overlap all or simply a part of that region.

A novel spatial data management system is described, which is capable of storing and retrieving records of image data regardless of their source. This system has been designed and implemented as part of the IIFS catalog. A new data structure, called a ***hypercylinder***, is central to the design. The hypercylinder is specifically tailored for data distributed over the surface of a sphere, such as satellite observations of the Earth or space. Operations on the hypercylinder are regulated by two expert systems. The first governs the ingest of new metadata records, and maintains the efficiency of the data structure as it grows. The second translates, plans, and executes users' spatial queries, performing incremental optimization as partial query results are returned.

1 Introduction

1.1 Needs of the scientific community

With the planned launching of the Earth Observing System (EOS) platforms and with the continuing generation of data by existing missions such as the Hubble Space Telescope (HST), NASA faces one of its greatest challenges yet: the cataloging of remote-sensor data in a manner that will allow users from a variety of scientific disciplines to quickly recover datasets of interest from the vast, constantly-expanding archive.

The ability to query or browse large catalogs of image data by the spatial characteristics of desired datasets is involved in solving what is referred to as the ***spatial data handling*** problem. Whether such a catalog contains downward-looking images of the Earth or outward-looking images of space, the spatial data structures resident in the catalog must support two basic spatial search operations required by the general scientific community (see Figure 1):

- *Window query*: given a region of interest, find all images that *overlap* the region.
- *Containment query*: given a region of interest, find all images that *completely contain* the region.

There is also a simple case of these queries, whose use is sometimes convenient:

- *Point query*: given a point of interest, find all images that overlap the point.

In addition, users require the ability to combine the above operations into more complex spatial queries via the operators AND, OR, and NOT.

1.2 Problems with existing approaches

Most attempts at spatial data handling in data catalogs encounter major difficulties from the start because the catalogs are implemented using relational database (RDB) packages. RDBs generally do not support data structures for handling anything other than linearly-ordered records. The object-oriented database (OODB) research of recent years provides a means of implementing spatial data structures directly inside data catalogs, and

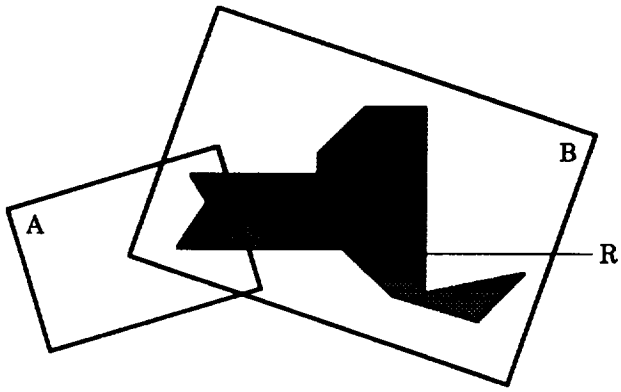

Figure 1: Images A and B both satisfy a window query on the shaded region R, but only image B satisfies a containment query on R.

OODB technology has thus been utilized in implementing the spatial data management system described in this paper.

Some catalogs circumvent most spatial data handling problems by virtue of only having to deal with queries involving a single instrument. By using information about the orbit of the instrument's platform, spatial queries are mathematically converted into sets of path-row coordinates that specify images satisfying the query, and these coordinates are used as the search keys for the images. The problem with this approach is the lack of both extensibility and flexibility. First, metadata from new platforms and instruments cannot be added without simultaneously authoring new spatial-search software. Second, images with identical path-row coordinates might not have identical locations due to fluctuations in the orbit of the platform, so a path-row-based spatial query system may falsely accept or reject images during a query.

Even catalogs that employ robust spatial data handling techniques encounter difficulties because they actually treat the globe not as a sphere but as a *planar surface*, a consequence of employing spatial data structures that use latitude-longitude based coordinate systems. The problem is that the surface of a sphere cannot be mapped onto a plane without introducing discontinuities and considerable distortion near the poles, as is evident in most cartographic projections. When using planar spatial data structures (such as quadtrees or k-d trees) to represent an inherently spherical domain, these anomalies present major difficulties in query processing and often result in an "unbalancing" of the data structure, leading to an overall drop in performance during search.

1.3 The application

The Intelligent Data Management group is conducting research into the development of data management systems that can handle the archiving and querying of data produced by Earth and space missions. Several unique challenges drive the design of these systems, including the volume of the data, the use and interpretation of the data's temporal, spatial, and spectral components, the size of the userbase, and the desire for fast response times.

The IDM group has developed an Intelligent Information Fusion System (IIFS) for testing approaches to handling the archiving and querying of terabyte-sized spatial databases (see Figure 2). Major components of the system are the mass storage and its interactions with the rest of the system [Camp91]; the real-time planning and scheduling for processing the data [Short91]; the extraction of metadata and subsequent construction of fast indices for organizing the data along various search dimensions [Camp89] [Cromp91] [Dorf91]; and the overall user interface.

The IIFS design is novel in a number of areas. Semantic data-modeling techniques are used to organize the mass storage system to reduce the transfer times of the data to on-line devices and the mechanical motions of the supporting robotics. Data percolates from near-line mass storage to on-line disk storage based upon its frequency of use. A combination of neural networks and expert systems defines how metadata is extracted to build up search indices to the underlying database. The metadata itself is organized in an object-oriented database which has special data structures for representing the multiple views of the data (such as temporal, spatial, spectral, project, sensor) without resorting to multiple copies of information. A special data structure that maps directly between the Earth and a sphere organizes the data for efficient spatial querying. The user interface is configured dynamically at run-time depending on the scientist's discipline and the current knowledge in the object database.

Experimentation with the IIFS design and implementation have shown that greater flexibility is needed in the spatial data handling routines so that images with a variety of coverage and orientation can be uniformly retrieved with respect to a user's region of interest. The remainder of the paper discusses the enhancements that have been made to the IIFS spatial data structures and describes an overall spatial data handling system that combines declarative and procedural knowledge for efficiently managing spatial queries.

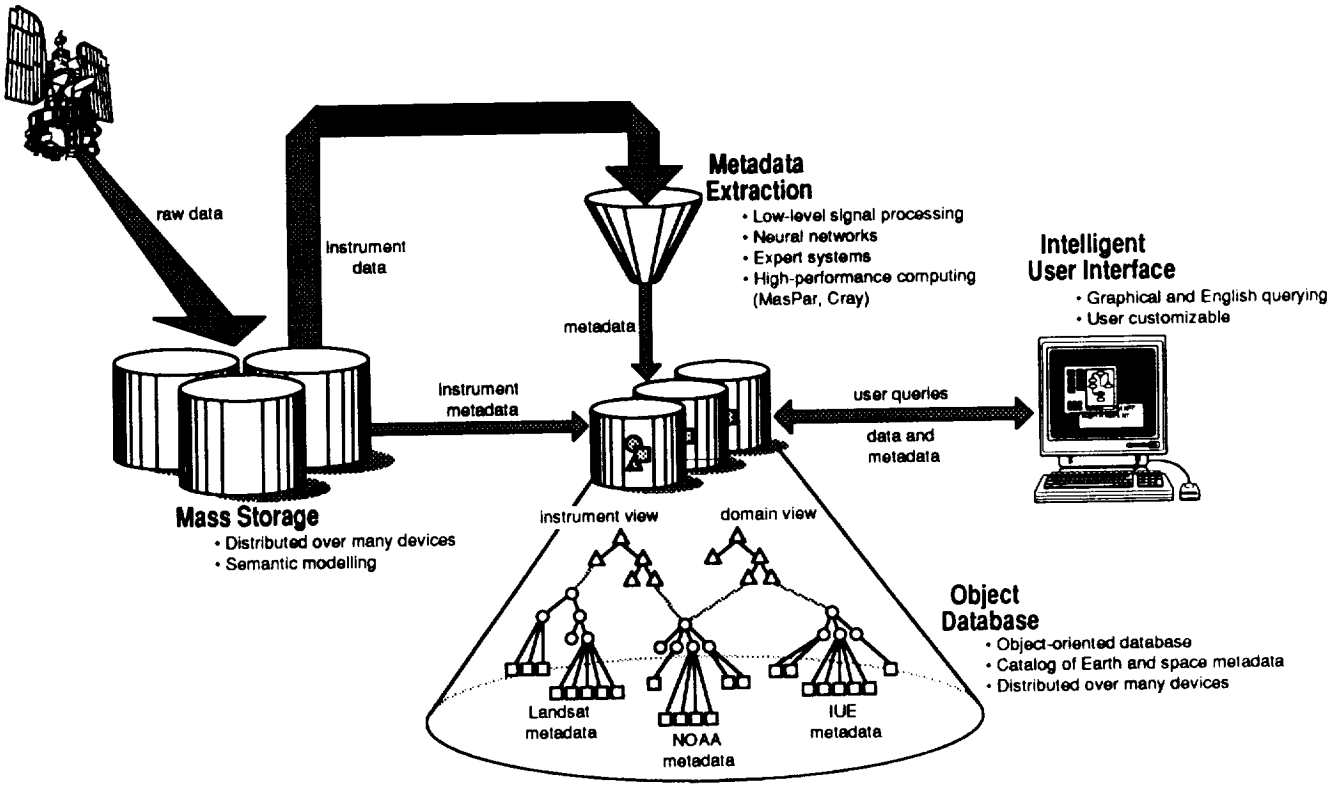

Figure 2: The high-level architecture of the Intelligent Information Fusion System.

1.4 A solution

A design for a spatial data structure suitable for a large, heterogeneous image database with global coverage must account not only for the goals of Section 1.1, but also the difficulties introduced by the richness of the remote sensing domain:

- *Multiple image orientations,* due both to different satellite orbits, and because there is no such thing as "fixed orientation" on the surface of a sphere.
- *Multiple image shapes,* due to the variety of sensors, the tilt of the individual spacecraft, and the alteration of the image border by geometric correction.
- *Multiple image sizes* in terms of the extent of the image boundaries on the surface of the sphere: e.g., sensors mounted on airplanes have smaller fields of view than similar sensors mounted on orbiting platforms.

The data structure described in this paper, together with the supporting expert systems for ingest and querying, addresses all these concerns. The result is a *spatial data handling system* which can handle NASA's next generation of image catalogs.

2 Simplifying spatial queries by a transformation scheme

2.1 The general concept

A variety of spatial data handling problems in complex spatial domains can be solved by mathematically transforming the domain D into a new domain D' where the corresponding queries can be handled more efficiently [Same90, p. 186]. Such transformations map a complex object in D (in this case, an image) into a single point in D': this point is referred to as the object's *representative point.* We are then left with the simpler goal of designing a data structure that can handle the storage and query of points rather than arbitrary shapes. Two difficulties with this approach can be encountered:

- A query region R in D must be transformed into its equivalent R' in D', and R' may be difficult to generate or to calculate with, even for simple R.
- The transformation may result in some loss of information about the stored objects, so that additional computation may be needed to exactly satisfy a spatial query.

These difficulties are dealt with in Section 4, where the implementation of the data structure is described.

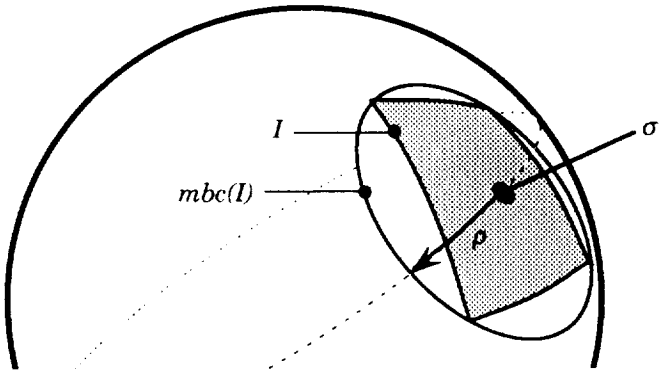

Figure 3: The minimal bounding circle of an image on the globe. Note that the radius is measured along a great-circle arc, like all distances on the surface of a sphere.

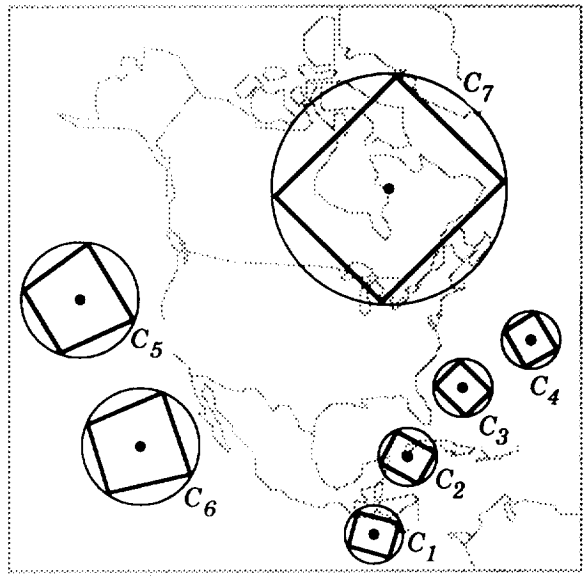

Figure 4: A portion of D, showing a group of images and their minimal bounding circles.

2.2 A transformation scheme for image data

In order to transform images into points, we discard the actual boundaries of the image and concern ourselves only with its minimal bounding circle, which we shall call the *representative circle* for the image (Figure 3). This is closely related to the approach taken by [Oost90], which takes the minimal bounding circles of objects on a planar surface instead of on a spherical surface. Note that the "representative circle" approach eliminates the problems of multiple image shapes and orientations.

By treating images as circles, we are able to describe every image by only two parameters: the location of the circle's center, which we shall denote as σ, and the radius of the circle, which we shall denote as ρ. Thus, every image can be treated as a simple point $\langle \sigma, \rho \rangle$. Under the terminology of [Hinr83], σ is the point's *location* parameter, and ρ is the point's *extension* parameter.

2.3 Visualizing the transformation

Consider a part of the globe over which several images have been taken, shown in Figure 4. For illustration purposes we will show only a small part of the globe so that it may be rendered as a simple plane, although it must be stressed that what is actually being shown is a portion of a curved surface. This would represent a scenario in D.

To map this scenario to D', we compute for each image I_i the center σ_i and radius ρ_i of its minimal bounding circle, and plot the resulting point $\langle \sigma_i, \rho_i \rangle$ in D' as shown in Figure 5.

To more compactly illustrate what the space of D' looks like, we must make some diagrammatic simplifications. Figure 6 shows how the surface of a sphere can be mapped onto the perimeter of a circle by means of a *space-filling curve*. This is a single curve that begins in the diagram at point A, passes through every point B, C, D, etc. on the sphere, and eventually returns to A (also labeled Z in the diagram). The curve places an ordering on the points: A is before B, B is before C, etc., and this ordering enables us to place every point on the sphere's surface onto the perimeter of the circle below. Note that points which are close to each other on the circle (like B and C) correspond to points which are close to each other on the sphere.

By using this mapping, the scenario of Figure 5 is depicted again in Figure 7. Here, D' is shown as the surface of a cylinder: the position on the vertical axis represents the ρ value, and the position along the circular perimeter represents the σ value.

Since individual sensors can be expected to produce large numbers of images of the same size, we expect the distribution of representative points for a large, heterogeneous image database not to be uniform, but instead to be concentrated in different strata along the ρ axis (Figure 8).

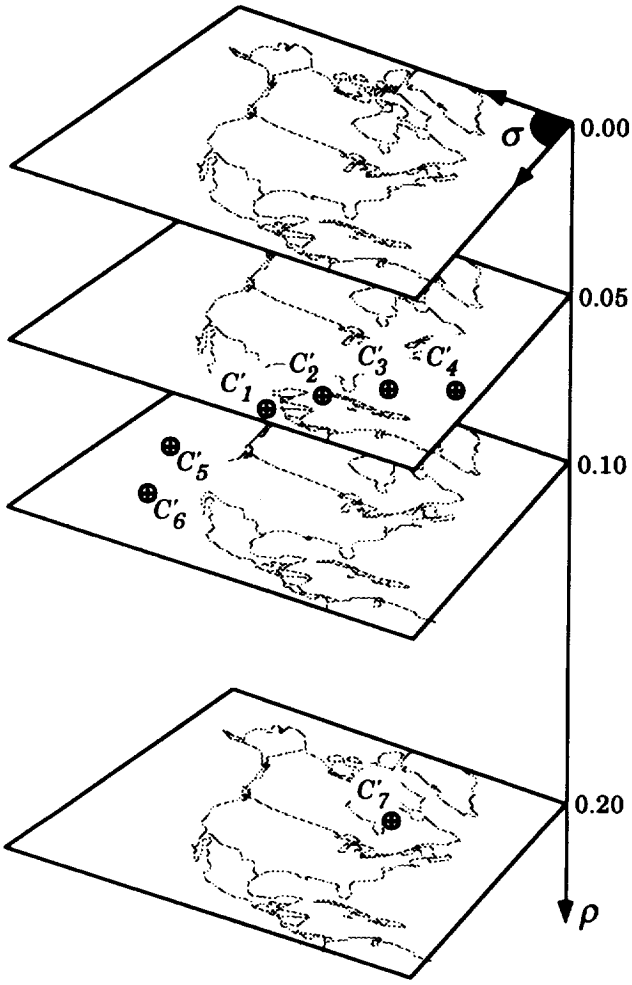

Figure 5: The representative points of the images in Figure 4, plotted in a portion of D'.

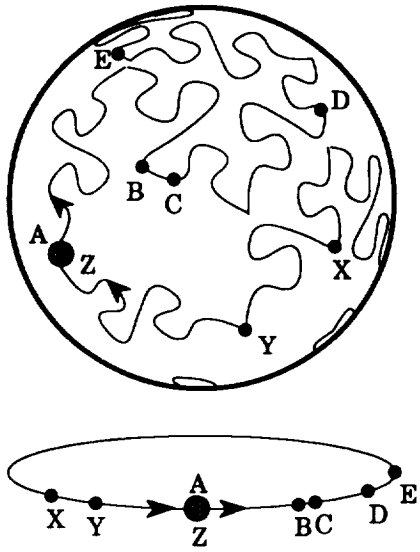

Figure 6: How the surface of a sphere (above) can be mapped onto the perimeter of a circle (below) by using a space-filling curve $A, B, \ldots, Y, Z$. For clarity, the curve on the sphere is not shown in its entirety.

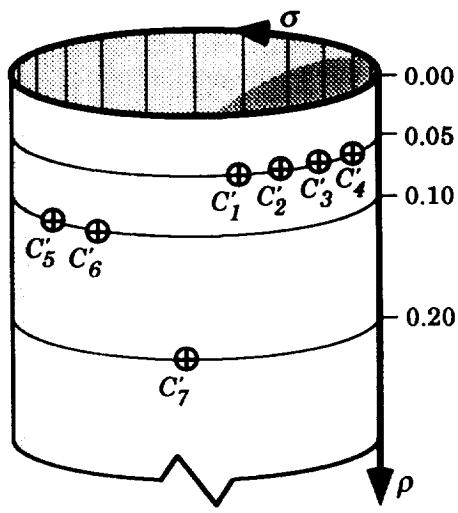

Figure 7: The same representative points as in Figure 5, this time plotted on a cylinder to represent D' more compactly. Every circular cross-section of this cylinder represents the entire surface of a sphere (the globe).

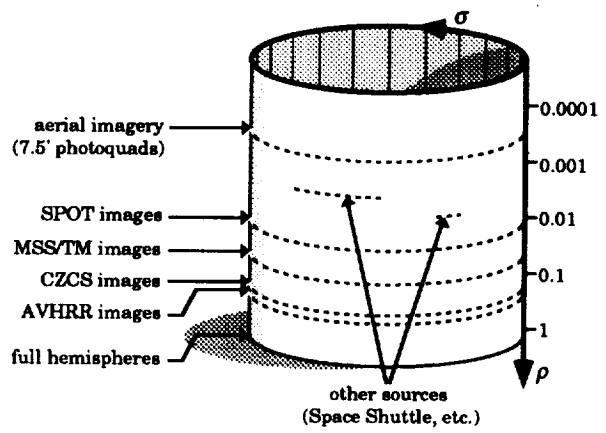

Figure 8: The expected distribution of representative points in D'. For convenience, ρ is shown on a logarithmic scale.

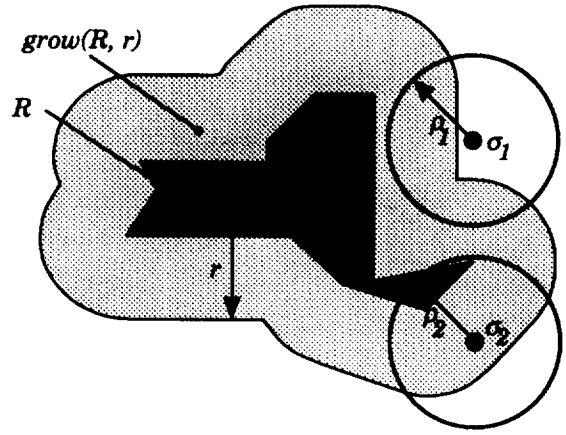

Figure 9: Criteria for a representative circle to overlap R. Both circles have the same radius, $\rho_1 = \rho_2 = r$, but different locations.

3 Processing queries in the transformed space

3.1 Processing window queries

Given a query region R on a sphere, we note that the further the center of a circle C is from R, the larger the radius of C must be if C is to overlap R. Let $grow(R, r)$ denote the locus of all points that are within a distance of r from R: read this as "grow R by radius r" A sample R and $grow(R, r)$ are depicted in Figure 9. We observe:

> A representative circle $C_i = \langle \sigma_i, \rho_i \rangle$ overlaps a region R if and only if its center σ_i falls inside $grow(R, \rho_i)$.

This rule is demonstrated in Figure 10, which depicts in D a query region R, the representative circles for four images $C_1 \ldots C_4$, and the region $grow(R, \rho_i)$ for the various image radii ρ_i. Note that:

- σ_1 is not inside $grow(R, \rho_1)$, and σ_3 is not inside $grow(R, \rho_3)$. Therefore, neither C_1 nor C_3 overlap R.
- σ_2 is inside $grow(R, \rho_2)$, and σ_4 is inside $grow(R, \rho_4)$. Therefore, both C_2 and C_4 overlap R.

Now, consider Figure 11. It depicts in D' the representative points $C_i' = \langle \sigma_i, \rho_i \rangle$ for the images in Figure 10. For each point the corresponding region $grow(R, \rho_i)$ has been plotted, on the same cross-section of D' where C_i' resides. Notice that, if $grow(R, r)$ had similarly been plotted for all r in ρ, the regions would trace out a cone-like solid in D'. In terms of formulating window queries in D', this means that:

> An image's circle in D overlaps a region R if and only if its representative point in D' falls within the cone in D' whose cross-section at $\rho = r$ is $grow(R, r)$.

If the region R is a single point p, then this becomes the definition for a point query, where $grow(p, r)$ is simply a circle with center s and radius r.

[Same90, pp. 187-192] observes that, when employing transformation schemes which represent stored objects by points that have distinct location and extension parameters, window queries and containment queries generally produce cone-like search regions. This is also true of the model described above (Figure 12), and for this reason we refer to the search regions in D' as *search cones*.

3.2 Processing containment queries

Up to this point we have dealt with window queries, which produce cone-like search spaces in D'. A containment query's search space is also a cone-like region, but differing in the way a cross-section of the cone is defined for a given value r of ρ: instead of its being the locus of all points p such that *any* point of R is within a radius of r from p, it is the locus of all points p such that *all* points of R are within a radius of r from p. Call this cross-section $cover(R, r)$.

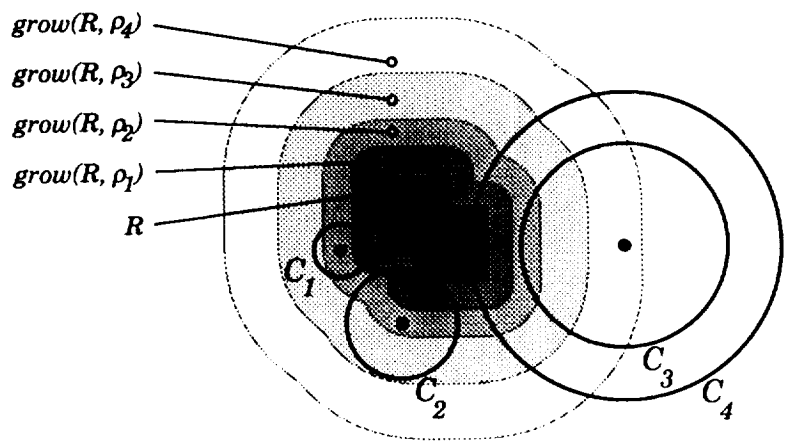

Figure 10: Four representative circles, as they would appear in D. Also shown are $grow(R, \rho_i)$ for each circle C_i.

Whereas $grow(R, r)$ is relatively easy to compute even on a sphere, $cover(R, r)$ is more complex. But, as it turns out, we need never compute $cover(R, r)$ directly to process queries.

To begin, notice that since an image with radius $\pi/2$ is a full hemisphere, we need not concern ourselves with images where $r > \pi/2$. It can be shown that for all $r \leq \pi/2$, if all the *vertices* of R are within a radius of r from a given point p, then the edges between those vertices are completely within a radius of r from point p, and thus *all* of R is within a radius of R from point p. So $cover(R, r)$ is actually the locus of all points within a radius of r from every vertex of R. Therefore, if R has n vertices, $cover(R, r)$ is the intersection of n circles of radius r whose centers are at the vertices of R (Figure 13).

Let R have vertices $v_1 \ldots v_n$. In terms of containment queries, this means that:

> An image's circle in D completely contains a region R if and only if its representative point in D' falls inside *all* the search cones $S_1 \ldots S_n$, where the cross-section of S_i at $\rho = r$ is $grow(v_i, r)$.

The search cone for the containment query can thus be defined as the intersection of n search cones: the search cones for the point queries on the n vertices of R (Figure 12).

4 The hypercylinder data structure

4.1 Design issues arising from implementation details

At the core of the spatial data handling system is the data structure that stores and retrieves points in D', named the *hypercylinder* because of the shape of the transformed space. To ensure that it is capable of efficiently processing queries in D, we must consider factors that place practical limitations on how the corresponding search regions in D' can be manipulated.

Although the cross-section of a search cone at a given value of ρ is easy to generate, computations involving the cone itself require a great deal more processing. Therefore we shall handle queries by dividing the search cones into cross-sections that may be dealt with individually (Figure 11 provides an illustration of "slices" of a search cone). The ramification for the data structure is that D' must be represented internally as a collection of slices that can be queried independently.

Since we must still compute cross-sections for each slice of the data structure, we need to divide D' into a manageable number of slices. If slices are infinitely thin (i.e., the data points in a given slice all have the same ρ value), then even small variations in the extents of images will result in a need for a large number of slices. We thus let each slice cover a range of ρ values.

To execute a query (Figure 14), we handle one slice of the search cone at a time, and then merge the results together to form the final result. For each slice, we com-

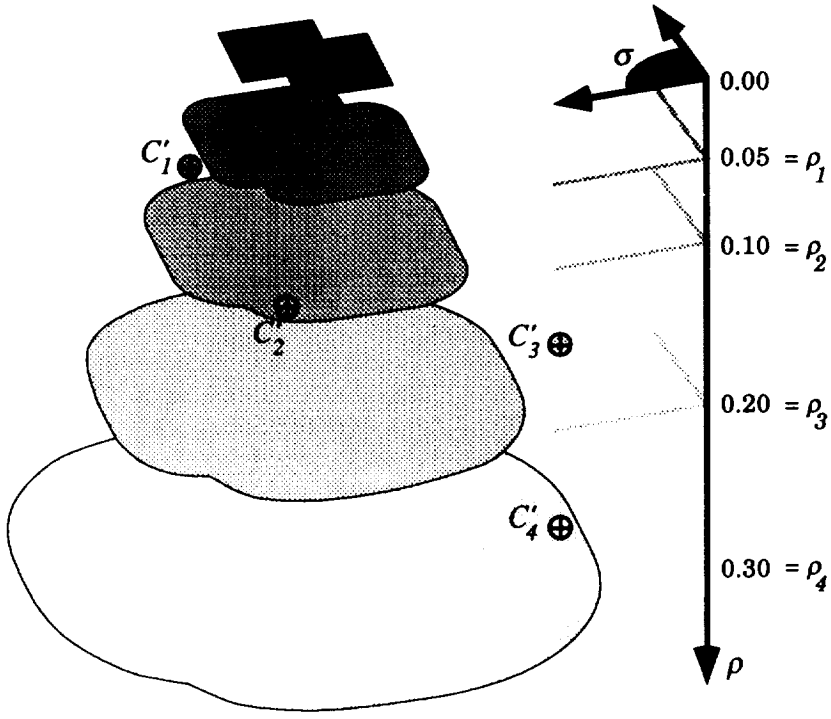

Figure 11: The representative points for the circles in Figure 10, as they would appear in D'. Also shown are $grow(R, \rho_i)$ for each circle C_i'.

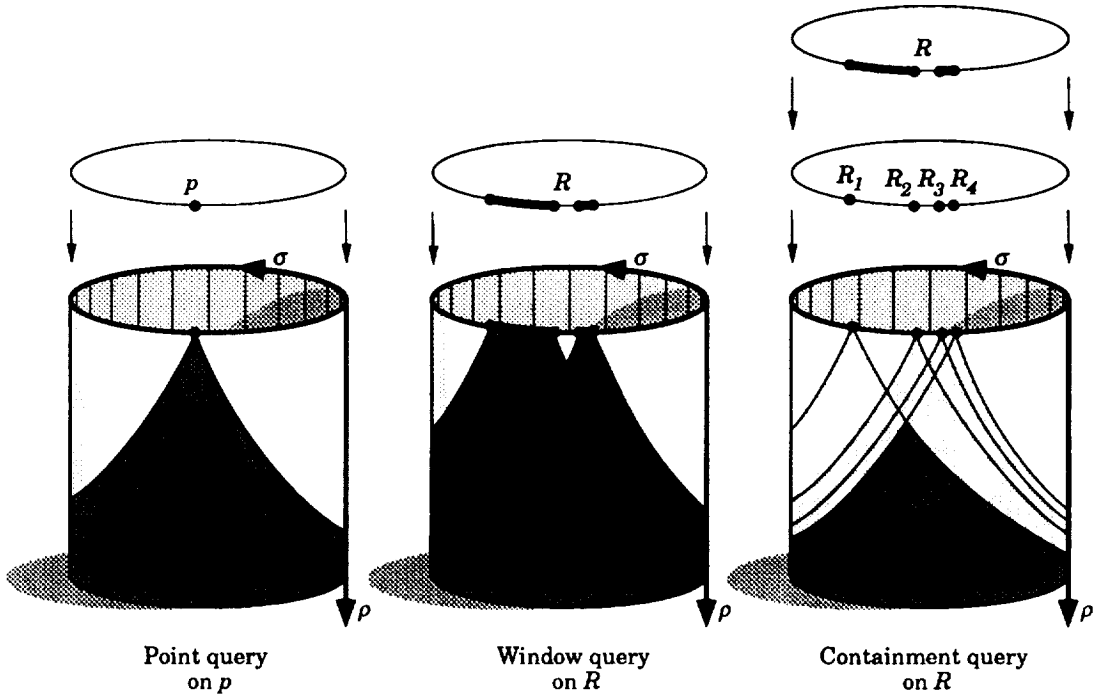

Figure 12: The search cones (shaded) in D' for a point query, window query, and containment query. Notice that the search cone for the containment query is the intersection of four search cones for point queries.

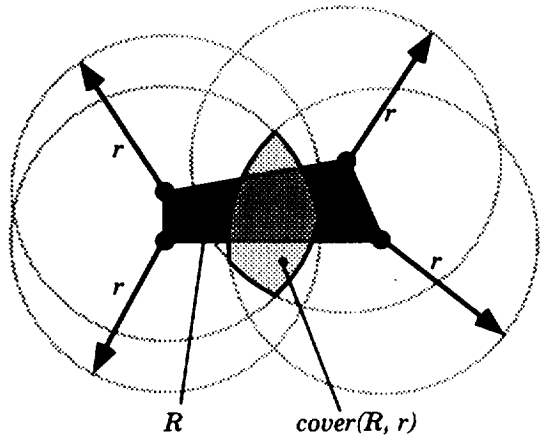

Figure 13: A region R and *cover*(R, r). Any circle of radius r in *cover*(R, r) will overlap all points in R.

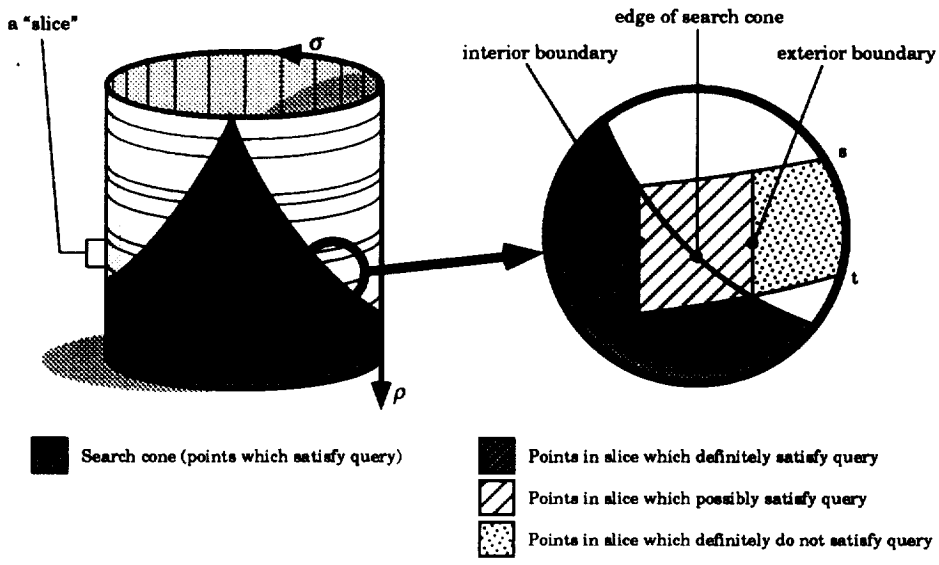

Figure 14: Slicing up D', and approximating the portion of a search cone inside the slice from $\rho = s$ to $\rho = t$. By computing the interior and exterior boundaries of the search cone in that slice, we can divide the points in that slice into three groups – those that definitely satisfy, possibly satisfy, and definitely do not satisfy the query.

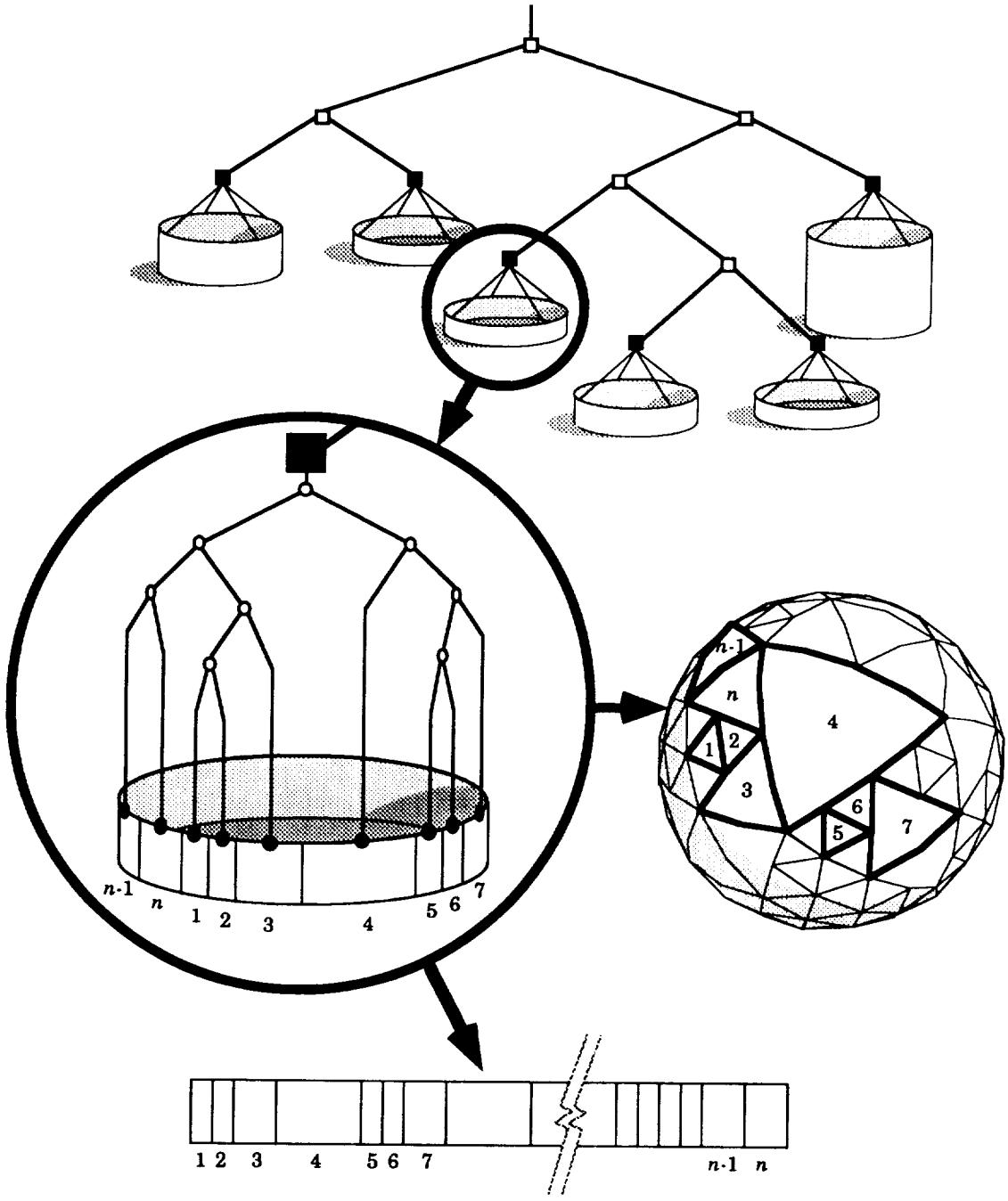

Figure 15: The hypercylinder data structure. Square nodes belong to the BST for ρ, circular nodes belong to the SQTs for σ, white nodes are internal nodes, and black nodes are leaf nodes. A close-up of one slice is depicted, with n leaf nodes in its SQT. Also shown is how some of the SQT leaf nodes (numbered) might look if the surface of the slice were "unrolled" (bottom), and how the corresponding trixels might look on the surface of a sphere (right).

pute *two* cross-sections of the query's search cone: one where the cone passes through the top of the slice, and one where it passes through the bottom of the slice. These two cross-sections give us, respectively, the interior and exterior boundaries of the search cone as it passes through the slice. We observe that:

- Data points from inside the interior boundary are *definitely* inside the search cone.
- Data points from between the interior and exterior boundaries are *possibly* inside the search cone, and must be tested on an individual basis.
- Data points outside the exterior boundary are definitely outside the search cone.

Notice that the thicker the slice, the greater the difference between the interior and exterior boundaries, and hence the more data points we can expect to have to test in this region – which we shall call the "possibly-satisfy" region - during a query. To maximize query efficiency, we must slice up the hypercylinder so that areas of D' with many data points are sliced thin, while areas of D' with very few data points may be covered by thick slices since fewer points will need to be tested in those areas. As revealed in Figure 8, we expect the representative points for images to be largely concentrated in different "strata" of D'. Unfortunately we cannot predict where all such strata will eventually lie, due to the continuous launching of new sources of image data.

As the number of points within a slice grows, eventually the density of points within that slice is such that excessive time is spent deciding whether to accept a point within the "possibly-satisfy" region. At this time, the slice must be split so that the collection of points within the subslices is more homogeneous. A heuristic approach to recognizing when this division should occur and where the division should be made is given in Section 5.1.

4.2 The data structure design

For an overview of the hypercylinder's design, refer to Figure 15. The top-level view is a binary search tree (BST), whose branches discriminate between values of ρ and whose leaves are the slices of D'. The data structure at each leaf is a sphere quadtree (SQT) [Feke84] [Feke90], a special variation of a quadtree designed for storing and retrieving points distributed on the surface of a sphere. The branches of a SQT discriminate between values of σ and the leaves represent triangular regions of the globe (Figure 16). The representative points of images are stored in the leaves of each SQT.

The sphere quadtree is a unique data structure in that it models the globe without introducing distortions

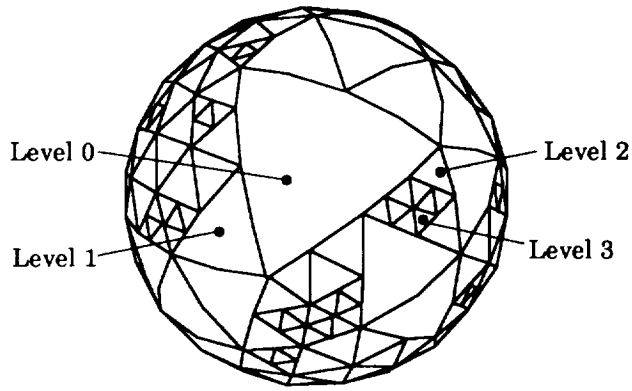

Figure 16: How a sphere quadtree divides the globe into triangular patches (called *trixels*). The higher the level number of a trixel, the deeper in the tree it is, and the smaller the area it covers.

or discontinuities, as other approaches such at latitude-longitude based schemes do (see Section 1.2). Conceptually, it divides the sphere into twenty identical equilateral triangles called *trixels*, where each trixel is a "bucket" for data points. When a trixel reaches its threshold number of data points, it is split into four nearly equal-area subtrixels. This subdivision is called *refinement* since it produces smaller trixels which, like the pixels in an image, can represent regions to a higher degree of resolution. As with most spatial data structures, refinement in a SQT can continue indefinitely: the result is that areas of the globe that are densely populated are more refined, so query regions in those areas are more accurately represented by the higher resolution trixels in the SQT.

Since satellite orbits generally provide global coverage, we expect the SQT for each slice to be fairly equally refined over most of the surface of the globe, i.e., the SQT is well balanced, and so spatial queries are handled with similar efficiency regardless of their location. But since satellite orbital paths are often designed to produce fully-overlapping images at each pass over a location, a clustering of the representative points occurs and results in a tree that is *globally* well-balanced, but *locally* unbalanced. A means for overcoming this problem exists, and is discussed in Section 7.1.

To ensure that the slices are split in an optimal manner when they achieve their threshold number of data points, a *profile* of how the data points are distributed in each slice is maintained. These profiles are used as heuristic devices to determine where the slices should be

subdivided. Their actual implementation is described in Section 5.1.

5 A spatial data management system

The primary motivation for designing an entire spatial data management *system* for the remote sensing domain, as opposed to just the custom-tailored spatial data structure described above, is that domain knowledge can often be employed to improve the overall performance of *any* data management scheme. For a remote-sensing catalog, such knowledge encompasses:

- *Model information:* what real-world entities and concepts (observations, sensors, geographic regions, scientific parameters, classification schema) are represented in the catalog, and what sorts of questions may be asked about them by end-users. This is mostly declarative information, intended for use by both the end-users and the system. It allows the users to ask the system about its contents, and it enables the system to translate users' natural-language and graphical queries into the system's internal representation.

- *Data structure information:* what data structures (e.g., the hypercylinder) exist in the database, under what conditions they should be used (e.g., spatial queries), and how data is distributed in them (e.g., the profiles mentioned in Section 4.2). This procedural and declarative metaknowledge is used by the catalog to construct plans for queries, to generate the necessary calls to the catalog's underlying database management system, and to optimize the query plans as intermediate results are returned.

- *Operational information:* the performance of the hardware devices over which the database is distributed, the anticipated system loads over the course of a typical day or week, the types of queries most frequently made, etc. This is largely declarative information, used by the catalog in performing a variety of tasks ranging from query optimization to automatic data structure reorganization.

The spatial data management system makes use of such information in its two supporting expert systems: the Spatial Ingest Expert System and the Spatial Query Expert System, both of which are discussed below.

5.1 The Spatial Ingest Expert System (SIES)

The primary function of the SIES is to govern the splitting of the slices of the hypercylinder, ensuring that each slice is divided so that dense strata of D' end up in thin slices, with thick slices covering the sparser expanses of D'. As mentioned in Section 4.2, each slice maintains a profile of the *current* distribution of the data points in the slice. In addition to this, the SIES incorporates information about the expected *future* distribution of points. Both provide a heuristic means of optimizing the hypercylinder as it is being built.

The primary requirements for profiles are that they must be easy to update during ingest, be implemented to allow rapid calculation during splitting, be large enough to adequately capture the distribution of points in a slice, and be small enough not to incur a large storage overhead. We have experimented with an approach based on incremental sampling of the representative points as they are stored in the slice: the profile consists of a small reservoir of the ρ values of sampled points, and as each new point is ingested there is a chance that one of the current elements of the reservoir will be replaced by the ρ value of the new point. During splitting, the profile is analyzed to determine where the ρ values are clustered, indicating emerging strata in D'.

The profiles are supplemented in the knowledge base by the *bias list*: a list of strata into which D' is expected to be organized. The bias list is updated whenever a new instrument is added to the knowledge base, and contains the expected minimum and maximum radii of images that the instrument will generate plus an estimate of how many images will be generated over the lifetime of the instrument. Although the bias list can supply information on where a slice is best split (or even whether to defer splitting a slice), its contents do not reflect the actual state of the data structure, and thus neither it nor the profiles are expected to provide maximal performance in isolation.

The hypercylinder initially consists of a single slice covering all of ρ. When the number of points stored in any slice reaches the threshold value for splitting, a strategy for dividing the slice is formulated from one of several alternatives, such as:

- Place the largest cluster into its own slice, and the spaces to either side of this cluster into two additional slices (Figure 17).

- Place the largest expanse of sparsely-populated space into its own slice, and the spaces to either side of it into two additional slices.

- Given an entry in the bias list whose minimum and maximum radii fall within the slice and whose esti-

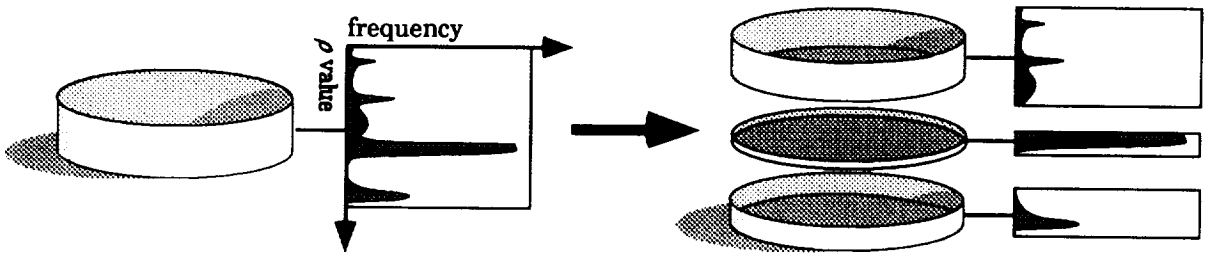

Figure 17: One strategy for splitting a slice, based on the distribution of points in the slice: this "fences in" the largest cluster of points, putting it into its own slice.

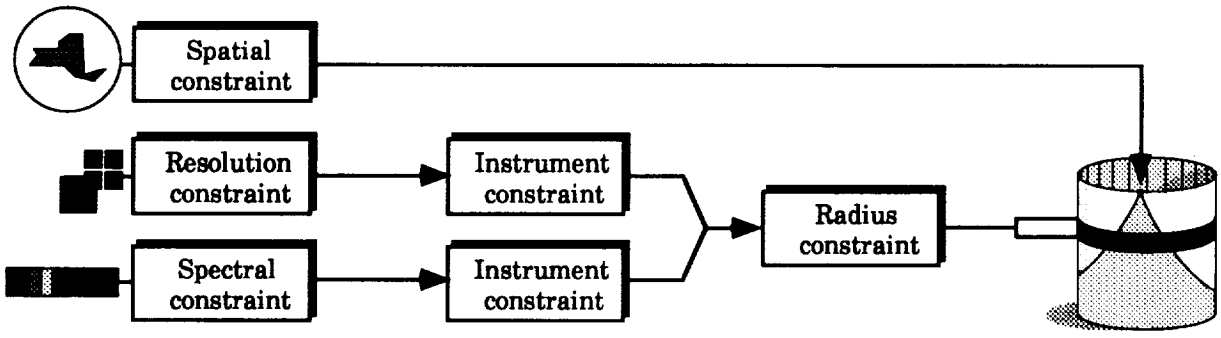

Figure 18: How non-spatial components of a query can place implicit spatial constraints. Only the shaded layers of the hypercylinder must be searched.

mated number of images is high, place that range of values into its own slice and the spaces to either side of it into two additional slices.

- Split the slice so that equal numbers of points are in each subslice.

The strategy chosen depends on factors such as how definite the clusters are, how widely they are distributed, and whether one cluster is significantly larger than the others. Each fact lends weight to one or more of the strategies during selection: these weights are then adjusted as more is learned about the performance of the SIES under the real-world environment.

5.2 The Spatial Query Expert System (SQES)

The SQES is actually a conceptual subset of the larger Query Processing Expert System, a mostly-procedural-knowledge base whose content is the model and data structure information described in Section 5, and whose purpose is the translation, optimization, and execution of user queries. The SQES handles those parts of the task that relate to spatial searches.

The first place the SQES is invoked is during the parsing of queries with symbolic spatial components. In natural-language and menu-driven queries, such components might appear as the names of geographic, political, or climatological regions on the Earth (or as the names of stellar objects or constellations, depending on the catalog type). The SQES translates these terms into geometric region descriptions, possibly invoking external information sources in the process, such as databases that house geopolitical boundaries, or that store the names of astronomical entities under different labelling schemes to allow translation from one scheme to another (e.g. the SIMBAD database).

Figure 18 shows how the SQES can optimize the spatial search by inferring additional spatial constraints from the user's query. The user's specification of desired ranges of image resolution and spectral bands constrains the set of instruments that might be sources of the desired data, which in turn limits the possible sizes of images that can be returned from the user's query, which in turn pinpoints the only slices of the hypercylinder that need to be searched.

The SQES also assists in planning complex spatial queries, where the order in which subparts of the query are executed can play a dramatic role in decreasing processing time. Consider the processing of a containment query: as noted above, containment queries are best handled as the intersection of a collection of point queries. Throughout the system, computing the intersection of a group of unknown sets is performed in a strategic manner: the members of the group are retrieved sequentially, from smallest to largest estimated size, and the most recently retrieved set is intersected with a running "result" set. The system stops and returns the empty set as the result of the intersection if any of the retrieved sets is the empty set.

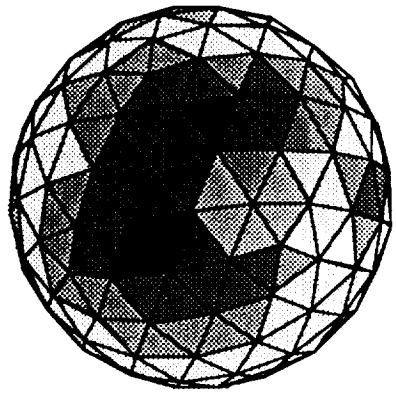

Figure 19: The profile used by the SQES. Darker trixels indicate that observations are more dense in those portions of the globe.

To allow the SQES to estimate the relative sizes of sets returned by the components of a spatial query, yet another profile is kept in the knowledge base. Whereas the previously-discussed profiles represent the distributions of the image *radii*, this new profile represents the distribution of the image *centers* on the globe, giving in effect the "density" of observations around the surface of the globe. Since it associates spherical locations with density values, this profile is implemented as a small spherical quadtree (Figure 19). When the SQES is confronted with a set of query regions that must be ordered by expected content, the area of each region is computed and multiplied by its average density from the profile to produce an estimate of the number of data points in the entire region, and it is by this estimate that processing order is determined. We intend to install similar profile-based approaches for aiding the construction of query plans on top of *all* the catalog's principle data structures.

6 Results

The spatial data handling system has been tested using a portion of the metadata stored in the Pilot Land Data

System (PLDS) catalog. Approximately 3,000 records of TM, MSS and AVHRR metadata were ingested from a flat file into the hypercylinder's SIES via a C program which extracted the appropriate fields for each image's location, boundaries, and primary key. To assess the performance and extendability of the SIES under different implementation languages, it was written both in Quintus Prolog and in CLIPS, an expert system shell developed by NASA's Johnson Space Center and capable of being linked into a C program and accessed via simple function calls. The SIES sends the appropriate ingest requests to the hypercylinder "server": a C++ program containing the hypercylinder data structure, accessible through a TCP/IP socket on a Sun-4. All of the spatial search routines, as well as the profiles for the hypercylinder's slices, were implemented in C++ and reside in the server. Query requests are sent to the SQES, a CLIPS/C module that uses a small, array-based version of the SQT (called a *linear SQT*) to store the SQES profile. The SQES performs the necessary query planning and sends the various partial spatial query requests to the hypercylinder server, which keeps track of execution times for various tasks.

The spherical quadtree components of the system and the supporting spherical geometry routines have been implemented and tested independently inside the IIFS catalog's database. The catalog uses the Smalltalk-based GemStone DBMS, a commercial object-oriented DBMS available from Servio Logic Corporation, which is capable of invoking external C and C++ functions.

The rationale for initially implementing the full hypercylinder as an in-core data structure rather than inside the catalog database was twofold. First, it enabled us to seamlessly integrate the hypercylinder with the C++ spherical geometry objects (such as query regions) and routines (such as *grow()*) necessary for spatial query handling. We found C++ to be an excellent programming platform for rapid data structure prototyping, and are currently using Oregon C++ from Oregon Software, which conforms to the base ANSI documents for this language and thus should produce highly portable code. Second, initial implementation and testing in core enabled us to take CPU-time measurements without concerning ourselves with the I/O and CPU overhead that would be introduced by interfacing with a DBMS.

The *grow()* routine performed well for any given radius: the algorithm is $O(n)$, where the inputs are the n vertices of the query region R and a radius of expansion r, and the outputs are the m vertices of $grow(R, r)$, where $n < m < cn$ for a predefined constant c. Unfortunately, the grown query region is almost always self-overlapping, and some necessary computations (such as determining whether a point is inside $grow(R, r)$) take $O(n^2)$ to process using our current algorithms. Removing the self-intersections from a spherical region appears to be an $O(n^2)$ operation in the best case: we are therefore focusing our attentions on developing more efficient algorithms for manipulating the self-overlapping regions.

7 Future research

7.1 The hypercylinder

The hypercylinder data structure, designed to meet stringent ingest and query requirements for large image catalogs, is nevertheless only one possible data structure and is specifically designed for image data. We hope in the near future to:

- Produce additional spatial data structures customized for efficient storage and retrieval of other types of observations, such as observations in atmospheric domains with additional spatial search criteria such as "altitude."
- Implement these data structures fully inside the catalog's database, ensuring that the hypercylinder's components are clustered so as to minimize page faults during tree traversal.
- Introduce tree compression techniques for the SQTs, as per [Ohsa83], to eliminate the clustering problem mentioned in Section 4.2.

7.2 The Spatial Ingest Expert System

In future implementations, we plan to expand the role of the SIES in the spatial data ingest process. The SIES will be empowered to:

- Periodically survey the data structure for conditions that would compromise efficiency, such as tree imbalance. If such conditions are detected, the SIES must determine how best to reorganize the data structure, and notify the database administrator (DBA) of the problem.
- Estimate the amount of system resources that a reoptimizing step will take, and, based on profiles of system loads, suggest to the DBA the best times for self-correction.
- Maintain a history of major decisions affecting the data structure: when a slice was split and why, when the data structure had to be reoptimized and why, etc. Alert the DBA if it is determined that some subset of the rules has contributed to poor decisions.

7.3 The Spatial Query Expert System

Much of the spatial query optimization is intended to be handled by the catalog's proposed Query Planning and Execution Module (QPEM), in which the SQES knowledge will reside. However, there are still spatial search strategies unique to the SQES that have yet to be explored:

- Transfer more spatial query processing control from the hypercylinder to the SQES. This would involve maintaining a collection of density profiles, each covering a different slice of the hypercylinder. Spatial queries would be handled and optimized independently by each slice of the hypercylinder, based on local profile information.

- Allow the user to specify different levels of spatial query processing. Since many stages of query processing in the data structure divide the tree into three types of branches – definitely satisfies query, possibly satisfies query, and definitely does not satisfy query – the user can be given the power to trade precision for execution time by deciding to either accept, reject, or vigorously test the "possibly satisfies query" branch.

8 Summary and conclusions

The research presented in this paper is intended to serve as the foundation for a new generation of spatial data management systems at NASA, tailored for the general remote-sensing domain and robust enough to support efficient spatial searches regardless of the shape or location of the user's area of interest.

The hypercylinder's two controlling expert systems, the SIES and the SQES, are as necessary as they are novel. By using rule firings in a supervisory expert system to activate data management tasks, the conditions under which different data management strategies are employed can be easily monitored, evaluated, and altered to fine-tune system performance. This is a major step beyond conventional catalog schemes, where inspection and evaluation of the underlying data structures are at best extremely difficult, and adjustment of the associated algorithms is traditionally impossible without down-time for code recompilation.

9 Acknowledgements

The authors would first like to thank William J. Campbell for his continual support of IDM, and for the foresight he has shared with us, which has helped make the IIFS into what we believe to be the finest prototype in existence for NASA's future data management systems. In addition, the research presented in this paper would not have been possible without the inspiration provided by three other researchers: Dr. George Fekete, who developed the spherical quadtree data structure, Nick Short, Jr., whose introduction of planning-and-scheduling technology to the IIFS led to the concept of the SQES, and finally Dr. Samir Chettri, whose expertise in statistics has contributed to the algorithms underlying the SIES.

References

[Camp89] W. J. Campbell, S. E. Hill and R. F. Cromp, "Automatic labeling and characterization of objects using artificial neural networks," *Telematics and Informatics*, Vol. 6, Nos. 3/4, pp. 259-271, 1989.

[Camp91] W. J. Campbell, N. M. Short, Jr., L. H. Roelofs and E. Dorfman, "Using semantic data modelling techniques to organize an object-oriented database for extending the mass storage model," *42nd Congress of the International Astronautical Federation*, Montreal, PQ, Canada, 1991.

[Cromp91] R. F. Cromp, "Automated extraction of metadata from remotely sensed satellite imagery," *Technical papers, 1991 ACSM-ASPRS Annual Convention Proceedings*, Vol. 3, pp. 111-120, 1991.

[Dorf91] E. Dorfman, "Architecture of a large object-oriented database for remotely sensed data," *Technical papers, 1991 ACSM-ASPRS Annual Convention Proceedings*, Vol. 3, pp. 129-143, 1991.

[Feke84] G. Fekete and L. S. Davis, "Property spheres: a new representation for 3-d object recognition," *Proceedings of the Workshop on Computer Vision: Representation and Control*, Annapolis, MD, 1984.

[Feke90] G. Fekete, "Rendering and Managing Spherical Data with Sphere Quadtrees," *Proceedings of the First IEEE Conference on Visualization*, IEEE Computer Society Press, Los Alamitos, CA, 1990.

[Hinr83] K. Hinrichs and J. Nievergelt, "The grid file: a data structure designed to support proximity queries on spatial objects," *Proceedings of the WG '83 (International Workshop*

on Graphtheoretic Concepts in Computer Science), M. Nagl and J. Perl, eds., Trauner Verlag, Linz, Austria, 1983.

[Ohsa83] Y. Ohsawa and M. Sakauchi, "The BD-tree – a new n-dimensional data structure with highly efficient dynamic characteristics," *Information Processing 83*, R. E. A. Mason, ed., North-Holland, Amsterdam, 1983.

[Oost90] P. van Oosterom and E. Classen, "Orientation insensitive indexing methods for geometric objects," *Proceedings of the 4th Intenational Symposium on Spatial Data Handling*, Vol. 2, Zurich, Switzerland, 1990.

[Same90] H. Samet, *The Design and Analysis of Spatial Data Structures*, Addison-Wesley, New York, NY, 1990.

[Short91] N. Short, Jr., "A real-time expert system and neural network for the classification of remotely sensed data," *Technical Papers 1991 ACSM-ASPRS Annual Convention*, Vol. 3, pp. 406-418, 1991.