EURASIP Journal on Applied Signal Processing 2005:15, 2514–2520

A Data-Driven Multidimensional Indexing Method for Data Mining in Astrophysical Databases

Marco Frailis
Dipartimento di Fisica, Università degli Studi di Udine, Via delle Scienze 208, 33100 Udine, Italy
Email: frailis@fisica.uniud.it

Alessandro De Angelis
INFN, Sezione di Trieste, Gruppo Collegato di Udine, Via delle Scienze 208, 33100 Udine, Italy
Email: de angelis@fisica.uniud.it

Vito Roberto
Dipartimento di Matematica e Informatica, Università degli Studi di Udine, Via delle Scienze 208, 33100 Udine, Italy
Email: roberto@dimi.uniud.it

Received 1 June 2004; Revised 2 March 2005

Large archives and digital sky surveys with dimensions of 10^{12} bytes currently exist, while in the near future they will reach sizes of the order of 10^{15}. Numerical simulations are also producing comparable volumes of information. Data mining tools are needed for information extraction from such large datasets. In this work, we propose a multidimensional indexing method, based on a static R-tree data structure, to efficiently query and mine large astrophysical datasets. We follow a top-down construction method, called VAMSplit, which recursively splits the dataset on a near median element along the dimension with maximum variance. The obtained index partitions the dataset into nonoverlapping bounding boxes, with volumes proportional to the local data density. Finally, we show an application of this method for the detection of point sources from a gamma-ray photon list.

Keywords and phrases: multidimensional indexing, VAMSplit R-tree, nearest-neighbor query, one-class SVM, point sources.

1. INTRODUCTION

At present, several projects for the multiwavelength observation of the universe are underway, for example, SDSS, GALEX, POSS2, DENIS, and so forth [1]. In the next years, new spatial missions will be launched (e.g., GLAST, Swift [2, 3]), surveying the wall sky at different wavelengths (gamma-ray, X-ray, optical).

In the astroparticle and astrophysical fields, data are mostly characterized by multidimensional arrays. For instance, in X-ray and gamma-ray astronomy, the data gathered by detectors are lists of detected photons whose properties include position (RA, DEC), arrival time, energy, error measures both for the position and the energy estimates (dependent on the instrument response), and quality measures of the events. Source catalogs, produced by the analysis of the raw data, are lists of point and extended sources characterized by coordinates, magnitude, spectral indexes, flux, and so forth.

Data mining applied to multidimensional data analyzes the relationships between the attributes of a multidimensional object stored into the database and the attributes of the neighboring ones. Typical queries required by this kind of analysis are the following: (i) *point queries*, to find all objects overlapping the query point; (ii) *range queries*, to find all objects having at least one common point with a query window; and (iii) *nearest-neighbor queries*, to find all objects that have a minimum distance from the query object. Another important operation is the *spatial join*, which in the astrophysical field is needed to search multiple source catalogs and cross-identify sources from different wavebands.

These multidimensional (spatial) data tend to be large (sky maps can reach sizes of terabytes) requiring the integration of the secondary storage, and there is no total ordering on spatial objects preserving spatial proximity [4]. This characteristic makes it difficult to use traditional indexing methods, like B+-trees or linear hashing.

2. AN OPTIMIZED R-TREE

The R-tree is a data structure meant to efficiently index multidimensional point data or objects with a spatial extent [5].

The structure of an R-tree is the following:

(i) an *internal node* of the R-tree has a number of entries of the form (cp, MBB), where cp is the address of a child node and MBB is the n-dimensional *minimum bounding box* of all entries in that child node;

(ii) a *leaf node* has a number of entries of the form (O_{id}, MBB), where O_{id} refers to a record in the database describing a particular object and MBB is the minimum bounding box of that object. For point data, the leaf entries can also have the form (*point, attributes*), where *point* is a coordinate in the n-dimensional space and *attributes* are data associated to that point.

A bounding box R is defined by the two endpoints S and T of its major diagonal in the n-dimensional data space:

$$R = (S, T), \tag{1}$$

where

$$S = (s_1, s_2, \ldots, s_n), \quad T = (t_1, t_2, \ldots, t_n), \quad s_i \le t_i \ \forall 1 \le i \le n. \tag{2}$$

The *level* (or *depth*) of a node x of the tree is the length (the number of nodes) of the path from the root r to the node x. The *fanout* of a node x is the maximum number of entries a node can have. The internal fanout is the fanout of nonleaf nodes (to be distinguished from the leaf fanout or capacity). Analogous to the B+-tree, the R-tree is a balanced tree and each node has a fanout dependent on the disk page size.

The dynamic R-tree (and its variant, the R*-tree [6]) defines particular insertion, deletion, and update operations to reduce the overlapping between sibling nodes and guarantee a minimum filling rate.

Usually, the analysis of astrophysical data is performed on a static dataset. In this case, using multiple insertions to build the index on the entire dataset is very slow: the cost is $O(N \log_B N)$ I/O operations, where N is the number of input MBBs and B is the number of MBBs fitting into a disk block.

An optimized index, in terms of construction time, memory occupied, and query performances, can be built using *a priori* information on the dataset by means of bulk loading algorithms. Several bulk loading techniques have been proposed in the literature [7]. With these algorithms, the index can be built with $O((N/B)\log_{(M/B)}(N/B))$ number of I/Os, where M is the number of MBBs fitting into main memory. The result is a near complete and balanced R-tree. The basic idea used in these algorithms is the following: input MBBs or point data are sorted or partially sorted according to a criterion that preserves spatial proximity between adjacent elements in the ordering, then they are placed in the leaves in that order. The rest of the tree is then built recursively in a bottom-up manner.

We have followed a top-down construction method called VAMSplit algorithm, described in [8], to build and optimized R-tree. This method preserves the spatial proximity between sibling nodes, resulting in a partition of the dataset with no overlapping between MBBs. Moreover, the volume of the data space covered by each node (at a particular level) is variable and dependent on data density. The main idea of this method is to recursively split the dataset on a *near median* element along the dimension with *maximum variance*. In particular, following the formalism given in [9], the index construction algorithm comprises the following subtasks:

(i) determine the tree topology: height and fanout of the internal nodes, and so forth;

(ii) compute the split strategy based on the tree topology;

(iii) use an external selection algorithm to bisect the data on secondary storage;

(iv) construct the index directory.

2.1. Determination of the tree topology

The topology of a tree includes the height of the tree, the fanout of the internal nodes in each tree level, the capacity of the leaf nodes, and the number of the data objects (i.e., records) stored in each subtree. The topology of the tree only depends on static information which is invariant during the construction such as the number of objects, the number of dimensions indexed, and the page capacity.

Let B be the maximum number of data objects in a data page (i.e., a page storing a leaf node) and F the fanout of a directory page (i.e., a page storing a nonleaf node). Then, using the floor and ceiling operations, respectively, indicated by $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$, we have that

$$B = \left\lfloor \frac{\text{page size}}{\text{size of (data object)}} \right\rfloor, \qquad F = \left\lfloor \frac{\text{page size}}{\text{size of (node entry)}} \right\rfloor. \tag{3}$$

The maximum number of data objects in a tree with height h is

$$C_{\max}(h) = B \cdot F^h. \tag{4}$$

Therefore, knowing the number N of data objects to be indexed, the height of the tree must be determined such that $C_{\max}$ is greater than N. More formally,

$$h = \left\lceil \log_F \left\lceil \frac{N}{B} \right\rceil \right\rceil = \left\lceil \log_F \frac{N}{B} \right\rceil. \tag{5}$$

This corresponds to the height of the root node. The fanout of the root node is evaluated considering its subtrees as complete trees with height $h - 1$ (the target index is a balanced tree). Hence,

$$\text{fanout}\,(h, N) = \left\lceil \frac{N}{C_{\max}(h-1)} \right\rceil. \tag{6}$$

2.2. The split strategy

Given the topology of the target disk-based index tree, the split strategy is represented by a linear-space tree. For the VAMSplit algorithm, the split strategy is implicitly represented by a binary tree, where, at each level, the dimension with maximum variance is chosen as the split dimension.

Then, a *near median* element is selected as the split value and computed by

$$\text{med} = \begin{cases} \text{gscap} \cdot \left\lfloor \frac{1}{2} \cdot \left\lceil \frac{N}{\text{gscap}} \right\rceil \right\rfloor & \text{if } N \le 2 \cdot \text{cscap}, \\ & \text{gscap} > 0, \\ \text{cscap} \cdot \left\lfloor \frac{1}{2} \cdot \left\lceil \frac{N}{\text{cscap}} \right\rceil \right\rfloor & \text{otherwise,} \end{cases} \tag{7}$$

where cscap stays for child subtree capacity and gscap stays for grandchild subtree capacity and

$$\begin{aligned} \text{cscap} &= C_{\max}(h-1), \\ \text{gscap} &= C_{\max}(h-2) = \frac{\text{cscap}}{F}. \end{aligned} \tag{8}$$

Hence, when $N > 2 \cdot \text{cscap}$, the split value is selected so that the *left subtrees* of the target index are fully utilized. When $N < 2 \cdot \text{cscap}$, the split based on the cscap value would generate a strongly biased split; thus in this case the near median element is evaluated by means of the granchild subtree capacity, but without introducing any extra page in the target index.

For large datasets, not fitting into main memory, an external selection algorithm using the secondary storage for partitioning the data around the median element is necessary. Our implementation uses a sampling strategy given by [10] to find a good pivot value and reduce the number of I/O operations; a caching strategy explained in [9] has been adopted to partition the data into the secondary storage. When the number of records covered by a subtree fits into main memory, its construction is continued in main memory, reducing further the number of I/O operations.

3. TESTS ON A PHOTON DATASET

To test the behavior of this method, we built the index on a list of gamma-rays simulated for the GLAST project. In particular, the optimized R-tree was built on the RA and DEC values while the other columns of each photon were considered as attribute data.

Figure 1 represents the structure of the R-tree built on the first two days of simulated photons (for a total of 1 847 588 photons). The background image represents the projection on the RA-DEC plane of the photon counts. The root node contains only two rectangles (child nodes) splitting the dataset on the RA value of the median element. For the rectangle on the right, the image shows the partition generated by the second level of the tree instead for the left rectangle the partition at the third level is shown. As one can notice, in regions where the flux is higher, the decomposition is finer.

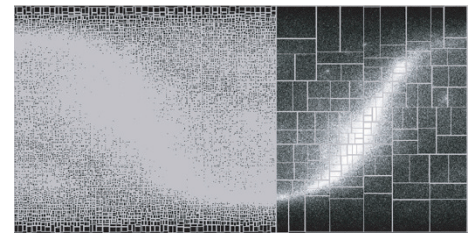

Figure 1: Minimum bounding boxes at different levels of the R-tree.

Then, to test query performances, we built an optimized R-tree on the photons generated by a fast simulation of an entire year of observation for a total of 40.1 millions photons. The size of each photon is of 165 bytes. Indexed attributes are again RA and DEC.

The system in which we run the tests is a Pentium IV 2400 MHz with 512 MB (DDR 266 Mhz), an 80 GB 7200 rpm Ultra ATA/100 hard disk. The operating system is a standard Red Hat 9.0 and the page size is 4096 bytes.

The building of the R-tree index on the entire dataset required 4 hours and 35 minutes. The result of the construction is a single index file with a size of 6.7 GB (it contains both the directory nodes and the data itself).

We performed 25 circular queries on the optimized R-tree, each one repeated 4 times. Each query is defined by a coordinate in RA and DEC together with a radius (of 15 degrees). Circular queries, on the R-tree, require a particular handling. We performed two types of queries. In the first type, the program converts a circular query into a rectangular query:

$$\begin{aligned} (\text{RA}, \text{DEC}, \text{radius}) \longrightarrow [&(\min \text{RA}, \min \text{DEC}), \\ &(\max \text{RA}, \max \text{DEC})], \end{aligned} \tag{9}$$

where the rectangle sides are tangent to the circular region. This way, the photons obtained by the rectangular query are a superset of the one obtained by the circular query.

The second type of query adds a filtering step to the first one, in which only photons inside the circular region are accepted. A Particular handling is required for circular queries intersecting the poles, but none of the 25 queries required it. The performances obtained are

(i) rectangular query average time: 10.06 seconds,
(ii) circular query average time: 10.47 seconds,
(iii) average number of elements retrieved by a rectangular query: 1.210.800,
(iv) average number of elements retrieved by a circular query: 973.239.

The hierarchical triangular mesh (HTM) [11] is another access method to index data characterized by a spherical distribution, which is used in several astrophysical experiments, like the Sloan Digital Sky Survey (SDSS) [12] and GLAST. To compare its performances with our indexing method, we used an HTM with 5 levels (the same configuration adopted

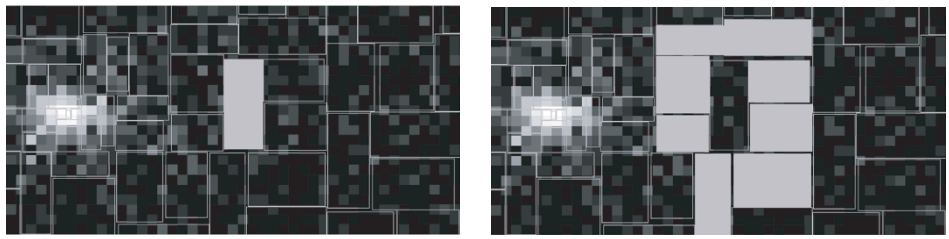

Figure 2: Adjacency between bounding boxes in the count map example.

in the SDSS project) to partition the photon dataset. A level-5 HTM decomposes the sky into 8 192 spherical triangles, each one associated to a different file on disk. The building of the HTM index required 1 hour and 27 minutes. Then we performed the same 25 circular queries used to test our R-tree index. Given a circular query, the HTM library returns a list of HTM IDs, each one identifying a spherical triangle intersecting the query region. The performances obtained are

(i) circular query average time: 140.72 seconds,
(ii) average number of level-5 triangles intersecting the query: 104.

4. NEIGHBORHOOD AND "WEAK" ADJACENCY

The structure of the optimized R-tree can help exploring the data and finding regions of interest. For this purpose, other information can be added to each node: the total number of data points covered by the node, their mean and variance, and other statistical moments.

Data mining techniques include clustering, classification, and density estimation tasks. The application of these techniques to large datasets involves the execution of multiple queries. Typical queries used for this tasks are nearest neighbor or similarity queries and adjacency queries. In particular, for cluster analysis or density estimation, it can be useful to define neighborhood or adjacency relations not only between data objects but also between the internal nodes of the optimized R-tree storing sufficient statistics. We use the definition of minimum distance between a point P and a bounding box R given by Roussopoulos et al. in [13]:

$$\mathrm{MINDIST}(P,R) = \sum_{i=1}^{n} |p_i - r_i|^2, \tag{10}$$

where

$$r_i = \begin{cases} s_i & \text{if } p_i < s_i, \\ t_i & \text{if } p_i > t_i, \\ p_i & \text{otherwise,} \end{cases} \tag{11}$$

which corresponds to the distance from the point to the nearest edge of the bounding box. Given a bounding box, its nearest neighbors are found by means of the *mindist* from its barycenter. An optimal algorithm, visiting only the nodes necessary for obtaining the nearest neighbors, is designed in [14]. This algorithm is also *incremental*, that is, the number of nearest neighbors to be retrieved is not known in advance.

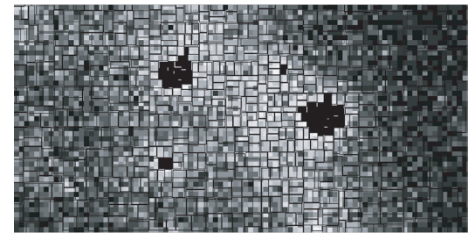

Figure 3: Local maxima obtained by bounding box sorting in the galactic anticenter.

Differently from space-partitioning data structures (like the kd-tree or the HTM) the R-tree has no adjacency relation between its nodes (i.e., usually edges are not shared between their bounding boxes). The adjacency relation is generally used in cluster analysis to find connected components. For point data characterized by an isotropic noise or background distribution, we define a *weak adjacency* between the R-tree bounding boxes as follows.

Definition 1 (weak adjacency). Two bounding boxes $U = (S,T)$ and $V = (S',T')$ are weakly adjacent if there exists $k \in \{1,\ldots,n\}$ such that

(i) $\neg(s_i \geq t'_i \vee t_i \leq s'_i)$ for all $1 \leq i \leq n$, $i \neq k$;
(ii) there does not exist $Z = (S'',T'')$ such that Z and U satisfy (i) and Z and V satisfy (i) and $(t_k \leq s''_k \leq s'_k \vee t'_k \leq s''_k \leq s_k)$.

In case of a regular grid in two dimensions, the above definition is equivalent to the 4-connectivity. Given an R-tree bounding box, the algorithm to find all its weakly adjacent bounding boxes is based on the incremental nearest-neighbor algorithm. Figure 2 shows an example of weakly adjacent bounding boxes found with this method.

In the optimized R-tree, the ratio between the number of elements n covered by a node and the volume V of its bounding box approximates the data density in that region. We use this information to find local maxima into the dataset. Given a rectangular (n-dimensional) region Q and a level l of the tree (chosen on the basis of the node resolution), the bounding boxes at level l overlapping Q are sorted in decreasing order of the n/V value. In Figure 3, the partition of the simulated photons in the galactic anticenter is shown: the first

90 bounding boxes in the ordering are filled, highlighting 4 densest areas which correspond to point sources in that region.

5. A STRATEGY FOR THE DETECTION OF POINT SOURCES

One of the major tasks, in the analysis of the data gathered by X-ray or gamma-ray detectors working in survey mode, is to distinguish point sources from diffuse background or extended sources. Point sources are mostly characterized by a stronger flux, with respect to the surrounding, focused on a small angular region. The area covered by a point source depends also on the instrument point spread function.

An optimized R-tree index can be built on a dataset including photons gathered in a certain range of time (we are using, for the analysis, a minimum interval of 6 days). To find static or strong variable sources (e.g., gamma-ray bursts), only a bidimensional indexing on the RA and DEC values is needed.

In the following, we propose a point source detection algorithm based on kernel methods [15], and in particular on the one-class SVM [16]. Standard kernel methods have memory and computational requirements that make them impractical for large datasets. In this work we show how to speed-up the training process by reducing the number of training data with the partitioning generated by our optimized R-tree.

5.1. One-class SVM

The one-class SVM algorithm estimates the support of a multidimensional distribution, that is, a binary function such that most of the data will live in the region where the function is nonzero. Given a dataset $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_\ell\}$, its strategy is to implicitly map the data into a high-dimensional feature space F using a kernel function, that is, a function k such that

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j) \rangle, \tag{12}$$

where $\boldsymbol{\phi}$ is the mapping from X to the (inner product) feature space F. Then, in F, it separates the data from the origin with maximum margin solving the following problem:

$$\min_{\mathbf{w},\rho,\xi} \frac{1}{2}\|\mathbf{w}\|^2 - \rho + \frac{1}{\nu\ell}\sum_i \xi_i \quad \text{s.t. } \mathbf{w}\cdot\phi(\mathbf{x}_i) \geq \rho - \xi_i, \quad \xi_i \geq 0,\ i = 1, \ldots, \ell, \tag{13}$$

where $\nu \in (0, 1]$ is a parameter of the problem. Constructing the Lagrangian and setting the derivatives to zero, we obtain the dual problem

$$\min_{\boldsymbol{\alpha}} \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \quad \text{s.t.} \sum_i \alpha_i = 1, \quad 0 \leq \alpha_i \leq \frac{1}{\nu\ell},\ i = 1, \ldots, \ell, \tag{14}$$

where the α_i are the Lagrange multipliers. This is a quadratic programming problem, solved by standard optimization techniques. After solving the dual problem, the support of the distribution is given by

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_i \alpha_i k(\mathbf{x}_i, \mathbf{x}) - \rho\right). \tag{15}$$

5.2. Scaling one-class method with the optimized R-tree

To scale the one-class method to a large dataset, the idea is to partition the data into pairwise disjoint convex subsets and use, in the one-class training, only one representative for each subset. An analogous method has been applied for classification tasks with the support vector machines (SVMs) [17]. Substituting the data in a subset X_0 with a representative is equivalent to adding the following constraint to the dual problem:

$$\alpha_{0,i} = \frac{\alpha_0}{\ell_0}, \quad \forall i = 1, \ldots, \ell_0, \tag{16}$$

where $\ell_0 = |X_0|$. Using the Gaussian kernel $k = \exp(-\lambda\|\mathbf{x} - \mathbf{z}\|^2)$, it can be shown that the best representative we can choose is the value, in the input space, satisfying

$$\min_{\mathbf{x}} \left\| \boldsymbol{\phi}(\mathbf{x}) - \frac{1}{\ell_0}\sum_{i=1}^{\ell_0} \boldsymbol{\phi}(\mathbf{x}_{0,i}) \right\|^2 = \min_{\mathbf{x}} 2 - \frac{2}{\ell_0}\sum_{i=1}^{\ell_0} \exp\left(-\lambda\|\mathbf{x} - \mathbf{x}_{0,i}\|^2\right), \tag{17}$$

that is, $\mathbf{x} = (1/\ell_0)\sum_{i=1}^{\ell_0} \mathbf{x}_{0,i}$. The partitioning we adopt is the one generated by one level of the optimized R-tree. We reduce the elements covered by a node of the partition to their mean value and train the one-class algorithm on such representatives. With respect to the standard one-class method, an approximate solution is found with a speedup that can be of two orders of magnitudes (depending on the level of detail in the partitioning).

5.3. Tests on the anticenter region

Instead of trying to detect directly the point sources, we use the accelerated one-class method to estimate the support of the background and the diffuse emission distribution. Being able to estimate such distribution, point sources are detected as outliers of the support found.

Our approach is to associate to each photon's coordinates the density of its surrounding area. We use the partition generated by indexing the data with the optimized R-tree to accelerate the training phase and get an approximate solution. Moreover, at a certain level of the R-tree, the decomposition is finer in the areas with higher density. Hence, we use the ratio between the volume of a bounding box and the number of photons it covers to approximate the density associated to each photon in that node.

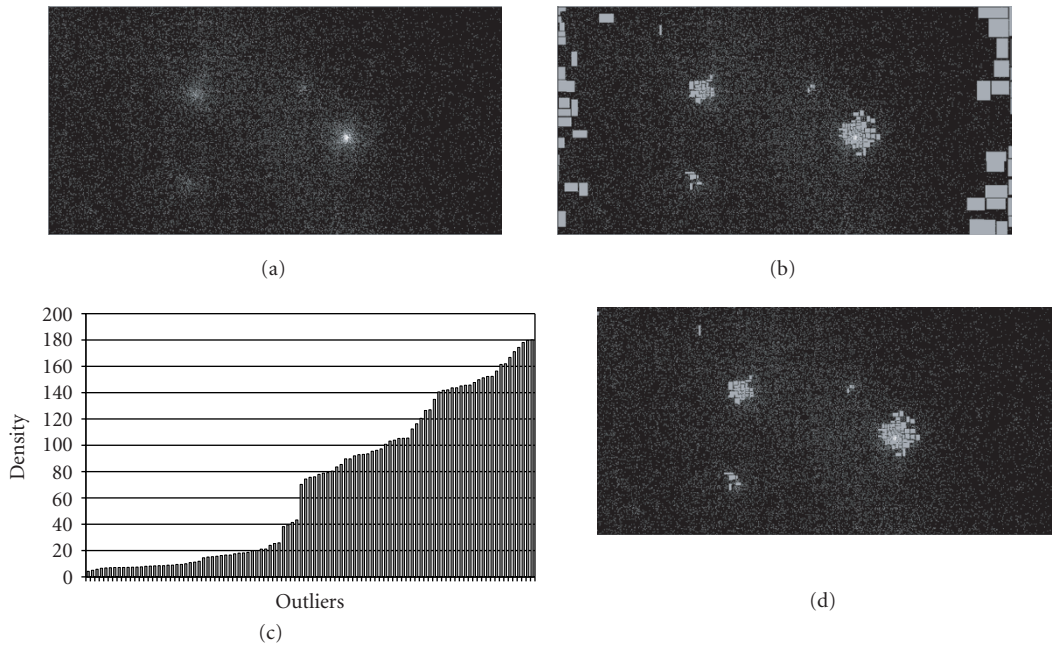

Figure 4: Point sources detection applying one-class SVM to the partition generated by the optimized R-tree. (a) A counts map of the anticenter region. (b) Outliers with respect to the diffuse emission. (c) Densities of the outliers in increasing order. (d) Result after filtering out the outliers with low density.

Putting together position and local density information generates some redundancy. In fact, in areas where the density is higher the mean distance between the photons is smaller. A solution for removing redundancy in a dataset is to perform the principal component analysis (PCA) [18], which gives, in a dataset, the directions of maximum variance. Generally, only a subset of the eigenvectors are kept, that is, the ones corresponding to the directions capturing most of the variance. Hence, the data are first projected into the subspace found:

$$\tilde{\mathbf{x}}_i = \mathbf{U}_k^T \mathbf{x}_i, \quad i = 1, \ldots, \ell, \tag{18}$$

where k is the number of eigenvectors used.

To test this method we have applied it again to the GLAST photon dataset and in particular to the anticenter region (including 25 890 photons). We have used the partition generated with the last level of the R-tree. The parameter values adopted are $\nu = 0.14$ and $\lambda = 0.0003$ (the width of the Gaussian kernel). The training has required 0.44 second.

The outliers detected are shown on Figure 4b. Apart from the four stronger sources, also areas with low density are highlighted as outliers. This is due to the one-class method itself: it finds the most dissimilar objects on the boundary of the decision function. In this particular task, the most dissimilar elements are the areas with a very high density with respect to the surrounding and the areas with very low density. This can be seen also by plotting the histogram of the density values (Figure 5.2) for the outliers. Hence, a simpler task remains to filter out the areas with low density (see Figure 5.2).

6. CONCLUSIONS

In this work, we have realized a multidimensional indexing method to efficiently access and mine large multidimensional astrophysical data. The index is based on a static version of the R-tree data structure, the VAMSRtree. We have fixed the original algorithm and adapted it to very large dataset, for which the partial sort cannot be performed in main memory. We have adopted an efficient incremental nearest-neighbor algorithm and defined a weak adjacency relation between the R-tree nodes. These algorithms, together with the optimized R-tree structure, allow to efficiently query the data (with point and n-dimensional range queries) and perform data mining tasks like clustering and density estimation. A fast novelty detection algorithm, based on the one-class SVM method, has been shown. We have used, as a running example, photon data gathered from a simulation for the Gamma-ray Large Area Space Telescope (GLAST).

REFERENCES

[1] R. J. Brunner, S. G. Djorgovski, T. A. Prince, and A. S. Szalay, "Massive datasets in astronomy," in *Invited Review for the Handbook of Massive Datasets*, J. Abello, P. Pardalos, and M. Resende, Eds., pp. 931–979, Kluwer Academic, Norwell, Mass, USA, 2002.

[2] *GLAST—Gamma-ray Large Area Space Telescope*, http://glast.gsfc.nasa.gov/.

[3] Swift mission, http://www.nasa.gov/mission_pages/swift/main/index.html.

[4] V. Gaede and O. Gunther, "Multidimensional access methods," *ACM Computing Surveys*, vol. 30, no. 2, pp. 170–231, 1998.

[5] A. Guttman, "*R*-trees: a dynamic index structure for spatial searching," in *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 47–57, Boston, Mass, USA, June 1984.

[6] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger, "The *R**-tree: an efficient and robust access method for points and rectangles," in *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 322–331, Atlantic City, NJ, USA, May 1990.

[7] L. Arge, K. Hinrichs, J. Vahrenhold, and J. S. Vitter, "Efficient bulk operations on dynamic *R*-trees," in *Proc. 1st International Workshop on Algorithm Engineering and Experimentation (ALENEX '99)*, vol. 1619 of *Lecture Notes In Computer Science*, pp. 328–348, Springer, Baltimore, Md, USA, January 1999.

[8] D. A. White and R. Jain, "Similarity indexing: algorithms and performance," in *Storage and Retrieval for Still Image and Video Databases IV*, vol. 2670 of *Proceedings of SPIE*, San Diego, Calif, USA, pp. 62–73, 1996.

[9] C. Böhm and H.-P. Kriegel, "Efficient bulk loading of large high-dimensional indexes," in *Proc. 1st International Conference on Data Warehousing and Knowledge Discovery (DaWaK '99)*, vol. 31, pp. 251–260, Florence, Italy, August–September 1999.

[10] C. Martínez and S. Roura, "Optimal sampling strategies in quicksort and quickselect," *SIAM Journal on Computing*, vol. 31, no. 3, pp. 683–705, 2001.

[11] P. Z. Kunszt, A. S. Szalay, and A. R. Thakar, "The hierarchical triangular mesh," in *Proc. of the MPA/ESO/MPE Workshop in Mining the Sky*, A. J. Bandy, S. Zaroubi, and M. Bartelmann, Eds., pp. 631–637, Garching, Germany, 2001.

[12] Sloan Digital Sky Survey, http://www.sdss.org/.

[13] N. Roussopoulos, S. Kelly, and F. Vincent, "Nearest neighbor queries," in *Proc. ACM-SIGMOD International Conference on Management of Data*, pp. 71–79, San Jose, Calif, USA, May 1995.

[14] G. R. Hjaltason and H. Samet, "Distance browsing in spatial databases," *ACM Transactions on Database Systems*, vol. 24, no. 2, pp. 265–318, 1999.

[15] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*, Cambridge University Press, Cambridge, UK, 2004.

[16] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, and A. J. Smola, "Estimating the support of a high-dimensional distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[17] D. Boley and D. Cao, "Training support vector machine using adaptive clustering," in *Proc. 4th SIAM International Conference on Data Mining (SIAM DM '04)*, Lake Buena Vista, Fla, USA, April 2004.

[18] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning*, Springer, New York, NY, USA, 2001.

Marco Frailis graduated in computer science in Udine in 2001. In November 2001, he started his Ph.D. degree at the Department of Mathematics and Computer Science, the University of Udine. His research work was carried out in collaboration with the Department of Physics and within the GLAST project (NASA). He received his Ph.D. degree in computer science in 2005 with a thesis entitled "Data Management and Mining in Astrophysical Databases" and he has a Postdoc position at the Department of Physics. His main research interests are pattern recognition and data mining techniques.

Alessandro De Angelis, a Professor of experimental physics at the University of Udine and at the IST of Lisboa, chairs the M.S. program in computational physics in Udine, and is a Member of INFN, CIFS, SIF. After classical high school, he graduated (cum laude) in physics in Padova in 1983, worked in the Group of Marcello Cresti, and was a Technical Officer at the Terrestrial Weapons headquarters, Rome, in 1983/1984. From 1984 to 1992, he studied properties of charmed particles with bubble-chamber detectors and worked in Padova and Udine on the preparation of the DELPHI experiment at the CERN LEP electron-positron collider. From 1993 to 1999 at CERN, Geneva, he worked in the Group of Ugo Amaldi, as a Research Associate and Staff Member, coordinating the data analysis software of DELPHI and the QCD Group, and was responsible for the software for the INFN project on artificial NN. Back to Italy in 1999, he founded in Udine a group on astroparticle physics, working on GLAST and MAGIC (detection of high-energy gamma rays with satellite and ground based, respectively), and giving a primary contribution to simulation, event display, and data acquisition. He is the author of more than 400 publications, referee for leading scientific journals, and organizer of several conferences in the field of astroparticle physics.

Vito Roberto was born in 1951, got his "Laurea" degree in physics in 1973 and is now a Full Professor in computer science. Since 1979, he has been working at the University of Udine, Italy, where he founded the Machine Vision Laboratory (http://mvl.dimi.uniud.it) and is now the Director of the Department of Mathematics and Computer Science. His main research interests are pattern recognition systems and network-based systems. Professor Roberto is the author of a book and over 80 scientific publications. He is a Member of the American Physical Society (APS) and the International Association of Pattern Recognition (IAPR).