Supporting Web-Based Visual Exploration of Large-Scale Raster Geospatial Data Using Binned Min-Max Quadtree

Jianting Zhang[1,2] and Simin You[2]

[1] Department of Computer Science
The City College of the City University of New York
138th Convent Avenue, New York, NY 10031
`jzhang@cs.ccny.cuny.edu`
[2] Department of Computer Science
The Graduate Center of the City University of New York
365 Fifth Avenue, New York, NY 10006
`syou@gc.cuny.edu`

Abstract. Traditionally environmental scientists are limited to simple display and animation of large-scale raster geospatial data derived from remote sensing instrumentation and model simulation outputs. Identifying regions that satisfy certain range criteria, e.g., temperature between [t1,t2) and precipitation between [p1,p2), plays an important role in query-driven visualization and visual exploration in general. In this study, we have proposed a Binned Min-Max Quadtree (BMMQ-Tree) to index large-scale numeric raster geospatial data and efficiently process queries on identifying regions of interests by taking advantages of the approximate nature of visualization related queries. We have also developed an end-to-end system that allows users visually and interactively explore large-scale raster geospatial data in a Web-based environment by integrating our query processing backend and a commercial Web-based Geographical Information System (Web-GIS). Experiments using real global environmental data have demonstrated the efficiency of the proposed BMMQ-Tree. Both experiences and lessons learnt from the development of the prototype system and experiments on the real dataset are reported.

Keywords: Binned Min-Max Quadtree, raster geospatial data, Web-GIS, visual exploration.

1 Introduction

Advancements in remote sensing technology and instrumentation have generated huge amounts of remotely sensed imagery. It has been estimated that remotely sensed imagery is acquired at the rate of several terabytes per day [1]. In addition to raw imagery data, derived data products targeting at domain-specific applications are fast growing as well. For example, regional and global coverage of Land Cover, Land Surface Temperature, Albedo and Vegetation Indices are among a fraction of MODIS data product table [2]. Still yet, numerous environmental models, such as Weather Research and Forecast

(WRF [3]), have generated even larger volumes of geo-referenced raster model output data with different combinations of parameters, in addition to increasing spatial and temporal resolutions. Traditionally environmental scientists are limited to simple display and animation of raster geospatial data in a desktop computing environment. Very little visual exploration functionality has been provided despite the continuous community software development efforts, such as the Integrated Data Viewer (IDV) from UCAR [4] and WorldWind from NASA [5]. A general purpose, high-performance spatial database backend is highly desirable in supporting visual explorations of large-scale raster geospatial data. Unfortunately, most of existing spatial database research and developments focus on vector geospatial data.

The limited support for binary raster data in the form of quadtree indexing that have been implemented in leading spatial databases, e.g., Oracle Spatial [6] and Microsoft SQL Server Spatial [7], are primarily designed for query filtering on vector spatial data rather than querying numeric raster data natively. We also argue that the pyramid-alike approaches, such as Oracle GeoRaster [8], ArcGIS tiled map services [9] and MapServer[10]/TileCache [11] are mostly designed for fast simple display purposes and are not suitable for supporting interactive visual explorations that involve ad-hoc queries on raster cell values. Efficient data structures, indexing techniques and coordination between browser-based applications and servers are essential to support such queries for interactive visual explorations in a Web environment.

In this study, we propose to use a binned min-max quadtree data structure to index integer or real value rasters to facilitate query-based visualization and visual explorations [12][13][14] [15][16] of large-scale raster geospatial data. By quantizing numeric raster cell values into a set of bins based on the histogram of a raster, the complexity of the resulting quadtree can be greatly reduced. The memory footprint of the quadtree can be sufficiently small to reside in main memory to efficiently answer exploratory queries, such as finding regions whose temperature are between [t1,t2) and precipitation are between [p1,p2). Our work along the direction is largely motivated by the binned bitmap indexing implemented in FastBit [17] with adaptation to more efficient support of finding Regions of Interests (ROIs) in large geo-referenced raster environmental datasets. We term the problem as ROI-finding query which obviously is an extension to finding individual data items as in [12][13]. The problem is similar to the one addressed in [18] and we have adopted a different approach. The data structure can be parallelized across multiple shared-nothing machines to effectively support visual explorations of large-scale geospatial raster data in a Web environment. A prototype that integrates our in-house developed query processing backend based on the proposed data structure and the commercial ArcGIS server software demonstrates the feasibility and effectiveness of the proposed approach. Our technical contributions cover the following three aspects:

- We have proposed a novel binned min-max quadtree data structure to index non-binary rasters. The data structure is efficient for processing queries in support of interactive visual explorations of large-scale raster data by taking advantages of the approximate nature of visualization related queries.
- We have developed a Web-based end-to-end system that allows users interactively and visually explore large-scale raster geo-referenced raster environmental data by coordinating Web clients and query processing backend.

– Experiments using a real environmental data have demonstrated the efficiency of the binned min-max quadtree data structure. The experiments also have provided experiences and lessons to further improve the performance of the Web-based system.

The remainder of the paper is structured as follows. Section 2 introduces related works on visual explorations of raster geospatial data and managing and indexing of raster geospatial data in a database environment. Section 3 presents the binned min-max quadtree data structure. Section 4 provides the architecture and the components of the prototype system to support Web-based visual explorations. Section 5 presents our experiments on the WorldClim global precipitation data at the 30 arc-seconds resolution. We report both experiences and lessons learnt from the development of the prototype system and experiments on the real dataset. Finally, Section 6 concludes the paper and outlines future research directions.

2 Background and Related Work

Visual explorations play an important role in seeking casual relationships among environmental factors and the possible relationships between human activities and their environmental consequences. As the majority of environmental data have a geospatial and a temporal component, research on exploratory spatial and spatiotemporal analysis [19] [20][21] can be generally applied. Quite a few research prototypes, such as GeoDa[22] and GeoVista [23] are available for vector geospatial data (including the associated attribute data) and they are quite successful in social, economic and health domains. On the other hand, while numerous image clustering, segmentation and classification algorithms have been applied to satellite imagery, only a handful research on visual explorations of the correspondences between pixel values and class labels have been reported [24][25]. The recently emerging object-oriented classification algorithms and their applications in remote sensing imagery have provided an opportunity for visual explorations of remotely sensed image data [26][27]. On the environmental model output side, most of existing developments focus on overlaybased display, map generation and animation. Little progresses have been evidenced on exploratory analysis with the exception of iso-surface generation. As an example, UCARs IDV[4] is capable of generating iso-surfaces from 3D atmospheric model outputs. However, the capabilities of linking geospatial phenomena (objects or events) with the underlying raster data (sensor observations or model outputs) are still lacking despite the availability of several pioneering theoretical works (e.g. [28]). It is beyond the scope of this paper to provide a comprehensive solution to visual explorations of raster geospatial data. Rather, our focus is on efficient implementation of a few generic operations to facilitate such visual explorations on large-scale raster geospatial data. More specifically, we target at the ROI-finding queries.

A few options are available to manage geospatial raster data in databases based on existing database technologies. First of all, each individual element can be treated as a tuple in relational table and traditional relational database technologies can be applied. Second, the raster data can be treated as multidimensional arrays and subsequently N-Array or nested table techniques can be applied in object-relational databases [29].

Finally, large raster data can be decomposed into small units and stored in databases as Large Binary Objects or BLOBs [30]. While each of the approaches has their suitable application scenarios, none of them can be used to process ROI-finding queries efficiently. Studies also have shown that multidimensional array supports implemented in mainstream databases are far less efficient than using native scientific data formats when scaling up to large datasets [29]. While the recently emerging column-store database techniques optimized for read-only or append only data [31][32] might provide an opportunity to reevaluate the suitability of relational and object-relational database technologies for geospatial raster data, it is likely that extending existing spatial indexing techniques remain to be the most effective way to process the ROI-finding queries.

We note that column-store based database layouts are similar to popular scientific data formats, such as NetCDF[33] and HDF5[34] in many aspects. We believe that one of the biggest problems in handling geo-referenced gridded environmental data in existing databases is the lacking of proper indexing mechanisms that take spatial or spatiotemporal autocorrelations into consideration. In fact, as discovered in [18], while sophisticated bitmap indexing techniques such as those implemented in FastBit [17] are very efficient in finding individual raster cells, it is more computationally expensive to assemble them into regions and return the regions as query results, i.e., assembling tuple IDs into regions. In contrast, our approach is based on spatial indexing that indexes regions and returns regions that satisfy searching criteria directly without incurring expensive post-processing cost. We also note that our work on indexing raster geospatial data complements existing works on array query definition language [35][36][37] and physical data layout of multidimensional data[38][39][40].

The most popular spatial indexing techniques include R-trees, quadtrees, octtrees and kd-trees. We refer readers to the overview papers and books for more detailed information [41][42]. Various quadtrees have been used to index both vector and binary raster data. On the other hand, interval tree [43],octree [44][45] and kd-tree [46][47] techniques have gained considerable popularity in deriving static and time-evolving iso-surfaces with or without ray-tracing. There are two problems in directly applying octree and kd-tree techniques to visual explorations of large scale gridded environmental data. First, techniques developed for iso-surface generation and/or ray-tracing usually have large memory footprints. While they are suitable for fine-scale offline rendering, it may be too costly for online visual explorations in a Web environment. Usually only limited resources are allocated to a Web browser. Data communication overheads between browsers and servers are much higher than those in tightly coupled desktop or cluster computing environments. Second, these techniques are mostly designed for tracing boundaries (iso-surfaces) and intersecting with linear objects (ray-tracing) and it is not straightforward to apply them to identify ROIs.

There are also works trying to extend quadtree techniques for binary rasters to grayscale or color images [48] [49][50]. However, most of existing studies along the direction focused on image encoding or compression with only few of them actually targeted at efficient query processing which are more relevant to visual explorations of scientific data. The difference between image compression and query processing is that the former focuses on the tradeoffs between disk storage and compression computation while the later focuses on the tradeoffs between the filtering and refinement in processing a query using an acceptable memory footprint. Previous research on quadtree based data structures and query processing focused on storage and manipulation of clusters of (overlapped) images. We refer readers to [51][52][53][54][55] and the review article by Manouvrier[56] for more details. These data structures are not tailored for read-only applications and they may not be suitable for supporting visual explorations of large-scale raster data, especially in a Web environment. Furthermore, most of the works utilized or followed a B-Tree indexing approach and assumed portions of the indices are dynamically loaded into memory. In contrast, our approach quantizes raw data into bins and tries to build a memory-resident index with desired memory footprint. The memory-resident index ensures fast filtering in query processing and returns regions that approximate true query results before subsequent refinement.

Our approach is largely motivated by the binned bitmap indexing reported in [57] and the multi-scale bitmap indexing reported in [58]. The binned bitmap indexing approach has been successfully applied to large-scale query-based interactive visualization using techniques such as Parallel Coordinate Plot or PCP [59]. A similar bin-hashing strategy has been applied to query driven visualization before rendering resulting records as point clouds [60]. One drawback of directly applying binned bitmap indexing for visual explorations of geospatial raster data is that when value ranges of a query are wide, it is likely that a large number of cells will be returned independently. It could be very expensive to transport the individual query results from query backend to client machines and paint these cells individually in a Web browser. The rendering speed in Web browsers could be improved by specifying regions instead of individual pixels. The advantages and practical needs of returning regions from a query are also identified by a recent work by Sinha[18] which is based on the FastBit. The authors proposed to identify regions from the bit vector of a query result which had been observed to be very computationally expensive. As such, while we recognize the generality of bitmap indexing that is suitable for both structured and unstructured data items, we argue that our approach is more suitable for ROI-finding queries for visual exploration purposes.

With respect to Web-based visual exploration of geospatial data, previous works focused on the mashup techniques by dynamically compositing images generated at the server sides [61][62]. However, very few of the reported works have taken query efficiency into consideration. Most of them assume that the server side programs are sufficiently fast and the bandwidth is large enough for interactive visualizations. As detailed in Section 3, the hierarchical nature of quadtree data structures allow pruning quadrants that do not intersect with the spatial extents that are under investigations by users. This not only speeds up query response times but also reduce the data volumes of query results need to be transported to clients. In addition, any lower level quadrants whose spatial extents are less than single pixel based on visualization scales at the clients can be pruned as well, which further reduce query response times and query result data volumes.

3 Binned Min-Max Quadtree

In this section, we first define the ROI finding queries before presenting the binned min-max quadtree data structure and its construction and query processing algorithms.

Given a set of rasters representing environmental variables $\{F_i|0 \leq i < n\}$ over a spatial domain D whose value ranges are $\left\{V_i^H\right\}$ and $\left\{V_i^L\right\}$ respectively, a ROI finding query Q identifies regions in D whose cells C_j satisfy the compound condition $\left\{C_j|V_{1j} \in [V_1^{QL}, V_{1j}^{QH}]\ \mathbf{op}\ V_{2j} \in [V_2^{QL}, V_2^{QH}] \ldots \mathbf{op}\ V_{kj} \in [V_k^{QL}, V_k^{QH}]\right\}$ where op can be either conjunctive and disjunctive and $0 < k < n$. V_i^{QL} and V_i^{QH} represent the lower and high bounds of query Q for variable i. While the formulation does not impose any relationships among the resulting raster cells, since we divide domain D into quadrants in quadtree indexing and evaluate the compound condition against relevant quadrants, the resulting cells are naturally reported as a collection of quadrants that satisfy the compound query criteria. Instead of reporting the cells individually from the query result, using a collection of quadrants may significantly save memory consumption and improve query results rendering for visualization purposes. Assuming that a quadtree for each environmental variable has been constructed, it is not difficult to see that evaluating the compound condition can be achieved by synchronized traversal of relevant quadtrees as described in (Manouvrier et al 2002) and will be omitted here due to space constraints. We next focus on the construction of a single quadtree and evaluating a single condition on a quadtree.

The Binned Min-Max Quadtree, or BMMQ-Tree for short, is directly motivated by both binned bitmap indexing [57] for scientific data and min-max octree/kd-tree for isosurface generation and ray tracing[47]. As discussed above, while min-max octrees and kd-trees are efficient in pruning tree branches that do not contain the iso-values being used, building a full octree or kd-tree using the finest resolution data may consume too much memory and slow down index construction and query processing. The binned bitmap indexing is more efficient and more suitable for large scale scientific data when compared with classic bitmap indexing. The downside is that the resulting cells in binned bitmap indexing do not naturally form regions and the post-processing to assemble cells into regions could be very computationally expensive. The proposed BMMQ-Tree combines the advantages of min-max octree/kd-tree indexing and binned bitmap indexing. Furthermore, our empirical studies have shown that, while environmental data are well-known for significant spatial autocorrelation due to the first law of geography [63], neighboring cell values are often slightly different. This makes traditional quadtree-based indexing techniques that require the uniformity of quadrants inappropriate. By binning cell values using proper boundary values, quadrant uniformity can be derived and mature quadtree indexing techniques (such as linearization and query processing in databases) can be applied. In this sense, our BMMQ-Tree data structure is an extension to traditional region quadtrees by associating each quadrant with a min and a max value of a quadrant. The min and max values are bin indices which can be bytes (8 bits for 256 bins) or short integers (16 bits for 65,536 bins), which normally are just a fraction of the storage requirement for a quadtree node. On the other hand, BMMQ-Tree branches can be efficiently pruned if the nodes min/max values do not overlap with the range of the query being evaluated. We next present the tree construction and query processing in more details.

Fig. 1 illustrates the process of constructing a BMMQ-Tree. The process first determines bin boundaries. While quite a few options are available, we decide to use a histogram based quantile approach for simplicity. The open source GDAL package

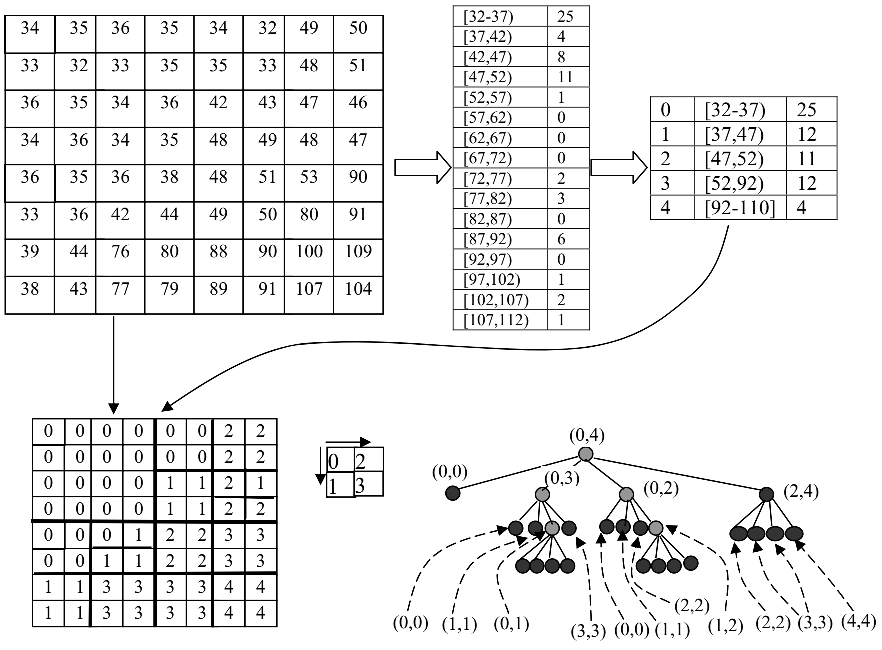


Fig. 1. Illustration of Binned Min-Max Quadtree Construction

[64] has provided an API to efficiently generate histograms for major image and raster data formats which further makes the histogram based approach desirable from implementation perspective. After the histogram with desired low/high boundary and bins has been retrieved, we loop through the histogram bins to determine the quadtree bins as the following. For each of the histogram bins, if its count is larger than the average count of the quadtree bins (calculated as the total valid raster cells divided by the desired quadtree bins - denoted as NA), then it is qualified as a quadtree bin; otherwise it will be combined with previous histogram bins until the combined count is larger than NA. Due to the bin formulation policy, the resulting number of bins can be less than the desired bins.

The main process of BMMQ-Tree construction is as the following. First, the 2D raster data being indexed is read into main memory following the sequence of BMMQ-Tree nodes that are being constructed. Each cell value is first quantized based on the bin boundary values. After the four cells that form a quadrant are processed, if their bin indices are exactly the same, then they will be combined to form a quadrant with the same min/max value. The process then moves to the next quadrant based on the quadtree space tessellation (row-major or column-major). The combination is also performed recursively to determine the min/max bin indices for each quadtree node. It is clear that the memory requirement in the BMMQ-Tree construction process is never larger than the size of the constructed quadtree plus four quadtree nodes and the data buffer to hold the values of raster cells being processed. We are in the process of exploring the possibilities of GPGPU based parallel constructions of BMMQ-Trees which is likely to

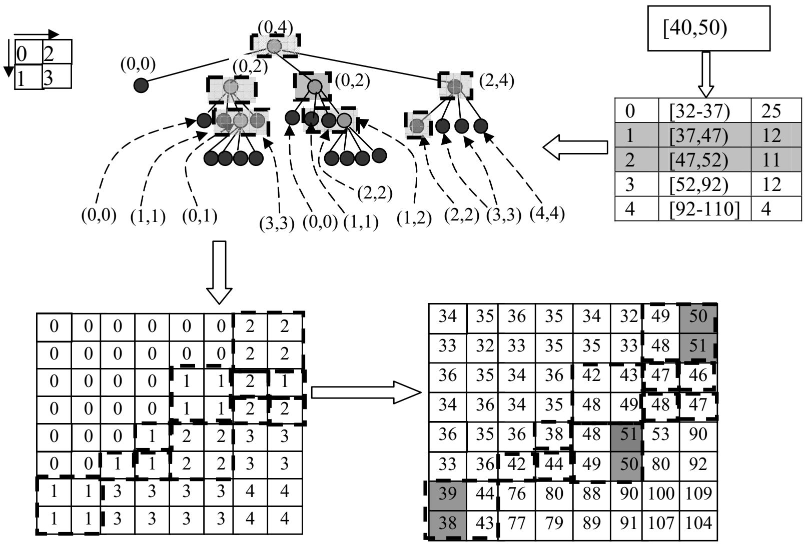


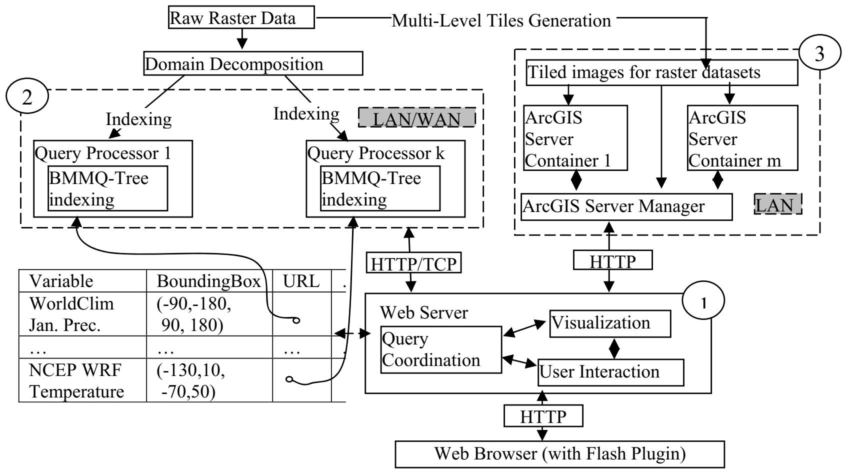
Fig. 2. Query Processing Based on Binned Min-Max Quadtree

reduce index construction times significantly. On the other hand, as most georeferred environmental data are read only, offline index creation and online query processing are independent of each other. While it is difficult to compare our approach with approach presented in [18] directly, we consider our approach trades offline index construction times with online query processing times by forming uniform quadrants which are intermediate between connected components and individual raster cells. We next turn to online query processing using BMMQ-Trees.

Query processing using BMMQ-Trees is illustrated in Fig. 2 using an example. First, the low and high boundary values [40,50) are mapped to the low and high ends of bin indices 1 and 2, respectively. The boundary values corresponding to the resulting bin indices [37,52) completely contain the query low and high boundary values to avoid true negatives. Second, starting from the root, each quadrant is recursively visited. If the min-max indices range does not overlap with the indices range corresponding to the query range, then all quadrants under the node corresponding to the current quadrant can be safely pruned. In Fig. 2 (top-left part), non-leaf nodes not in the shaded rectangles have been pruned. The recursive query processing stops under two conditions. First, a leaf node is reached which indicates that either a raster cell at the finest resolution is reached or all the raster cells in the quadrant corresponding to the node have the same bin index. Second, the quadrant corresponding to the node has a spatial extent less than a single pixel based on visualization scales at the client side. Assuming the width and height of visualization canvas at the client side are Wc and Hc and the width and height measured as the real-world coordinate system (e.g., latitude/longitude) are Wr and Hr, respectively, the quadtree level to stop recursive query processing can be easily calculated as $L_{stop} = floor(log_2(min(Wr/Wc, Hr/Hc)))$. It is clear that false positives may be introduced (the shaded cells in the lower-right part of Fig. 2) due to the binning nature of the data structure and the less-than-single-pixel stopping policy. While this may be a problem for querying databases exactly, it turns out that allowing false positives reduces the structural complexities of query results and hence the data volumes to be transported to the clients as well as rendering times at the client side. More discussions are provided in the experiments section.

4 System Architecture and Implementation

The primary focus of this research is to develop an end-to-end system that can effectively facilitate environmental scientists to explore large-scale geo-referenced raster data. Compared to desktop applications, a Web-based interface is more preferable. However, normally only limited resources are allocated to a Web browser and graphics rendering accelerators are usually not accessible to browser-based applications. Data communication overheads between browsers and servers are much higher than those in tightly coupled desktop or cluster computing environments. As such, careful design to take the characteristics of both client and server sides into consideration is essential in visual exploration of large-scale raster geospatial data in a Web environment. In our prototype system, the client side is based on the Rich Internet Application (RIA) framework using Adobe Flex programming [65]. The server side is a combination of commercial ArcGIS server technologies [9] and our in-house developed distributed query processing modules that implement the binned min-max quadtree based indexing as described above. The overall architecture is shown in Fig. 3 and more details on each of the components will be provided subsequently. Note that some components in Fig. 3 involve both offline and online parts and both of them will be introduced in the respective subsections.



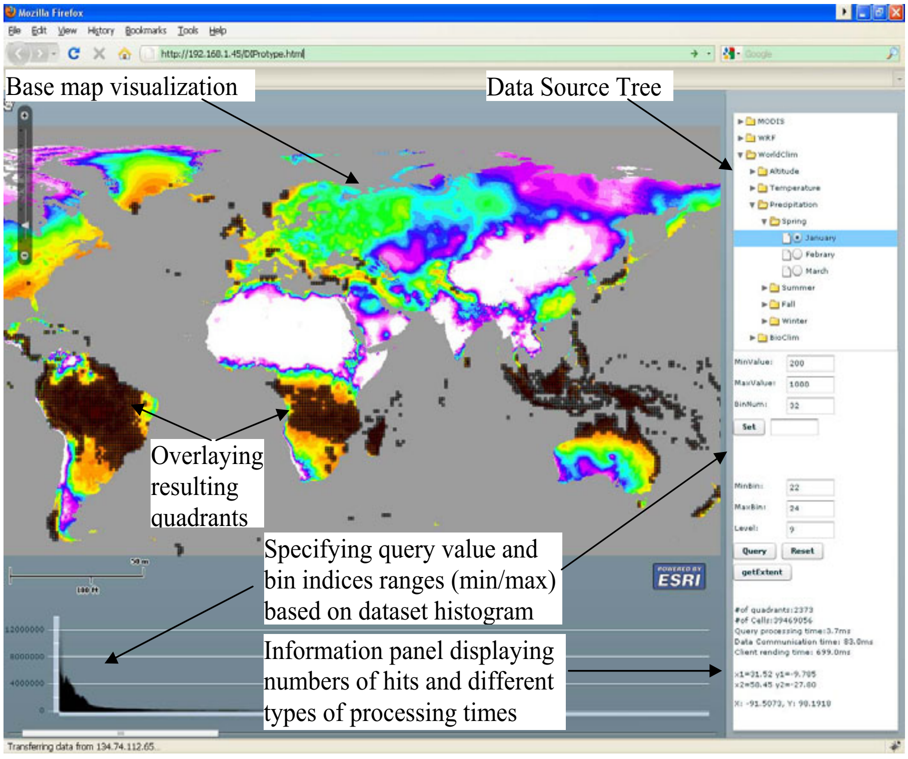
Variable	BoundingBox	URL	.
WorldClim Jan. Prec.	(-90,-180, 90, 180)		
…	…	…	
NCEP WRF Temperature	(-130,10, -70,50)		

Fig. 3. Prototype System Architecture

4.1 Visual Exploration Client Module

We have adopted Adobe Flex programming [65] to develop the client module for the following considerations. First, the Flex RIA framework has built-in rich graphics functionality and allows users control rendering canvas at the pixel level which is much more powerful and flexible than traditional Ajax based solutions. Second, it also allows more complex and immediate interactions with users which are crucial in visual explorations. Third, quite a few popular visualization packages, such as ArcGIS [66], have provided Flex APIs which makes the choice very attractive compared to the similar frameworks. Nevertheless, we believe that our design is also applicable to other RIA frameworks, should an alternative be more desirable for practical reasons.

The client module provides graphics user interfaces (GUIs) and interacts with users. Fig. 4 provides a snapshot shows the layout of the GUIs. A tree interface shows the catalog of available raster datasets based on their semantic classifications. Users can search and subsequently select a subset of rasters as the candidates for explorations. For the chosen rasters, histograms can then be visualized to help users determine ranges of values for subsequent visual explorations. The "Query" button serves as the entry point to translate the query criteria represented by GUIs into a query string to be passed to


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://192.168.1.45/DIProtype.html
Base map visualization
Data Source Tree
MODIS
WRF
WorldClim
Altitude
Temperature
Precipitation
Spring
January
Febrary
March
Summer
Fall
Winter
BioClim
Overlaying resulting quadrants
Specifying query value and bin indices ranges (min/max) based on dataset histogram
Information panel displaying numbers of hits and different types of processing times
Set
Query
Reset
getExtent
ESRI
12000000
8000000
4000000
0
Transferring data from 134.74.112.65...

MinValue: 200
MaxValue: 1000
BinNum: 32

MinBin: 22
MaxBin: 24
Level: 9

```
#of quadrants:2373
#of Cells:39469056
Query processing time:3.7ms
Data Communication time: 83.0ms
Client rending time: 699.0ms

x1=31.52 y1=-9.705
x2=50.45 y2=-27.00

X: -91.5073, Y: 90.1910
```

Fig. 4. Prototype System Snapshot and Web Client GUI Layout

the appropriate query processing backend. The query string currently includes spatial extent, bin index range and desired level of details. There are different options to visualize query results at the client side. We have chosen to ask the backend to send back the results in the form of a collection of (x, y, level) triples, convert these triples into geographical coordinates and generate polygons so that they can be easily visualized in Flex-enabled runtime engines embedded in Web browsers through ArcGIS Flex API. While this approach is simple to implement and works well when the numbers of resulting quadrants are in the order up to a few thousands, we have found that the performance degrades as the numbers of resulting quadrants increase when the desired levels of details are high in some tests. We are in the process of exploring an image-based option that clients ask servers send back the query results in the form of binary images so that they can be visualized in Web browsers using less memory and computation resources. The visualization module is also responsible for requesting proper tiles from ArcGIS Server Manager and displays the returned images representing raw data with a predefined coloring schema. Note that client requests to the query processing backend and ArcGIS servers are completely independent. More information on this part is provided in Section 4.3.

4.2 Distributed Query Processing

As discussed before, the BMMQ-Tree based indices can be distributed across multiple shared-nothing machines to make visual exploration related query processing scalable. While there are quite a few options for workload distribution in facilitating parallel computing, the approach used in this study is based on a combination of spatial and categorical criteria which certainly lefts rooms for further refinements. As an example, the GLCF MODIS data from the University of Maryland comes with five parts [67], namely Africa, EuraAsia, North America, South America and Oceania. Each part has images at about 60 time periods and each image for a single period has 7 bands. Thus it is natural to decompose the spatial domain by the five parts. Depending on the available machines, the raster images in each part can be further decomposed by time and/or by bands. There are advantages and disadvantages of using fine and coarse level parallelization in this context. Using more computer nodes certainly reduces workload per node but also increases monetary costs. Furthermore, for processing conjunctive queries that involve multiple rasters (layers), if these rasters are indexed by a same computer node, query optimization is possible by early termination during synchronized traversals of co-located quadtrees.

4.3 Tiled Map Visualization Using ArcGIS

Fast visualizing and examining raw data are among the basic requirements for visual explorations of environmental data. Major Internet map providers, such as Google map and Microsoft Live map use the tile cache technology to speed up map rendering. GIS server software, such as ArcGIS [9] and MapServer+TileCache[10][11] provide similar functions to allow users to publish their own data as tiled maps that can be visualized in a variety of client software. We choose ArcGIS due to its technical maturity and easy of use. Map tiles at the different scales for all rasters have been generated in ArcGIS server. The ArcGIS Flex API [66] has made it easy to use tiled maps hosted by

ArcGIS servers in Web-based client applications. Note that, in our prototype design, visualizing the raw data is completely independent of processing ROI-finding queries. The resulting regions derived from ROI-finding queries can be overlaid on top of base maps to maximize the benefits of visual explorations. For example, users can compare raster cells inside the resulting ROIs with those nearby. While processing ROI-finding queries involves identifying cells from all relevant rasters which can be computationally intensive when sophisticated indexing technqiues are used, displaying base map only involves determining appropriate pre-generated image tiles under the spatial extent of the current active view, which is much less computationally expensive.

5 Experiments and Evaluation

5.1 Data and Experiments Setup

The experiments are designed for two purposes. First, we would like to examine the efficiency of the proposed binned min-max quadtree. It is clear that the sizes of the resulting quadtrees increase when the numbers of quadtree bins increase. At the same time, the false positives will decrease and the numbers of returned ROIs will increase. There are tradeoffs among disk/memory consumptions of constructed indices, rates of false positives, data communication costs and client visualization rendering costs. The later two are proportional to the number of returned ROIs. As we are not able to provide an analytical cost model for the binned min-max quadtree at this moment, the empirical results are important in understanding the efficiency of the data structure in real applications. Second, we would like to examine the overall performance of the end-to-end prototype system.

The data communication time also includes the overheads of data passing through different protocol stacks in the middleware and Web server in addition to data transport time over the network. To minimize the network traffic instabilities, in our experimens, the client machines and the server machines are co-located within our campus network. The client visualization rendering time includes parsing returned quadrants data of triples (x, y, level), converting the triples to squares in the real-world coordinate system that is being used at the client side and rendering the squares to the client visualization canvas embedded in a Web browser. We plan to use the current climate data published by WorldClim [68][69]. The dataset is the interpolations of observed data from 1950-2000 and includes monthly precipitation, minimum temperature and maximum temperature (12 month) and 18 derived bioclimatic variables at the global 30 arc-seconds (1 kilometers) resolution. Thus the number of grid cells is 432000*216000 for each of 12*3+18=54 rasters. Due to space limit, we only report the experiment results using the January precipitation data. For the January precipitation data, the number of cells with valid values is 222,265,892, which is about 23.82% of the total number of cells in the raster dataset. The percentage is slightly less than the percentage of land over the earth surface as there are no data for cells to the south of 60 degrees. The minimum and maximum precipitation values are 0 and 1003 millimeter, respectively. We have set the maximum level of the BMMQ-Tree to 16 as $2^{16} = 65536$ is already larger than 432000. We have used three bin numbers (8, 16 and 32) in our experiments for the reasons discussed above. The numbers of the leaf nodes of the corresponding quadtrees are

8,491,370, 15,036,155 and 25,877,417, respectively. These numbers translate to 110:1, 62:1 and 36:1 compression rates. We have performed extensive tests using different combinations of value ranges, spatial extents and quadtree bin numbers.However, due to space limit, only the results for precipitation range [90,300] with eight different spatial extents are reported in the following subsections. The cell selectivity rates of the eight queries vary from 0.23% to 5.76% using the total number of valid cells as the denominator.

5.2 Results of Query Processing

Results show that the query response times for the eight queries measured at the server side using the three bin numbers are 51-160, 42-162 and 47-252 milliseconds, respectively. All the response times are just fractions of a second which clearly shows the efficiency of the proposed BMMQ-Tree data structure in facilitating ROI-finding queries. As discussed previously, queries based on BMMQ-Tree are approximate in nature. Our experiments show that the larger bin sizes the smaller false rates, which is expected. The minimum and maximum false rates for B=32 are 12% and 53%, respectively, with an average of 23%. Visual inspections show that false positives are close to true positives and the resulting quadrants highlights the locations of true positives in a simplified manner which are often desirable in visual explorations.

5.3 Results of End-to-End Performance

As discussed before, the end-to-end performance of each query-driven visual exploration operation has three components: server query response time (T_{Server_Query}), data communication time (T_{Data_Comm}) and visualization rendering time ($T_{Vis\_Rendering}$). Due to space limit, we only report the three components for the eight queries using bin size 32 and the queries stop at the quadtree level 12 (determined based on ess-than-single-pixel stopping policy discussed in Section 3). In addition to showing the three categories of times for the eight queries in Table 1, we also report the resulting numbers of quadrants at the first column of the table (N_Q).

From Table 1, it is clear that both the data communication times and the visualization rendering times are proportional to the resulting number of quadrants which is expected. Compared with the server query response times, data communication times are generally one order greater and the visualization rendering times are more than one order

Table 1. End-to-end Performance Measurements with B=32 and MaxLevel=12 (in milliseconds)

Query ID	N_Q	$T_{Server-Query}$	$T_{Data-Comm}$	$T_{Vis-Rendering}$
Q1	13990	18.35	180.7	5329
Q2	12941	16.39	163.7	4741
Q3	31936	36.49	340.3	21667
Q4	13904	18.21	341.3	5790
Q5	3771	5.35	108.7	1262
Q6	15313	18.67	300.0	6647
Q7	21507	25.66	306.7	11705
Q8	11143	13.82	174.7	3896

greater than the data communication times. While the server query response times are excellent and the data communication times are adequate for visual explorations (less than 1 second), the visualization rendering times may be too large for the purpose.

To further investigate the rendering bottleneck incurred by ArcGIS Flex API, we have performed additional experiments on querying a value range of [200,1000] at the tree level 9 to 12 using the globe as the spatial extent. Note the value range corresponds to bins [22-14] with B=32. The results show that the visualization rendering time is acceptable for level=9 (0.693s) and level=10 (2.238s) while it becomes impractical for level=11 (8.2s) and level=12 (73.9s). We suspect that ArcGIS Flex API may be the bottleneck in rendering tens of thousands of squares. We are in the process of implementing our native square rendering algorithms in Adobe Flex. Another option would be requesting the query processing backend send back compressed images rather than individual quadrants as discussed in Section 4.1. On the other hand, we note that quadrants at level 10 have an approximate spatial resolution of 0.5 degree which is comparable to many datasets being used for global studies.

5.4 Discussions

As BMMQ-Tree and binned bitmap indexing [57] share many similarities, it is desirable to compare the two indexing approaches. However we found that direct comparisons are both inappropriate and difficult. First of all, binned bitmap indexing seeks to answer queries exactly at the individual data element (raster cell) level while BMMQ-Tree is primarily designed for visual explorations that allow approximate queries. Second, BMMQ-Tree query results naturally approximates connected components which are more suitable for further analysis of raster geospatial data. In contrast, binned bitmap requires expensive post-processing to find connected component [18]. BMMQ-Tree query results represent an intermediate step between individual raster cells and more complex connected components. Third, binned bitmap may generate large numbers of individual data element identifiers whose data volumes generally are too big to be transported to Web browsers for visualization purposes. Differently, BMMQ-Tree allows specify a maximum query level for a quadtree and thus reduce resulting data volumes at the expenses of higher false positive rates. The feature may be desirable in many visual exploration applications.

We believe exact queries based on FastBit can enhance our prototype system in the following way. For each client query, we first direct the query to FastBit and obtain the exact number of cells in the query result. We then direct the query to our query processing backend based on BMMQ-Tree indexing using a stopping quadrant level determined based on the less-than-single-pixel stopping policy discussed at the end of Section 3 and calculate the false positive ratio. Users then can determine whether to increase the stopping quadrant level based on the false positive rate. When the exact number of cells in a query result set is less than a predefined number (e.g., 10,000), the coordinates of the cells can be transported to the client side and visualized in Web browsers directly. Integrating FastBit with our prototype to allow visualization of exact query results is underway.

6 Conclusions and Future Work

Lacking interactive query-driven visualization supports for large-scale raster geospatial data in a Web environment has motivated us to develop a binned min-max quadtree data structure to index raster geospatial data. A prototype system that integrates commercial ArcGIS system with our query processing backed has been developed to demonstrate the feasibility and identify pitfalls. Experiments show a mixture of successes and failures. The BMMQ-Tree data structure is successful in the sense that it takes advantages of the approximate query nature of visual exploration based applications in both spatial and value dimensions. Our experiments show that BMMQ-Tree based indexing is able to process the WorldClim January Precipitation with nearly 1 billion cells in less than a quarter of a second for a query value range across multiple bins using a 32-bin, 16-level BMMQ-Tree. The resulting numbers of quadrants are in the order of a few thousands to a few hundreds of thousands whose data transport delays between servers and Web browsers range from excellent to acceptable. However, our experiment results also revealed that the ArcGIS Flex API that we used to render the resulting quadrants in Web browsers perform poorly for numbers of quadrants beyond a few thousands.

For the future work, our focus would be improving the visualization rendering speed at the client side. We will explore both options discussed in the text. We also want to integrate FastBit exact query with our prototype so that users can be aware of the quality of approximate query based visual explorations and retrieve exact query results when necessary. Finally, we are interested in profiling users queries and select better quadtree bin boundary values, in addition to providing more choices in automatically generating quadtree bin boundary values. We are expecting to work with environmental scientists more closely on this matter.

References

1. Li, Y.K., Bretschneider, T.R.: Semantic-sensitive satellite image retrieval. IEEE Transactions on Geoscience and Remote Sensing 45(4), 853–860 (2007)
2. USGS: Modis product table, `https://lpdaac.usgs.gov/lpdaac/products/modis_products_table`
3. WRF: Weather research and forecast, `http://www.wrf-model.org`
4. UCAR Unidata: Unidata integrated data viewer (idv), `http://www.unidata.ucar.edu/software/idv/`
5. NASA: Worldwind, `http://worldwind.arc.nasa.gov/java/`
6. Kothuri, R.K.V., Ravada, S., Abugov, D.: Quadtree and r-tree indexes in oracle spatial: a comparison using gis data. In: SIGMOD 2002, pp. 546–557 (2002)
7. Fang, Y., Friedman, M., Nair, G., Rys, M., Schmid, A.E.: Spatial indexing in microsoft sql server 2008. In: SIGMOD 2008, pp. 1207–1216 (2008)
8. Oracle: Georaster, `http://download.oracle.com/docs/html/B10827-01/geor-intro.htm`
9. ESRI: Arcgis server, `http://www.esri.com/software/arcgis/arcgisserver`
10. MapServer: Mapserver open source mapping, `http://mapserver.org/`
11. TileCache: Tilecache - web map tile caching, `http://tilecache.org/`

12. Wu, K., Koegler, W., Chen, J., Shoshani, A.: Using bitmap index for interactive exploration of large datasets. In: SSDBM 2003, pp. 65–74 (2003)
13. Stockinger, K., Shalf, J., Wu, K., Bethel, E.W.: Query-driven visualization of large data sets. In: IEEE Visualization, p. 22 (2005)
14. Glatter, M., Mollenhour, C., Huang, J., Gao, J.Z.: Scalable data servers for large multivariate volume visualization. IEEE TVCG 12(5), 1291–1298 (2006)
15. Kendall, W., Glatter, M., Huang, J., Peterka, T., Latham, R., Ross, R.: Terascale data organization for discovering multivariate climatic trends. In: Bergel, A., Fabry, J. (eds.) SC 2009. LNCS, vol. 5634, pp. 1–12. Springer, Heidelberg (2009)
16. Fuchs, R., Hauser, H.: Visualization of multi-variate scientific data. Computer Graphics Forum 28(6), 1670–1690 (2009)
17. Lawrence Berkeley National Laboratory: Fastbit, `https://sdm.lbl.gov/fastbit/`
18. Sinha, R.R., Winslett, M., Wu, K.: Finding regions of interest in large scientific datasets. In: SSDBM 2009, pp. 130–147 (2009)
19. Maceachren, A.M., Wachowicz, M., Edsall, R., Haug, D., Masters, R.: Constructing knowledge from multivariate spatiotemporal data: integrating geographical visualization with knowledge discovery in database methods. International Journal of Geographical Information Science 13(4), 311–334 (1999)
20. Andrienko, N., Andrienko, G., Gatalsky, P.: Exploratory spatio-temporal visualization: an analytical review. Journal of Visual Languages and Computing 14(6), 503–541 (2003)
21. Guo, D.S., Chen, J., MacEachren, A.M., Liao, K.: A visualization system for space-time and multivariate patterns (vis-stamp). IEEE TVCG 12(6), 1461–1474 (2006)
22. Anselin, L., Syabri, I., Kho, Y.: Geoda: An introduction to spatial data analysis. Geographical Analysis 38(1), 5–22 (2006)
23. Gahegan, M., Takatsuka, M., Wheeler, M., Hardisty, F.: Introducing geovista studio: an integrated suite of visualization and computational methods for exploration and knowledge construction in geography. Computers, Environment and Urban Systems 26(4), 267–292 (2002)
24. Rushing, J., Ramachandran, R., Nair, U., Graves, S., Welch, R., Lin, H.: Adam: a data mining toolkit for scientists and engineers. Computers & Geosciences 31(5), 607–618 (2005)
25. Zhang, J.T., Gruenwald, L., Gertz, M.: Vdm-rs: A visual data mining system for exploring and classifying remotely sensed images. Computers & Geosciences 35(9), 1827–1836 (2009)
26. Benz, U.C., Hofmann, P., Willhauck, G., Lingenfelder, I., Heynen, M.: Multi-resolution, object-oriented fuzzy analysis of remote sensing data for gis-ready information. ISPRS Journal of Photogrammetry and Remote Sensing 58(3-4), 239–258 (2004)
27. Liu, Y., Guo, Q.H., Kelly, M.: A framework of region-based spatial relations for non-overlapping features and its application in object based image analysis. ISPRS Journal of Photogrammetry and Remote Sensing 63(4), 461–475 (2008)
28. Goodchild, M.F., Yuan, M., Cova, T.J.: Towards a general theory of geographic representation in gis. Int. J. of Geographical Information Science 21(3), 239–260 (2007)
29. Cohen, S., Hurley, P., Schulz, K.W., Barth, W.L., Benton, B.: Scientific formats for object-relational database systems: a study of suitability and performance. SIGMOD Rec. 35(2), 10–15 (2006)
30. Stancu-Mara, S., Baumann, P.: A comparative benchmark of large objects in relational databases. In: IDEAS 2008, pp. 277–284 (2008)
31. Abadi, D.J., Madden, S.R., Hachem, N.: Column-stores vs. row-stores: how different are they really? In: SIGMOD 2008 (2008)
32. Pavlo, A., Paulson, E., Rasin, A., Abadi, D.J., DeWitt, D.J., Madden, S., Stonebraker, M.: A comparison of approaches to large-scale data analysis. In: SIGMOD 2009, pp. 165–178 (2009)
33. UCAR Unidata: Netcdf, `http://www.unidata.ucar.edu/software/netcdf/`

34. The HDF Group: Hdf5, `http://www.hdfgroup.org/HDF5/`
35. Baumann, P., Furtado, P., Ritsch, R., Widmann, N.: The rasdaman approach to multidimensional database management. In: SAC 1997, pp. 166–173 (1997)
36. Marathe, A.P., Salem, K.: Query processing techniques for arrays. The VLDB Journal 11(1), 68–91 (2002)
37. Baumann, P.: Designing a geo-scientific request language - a database approach. In: SSDBM 2009 (2009)
38. Sarawagi, S., Stonebraker, M.: Efficient organization of large multidimensional arrays. In: ICDE 1994, pp. 328–336 (1994)
39. Otoo, E.J., Rotem, D.: Efficient storage allocation of large-scale extendible multi-dimensional scientific datasets. In: SSDBM 2006, pp. 179–183 (2006)
40. Kim, J., JaJa, J.: Component-based data layout for efficient slicing of very large multidimensional volumetric data. In: SSDBM 2007, p. 8 (2007)
41. Gaede, V., Gunther, O.: Multidimensional access methods. ACM Computing Surveys 30(2), 170–231 (1998)
42. Samet, H.: Foundations of Multidimensional and Metric Data Structures. Morgan Kaufmann Publishers Inc., San Francisco (2005)
43. Cignoni, P., Marino, P., Montani, C., Puppo, E., Scopigno, R.: Speeding up isosurface extraction using interval trees. IEEE TVCG 3(2), 158–170 (1997)
44. Wilhelms, J., Vangelder, A.: Octrees for faster isosurface generation. ACM Transactions on Graphics 11(3), 201–227 (1992)
45. Wang, C., Chiang, Y.J.: Isosurface extraction and view-dependent filtering from time-varying fields using persistent time-octree (ptot). IEEE TVCG 15(6), 1367–1374 (2009)
46. Gress, A., Klein, R.: Efficient representation and extraction of 2-manifold isosurfaces using kd-trees. Graphical Models 66(6), 370–397 (2004)
47. Hughes, D.M., Lim, I.S.: Kd-jump: a path-preserving stackless traversal for faster isosurface raytracing on gpus. IEEE TVCG 15(6), 1555–1562 (2009)
48. Lin, T.W.: Compressed quadtree representations for storing similar images. Image and Vision Computing 15(11), 833–843 (1997)
49. Chan, Y.K., Chang, C.C.: Block image retrieval based on a compressed linear quadtree. Image and Vision Computing 22(5), 391–397 (2004)
50. Chung, K.L., Liu, Y.W., Yan, W.M.: A hybrid gray image representation using spatial- and dct-based approach with application to moment computation. Journal of Visual Communication and Image Representation 17(6), 1209–1226 (2006)
51. Vassilakopoulos, M., Manolopoulos, Y., Economou, K.: Overlapping quadtrees for the representation of similar images. Image and Vision Computing 11(5), 257–262 (1993)
52. Manolopoulos, Y., Nardelli, E., Papadopoulos, A., Proietti, G.: Mof-tree: a spatial access method to manipulate multiple overlapping features. Inf. Syst. 22(9), 465–481 (1997)
53. Nardelli, E., Proietti, G.: An efficient spatial access method for spatial images containing multiple non-overlapping features. Information Systems 25(8), 553–568 (2000)
54. Tzouramanis, T., Vassilakopoulos, M., Manolopoulos, Y.: On the generation of time-evolving regional data. Geoinformatica 6(3), 207–231 (2002)
55. Manolopoulos, Y., Nardelli, E., Proietti, G., Tousidou, E.: A generalized comparison of linear representations of thematic layers. Data & Knowledge Engineering 37(1), 1–23 (2001)
56. Manouvrier, M., Rukoz, M., Jomier, G.: Quadtree representations for storage and manipulation of clusters of images. Image and Vision Computing 20(7), 513–527 (2002)
57. Wu, K., Stockinger, K., Shoshani, A.: Breaking the curse of cardinality on bitmap indexes. In: Ludäscher, B., Mamoulis, N. (eds.) SSDBM 2008. LNCS, vol. 5069, pp. 348–365. Springer, Heidelberg (2008)
58. Sinha, R.R., Winslett, M.: Multi-resolution bitmap indexes for scientific data. ACM Trans. Database Syst. 32(3), 16 (2007)

59. Rubel, O., Wu, K., et al.: High performance multivariate visual data exploration for extremely large data. In: Pautasso, C., Tanter, É. (eds.) SC 2008. LNCS, vol. 4954, pp. 1–12. Springer, Heidelberg (2008)
60. Gosink, L.J., Anderson, J.C., Bethel, E.W., Joy, K.I.: Query-driven visualization of time-varying adaptive mesh refinement data. IEEE TVCG 14(6), 1715–1722 (2008)
61. Wood, J., Dykes, J., Slingsby, A., Clarke, K.: Interactive visual exploration of a large spatio-temporal dataset: Reflections on a geovisualization mashup. IEEE TVCG 13(6), 1176–1183 (2007)
62. Dork, M., Carpendale, S., Collins, C., Williamson, C.: Visgets: Coordinated visualizations for web-based information exploration and discovery. IEEE TVCG 14(6), 1205–1212 (2008)
63. Tobler, W.: A computer model simulating urban growth in the detroit region. Economic Geography 46(2), 234–240 (1970)
64. GDAL: Geospatial data abstraction library, `http://www.gdal.org/`
65. Adobe: Adobe flex api, `http://www.adobe.com/products/flex/`
66. ESRI: Arcgis flex api, `http://www.adobe.com/products/flex/`
67. University of Maryland: Global land cover facility (glcf), `ftp://ftp.glcf.umiacs.umd.edu/modis/500m/`
68. Hijmans, R.J., Cameron, S.E., Parra, J.L., Jones, P.G., Jarvis, A.: Very high resolution interpolated climate surfaces for global land areas. International Journal of Climatology 25(15), 1965–1978 (2005)
69. WorldClim: Worldclim current conditions data 1950-2000, `http://www.worldclim.org/current`