

# Multiresolution Spatial Databases: Making Web-based Spatial Applications Faster

Xiaofang Zhou, Sham Prasher, Sai Sun and Kai Xu

School of Information Technology and Electrical Engineering  
University of Queensland, Australia  
{zxf, sham, sunsai, kaixu}@itee.uq.edu.au

**Abstract.** Spatial data has now been used extensively in the Web environment, providing online customized maps and supporting map-based applications. The full potential of Web-based spatial applications, however, has yet to be achieved due to performance issues related to the large sizes and high complexity of spatial data. In this paper, we introduce a multiresolution approach to spatial data management and query processing such that the database server can choose spatial data at the right resolution level for different Web applications. One highly desirable property of the proposed approach is that the server-side processing cost and network traffic can be reduced when the level of resolution required by applications are low. Another advantage is that our approach pushes complex multiresolution structures and algorithms into the spatial database engine. That is, the developer of spatial Web applications needs not to be concerned with such complexity. This paper explains the basic idea, technical feasibility and applications of multiresolution spatial databases.

## 1 Introduction

A very large number of websites now support various spatial applications. Maps are used for assisting users to formulate location-based queries and as a background for displaying search results [15]. Many ‘location-aware’ websites allow the users to interact with a map, to zoom and pan on the client side, and to click spatial objects to request further information. More sophisticated map-based applications can generate dynamic maps according to the user’s intension and display devices [11].

Many Web-based spatial applications nowadays are enabled by a spatial database engine on the server side, which manages spatial data and supports non-trivial spatial applications with powerful spatial query processing capability and a simple API. Comparing to standalone Geographical Information Systems (GIS) applications and traditional, non-spatial Web applications, Web-based spatial applications encounter a number of technical issues such as network bandwidth constraint, interoperability, and integration of spatial services with other enterprise applications. This paper focuses on performance issues of spatial Web applications, proposing a new approach to spatial data management and query

processing by exploring the multiresolution nature of spatial data. Resolution is an important concept associated with spatial data. It is usually defined as the minimum geometric measure that an object (or a part of) must have in order to be of interest. For spatial data in vector format (a spatial data format representing the geometry of spatial objects as a sequence of points, more compact and versatile than raster image), a lower resolution means that a smaller number of points may be sufficient to describe the geometry of spatial objects, as some different points at a high resolution level may become indistinguishable or insignificant at a lower resolution level thus can be merged for some applications. While spatial data is typically stored in a spatial database system with the highest level of resolution available, it is obvious that not all applications using the same database require the same level of resolution. For example, when the area of a suburb is displayed on a hand-held device for navigational purpose, a highly simplified set of spatial data is more preferable than the original data that might contain too many data objects and too much detail to be clearly displayed on that device. Excessive amount of detail is not very useful to an application that does not need that level of detail (and sometimes counter productive in terms of the quality of rendered image). However, it does waste resources for retrieving, transferring and processing data. Therefore, it is highly desirable to use a set of spatial objects with both the number of objects and the complexity of individual objects minimized for a given application. By doing so, data retrieval operations and other operations (e.g., data transferring, rendering and manipulation) can be performed more efficiently without sacrificing the data quality required by the application. This issue becomes more relevant and important for Web-based spatial applications, which are much more diversified than standalone spatial applications. Clearly, most advantages can be gained if such data simplification can be performed within the spatial database system, as spatial query processing is often one of the most time-consuming operations and the benefit of a reduced query result size will follow on to other post-query operations, such as data manipulation and transfer.

We propose to use multiresolution spatial databases for Web-based spatial applications. A multiresolution spatial database system has built-in spatial simplification and generalization capability. In this paper, we examine basic data structures for storing spatial data in the database. As spatial objects will be used in different and simplified forms for different applications (i.e., their geometry may be generalized), it is necessary to investigate the feasibility of recording a spatial object with finer granularity, such as line segments or points. We propose to use a basic vector format based on points for storing spatial objects, and investigate the overhead associated with fragmented data storage in the context of simplifications when retrieving at lower resolution levels. This idea will then be tested for a number of common Web-based spatial applications, including map generalization, progressive vector data transfer, and approximate spatial query processing. A significant overall saving can be achieved for these important applications, and at the application level the spatial queries to a multiresolution spatial database remain simple. One main contribution of this paper is the argu-

ment that spatial objects can no longer be considered as an opaque structure as used by most spatial database systems today, and spatial simplification and generalization operations can be performed within the database system, as opposed to the traditional approach of post-database processing which in favors of spatial analytical operations that typically need the most accurate data available and treat spatial objects in entirety. For emerging applications such as Web-based spatial applications, a finer level of structures for storing spatial objects make it possible to have significant performance gains without retrieving whole objects first and then simply them later in application programs.

The remainder of this paper is organized as follows. In section 2, we compare multi-representation and multiresolution spatial databases. The multiresolution approach is introduced in section 3, proposing a scaleless spatial data structure and query processing models. A number of Web-based spatial applications that can benefit from the proposed approach are discussed in section 4. We conclude this paper in section 5.

## 2 Multiresolution and Multi-representation

For a spatial database system to support applications that require variable level of resolution, one approach, called the *multi-representation* approach, is to pre-generate and store spatial data at different resolution levels. The other approach, called the *multiresolution* approach, is to store only the data at the highest level of resolution and simplify and generalize data dynamically. In this section, we explain these two approaches and compare their advantages and disadvantages.

The multi-representation approach physically stores data at different scales in a database [3]. Data at different level of resolutions are pre-generated, and issues related to topological, metric and geometric correctness can be solved at the time of pre-processing by human experts. It incurs no overhead associated with on-the-fly spatial generalization, thus can be more efficient. This concept is similar to have multiple paper-based maps at different scales and for different purposes (such as tourist maps and military maps). A higher storage overhead and difficulties arising from object updates are among the problems related to this approach; however, a more significant problem preventing this approach from being generally useful is that the level of details (LoDs) required by different applications, or the same application at different stages, can vary. It is known that LoDs should meet the law of *constant information density* [3]. One way to state the law is that the rendered image on a target display should maintain a constant ratio of the number of pixels for objects over the number of pixels for the background (in other words, information is not too visually crowded). Information density, nonetheless, is dependant on a query (e.g., the size of the area of interest, and the spatial and non-spatial conditions for object selection in the query), as well as the resolution of the target display. The required LoD of the resultant spatial data from a query can only be determined when all these parameters are known. The number of possible combinations of spatial and non-spatial query conditions and target display sizes can be prohibitively

large for any approach based on materialization. Another reason that a spatial dataset cannot always be generalized independent of queries is that spatial data typically consists of a large number of thematic layers (e.g., roads, soil types, water systems, vegetations etc). Spatial generalization requires validating spatial objects with their surrounding objects, which vary depending on which layers are required. Validating against all layers in the database (up to 100 for some environmental applications) may over-constrain spatial objects and impact on generalization quality adversely.

The approach of multiresolution spatial databases, on the other side, attempts to provide support for deriving proper LoDs on demand [5, 2]. With the data at the finest available LoD stored in the database, a multiresolution database system is capable to dynamically reduce LoDs according to applications. Such on-demand deviation of spatial data can be done by a process known as cartographical generalization, which is a process to “derive, from a data source, a symbolically or digitally encoded cartographic data set ... to reduce in scope, the amount, type, and cartographical portrayal of the mapped or encoded data consistent with the chosen map purpose and intended audience; and to maintain clarity of presentation at the target scale” [8].

Generalization of spatial data is a non-trivial task. It is a step beyond object selection, involving object simplification and other operations. Simplification of spatial objects may lead to alternation of object geometry, dimensionality (e.g., rivers with both banks may simplified into a polyline), and sometimes existence (i.e., an object is not displayed at all). There are three aspects to be taken into account for a proper generalization-based approach. Firstly, the generalized data should be “correct”. This means that spatial relationships, which can be topological (e.g., containment and connectivity), directional (e.g., left, right, above and below) and metrical (e.g., relative distance between two objects from any given object), among the objects before and after generalization should be the same or at least “similar”. Generalization of an object in the vector format will remove some vertices from its geometry. While there are many generalization algorithms that attempt to make a generalized object bear maximum geometric similarity to the original object, an important yet very much under investigated goal is to maintain all spatial relationships among any pair of objects. This goal can only be met by treating a spatial data set as a whole. It is much harder to achieve such a holistic goal than preserving geometric similarity of individual objects. Secondly, the generalized data should maintain the general look and feel of the original data set. This visual aspect of generalization not only requires individual objects to maintain their original geometric properties (locations and shapes), but also the overall impression perceived by a human user. One goal of spatial generalization is to enhance the clarity of a map. This often involves operations as tokenization (replacing geometrically small but semantically important objects, such as telephone boxes and police stations, by a token), and amalgamation (merging similar objects, such as several neighboring shops, into a large object). Thirdly, generalization should be done efficiently for it to be useful for interactive applications such as map-based navigation for WWW ap-

plications. This performance aspect has been a secondary issue in cartographic generalization, which mainly concerns the production of paper-based maps. In the context of spatial database based generalization, it is highly desirable to retrieve a minimum amount of data from the database, as explained before. Any brute-force approach retrieving objects from database first and then discarding many of them should be avoided whenever possible. Unfortunately, there is little work in literature on spatial database support for generalization; and most work on spatial generalization does not consider database issues either.

It is clear that much research work is needed for either approaches to be practically useful, in particular for Web-based spatial applications where the resolution level required vary significantly (consider an application using map-based navigation starting from a large region and zooming into street level).

### 3 Multiresolution Spatial Databases

We observe that a spatial dataset in the database is always an approximation of real world objects to a certain level of accuracy. A higher resolution leads to a more accurate approximation, but at the same time can cause higher overhead for both data retrieval and processing. The goal of a multiresolution spatial database system is, therefore, to find a systematic way to *generalize* the spatial data stored in the database such that the spatial data required by an application is returned with the right LoD for the application. By minimizing the amount of data required by an application, this approach can speed up operations in data retrieval and complement many other performance improvement strategies and algorithms with a reduced size and complexity of input data. In this section, we will propose a foundation for a multiresolution spatial database based on the relational database technology as most spatial database systems products do (such as Oracle).

#### 3.1 Scaleless Spatial Data Structures

Let us consider the internal representation of polygonal data in a spatial database system. The most common approach is to store the sequence of points of a polygon together, essentially as an opaque block (using either the relational or object-relational model) [4]. This approach is typified by Oracle in its Spatial Data Cartridge [13]. The advantage of this structure is that one object can be accessed directly to avoid the overhead of accessing otherwise fragmented components and for object reconstruction. This approach, while being the dominate structure used by modern spatial database systems, limits the scope of spatial generalization not allowing in-database generalization of individual objects. Therefore, spatial generalization can only be done as a post-query process.

An alternative view, called topology-based structure and used by many GIS products such as ARC/INFO, is centered on line segments where each line segment is the unit of data storage and comes with IDs for the polygons on each side of the line. This approach requires slightly less storage than the polygon-based

approach above, as a line shared by two polygons is recorded once. More significant benefit of this structure is that it is very efficient for topology-based spatial analysis such as disjoint, connectivity and adjacency, which are important applications for GIS. The disadvantages of this approach include high maintenance costs (insertion, modification or deletion of an object may cause updating of other objects), and higher costs for reconstruct a complete object.

One simplest and oldest way to represent spatial objects is to use point as the unit of storage. This has been used in the early days to use relational databases to store spatial objects (i.e., each point is stored as a record and the points of the same polygon are linked using the polygon ID and are ordered explicitly). This approach was quickly abandoned as it has very high overhead to fetch and combine many records for an object [4].

Now we revisit the issue of basic data storage in the light of multiresolution databases. Spatial generalization to reduce resolution levels involve a selection of objects (which objects will be selected, typically according to non-spatial attributes but also possible to select according to spatial attributes), as well as a selection of parts of objects (i.e., generalization of spatial objects according to resolution levels). The polygon-based structures are good in support object selections. But for spatial generalization, they require fetching whole objects out of the database and then simplifying them, causing extra I/O costs for retrieving extraneous data and extra CPU time for simplifying them. They are suitable for systems that consider one resolution level only. For line- and point-based structures, they are more flexible as lines or points can be selected to reconstruct objects at different resolutions. However, the overhead of accessing object fragments and assembling them have to be justified comparing to possible savings from reduced amount of data retrieved, and there must be a set of efficient algorithms to maximize such benefits.

There are two aspects of data reduction from the multiresolution concept. First, the number of points may be reduced when the level of resolution reduces. Second, for each point, the precision for point data representation (i.e., the numerical precision used to record the  $x$  and  $y$  coordinates of a point) may also be reduced. Now we propose a scaleless data structure for polygonal data such that the number of points and numeric precision of points can be automatically changed with the level of resolution. This scaleless data structure is based on the well-known  $z$ -values [12], which works as the follows. The whole data space is divided into four quadrants in the center, with the four quadrants numbered as 0 to 3 following certain order. These quadrants can be further divided and numbered recursively. The  $z$ -value of a quadrant is determined using the following recursive steps: (1) The  $z$ -value of the initial space is 1; and (2) The  $z$ -value of the  $i^{th}$  sub-quadrant,  $0 \leq i \leq 3$ , of a quadrant whose  $z$ -value is  $z$ , is  $z \cdot i$  (i.e., appending  $i$  at the end of  $z$ ). A point can be approximated by a small enough rectangle obtained by the above recursive decomposition method, thus its  $(x, y)$  coordinates can be expressed using a  $z$ -value. The length of a  $z$ -value indicates the number of decompositions from the root space to the quadrant represented by the  $z$ -value, and indicates the size of that quadrant too (relative to the size of

the root space). For most applications, a decomposition level of 20 to 30 is sufficient to represent data points with sub-centimeter accuracy. The storage cost for representing 20-30 base-4 digits is comparable and sometimes smaller than that for storing two coordinate numbers (with float or double precision). Truncating a z-value from the right end increases the size of the corresponding quadrant; in other words, reduces its resolution. In addition, those points that fall into the same quadrant after truncating will have the same truncated z-values, thus can be identified easily and choose one representative (as others are not distinguishable from that point at the reduced resolution level). This property is precisely what we need in the multiresolution database approach.

In its simplest form, a scaleless data structure for representing polygonal data can be defined as `pointPolygon (pid, ind, z)`, where `pid` is the polygon id, `z` is the z-value for a point (at the highest resolution level for the dataset), and `ind` is the index number of the point in the polygon. The following SQL query, albeit inefficient in execution and not robust in object geometry (see discussions in Section 4), gives an idea about how this scaleless data structure works to support multiresolution data retrieval inside the spatial database system:

```

select pid, min(ind), substr(z, 1, k)
from pointPolygon
where /* other spatial and non-spatial query conditions */
group by substr(z, 1, k);

```

where `substr(z, 1, k)` is the SQL string function returns the first  $k$  characters of string  $z$ ,  $k$  in this query is the level of decomposition that leads to a quadrant whose size is the smallest one larger than the acceptable resolution level. To reconstruct the simplified polygon, the selected points will be connected in ascending order of `min(ind)`, as some consecutive points might become indistinguishable at resolution level  $k$  thus only one representative point is fetched. This crude table structure and SQL query will be refined for different applications in section 4.

For an application that only require to access spatial data at a particular resolution, the most efficient way is, of course, to have the data pre-generalized at that resolution level (i.e., adopting the multi-representation approach). It is a trade-off between data management cost (storage and maintenance overhead) and on-the-fly generalization cost (data retrieval and object assembling overhead for the multiresolution approach). We have experimented with various simulated and real spatial data sets of variable complexity (i.e., the number of points and complexity of geometric shapes) and for various spatial operations (see Section 4). We found that the overhead of simplify-while-retrieve approach based on the scaleless data structure is significant but not very large. When the level of resolution is low, this overhead can be much smaller than the savings from reduced object sizes.

Notice that the z-value encoding scheme is just one hierarchical encoding methods, and other methods, such as the QTM model proposed in [2], are possible (and conceptually similar to our discussions based on z-values).

### 3.2 Multiresolution Spatial Query Processing

The proposed scaleless data structure above essentially puts *raster-like* properties (efficient for visualization, overlay and other space-related operations) into vector spatial data (good for object-centric operations), thus has the potential to have the advantages of the both types of spatial data representation formats. They also provide a natural way of supporting *multiple approximations* of spatial objects and maintaining vertical links among the same data at different (virtual) resolution levels.

It is not practical for the Web user to specify the resolution level they need when accessing the spatial database. We need to find a concept that is both natural to the end user and easy to be translated into SQL queries to access the multiresolution database. The following accuracy models can be used by the user for different application scenarios:

1. *the visual quality model*: for applications that render the query results into a map image and display in a target device (such as a PDA or a Web browser), a natural accuracy model is that the resolution level for the spatial data to be retrieved from the multiresolution database can be lowered to a level that the rendered image, after being displayed on the target device, looks no differently to the image rendered using any higher resolution data on that device. This model is suitable for most applications that use spatial data to draw a map for the user to read or to interact with. Those topological, metric or geometric problems usually found in spatial generalization may still exist but are tolerable as far as visual quality is concerned. Data reduction comes from the fact that a pixel on the target device may correspond to a large area in the actual data space and the details inside such an area are not discernible thus can be suppressed before spatial objects are fetched out from the database.
2. *the aggregation model*: for those applications that require only aggregated information from a spatial query, such as the total number or size of the objects interact with, contained by or within a distance of another object or each other, the user may specify that a percentage of deviation is acceptable as long as such inaccuracy can be guaranteed within the specified range.
3. *the probability model*: each object returned by the multiresolution database for a query using lower resolution data comes with a probability to indicate the likelihood of that object in the query result should the highest resolution data is used. The user can specify a probability cut-off value for filtering the query results.

For both the aggregation model and the probability model, a 100% value from the user implies the need to use the highest resolution data. While these models are intuitive for the end-users, it requires many meta-data to be maintained by the database system in order to estimate the accuracy or probability variation when resolution level changes for queries involving different spatial operations. This problem is in general very difficult and no research has been done in assisting such estimation until very recently [7].

## 4 Applications

In this section, we discuss some common Web-based spatial applications that can benefit from multiresolution spatial databases, using different accuracy models. Due to the limit of paper pages, detailed performance analyses are not discussed in this paper.

### 4.1 Automatic Map Generalization

Map generalization is the process of deriving small-scale maps (i.e., at a lower resolution) from larger ones while maintaining acceptable level of topological, metric and geometric resemblance to the original [8]. It is difficult to algorithmically define the correctness criteria of map generalization that results to changes in not only object shapes but also topological and other relationships with other objects, thus this process is regarded an interactive process which needs human cartographers involved in judging the correctness and quality of a generalized map and making fine tuning and necessary corrections.

A concept of perfect generalization is proposed in [9], which adopts the visual quality accuracy model that bears similarity to raster image resolution reduction. This concept is built on the well-known Li-Openshaw line generalization algorithm [6]. Given the spatial extent of a query (i.e., the minimum rectangle of  $w \times h$  units containing all spatial objects of the query) and the resolution of the display device (of  $x \times y$  pixels), a *data pixel* can be defined as an area of size  $x' \times y'$  where  $x' = \text{ceil}(w/x)$  and  $y' = \text{ceil}(h/y)$ . Then, all points in a single data pixel are not distinguishable after the data is rendered in the display device, thus can collapse into a single point without causing visually-detectable metric, topological or geometric distortions for the given query and display device, as long as some special cases such as the points of different objects and the non-consecutive points of one object remain separate. We extend the scaleless data structure discussed in section 3.1 by adding a number  $\delta$  to each point record, where  $\delta$  is a number from which the z-values for the point itself and the next point in the polygon start to differ (counting from the left). That is, this point is one end point of a line segment that crosses the boarder of a quadrant obtained by  $\delta - 1$  times decomposition. Thus, if a data pixel is at level  $k$ , only those points with  $\delta > k$  needs to be retrieved. In such a way, the efficiency problem caused by using the substring and aggregate functions (which cannot make use of database indexes efficiently) and possible errors caused by the same polygon enters the same data pixel multiple times are solved. In other words, the perfect generalization of polygonal data (and the Li-Openshaw algorithm) can be implemented using a single SQL query with only one extra condition added which is  $\delta > k$ . Map generalization is one of the most important functions to support Web-based spatial applications. The performance of using the revised scaleless data structure and a single SQL query for both data retrieval and simplification is shown to be significantly better than the traditional retrieval-then-simplify approach [11].

## 4.2 Progressive Vector Data Transfer

Several most popular raster image formats, such as GIF and JPEG, have a property called *progressive encoding*, that allows creating a rough image preview while the image is downloaded, and the clarity improves gradually when more data is downloaded. This property is very important for Web applications, as it holds the users attention while a large image is being downloaded, and allows the user to click to other pages if they have obtained sufficient information before the download is complete. This behavior is, however, very difficult to support for vector spatial data, as adding more points can change geometric and topological relationships. A rough image rendered using incomplete vector data therefore can be misleading. Progressive vector data transfer is a highly desirable property for spatial data, but there is no satisfactory solution to this problem. A basic common assumption for progressive vector data transfer is that all the points received earlier must be a subset of the points of the final object, so the newly added points can be used together with, not overwrite, those already received. It is easy to see that all points must carry information about their index in the final object [1].

Now we give a simple and efficient solution to this problem using the idea of the scaleless data structure, same to what used in map generalization. Points of a polygon are sent to the client according to their  $\delta$  in descending order, and the image on the client side is rendered in a ‘blocky’ way according to the current  $\delta$  being transferred, and the granularity of ‘blocks’ are reduced to improve image quality on the client side when more data is received. This approach ensures that the rough image do not give the user misleading information and at the same time, allows transferring of vector data in a progressive way. As images of very large ‘blocks’ are not useful and can be annoying to the user, the initial  $\delta$  should not start from 1, but from a more reasonable number that considers both the client-side block size and the amount of data initially transferred.

We are currently evaluating different vector data compression methods such that the data with a large  $\delta$  value can be sent with low precision too. Other important issues that need to be considered to make progressive vector data transfer really useful include client and proxy side vector data caching and integration with W3Cs standard for vector data exchange (such as SVG).

## 4.3 Approximate Spatial Query Processing

Spatial join operations combine two or several spatial datasets using spatial relationships such as overlap, adjacent, encloses or within distance [4]. They are among the most time-consuming yet most important operations in a spatial DBMS. Traditionally the filter-and-refine approach is used to minimize spatial join processing cost. It has two steps: a filter step which applies some simple operations on approximations of spatial objects, followed by a refinement step where a full test using complete geometry is applied to those objects survived the filter condition. For example, the minimum bounding box intersection is often used

as the filter condition for polygon intersection. Typically, object approximations are used together with a spatial index.

Multi-resolution database makes it possible to extend the principle of the filter-and-refine approach further. Firstly, the full geometry used in the refinement stage needs not always to be the highest resolution data in the database; a suitable level of resolution, depending on the accuracy models used (either the aggregation model or the probability model), will be determined for a given application and the data at that level can be used for refinement. This will reduce the cost of refinement for the applications that do not require the highest level of resolution. Secondly, the data at a lower resolution level can be used as approximations for its counter parts at a higher resolution level. Thus, multiple filters are available in a multi-resolution database, and can be used to minimize the need to use more complex data. Using the scaleless data structure introduced before, a spatial object can be approximated differently when  $\delta$  changes. The traditional filter-and-refine strategy becomes a simple-box-or-full-geometry special case. Scaleless data structures can produce similar but more flexible approximations, with virtually no extra storage or maintenance overhead.

It should be pointed out that scaleless data based approximation is neither fully conservative nor fully progressive, thus may not be suitable for applications that require precise results based on the highest resolution data. We are working on defining relevant metadata to be collected to find methods to determine the lowest acceptable level of resolution for a query based on the aggregation model and the probability model. Using the aggregation model (which is simpler but less useful than the probability model), we have observed up to 75% reduction of disk access costs while the accuracy is still within 80%. A more comprehensive empirical evaluation is underway now, and research to develop a probability model has just been started.

#### 4.4 Other Applications

The scaleless data structure has also been applied to many other applications in our lab, including view-dependant multiresolution terrain visualization using the visual quality model [14], 3D spatial analysis (in particular, surface  $k$  nearest neighbor queries) using the aggregation model, spatial data amalgamation for spatial data mining and warehousing applications [10], and applications for environmental analysis and location-based services (mainly spatial generalization and tokenization). The underlying data structures for these projects, while all similar and all based on the scaleless data structure we introduced in this paper, are customized to specific application problems. It is necessary to consider in the near future how to unify these optimized structures for different applications in a general-purpose spatial DBMS.

## 5 Conclusions

Spatial data has the multiresolution nature. This important property has not been adequately considered in the past to improve the performance of spatial

data management and query processing, which now becomes a primary barrier to more efficient and sophisticated spatial application in the Web environment where bandwidth can be limited and applications are much more diversified. In this paper, we address this key technical issue, proposing a novel solution to improve the performance of spatial applications by exploring the multiresolution nature of spatial data. Spatial objects are stored using scaleless data structures. We have shown that the overhead of assembling spatial objects can be compensated by a reduced amount of data to be retrieved from the database. The benefits of in-database simplification of spatial objects flow on to database retrievals, spatial operation processing, post-query processing and server-to-client data transfer. It also makes it possible, for the first time, to support real progressive spatial data transfer and semantic spatial caching. We have discussed a number of important spatial applications to demonstrate that multiresolution spatial databases are ideal for Web-based spatial applications.

**Acknowledgment:** The work reported in this paper has been partially supported by grant DP0345710 from the Australian Research Council.

## References

1. M. Bertolotto and M. Egenhofer. Progressive vector transmission. In *ACM GIS*, pages 152–157, 1999.
2. G Dutton. Digital map generalization using a hierarchical coordinate system. In *Auto Carto*, 1997.
3. A. U. Frank and S. Timpf. Multiple representations for cartographical objects in a multi-scale tree - an intelligent graphical zoom. *Computers and Graphics*, 18(6):823–829, 1994.
4. R. H. Güting. An introduction to spatial database systems. *VLDB Journal*, 3(4):357–399, 1994.
5. C. B. Jones and D. B. Kinder. Database design for a multi-scale spatial information system. *J. GIS*, 10(8):901–920, 1996.
6. Z. Li and S. Openshaw. Algorithms for automated line generalization based on a natural principle of objective generalization. *J. GIS*, 6(5):373–389, 1992.
7. X. Lin, Q. Liu, Y. Yuan, and X. Zhou. Multiscale histograms: Summarizing topological relations in large spatial datasets. In *VLDB*, pages 814–825, 2003.
8. R. B. McMaster and K. S. Shea. *Generalization in Cartography*. Association of American Geographers, Washington, D.C., 1992.
9. P Prasher. Perfect cartographic generalisation and visualisation. In *VDB*, 2002.
10. S. Prasher and X. Zhou. Multiresolution amalgamation: Dynamic spatial data cube generation. In *ADC*, pages 103–111, 2004.
11. S. Prasher, X. Zhou, and M. Kitsuregawa. Dynamic multi-resolution spatial object derivation for mobile and WWW applications. *J. WWW*, 6(3):305–325, 2003.
12. H. Samet. *Applications of Spatial Data Structures*. Addison-Wesley, 1990.
13. J. Sharma. *Oracle Spatial: an Oracle technical white paper*. Oracle Technical Network, 2002.
14. K. Xu, X. Zhou, and X. Lin. Direct mesh: an multiresolution approach to terrain visualization. In *ICDE*, page to appear, 2004.
15. J. Zhou, X., Yates, and G. Chen. Using visual spatial search interface for WWW applications. *Info. Sys.*, 6(2):61–74, 2001.