An SVG-based method to support spatial analysis in XML/GML/SVG-based WebGIS

Haosheng Huang, Yan Li, Georg Gartner & Yunpeng Wang

Abstract:

XML/GML/SVG based approaches are promising for building Web-based geographic information system (WebGIS). However, current XML/GML/SVG based WebGISs are lacking in spatial analysis. Some of them are designed for web mapping only. Others adopt a server-side solution for spatial analysis, which suffers from the “bottleneck” problem, and results in a high network transmission load. Load balancing spatial analysis between server side and browser side can be used to solve the above problems. This paper focuses on one of the key building blocks of load balancing spatial analysis, i.e., SVG-based spatial analysis which enables spatial querying and analysis directly on SVG (on the browser side). After analyzing the workflow of spatial analysis, we identify and focus on two key issues in providing spatial analysis on SVG: SVG-based spatial information representation and SVG-based Spatial Extended SQL (SSESQL). For the first issue, a theoretical foundation is set up to develop an SVG-based spatial information representation model. Some spatial operators are designed and integrated into an SSESQL to support spatial querying on SVG. Finally we design and implement two case studies. The results of these case studies show that the proposed method is feasible and operable in supporting spatial analysis directly on SVG on the browser side. The proposed method can be easily incorporated with some existing methods (e.g., GML-based spatial analysis on the server side) to provide load balancing spatial analysis (load balancing between server side and browser side) in XML/GML/SVG based WebGIS. As a result, users can access high performance spatial analysis simply via a web browser (such as, Internet Explorer and Firefox).

Keywords:

XML/GML/SVG based WebGIS; spatial data modeling; spatial information representation; SVG-based Spatial Extended SQL; Web-based spatial analysis

1. Introduction

Technological advances in the Internet/Web have triggered a move towards Web-based geographic information system (WebGIS). WebGIS aims at providing GIS functions (e.g., web mapping and spatial analysis) to users through a common web browser, such as Internet Explorer and Firefox. XML (Extensible Markup Language) /GML (Geography Markup Language)[1]/SVG (Scalable Vector Graphics)[2] based solutions have been proven to be a promising approach for building WebGIS (Peng and Zhang, 2004; Chang and Park, 2006; Lu et al., 2007; Huang et al., 2009). In these solutions, GML is used as a coding, storing and transmitting standard of spatial data on the server side, while SVG is considered as a rendering tool for displaying spatial data on the browser side. Recently, as Google published a JavaScript library[3] to provide "native" SVG support on many browsers (e.g., Internet Explorer, Firefox, and Safari), XML/GML/SVG based WebGIS has become increasingly popular. However, many of the XML/GML/SVG based WebGISs have been designed for visualization (web mapping) only, but "avoid the access to spatial analysis functions" (Lin and Huang, 2001), e.g., Carto:net[4], Peng and Zhang (2004), and Köbben (2007). To meet the increasing demand of complicated GIS applications in the Web environment, spatial analysis should be introduced into XML/GML/SVG based WebGIS.

There are also many XML/GML/SVG based WebGISs adopting a server-side solution to provide spatial analysis, i.e., executing all the spatial analytical tasks on GML on the server side, and sending the results to the browser side for visualization in SVG. This server-side solution becomes impractical, as servers may not be able to handle a large volume of concurrent users. Additionally, spatial analysis is a complex task; users often have to try

[1] GML, http://www.opengeospatial.org/standards/gml
[2] SVG, http://www.w3.org/Graphics/SVG/
[3] svgweb, http://code.google.com/p/svgweb/
[4] Carto:net, http://www.carto.net/

different query solutions before they are satisfied with the results. As spatial queries often result in a large amount of data, there will be a high transmission load between the server side and the web browser side. What's more, in order to improve performance, not all the spatial queries should be implemented on the server side. For example, the "Buffer" operation often results in more data output than input. As the processor performance of a normal PC (personal computer) has been improved drastically, "Buffer" may be more suitable to be implemented on the browser side to reduce the network transmission load.

A promising solution for the above problems is to provide load balancing for spatial analysis between server and browser side. Load balancing spatial analysis executes spatial querying and analysis either on the server side (GML) or on the browser side (SVG) depending on execution costs (i.e., network transmission costs and computation costs) (Huang et al., in press). In order to implement this load balancing spatial analysis, GML-based spatial analysis (for the server side) and SVG-based spatial analysis (for the browser side) should be developed. There are some researches focusing on spatial querying on GML (Guan, 2006). However, to the best of our knowledge, none of the research provides spatial querying and analysis directly on SVG.

Compared to GML which is designed for geospatial applications, SVG is developed primarily as a rendering tool for 2D graphics on the Web (W3C, 2003). As a result, methods of GML-based spatial analysis cannot be directly applied to SVG. Recognizing this limitation, this paper attempts to develop an approach to enable spatial querying and analysis directly on SVG.

In this paper, after analyzing the workflow of spatial analysis, we identify the key issues of providing spatial analysis on SVG (section 2): SVG-based spatial information representation (section 3), and SVG-based Spatial Extended SQL (SSESQL) (section 4). We attempt to find a theoretical foundation for the first issue based on the theory of spatial data modeling. On this basis, we next propose an SVG-based spatial information representation model to represent spatial data. Based on the representation model, some spatial operators are designed and integrated into an SSESQL. With the proposed method, users can easily execute spatial querying and analysis directly on SVG. Two case studies are implemented in section 5 to evaluate the proposed method. Also, comparisons with the server-side solution are

provided in section 5. Finally, in section 6 we draw conclusions and present future work.

2. Workflow of Spatial Analysis

Spatial analysis is “a set of methods whose results change when the location of the object being analyzed changes” (Longley et al., 2005). It plays a key role in GIS (Goodchild and Gopal, 1989). According to Wu (2002), the workflow of spatial analysis includes four steps: 1) define the goal and evaluation criteria; 2) represent the needed spatial dataset; 3) carry out spatial querying and analysis with GIS tools; 4) appraise and explain results. Step 1 and step 4 require domain knowledge and are mainly carried out by domain experts. For step 2 and step 3, GIS tools are needed to support/assist human-computer interaction.

The proposed method is mainly designed to support/assist step 2 and step 3. For step 2, we design an SVG-based spatial information representation model which can be used to represent the needed spatial dataset in SVG (section 3). For step 3, we design and implement some spatial operators and integrate them into an SVG-based Spatial Extended SQL (SSESQL) to support spatial querying and analysis on spatial dataset represented in SVG (section 4). Figure 1 depicts the workflow of spatial analysis on SVG.

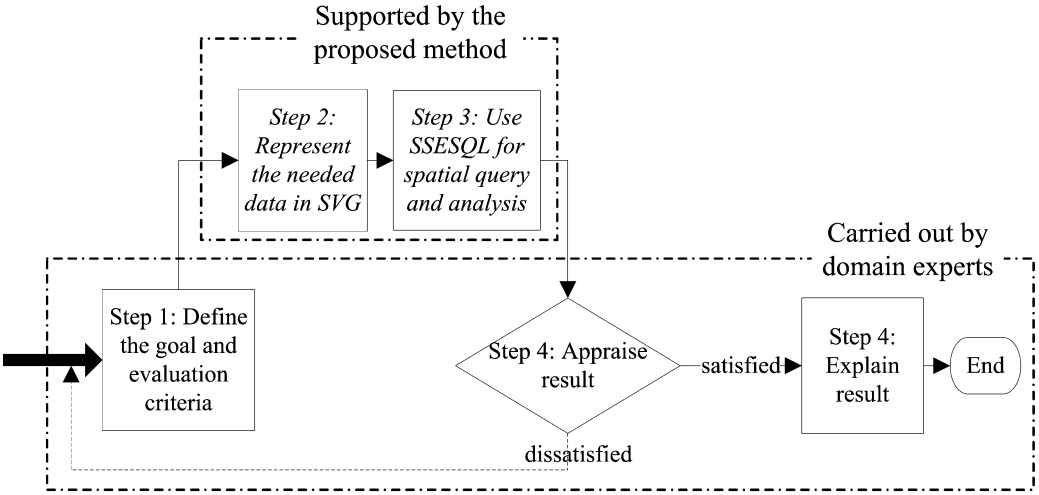

Figure 1 Workflow of spatial analysis on SVG

3. SVG-based spatial information representation

SVG is developed primarily as a rendering tool for 2D graphics. However, spatial information has a particular way in representing and organizing spatial features and their relationships (such as hierarchical structure of map - layer - spatial object, spatial attributes vs. non-spatial attributes). To represent the needed spatial dataset in SVG, a model considering the characteristics of spatial information has to be developed. Unfortunately, little work has been done on this topic. In this section, we introduce theory of spatial data modeling to design a model for SVG-based spatial information representation.

3.1 Theory of Spatial Data Modeling

Spatial data modeling is the process of abstracting the real world (identifying the relevant objects and phenomena) and representing it in an appropriate form which can be recognized by computers (Chen et al., 2001). SVG is one of the forms which computers use to represent relevant objects or phenomena of the real world. Therefore, we can utilize the theory of spatial data modeling, to discuss SVG-based spatial information representation to represent these kinds of information.

Three models play a role in spatial data modeling (Figure 2): conceptual model, logical model, and physical model. According to the content and requirements of these models, spatial data modeling includes three steps (Chen et al., 2001): 1) choose a conceptual model which can abstract the real world most appropriately, 2) choose a suitable data structure to represent the conceptual model, 3) design a file format, or an appropriate method to record or store the data structure from step 2.

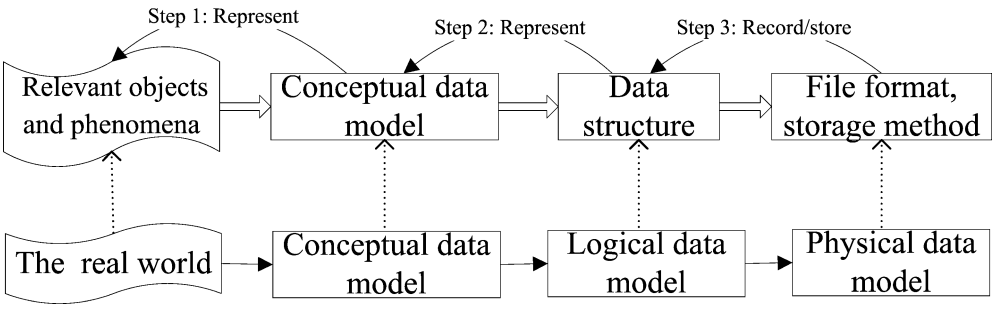

Figure 2 Content and steps of spatial data modeling

In the following, the above mentioned steps are used to develop an SVG-based spatial information representation model. First, we make some extensions to the Open Geospatial Consortium (OGC)'s Geometry Object Model (OGC, 1999), and take the extended model as our conceptual model. Next a spatial data structure is designed to describe the conceptual model based on object-oriented design (OOD). Finally, the SVG standard (file format) is employed to represent the above spatial data structure.

3.2 Spatial Conceptual Data Model

Spatial conceptual data models can be categorized into raster data model and vector data model. The latter treats the world as a surface littered with recognizable spatial objects (e.g., cities, rivers), which exist independently of their locations (Shekhar et al., 1997). As SVG is developed to represent vector graphics, this paper focuses on the vector data model.

In the vector data model, spatial data are organized as a hierarchical structure: spatial entity (object), layer, and map. A spatial entity refers to an object or phenomenon with a geometrical shape. It has two types of attributes: spatial attributes that describe geometry and topology of the spatial entity, and non-spatial attributes which define the semantics (name, theme, etc.) of the spatial entity. Spatial entities belonging to the same theme often have similar non-spatial attributes or geometrical types, and thus are grouped into a layer. A map is often constituted by different layers which describe the same region (area) of the real world.

OGC's Geometry Object Model (GOM) is one of the most popular vector models. It abstracts spatial entities as Point, Curve, Surface, Multipoint, Multicurve and Multisurface according to their geometries. Line is a special type of Curve, and Polygon is a special type of Surface. As GOM only defines spatial attributes of spatial entities, we extend it by adding some concepts like map, layer and non-spatial attributes. Also we don't depict the relationships between Point and Curve, Curve and Surface. The reason is that we want to have a relatively simple SVG structure (without too many "xlink:href"), which can improve the rendering and querying performance of SVG document. Figure 3 shows the extended conceptual model.

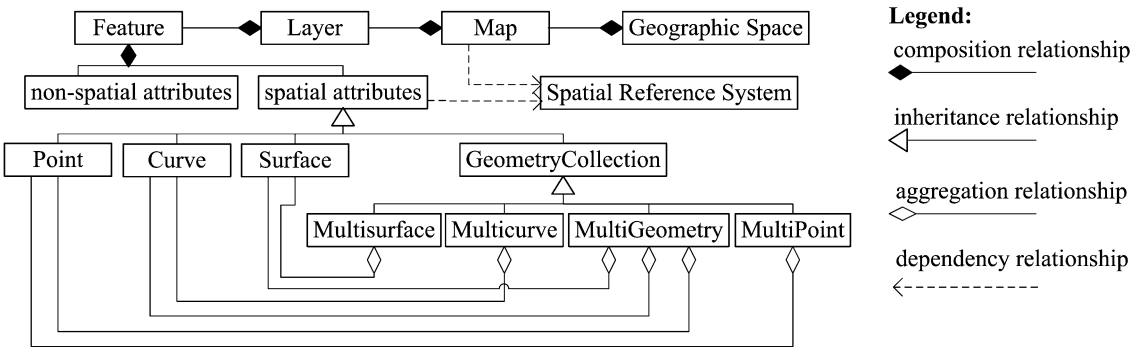

Figure 3 The spatial conceptual data model (using UML notations) which is an extension of OGC's Geometry Object Model (GOM) (adapted from OGC (1999))

3.3 Spatial Data Structure

In this section, a spatial data structure is designed to represent the above conceptual model based on object-oriented design (OOD). Figure 4 depicts the spatial data structure.

In this data structure, a spatial entity is designed as an abstract class Feature. The non-spatial attributes of spatial entity are designed as data members of the class Feature. Point, Curve, Surface, Multipoint, Multicurve, Multisurface and Multigeometry are inherited from the class Feature. Map is described as the class Map, which includes data members such as x1, y1, width and height (the bounding of the map) and SRS (Spatial Reference System).

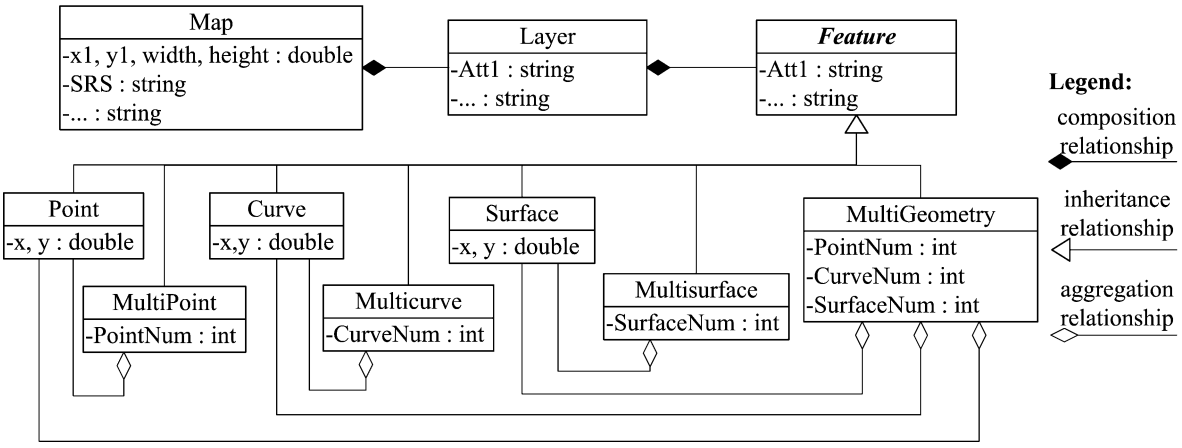

Figure 4 The class hierarchy of spatial data structure which represents the conceptual model in Figure 3

3.4 SVG-Based Spatial Information Representation Model (File Format)

In this section, we discuss how to use the SVG standard (1.1) to represent (store) the above spatial data structure. The basic strategies are as follows: 1) use a <svg> element to represent the class Map, and use the viewBox attribute to represent data members x1, y1, width and height, 2) the abstract class (Feature) doesn't need to be represented, and its data members are represented in its inherited classes, 3) data members (spatial attributes and non-spatial attributes) in a class are represented as corresponding SVG element's attributes, 4) if class B is PART-OF class A (composition/aggregation relationship in the spatial data structure), use a <g> element to represent the class A and group its members (e.g., class B) together. For example, the class Feature is PART-OF the class Layer, so we use a <g> element to represent the class Layer and group the class Feature (spatial entity) together. According to these strategies, <g> element is used to represent the class Layer, Multipoint, Multicurve, Multisurface and Multigeometry.

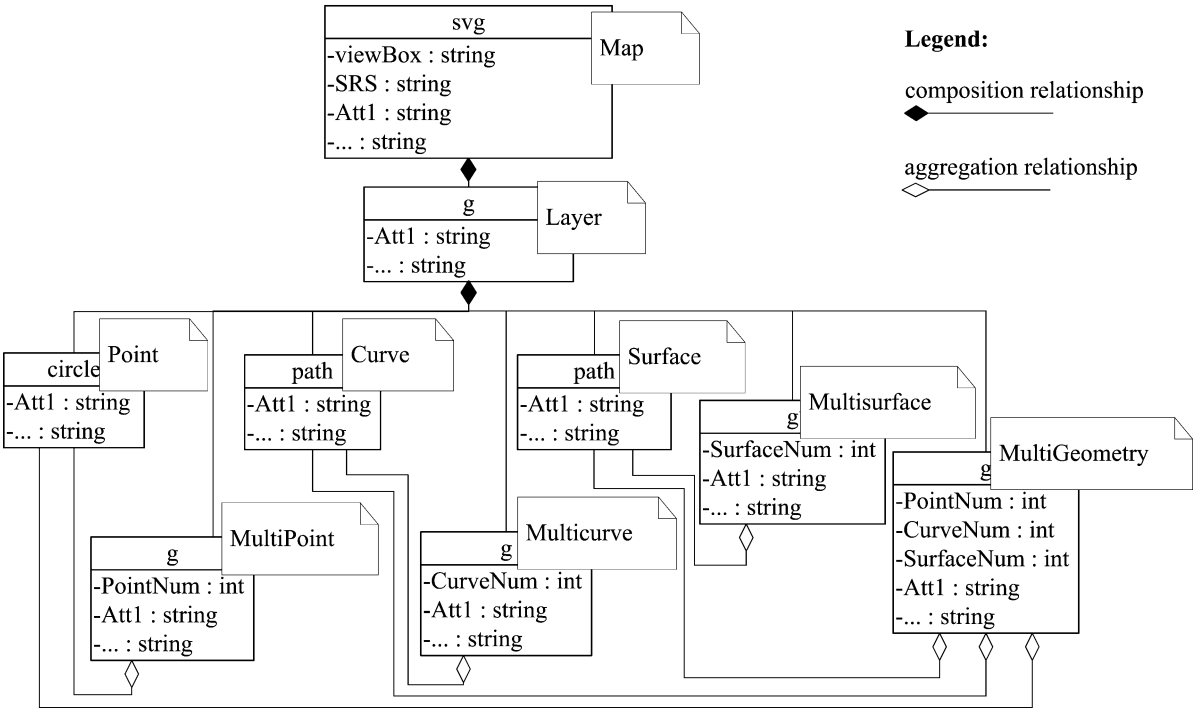

Figure 5 SVG-based spatial information representation model

Figure 5 depicts the SVG-based spatial information representation model. If B is PART-OF A (i.e., composition/aggregation relationship), B will be represented as a child element of A in

this model. For example, Layer is PART-OF Map, so the <g> element which represents Layer is a child element of the <svg> element which represents Map. With this model, users can use SVG to represent the needed spatial information. Please refer to section 4.2 for an example.

4. Spatial Operators and SSESQL for Spatial Analysis on SVG

This section focuses on how to carry out a spatial data query on SVG. It is important to note that XQuery is often employed to query XML-based data, and is also suitable for querying SVG data. However, XQuery has a very special and complicated syntax, and is not familiar to many people. In contrast, SQL (Structured Query Language) has a relative simple and intuitive syntax, and is familiar to many technical users. More importantly, SQL is more powerful in querying data. Thus, in consideration of end users' acceptance and the processing capabilities, extended SQL is employed for spatial data querying on SVG.

Many researchers have designed their spatial extended query language for spatial querying (Lin and Huang, 2001; Egenhofer, 1994). These SQL-like languages introduce spatial data types (e.g., point, line and polygon) and spatial operators. They allow users to inquire spatial features, primarily in terms of spatial relationships and metric constraints (Lin and Huang, 2001). It is widely recognized that these spatial operators and SQL-like languages can be used to support spatial analysis.

According to section 3, spatial information is organized as map - layer - spatial object. A map represented by an SVG document can be viewed as a database, the layers as tables of the database, the attributes (spatial and non-spatial) of spatial objects in the layer as columns of corresponding table, and spatial objects as records of corresponding tables. Therefore, SQL like languages can be used for spatial querying on SVG.

In this section, some spatial operators are designed and integrated into an SVG-based Spatial Extended SQL (SSESQL). This SSESQL uses the basic spatial data types discussed in section 3 (i.e., Point, Curve, Surface, Multipoint, Multicurve, Multisurface and Multigeometry).

4.1 Spatial Operators

Spatial operators are mainly designed to access spatial attributes, calculate spatial relationships, and perform geometrical operations. The following operators are designed:

1) Attribute access operators: They include Centroid, Length, Area and Envelope. They are used to calculate centroid, length, area and bounding of a spatial object.

2) Spatial topological operators: Spatial topological relationships are very important for spatial querying and spatial analysis. Two different approaches are used in describing spatial topological relationships: DE-9IM (the Dimensionally Extended 9 Intersection Model) and RCC-8 (Region Connection Calculus). According to Renz et al. (2002), these two completely different approaches lead to an identical set of topological relations. This paper uses the smallest complete set of topological relationships based on DE-9IM (Disjoint, Touch, Crosses, Within and Overlap), and implements them as our spatial topological operators. For convenience, the Contain operator is also included as the opposite of the Within operator.

3) Spatial metric operators: They include a Distance operator to calculate the distance between two spatial objects.

4) Geometrical operators: Sometimes, spatial analysis needs to create new spatial features with some geometrical operations. In order to support this kind of operation, Intersection, Union, and Difference are defined as geometrical operators. A Buffer operator is designed to support buffer generation around a spatial object.

These five kinds of operators meet the basic requirements of spatial analysis. For network analysis, Touch operator and Length operator can be used to find out the touched spatial object (e.g., road) and the distance. Buffer operator and topological operators can be used to carry out buffer analysis, such as “which cities are around 100 km from the Yangtze River”. Also, Intersection, Union and Difference operators can be used to carry out overlay analysis.

4.2 SSESQL and Some Query Examples

As SSESQL is designed for spatial querying, there is no need to consider data insert, update and delete. All we need to do is integrating the above spatial operators to the original

SELECT clause of SQL. Huang (2006) provided the EBNF (Extended Backus-Naur Form) of SELECT clause of SSESQL.

The following SVG codes show a map of Guangdong Province in South China based on the proposed model (Figure 5). This map includes two layers: cities and rivers.

```
<svg viewBox="94928 2172873 790615 595213" SRS="xi'an80">
    <g id="city">
        <path id="C1" pop="1500000" d="…"/>
        …
    </g>
    <g id="river">
        <path id="R1" length="100" d="…"/>
        …
    </g>
</svg>
```

The two layers represented in the above SVG codes can be viewed as the following tables: city (id, pop, d) and river (id, length, d). Below are some query examples.

1) Query example 1: List the cities which are crossed by the river "R1".

```
SELECT c.id AS cid FROM river r, city c
((WHERE r.id='R1') AND (Crosses(r.d, c.d)=true));
```

2) Query example 2: There is some toxic contamination throughout the River "R1". This toxic contamination will affect cities, in a 100 km river stretch. List these cities.

```
SELECT cy.id AS id FROM city cy, river r
WHERE ((Overlap(cy.d, Buffer(r.d, 100000))=true) AND( r.id='R1'));
```

5. Implementation, Case Studies and Discussion

5.1 Implementation of Spatial Operators and SSESQL

Algorithms of computational geometry can be utilized to implement the spatial operators. Many open source libraries can be also used to implement them, among which are JTS Topology Suite[5] and NetTopologySuite[6]. We implement the spatial operators with JTS Topology Suite, and develop them as Java Applet. For the SSESQL, a compiler is needed to execute syntax, sentence, and semantic analysis for SSESQL sentences. Google Gears API provides an SQLite compiler with JavaScript based API. We therefore implement a JavaScript based SSESQL compiler with the Gears API. For the spatial operators embedded in the SSESQL sentences, JavaScript involves the developed Java applet to execute the corresponding spatial operations.

The JavaScript-based SSESQL compiler and the developed Java applet are embedded in a HTML web page. An SVG document, which represents the needed spatial data based on the model shown in Figure 5, is translated from GML and also embedded in the web page. A user interface is embedded in the HTML web page for inputting SSESQL sentences, showing SSESQL results as well (Figure 6). Users can access spatial analysis functions simply with a web browser (such as IE) with SVG support. For different spatial analysis tasks, users only need to represent the needed spatial data in an SVG document, and embed it into the web page.

5.2 Case Studies

In the following, we present two case studies to evaluate the suggested method as proof of concept. As SVG document, SSESQL compiler, and the Java applet are embedded in a HTML web page, and will be cached locally on the browser side when users open the web page, the current case studies can be considered as a browser-side spatial analysis (directly on SVG).

[5] http://www.vividsolutions.com/jts
[6] http://nts.sourceforge.net/

The case studies investigate changes of land use and economic development with the growth of a transportation network. It is important to note that a good procedural design of spatial analysis is vital for a spatial analysis task. This means that users do have to plan the analysis task carefully even if they have the best spatial analysis tool.

5.2.1 Case study 1

The case study 1 focuses on changes of land use along railways and freeways between 1987 and 1996. To investigate this issue, we need to define a buffer for the railways and freeways, find out which cities are located in this buffer, and then analyze the changes of land use for these cities. We carry out this task based on the steps described in Figure 1. First, the required data and evaluation criteria are identified. Next an SVG document, which represents the needed spatial data (i.e., district boundary, railway, freeway, and city center) based on the proposed model (Figure 5), is translated from GML and also embedded in the web page. We then submit SSESQL sentences on the HTML web page to execute the spatial queries on this SVG by the following steps: 1) calculate a 20 km buffer of railways and freeways (using the Buffer operator), 2) identify the cities whose centers are located in this buffer (using the Within operator). Finally, a statistics function is employed to generate the bar graphs of changes of land use for every relevant city. The SSESQL sentences for this case study are listed in Appendix A.

Figure 6 shows the user interface and results of this case study. The right lower box lists the names of cities whose centers are within the buffer.

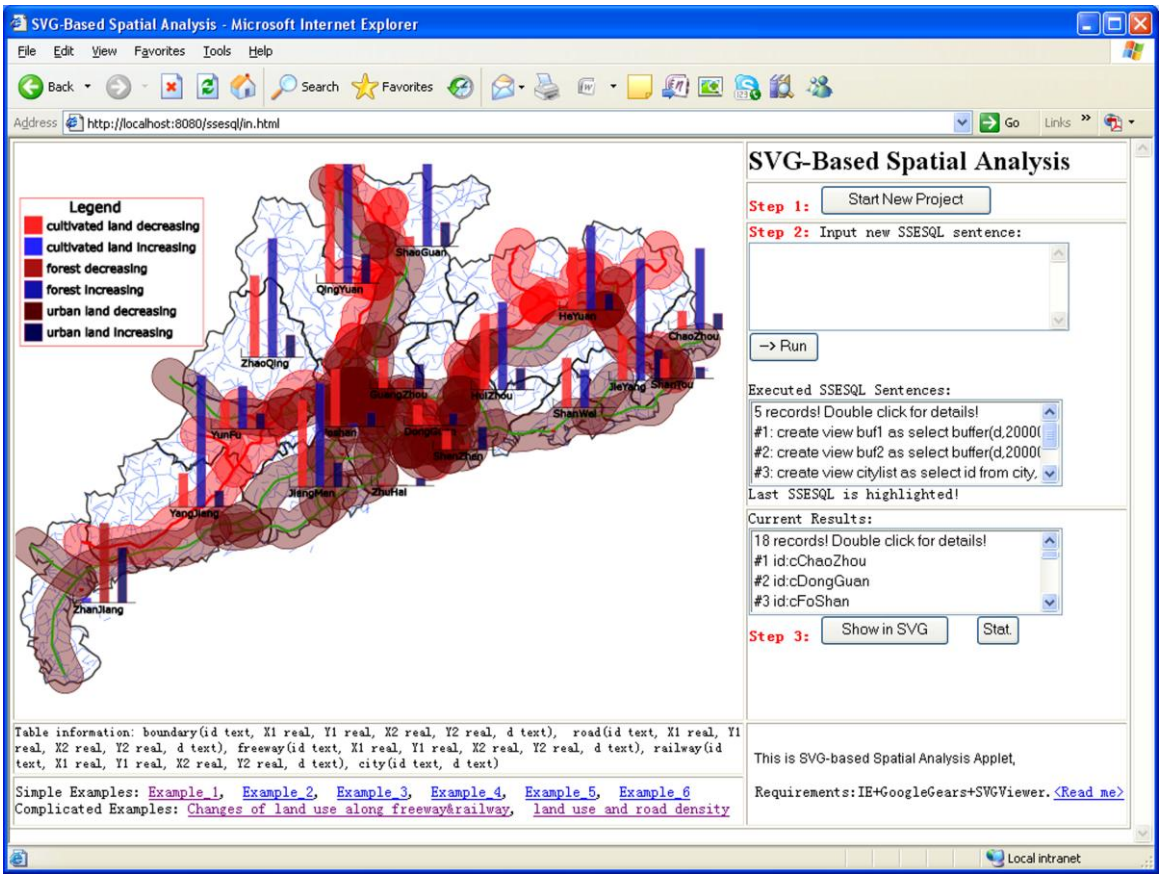

Figure 6 Case study 1: Land use changes along the railways and freeways between 1987 and 1996 in Guangdong Province, China. (Legend for bar graphs: red: decrease of cultivated lands; blue: increase of cultivated land; dark red: decrease of forest; dark blue: increase of forest; brown: decrease of urban land; dark grey: increase of urban land)

As can be seen from the bar graphs, the area of cultivated land decreases for all cities, whereas area of urban land increases between 1987 and 1996. These results coincide with the situation of Guangdong Province between 1987 and 1996. During that period, with the growth of the transportation network, especially railways and freeways, all cities expanded greatly by changing cultivated land into urban land. However, area of forested land increases between 1987 and 1996. An explanation for this is the governmental announced policy for forest increase in 1985: “Create a Green Guangdong in 10 Years” (Guangdong Forest 2008). From this case study, we get a rough concept that land use changes with the growth of a transportation network. To make some quantitative conclusions, more case studies may be done by using the proposed method.

As mentioned in section 5.1, SVG document, SSESQL compiler, and the Java applet are automatically cached on the browser side when users open the web page, and thus a browser-side spatial analysis is adopted where spatial queries are executed directly on SVG. To illustrate the advantages, we make a comparison between the proposed method and existing server-side solution (i.e., spatial querying on GML on the server side) in XML/GML/SVG based WebGIS for carrying out this task. We mainly compared the data amount of network transmission (excluding the request/response sentences) between the server side and the browser side. Table 1 depicts the comparison.

Table 1 Comparison of spatial analysis on GML and spatial analysis on SVG for case study 1 (data amount)

	Spatial analysis on GML	Spatial analysis on SVG
Step1: Buffer	126 976B	0B
Step2: Within	171B	0B
Step3: Stat.	2 168B	2 168B
Total	129 315B	2 168 B

As can be seen from Table 1, compared to the server-side solution (i.e., spatial analysis on GML), the proposed method has a smaller network transmission load between the server side and the browser side.

5.2.2 Case study 2

The case study 2 tries to identify a relationship between economic development and road (i.e., freeway and railway) density in Guangdong Province. Similar to the case study 1, we represent the needed spatial data (railway, freeway, and district boundary) in an SVG document and deliver it to the web browser side as the initial UI. The task is carried out by the following steps: 1) calculate the total road length for every city (using Interaction and Length operators), 2) calculate every city's area (using the Area operator), 3) calculate every city's road density. Finally, we color every city according to its road density. The SSESQL sentences for this case study are listed in Appendix B.

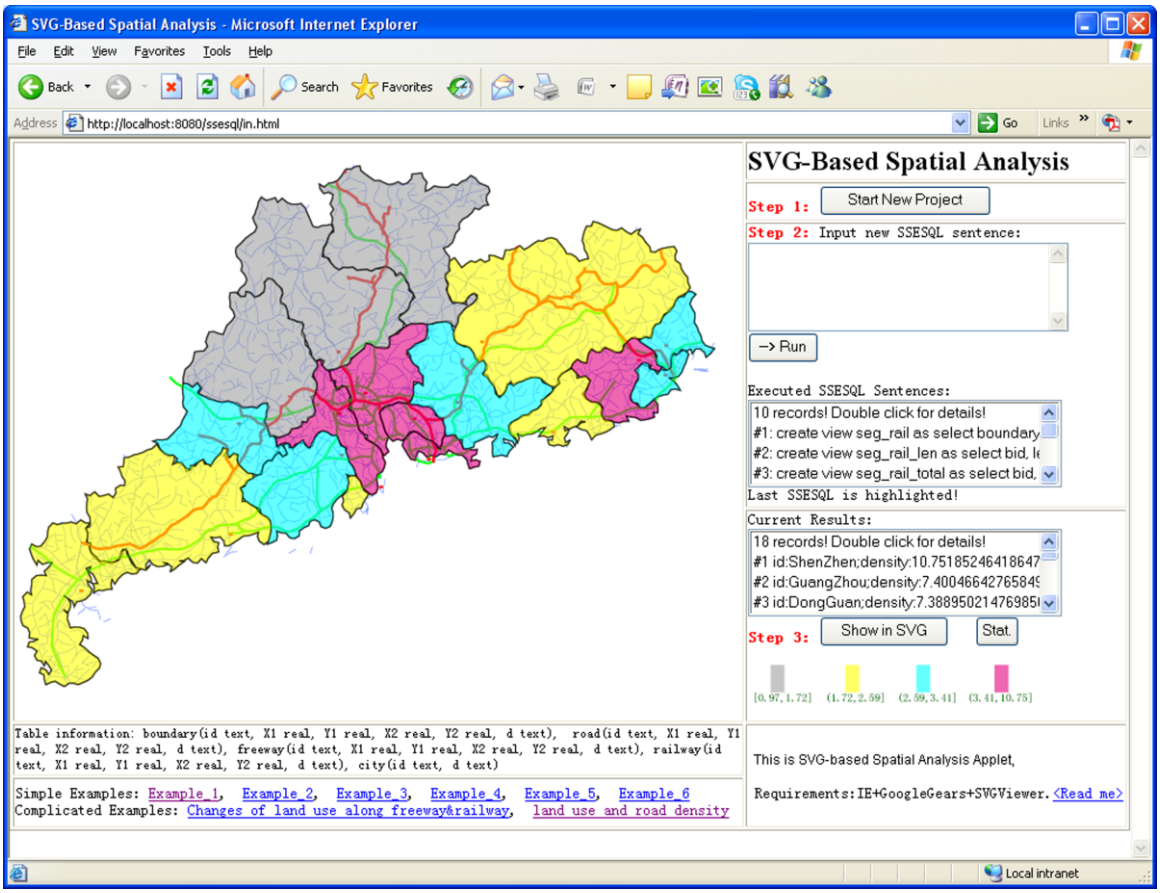

Figure 7 Case study 2: The relationship between economic development and road density in Guangdong Province, China

Figure 7 depicts the results for case study 2. The cities' names and their road densities are listed at the right lower box. Every city is marked with a specific color according to its road density.

The results show that Shenzhen is the city with the highest road density (in the Pearl River Delta) 10.75 km/100km^2. Note that Shenzhen has the fastest economic growth since the middle of 1980s. Other cities (Guangzhou, Dongguan and Foshan) in the Pearl River Delta (one of the most economically dynamic region in China) have very high road densities. Some cities (Jieyang, Yunfu and ChaoZhou) in eastern and western Guangdong also have a high road density, but surprisingly a relative slower economic growth. A possible explanation is their distance away from Hong Kong. It seems that the transportation density and contiguity with Hong Kong play a key role in the economic development. More case studies have to been done to test this hypothesis.

Similar to the case study 1, we compare the data amount of network transmission in the

proposed solution and existing server-side solution (spatial querying on GML on the server side). Table 2 depicts the comparison.

Table 2 Comparison of spatial analysis on GML and spatial analysis on SVG for case study 2 (data amount)

	Spatial analysis on GML	Spatial analysis on SVG
Step1: Length	1 318B	0B
Step2: Area	1 345B	0B
Step3: Density	1 756B	0B
Step4: Stat.	2 168B	2 168B
Total	6 587B	2 168 B

As can be seen from Table 2, the network transmission load between server side and browser side is eased with the proposed method.

5.3 Discussion

The two case studies show that the proposed method is feasible and operable to enable spatial querying and analysis directly on SVG (on the browser side). It also shows that this browser side solution may greatly ease the network transmission load between the server side and browser side during the process of spatial analysis. In the case study 1 we used a 20 km buffer size. However, in real world applications, users often have to try different buffer sizes before they are satisfied with the results. This often results in lots of intermediate data users may not need. When using existing methods (e.g., spatial querying on GML on the server side), there will be a high transmission overload between the server side and the browser side. For example, if a user tries a 50 km buffer size followed by a 5 km buffer size, and finally she/he decides to use a 20 km buffer size, the total amount of 375 808B data (i.e., 136 192B for 5 km buffer, 112 640B for 50km buffer, and 126 976B for 20 km buffer) have to be transmitted to the browser side, whereas only 126 976B (i.e., data amount of the 20 km buffer) are needed for the current task. The proposed method doesn’t have this problem. As

the spatial data of railway and freeway are cached locally, and SSESQL compiler works directly on SVG, all these buffer operations are solely executed on the web browser side. Thus the method proposed here greatly eases the transmission load between the server side and the browser side.

However, it is important to note that in the above case studies all the needed spatial data are represented in SVG and automatically cached on the browser side when users open the web page. For some complicated spatial applications, only some of the spatial data will be delivered to the browser side for visualization. As a result, load balancing spatial analysis between server side and browser side should be introduced into XML/GML/SVG based WebGIS. Load balancing spatial analysis executes spatial querying and analysis either on the server side (on GML) or the browser side (on SVG) depending on network transmission costs and computation costs. The proposed method (i.e., SVG-based spatial analysis on the browser side) provides a basis for this issue, and can be easily incorporated with existing methods (e.g., GML-based spatial analysis on the server side) for developing load balancing spatial analysis. As a result, users can access high performance spatial analysis functions simply via a web browser (such as, IE and Firefox).

6. Conclusions and future work

Spatial analysis plays a key role in GIS applications. However, most of the XML/GML/SVG based WebGISs (also lots of other WebGIS applications) only offer web mapping functions, but avoid providing spatial analysis functions. Others adopt a server-side solution for spatial analysis, i.e., executing all the spatial analysis tasks on GML on the server side, and sending the results to the browser side for visualization (in SVG). This server-side solution suffers from the “bottleneck” problem, and results in a high network transmission load between server side and browser side. Load balancing spatial analysis between server and browser side is a promising solution for the above problems. SVG-based spatial analysis is one of the key issues of load balancing spatial analysis. This paper focused on designing approaches to enable spatial querying and analysis directly on SVG (on the browser side).

The contributions of this paper are: 1) identifying the key issues of SVG-based spatial analysis,

2) setting up a theoretical foundation for developing SVG-based spatial information representation model, 3) designing some spatial operators, and integrating them into the SVG-based Spatial Extended SQL (SSESQL) to support spatial querying directly on SVG on the browser side, 4) implementing two case studies to evaluate the proposed method, and making comparisons with existing server-side spatial analysis (e.g., GML-based spatial analysis). The results of the case studies show that the proposed method is feasible and efficient to support spatial analysis in the Web environment.

Our final goal is to provide load balancing spatial analysis in XML/GML/SVG based WebGIS. The proposed method provides a basis for this issue, and can be easily incorporated with some existing methods of other building blocks (e.g., GML-based spatial analysis on the server side) to provide load balancing spatial analysis. As a result, our next step is to develop a load balancing middleware to dispense spatial queries to either server side (on GML) or browser side (on SVG) based on their execution costs (i.e., network transmission costs and computation costs). Also, we are interested in designing more complex case studies to evaluate the proposed solution.

References

Chang, Y. and Park, H., 2006. XML Web Service-based development model for Internet GIS applications. *International Journal of Geographical Information Science*, 20(4), 371- 399.

Chen, S., Lu, X. and Zhou, C., 2001. *Introduction of GIS*, Science publish, Beijing, 28-30 [in Chinese].

Egenhofer, M., 1994. Spatial SQL: A Query and Presentation Language. *IEEE Transactions on Knowledge and Data Engineering*,6(1), 86-95.

Goodchild, M F. and Gopal, S., 1989. Modeling error in objects and fields. *The Accuracy Of Spatial Databases*, 1(4), 107-114.

Guan, J., 2006. GQL: Extending Xquery to query GML documents. *Geo-Spatial Information Science*, 9(2), 118-126.

Guangdong Forest, 2008. Guangdong Forest in 30 Years http://www.gdf.gov.cn/index.php?controller=front&action=view&id=10005572 [In Chinese, Accessed on 20 May 2010]

Huang, C., Chuang, T., Deng, D. and Lee, H., 2009. Building GML-native web-based geographic information

systems. *Computers & Geosciences*, 35(9), 1802-1816.

Huang, H., 2006. Key Issues of SVG-based Network Spatial Information Publishing. Unpublished M.Sc. Thesis, South China Normal University, Guangzhou.

Huang, H., Li, Y. and Gartner, G., in press. A load balancing method to support spatial analysis in XML/GML/SVG-based WebGIS. In: Li, S., Dragicevic, S., Veenendaal, B. (Ed.) *Advances in Web-based GIS, Mapping Services and Applications*, CRC Press.

Köbben, B., 2007. RIMapperWMS: a Web Map Service providing SVG maps with a built-in client. In: Fabrikant, S.I., Wachowicz, M. (Ed.) *The European Information Society: Leading the Way with Geo-information*, Springer Berlin Heidelberg, 217-230.

Lin, H. and Huang, B., 2001. SQL/SDA: A query language for supporting spatial data analysis and its Web-based Implementation. *IEEE Transactions on Knowledge and Data Engineering,* 13(4), 671-682.

Longley, P.A., Goodchild, M.F., Maguire, D.J. and Rhind, D.W., 2001. *Geographic Information Systems and Science*. John Wiley & Sons, Chichester, UK, 472 pp.

Lu, C., Santos, R., Sripada, L. and Kou, Y., 2007. Advances in GML for Geospatial Applications. *GeoInformatica*, 11(1), 131-157.

OGC, 1999. Open GIS simple feature specification for SQL, http://www.opengeospatial.org/standards/sfs, [Accessed on 20 May 2010].

Peng, Z. and Zhang, C., 2004. The roles of geography markup language (GML), scalable vector graphics (SVG), and Web feature service (WFS) specifications in the development of Internet geographic information systems (GIS). *Journal of Geographical Systems*, 6(2), 95-116.

Renz, J., Rauh, R. and Knauff, M., 2002. *Towards Cognitive Adequacy of Topological Spatial Relations, Spatial Cognition II*, LNCS 1849, Springer Berlin / Heidelberg, 184-197.

Shekhar, S., Coyle, M., Goyal, B., Liu, D. and Sarkar, S., 1997. Data models in geographic information systems. *Communications of the ACM*, 40(4), 103-111.

W3C, 2003. Scalable Vector Graphics (SVG) 1.1 Specification, http://www.w3.org/TR/SVG11/, [Accessed on 20 May 2010].

Wu, X., 2002. *Principles and methods of GIS*, Publishing House of Electronics Industry, Beijing, 156-157 [in Chinese].

Appendix A: the SSESQL sentences for the case study 1.

1) Calculate a buffer of railway and freeway (using Buffer operator)

create view buf1 as select buffer(d,20000) as buf from railway;

create view buf2 as select buffer(d,20000) as buf from freeway

2) List cities whose center is located in this buffer (using Within operator)

create view citylist as select id from city, buf1 where (within(city.d, buf1.buf)=true)

create view citylist as select id from city, buf2 where (within(city.d, buf2.buf)=true)

*create view finalresult as select distinct * from citylist*

Appendix B: the SSESQL sentences for the case study 2.

1) Calculate the total road length in every city (using Interaction and Length operators)

create view seg_rail as select boundary.id as bid, Intersection(boundary.d, railway.d) as int from boundary, railway;

create view seg_rail_len as select bid, length(int) as len from seg_rail;

create view seg_rail_total as select bid, sum(len) as total_len from seg_rail_len group by bid;

create view seg_free as select boundary.id as bid, Intersection(boundary.d, freeway.d) as int from boundary, freeway;

create view seg_free_len as select bid, length(int) as len from seg_free;

create view seg_free_total as select bid, sum(len) as total_len from seg_free_len group by bid;

create view totalroad as select seg_rail_total.bid as id, seg_rail_total.total_len +seg_free_total.total_len as total from seg_rail_total, seg_free_total where (seg_rail_total.bid=seg_free_total.bid);

2) Calculate the area of every city (using the Area operator)

create view citysize as select id, area(d) as size from boundary;

3) Calculate the road density of every city

*create view roaddensity as select citysize.id as id, totalroad.total*1000*100/citysize.size as density from citysize, totalroad where (citysize.id=totalroad.id);*

*create view density as select * from roaddensity order by density desc;*