SVG-based Spatial Information Representation and Analysis

Haosheng Huang[1,1], Yan Li[2] and Georg Gartner[1]

[1] Institute of Geoinformation and Cartography, Vienna University of Technology, Austria

[2] School of Computer, South China Normal University, China

huanghaosheng@gmail.com, yanli@scnu.edu.cn, georg.gartner@tuwien.ac.at

Abstract. This paper tries to make some spatial extensions to W3C's Scalable Vector Graphics (SVG) Specification to support SVG-based spatial information representation and analysis in the Web environment. Based on the theory of spatial data modeling, this paper attempts to find a theoretical foundation for SVG-based spatial information representation. And then, based on this theoretical foundation, we propose an SVG-based spatial information representation model for spatial data publishing. Furthermore, this paper designs and implements some spatial operators, and integrates them into an SVG-based Spatial Extended SQL to support spatial analysis on the browser side, which improves the functions of current WebGIS applications, most of which have been only employed for visualization. Finally, this paper designs some case studies. The results of the case studies prove that the suggested methods are feasible and operable for spatial information publishing and analysis in the Web environment.

Keywords: Spatial data modeling, SVG-based spatial information representation, Web-based spatial analysis, SVG-based Spatial Extended SQL (SSESQL), SVG-based WebGIS

1 Introduction

Technological advance in Internet/Web triggers a move towards WebGIS. More and more researchers pay attentions to the problems of spatial information integration and sharing in WebGIS. XML/GML/SVG-based solution is viewed as a feasible solution for solving these problems. In general, Geography Markup Language (GML) [1] is used as a coding, storing and transmitting standard of multiple spatial data on the server side; while Scalable Vector Graphics (SVG) [2] is used as the representing tool for displaying and interacting with vector data on the browser side.

Since SVG was introduced by W3C in 2001 [2], SVG has become an active area of research in WebGIS. Some of the researchers use SVG's shape elements (such as *line*, *polygon*, *path*, etc.) and graphic styles (such as *fill*, *stroke*, etc.) for spatial information visualization and developed some prototype systems [3], [4] and [5]. Furthermore, there are also some researches focusing on GML to SVG transformation [6] and [7]. As SVG is developed primarily as a XML-based publishing tool of 2D graphics, and also spatial information has its own characteristics in representing and organizing spatial objects and their relationships (such as hierarchical structure of map - layer - spatial object, spatial attributes vs. non-spatial attributes), in order to use SVG for spatial information publishing, an SVG-based spatial representation model which takes the characteristics of spatial information into account has to be developed. Unfortunately, little work has been done on this topic.

[1] corresponding author

At the same time, currently, lots of SVG-based spatial information applications have been designed for visualization only, but avoid the access to spatial analysis functions such as spatial topological query, map overlay, and buffer that are vital for spatial applications [8]. This is mainly due to the fact that there are no effective functions to support spatial query and analysis directly on SVG.

This paper attempts to solve the two problems mentioned above. First, based on the theory of spatial data modeling, this paper tries to find a theoretical foundation for SVG-based spatial information representation. And then, based on this theoretical foundation, we propose an SVG-based spatial information representation model for publishing spatial data in the web environment. In order to support SVG-based spatial analysis, we design and implement some spatial operators, and integrate them into an SVG-based Spatial Extended SQL (SSESQL). With this suggested model and SSESQL, users can easily carry out spatial information publishing and analysis tasks in the web environment.

The paper is arranged as follows. In section 2, based on the theory of spatial data modeling, we develop an SVG-based spatial information representation model for publishing spatial data. Section 3 designs and implements some spatial operators, and integrates them into SSESQL language, and then discusses how to use SSESQL for spatial analysis. In section 4, some case studies are implemented to evaluate our suggested methods. Finally, the conclusion and future work will be described in section 5.

2 Spatial Data Modeling for SVG-Based Spatial Information Representation

2.1 Theory of Spatial Data Modeling

Spatial data modeling is the process of abstracting the real world (identifying the relevant objects and phenomena in the real world) and representing the relevant objects or phenomena in an appropriate form which can be recognized by computers [9]. As SVG is one of the forms which computers use to represent relevant objects or phenomena of the real world, thus we can utilize the theory of spatial data modeling, to discuss SVG-based spatial information representation model which attempts to describe and represent the relevant objects and phenomena in the real world.

There are three models during spatial data modeling (Fig. 1): conceptual model, logical model, physical model. According to the contents and requirements of these models, spatial data modeling includes three steps [9]: 1) choosing a conceptual model which can abstract the real world most appropriately, 2) choosing an appropriate data structure to represent the conceptual model, 3) designing a file format, or the appropriate method of record or storage for the data structure in step 2.

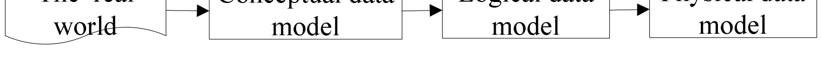


Fig. 1. Three models in data modeling

In these following sections, we use the above steps to develop an SVG-based spatial data representation model. First, we make some extensions to the Open Geospatial Consortium (OGC)'s Geometry Object Model [10], and take the extended model as our conceptual model. And then we design a spatial data structure for the above conceptual model based on object-oriented design. Finally, we use SVG standard to represent the above spatial data structure, and propose our SVG-based spatial information representation model.

2.2 Spatial Conceptual Data Model

Spatial conceptual data models can be categorized into field-based models and object-based models. Field-based models (raster data model) see the world as continuous surface over which features (e.g. elevation) vary. Object-based model (vector data model) treat the world as surface littered with recognizable spatial objects (e.g. cities, rivers), which exist independently of their locations [11]. As SVG is developed to represent vector graphic, we will focus on vector data models.

In vector data models, spatial data which describe the real world are organized as a hierarchical structure: *spatial entity (object)*, *layer* and *map*. *Spatial entity* refers to the thing or phenomenon which has geometrical shape. It has two types of attributes: *spatial attributes* that describe the 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 always have similar non-spatial attributes or geometrical type, and thus are grouped into a *layer*. A *map* is always constituted by different layers which describe the same region of the real world.

OGC's Geometry Object Model (GOM) [10] is one of the most popular vector models. It abstracts spatial entities as *Point*, *Linestring*, *Polygon*, *Multipoint*, *Multiline* and *Multipolygon* according to their geometries. As GOM only defines spatial attributes of spatial entities, we extend it by adding some concepts like map, layer and non-spatial attributes. Fig. 2 shows our conceptual model.

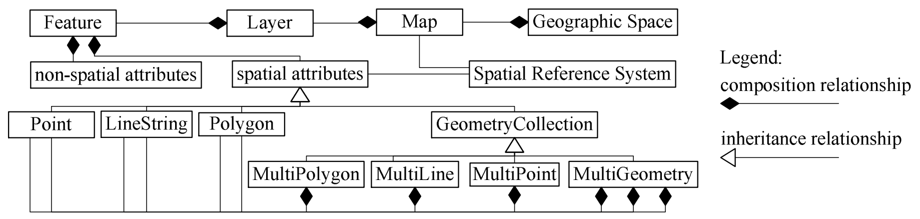


Fig. 2. The conceptual model (using UML notations) which is extended from Geometry Object Model (after [10])

2.3 Spatial Data Structure

In this section, we design a spatial data structure for the above conceptual model based on object-oriented design. Fig. 3 depicts our spatial data structure. In this data structure, spatial entity is designed as class *Feature* which is an abstract class. The non-spatial attributes of spatial entity are designed as data members of class *Feature*. *Point*, *Line*, *Polygon*, *Multipoint*, *Multiline*, *Multipolygon* and *Multigeometry* are inherited from class *Feature*. Map is described as class *Map*, which includes *x1, y1, width, height* (the bounding region of the map) and *SRS* (Spatial Reference System) attributes.

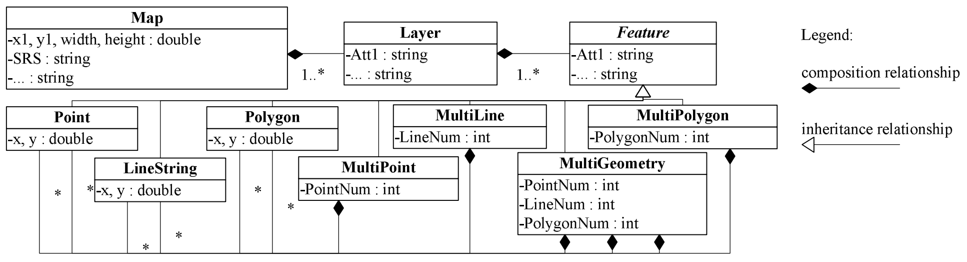


Fig. 3. The class hierarchy of spatial data structure which represents the conceptual model

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

In this section, we discuss how to use SVG standard to represent (store) the above spatial data structure. The basic principles are as follows:

1) Use <*svg*> element to represent class *Map*, and use *viewBox* attribute to represent data members *x1, y1, width, height.*

2) The abstract class (*Feature*) doesn't need to be represented; its data members are represented in its derived 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 relationship in the spatial data structure), use <*g*> element to represent class A and group members (e.g., class B.) of class A together. For example, class *Feature* is PART-OF class *Layer*, so we use <*g*> element to represent class *Layer* and group class *Feature* (spatial entity) together. According to the above principles, <*g*> element is used to represent class *Layer*, class *Multipoint*, *Multiline*, *Multipolygon* and *Multigeometry*.

Fig. 4 depicts the SVG-based spatial information representation model. In this model, if B is PART-OF A, B will be represented as child element of B. For example, *Layer* is PART-OF *Map*, so <*g*> element which represents *Layer* is a child element of <*svg*> element which represents the *Map*.

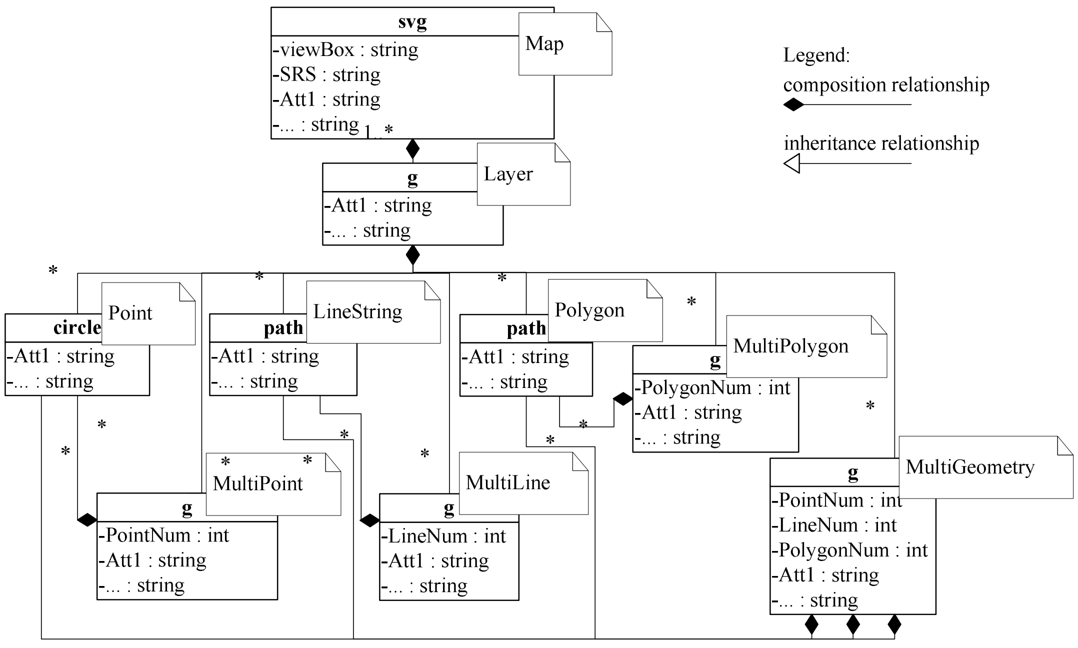


Fig. 4. SVG-based spatial information representation model

With this SVG-based spatial information representation model, users can easily use SVG to publish spatial information in the Web environment. Please refer to section 3 for some examples.

3 SVG-Based Spatial Analysis

The ability of spatial analysis is viewed as one of the key characters which distinguish GIS from CAD systems or other information systems. Currently, SVG-based WebGIS has been employed for visualization; but avoided the access to spatial analysis functions. This is mainly due to the fact that there are no effective web-based functions to support spatial query, and analysis on SVG.

Since the need for spatial query languages has been identified in 1982 [12], many authors have designed their spatial extended database language [8], [13] and [14]. These SQL like languages introduce spatial data types (e.g., point, line and polygon) and spatial operators, allow users to inquire spatial features, primarily in terms of spatial relationships and metric constraints [8]. It is widely acknowledged that these spatial operators and SQL like languages can be used for spatial analysis.

According to section 2, spatial information is organized as a hierarchical structure: map - layer - spatial object. A map represented by SVG can be viewed as a database, the layers as tables of the database, the attributes (spatial attributes and non-spatial attributes) of spatial entities in the layer as columns of corresponding table, spatial entities as the records of corresponding table. As a result, SQL like languages can be used to support spatial analysis on SVG.

In this section, we design some spatial operators and integrate them into a spatial extended SQL, which we define as SVG-based Spatial Extended SQL (SSESQL). This SSESQL uses the basic spatial data types discussed in section 2: *Point*, *Linestring*, *Polygon*, *Multipoint*, *Multiline*, *Multipolygon* and *Multigeometry*. The proposed SSESQL can be used on server side or browser side for spatial query and analysis on SVG.

3.1 Spatial Operators

In order to carry out a spatial analysis task, we always need to query the spatial object in terms of attributes, spatial topological relationships and metric constraints, and then maybe use some geometrical operations or feature fusion to handle these spatial data.

Spatial analysis basically includes buffer analysis, overlay analysis and network analysis. After analyzing their requirements, we design the following spatial operators:

1) Attribute access operators

This kind of operators is mainly used to calculate length, area and centroid of the spatial object. Those are *GeometryType*, *Centroid*, *Length* , *Area*, *Envelope*.

2) Spatial topological operators

Spatial topological relationship is very important to spatial query and spatial analysis. There are two different models describing the spatial topological relationship: DE-9IM [15] and RCC-8 [16]. According to Renz et al. [17], these two completely different approaches to topological relationships lead to exactly the same set of topological relations. This paper uses the smallest complete set of topological relationship based on DE-9IM: *Disjoint*, *Touch*, *Crosses*, *Within* and *Overlap*. As a result, spatial topological operators include *Disjoint*, *Touch*, *Crosses*, *Within* and *Overlap*. For convenience, *Contain* operator is also included as the opposition of *Within* operator.

3) Spatial order operators

People always use spatial order relationship to describe the position of the spatial object, e.g., the university is in the north of the river. In order to perform spatial order relationship query, spatial order operators include: *East*, *East_South*, *South*, *West_South*, *West*, *West_North*, *North* and *East_North*.

4) Spatial metric operators

Spatial distance is the most important relation of metric relationship. Spatial metric operators include: *Max_Distance*, *Min_Distance* and *Average_Distance*.

5) Geometrical operators

Sometimes, spatial analysis needs to create new spatial features with some geometrical operation. As proved in mathematics, *{Intersection, Union, Difference}* is a complete set for 2D geometrical operation. So we define *Intersection*, *Union*, *Difference* as geometrical operators. In order to support Buffer analysis, *Buffer* operator is also designed for this kind of operators.

The above five kinds of operators can meet the basic requirements of spatial analysis. For network analysis, we can use *Touch* operator and *Length* operator to find out the touched spatial object (e.g., road) and the distance. We can use *Buffer* operator and topological operators to carry out buffer analysis. Also, we can use *Intersection*, *Union* and *Difference* operators to carry out overlay analysis.

3.2 SSESQL and Some Query Examples

As SSESQL is designed for spatial query, there is no need to consider data insert, update and delete. All we should do is integrating the above spatial operators to the original SELECT clause of SQL. The following EBNF (Extended Backus-Naur Form) is the grammar of SELECT clause of SSESQL.

```
<select_state>:=SELECT <selectexpression> FROM <fromexpression>
                 WHERE <whereexpression> GROUP BY<groupbyexpression>
<selectexpression>:=<select_list>[,<Derivated_att>]|<Derivated_att>
<Derivated_att>:=<spatial_operator> AS <Att_name>[,<Derivated_att>]*
<spatial_operator>:=<att_operator>|<topo_operator>|<order_operator>
                     |<dist_operator>|<geometical_operator>
<att_operator>:=<att_operator_v>|<att_operator_g>
<att_operator_v>:=<Att_function1>(<att_geom>)
<Att_function1>:=GeometryType|Length|Area
<att_geom>:=expression|<analysis_operator>|<att_operator_g>
<att_operator_g>:=<Att_function2>(<att_geom>)
<Att_function2>:=Centroid|Envelope
<topo_operator>:=<topo_function>(<att_geom>,<att_geom>)
<topo_function>:=Disjoint|Touch|Crosses|Within|Overlap|Contain
<order_operator>:=<order_function>(<att_geom>,<att_geom>)
<order_function>:=East|West|North|South|West_South|East_North
                   |West_North|East_South
<dist_operator>:=<dist_function>(<att_geom>,<att_geom>)
<dist_function>:= Max_Distance| Min_Distance| Average_Distance
<geometrical_operator>:=<geometricaloperator>|<Bufferoperator>
<geometricaloperator>:=<analysis_function>(<att_geom>,<att_geom>)
<analysis_function>:=Intersection|Union|Difference
```

```
<Bufferoperator>:=Buffer(<att_geom>,<d_value>)
<d_value>:=expression|<att_operator_v>
<fromexpression>:=<table_source>[,<table_source>]*
<whereexpression>:=<select_condition>[AND<op_exp>|OR<op_exp>]*
                   |<op_exp>[AND<op_exp>|OR <op_exp>]*
<op_exp>:=<spatial_operator><condition>expression
<condition>:=>|<|>=|<=|=
```

The following SVG codes show a map of Guangdong province in China which is based on our suggested SVG-based spatial information representation model (section 2). This map includes two layers: city and river. We will carry out some spatial queries on this SVG map.

```
<svg viewBox=”94928 2172873 790615 595213” SRS=”EPSG:4326”>
   <g transform=”translate(0,4940959),scale(1,-1)”>
      <g id=”city”>
         <path id=”C1” population=”7.5” d=”…”/>
         …
      </g>
      <g id=”river”>
         <path id=”R1” length=”100” d=”…”/>
         …
     </g>
   </g>
</svg>
```

The above SVG map includes two layers (city and river), which can be viewed as the following tables: city (id, population, d) and river (id, length, d). In the following there are some query examples.

1) Query example 1: List the cities which are the neighbors of city “C1”.

```
SELECT A.id AS “Neighbors_of_C1”
FROM city A, city B
WHERE Touch (A.d, B.d) =True AND B.id=”C1”
```

2) Query example 2: 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
```

3) Query example 3: River “R1” can supply water for the cities, which are 40KM around the river, List this kind of cities.

```
SELECT cy.id
FROM city cy, river r
WHERE Overlap(cy.d, Buffer(r.d, 40))=True AND r.id=”R1”
```

3.3 SSESQL-Based Spatial Analysis in the Web Environment

In this section, we concentrate on using SSESQL for spatial analysis. First, we will introduce the workflow of spatial analysis, and then consider how to use SVG-based spatial information representation model (section 2) and SSESQL to support spatial analysis.

The workflow of spatial analysis includes four steps [18]: 1) defining the goal and evaluation criteria; 2) preparing and representing the needed spatial data; 3) carrying out spatial query and analysis with GIS tools; 4) result appraisal and explanation. Step1 and step 4 need domain knowledge and are mainly carried out by experts. Step 2 and step 3 need GIS tools to support human-computer interaction.

For step 2, we can use SVG to represent the needed spatial data based on the SVG-based spatial data representation model (section 2); while for step 3, SSESQL can be utilized to carry out spatial query. Fig. 5 shows the workflow of SVG-based spatial analysis.

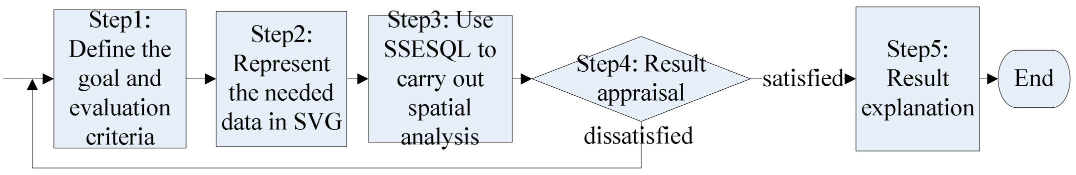

Step1: Define the goal and evaluation criteria
Step2: Represent the needed data in SVG
Step3: Use SSESQL to carry out spatial analysis
Step4: Result appraisal
satisfied
dissatisfied
Step5: Result explanation
End

Fig. 5. Workflow of SVG-based spatial analysis

Here is an example that utilizes the SSESQL for site selection. Suppose city “Z” wants to choose a site for a theme park. According to the above workflow, we carry out this task based on SSESQL:

Step 1, define the goal and evaluation criteria. The selection criteria are: 1) Site must lie in unused land; 2) Site must be beyond 250 meters of the road; 3) Site must be within 1200 meters of rivers.

Step 2, analyze these selection criteria and use the methods discussed in section 2 to represent the needed spatial data (for this task, including road, river, and landuse).

The following is the needed spatial data represented by SVG.

```
<svg viewBox=”…” SRS=”…”>
  <g transform=”…”>
    <g id=”landuse”>
      <path id=”A” type=”unused” d=”…”/>
      …
    </g>
    <g id=”road”>
      <path id=”C1” d=”…”/>
      …
```

```
        </g>
        <g id=”river”>
            <path id=”R1” d=”…”/>
            …
        </g>
    </g>
</svg>
```

Step 3, utilize SSESQL to carry out spatial analysis. First, we utilize “*Buffer*” operator to create buffer of road and river, then use “*Intersection*” and “*Disjoint*” operators to carry out overlay operation of landuse, buffer of road and buffer of river. The query sentences are as follows:

```
CREATE VIEW ribuf AS SELECT id,Buffer(d,1200)AS buf FROM river;
CREATE VIEW robuf AS SELECT id, Buffer(d,250) AS buf FROM road;
CREATE VIEW landriver AS
                    SELECT Intersection(landuse.d,ribuf.buf) AS temp1
                    FROM   ribuf, landuse
                    WHERE   landuse.type=”unused” AND
                            Disjoint(landuse.d,ribuf.buf)=False;
SELECT  Intersection(landriver.temp1,robuf.buf) AS site
FROM    landriver, robuf
WHERE   Disjoint(landriver.temp1,robuf.buf)=False;
```

Step 4, result appraisal and explanation. After executing the above sentences, the appropriate site will be highlighted on the SVG map. If users are not satisfied with the result, they can return to step 1.

4 Implementation and Case Studies

4.1 Implementation of Spatial Operators and SSESQL

It is widely acknowledged that load-balancing between server side and browser side is a feasible technology for Web-based spatial analysis [19]. WebGIS will carry out spatial operation on server side or browser side based on the communication cost versus the computational cost. As a result, we need to develop the spatial operators and SSESQL for both server side and browser side.

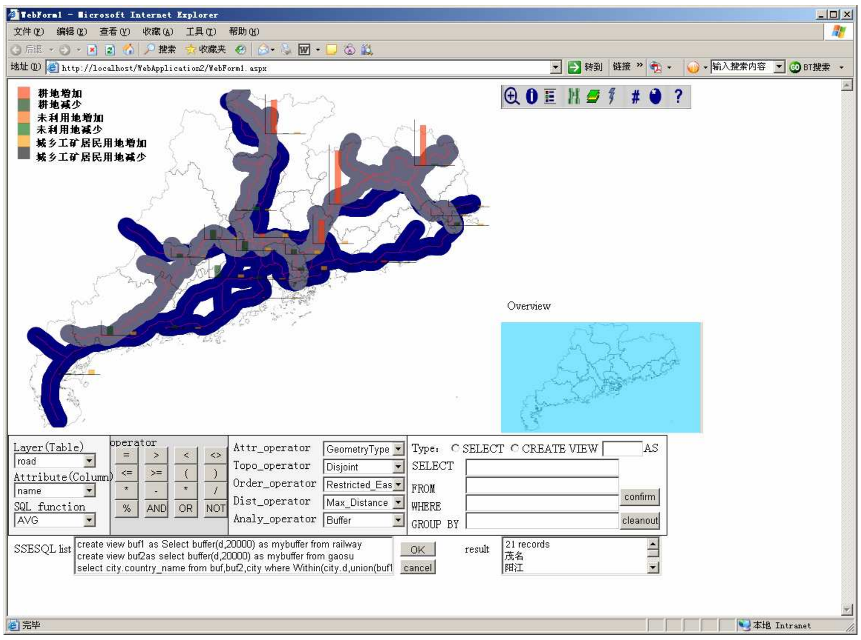
Algorithms of *computational geometry* can be utilized to develop our spatial operators. As Java has provided some basic computational geometry APIs (Java 2D API), we implement the spatial operators with algorithms of *computational geometry* and Java 2D API. For the SSESQL, we develop a compiler to carry out syntax, sentence, and semantic analysis for SSESQL sentences. For the server side implementation, spatial operators and SSESQL compiler are designed as Java servlets. For browser side implementation, spatial operators and SSESQL compiler are developed as Java applets, and embedded in HTML; DOM APIs of HTML and SVG are used to access SVG document and invoke the Java applets; a user interface is embedded in HTML for inputting SSESQL query sentences (Fig. 6).

In the following section, we design two case studies to evaluate the suggested methods on SVG-based spatial information representation and analysis. As load-balancing between server side and browser side is not our research focus in this paper, we carry out spatial analysis based on the browser side implementation (Java applets).

4.2 Case Studies

The cultivated lands are very sensitive in a highly populated country like China, especially, in the most quickly developing province of Guangdong. People may consider whether the cultivated lands change with the growth of transportation network. We choose two issues to analyze and discuss.

The first issue is how the cultivated lands along the railway and highway change. Let's take 15km extent and analyze the cultivated land change between 1987 and 1999. We carry out this analysis based on the method described in section 3.3. First, based on the suggested model in Fig. 4, we use SVG to represent the needed spatial data (railway, highway, district map, and different years' statistics on land use). And then use SSESQL to carry out spatial queries on this SVG. Fig. 6 shows the user interface and result of this case study. Functions of zoom in/out, roam, layer control, query, and statistics are also added to this interface. This task uses the "*Buffer*", "*Union*", "*Within*" operators and SSESQL in the analyzing procedure. The right low box lists the names of cities involved in the calculation. At last, we use the statistics function to generate the bar graphs for every relevant city.



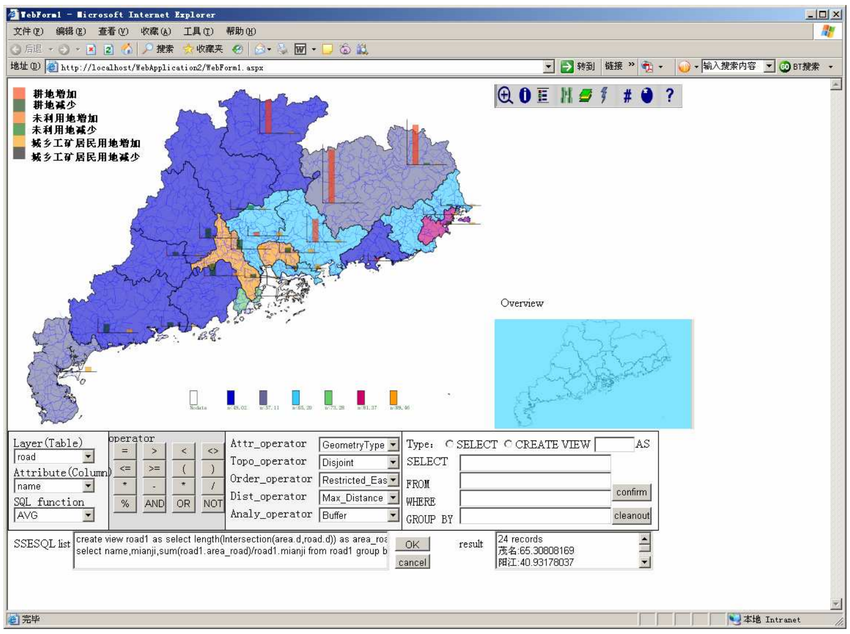
Layer(Table): road
Attribute(Column): name
SQL function: AVG
Attr_operator: GeometryType
Topo_operator: Disjoint
Order_operator: Restricted_Eas
Dist_operator: Max_Distance
Analy_operator: Buffer
Type: ○ SELECT ○ CREATE VIEW AS
SELECT
FROM
WHERE
GROUP BY

SSESQL list
```
create view buf1 as Select buffer(d,20000) as mybuffer from railway
create view buf2as select buffer(d,20000) as mybuffer from gaosu
select city.country_name from buf,buf2,city where Within(city.d,union(buf1
```
result
```
21 records
茂名
阳江
```

Fig. 6. The cultivated lands change along the railway and main highway in 15km extent (Legend for bar graphs: dark orange for cultivated lands increasing; dark green for cultivated lands decreasing; middle orange for unused land increasing; middle green for unused land decreasing; light orange for residential area increasing; dark grey for residential area decreasing)

Another issue is what has happened to cultivated lands with the increasing of the road density (railway, highway, and province-level road). We use the "*Interaction*", "*length'*, "*area*" operators and

SSESQL to carry out this task step by step. First, we utilize the "*intersection*" and "*length*" operators to calculate the total road length for every city, and use "*area*" operator to calculate every city's area, and then calculate every city's road density. Fig. 7 depicts the result of this issue. The result lists the name of cities and their road density at the right low box. Every city is marked with different color according to its road density, and with a statistic bar graph on it.



SSESQL list

```
create view road1 as select length(Intersection(area.d,road.d)) as area_roa
select name,mianji,sum(road1.area_road)/road1.mianji from road1 group b
```

result

```
24 records
茂名:65.30808169
阳江:40.93178037
```

Fig. 7. The cultivated land changes with the increasing of road density. (The Legend is the same as that in Fig. 6)

The implementations of the above case studies show that our suggested methods for SVG-based publishing and analysis is feasible and operable, it can greatly support the publishing and analysis of spatial information in the Web environment. The suggested SSESQL supports the user to carry out spatial analysis task on the browser side.

5 Conclusions and Future Work

This paper tries to make some spatial extensions to W3C's SVG Specification to support SVG-based spatial information representation and analysis in the Web environment. Based on the theory of spatial data modeling, this paper sets up a theoretical foundation for SVG-based spatial information representation, and develops an SVG-based spatial data representation model based on this theoretical foundation. Furthermore, we design and implement some spatial operators, and integrate them into SSESQL language, and then discuss how to use SSESQL for spatial analysis in Web environment. In order to evaluate our suggested methods, this paper designs and implements two case studies. The results of the case studies show that our proposed methods are feasible and operable to support spatial information publishing and analysis in Web environment.

Our next step is to consider the load-balancing technology between server side and browser side for SVG-based spatial analysis. We are also interested in the problem of completeness of the spatial operators for spatial analysis task.

References

1 Geography Markup Language, http://www.opengeospatial.org/standards/gml

2 Scalable Vector Graphics (SVG), http://www.w3.org/Graphics/SVG/

3 Cartographers on the net, http://ww.carto.net

4 Peng, Z-R., and Zhang, C.: 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), pp. 95-116 (2004)

5 Li, Q., Xie, Z., Zuo, X., and Wang, C.: The spatial information description and visualization based on SVG (in Chinese). Acta geodaetica et cartographica sinica, 34(1), pp. 58-63 (2005)

6 Guo, Z., Zhou, S., Xu, Z., and Zhou, A.: G2ST: A novel method to transform GML to SVG. in Proceedings of the 11th ACM international symposium on advances in geographic information systems, pp.161-168 (2003)

7 Tennakoon, W.T.M.S.B.: Visualization of GML data using XSLT. Diploma Thesis, International Institute for Geo-Information Science and Earth Observation (2003)

8 Lin, H. and Huang, B.: SQL/SDA: A query language for supporting spatial data analysis and its Web-based Implementation. IEEE transactions on knowledge and data engineering, 13(4), pp.671-682 (2001)

9 Chen, S., Lu, X., and Zhou, C.: Introduction of GIS (in Chinese). Beijing: Science publish, pp.28-30 (2001)

10 Open GIS simple feature specification for SQL, http://www.opengeospatial.org/standards/sfs

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

12 Frank, A.: Mapquery-database query languages for retrieval of geometric data and its fraphical representation. ACM Computer Graphics, 16(3), pp.199-207 (1982)

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

14 Ingram, K.J., and Phillips, W.W.: Geographic Information Processing using a SQL Based Query Language, Proc. AutoCarto8, pp.326-335 (1987)

15 Egenhofer, M.: Reasoning about binary topological relations. In Gunther, O. and Schek, H.-J. (Eds.): SSD'91, LNCS 525, pp. 43–160 (1991)

16 Randell, D., Cui, Z., and Cohn, A.: A spatial logic based on regions and connection. In: Nebel, B. et al. (Eds.): Proc. of the Knowledge Representation and Reasoning, pp. 165~176 (1992)

17 Renz, J., Rauh, R., and Knauff, M.: Towards Cognitive Adequacy of Topological Spatial Relations, in Freksa, C. et al. (Eds.): Spatial Cognition II, LNCS 1849, pp. 184-197 (2002)

18 Wu, X.: Principle and methods of GIS (in Chinese), Beijing: Publishing House of Electronics industry, pp.156-157 (2002)

19 Vatsavai, R., Shekhar, S.m Burk, T., Lime, S.: UMN-MapServer: A High-Performance, Interoperable, and Open Source Web Mapping and Geo-spatial Analysis System. In Raubal, M. et al. (Eds.): GIScience 2006, LNCS 4197, pp. 400–417 (2006)