Design of a Visual Query Language for Geographic Information System on a Touch Screen

Siju Wu, Samir Otmane, Guillaume Moreau, Myriam Servières

Design of a Visual Query Language for Geographic Information System on a Touch Screen

Siju Wu[1], Samir Otmane[1], Guillaume Moreau[2], Myriam Servières[2]

IBISC, Université d'Evry Val d'Essonne, France
{siju.wu,samir.otmane}@ibisc.univ-evry.fr
LUNAM Université, Ecole Centrale de Nantes – CERMA, France
{guillaume.moreau,myriam.servieres}@ec-nantes.fr

Abstract. This paper presents two spatial query methods for a Geographic Information System (GIS) that runs on a touch screen. On conventional GIS interfaces SQL is used to construct spatial queries. However keyboard typing proves to be inefficient on touch screens. Furthermore, SQL is not an easy-learning language, especially for novices to GIS. To simplify query construction, firstly we have designed a map interaction based query method (MIBQM). This method allows users to make simple queries by selecting necessary layers, features and query operators directly on the interface. To allow users to construct complex queries, a sketch drawing based query method (SDBQM) is proposed. Spatial query concepts can be represented by sketches of some symbolic graphical objects. It is possible to add spatial conditions and non-spatial conditions to describe query concepts more precisely. An evaluation has been made to compare SQL and MIBQM. We have found that for simple queries, MIBQM takes less time and proves to be more user-friendly.

Keywords: GIS, Touchable Interface, Visual Query Language, Spatial Query

1 Introduction

Geographic Information System (GIS) is a system which is able to capture, store, analyze, manage and present the geographic referenced data [1]. Because of its powerful capability of data processing, now it has become an inevitable tool in many domains. In recent years touch screen technology has developed rapidly and has been widely used on Smartphones and Tablet PCs. Since the touch screen is apt to provide natural user experience, nowadays some GIS interfaces are redefined to adapt to gesture interaction. However, only some fundamental functions can be accomplished on these interfaces. There is no satisfying solution for spatial query construction. On conventional GIS interfaces such as ArcGIS [2] or OrbisGIS [3], SQL is used to make spatial queries. Unfortunately typing on a keyboard still remains inefficient on the touch screen. Besides this, users may find it difficult to translate query concepts into SQL statements in an intuitive way. A mature query construction method is still to be proposed.

In this paper we propose two spatial query methods which are apt to be used on touch screens. The first method is named map interaction based query method (MIBQM). This method allows users to construct simple queries by selecting necessary layers, features and query operators. If a specific feature is concerned in a query, it can be selected directly on the map. The second method is the sketch drawing based query method (SDBQM). By making use of a new visual query language, users can draw sketches to formulate queries. Different from MIBQM, spatial conditions and non-spatial conditions can be added to describe query concepts more precisely. These query methods are designed for specialists to improve their working efficiency. By applying these methods, query functions of GIS are also available for novices.

The remainder of this paper is organized as follows. In Sect. 2, we review related work about the design of spatial query methods. Then in Sect. 3 and Sect. 4 we present how to make queries by using our methods and we provide a brief introduction of the touchable interface in Sect. 5. In Sect. 6 we present a usability study which evaluates MIBQM by comparing it with SQL. Finally, we draw some conclusions and give a discussion about future research.

2 Related works

On conventional interfaces, SQL is widely used as a tool to search spatial information. To give users the capability of dealing with spatial attributes of layers, different database systems have provided their own series of spatial functions. These functions can be used to construct SQL query statements. Although these abundant functions are powerful, some users may find it hard to remember all their names and application methods.

Besides SQL, there also exist other spatial query methods. One type of methods allows users to draw a flowchart to make a query. Users place flowchart elements, which can be a layer or a query operator, in a design space and then connect them in a manner consistent with the interface to construct a model [4] [5] [6]. A flowchart can express query concepts precisely, but it lacks visual explanation of the query concept. The flowchart may be complex to understand when the query is complex. To make GIS more accessible for non-trained users, visual query languages have been proposed. For many database management systems, visual query languages are often designed to improve the effectiveness of human-computer communication [7]. Some visual query languages allow users to use some predefined icons to compose spatial queries [8] [9]. Each icon thus represents a specific layer. Spatial relations between two layers can be represented by a relationship operator and query operators are used to indicate what kinds of data to search. One disadvantage of these methods is that once a new layer is added in the database, a new icon should be defined. Some other visual query languages offer more liberty to draw sketches. To represent a layer, users can draw a symbolic object and add a name to it [10] [11]. A spatial condition can be represented by the spatial relationship between two symbolic objects. The use of graphical representations offers an intuitive and incremental view of spatial queries, but it sometimes causes different interpretations of the same query. Users' query con-

cepts may be misunderstood by the system. Though most of the methods mentioned above are not designed for touch screens, some design concepts can still be borrowed. After analyzing these methods, we have proposed our spatial query methods.

3 Map Interaction Based Query Method

When making a spatial query, three kinds of information should be offered: what kind of data to search, which layers the data comes from and what conditions should be respected. The first two kinds of information are mandatory while the third one is optional. Inspired by a query method which allows users to find articles related to a certain subject directly through gesture interaction with the map [12], we have proposed MIBQM. Users can construct a spatial query by selecting necessary information which can be a layer, a feature or an operator on the interface and through combination of selected information, the system can automatically create a SQL statement.

In this method, two kinds of query operators are provided: unary operators (applicable to a single layer) and binary operators (that use two layers). Unary operators are used to get spatial attributes of a layer, such as the start points of rivers or the boundaries of provinces. Binary operators are for example called to calculate the topological relationships between features from two layers. To apply a unary operator, firstly a layer should be selected and the operator can be chosen. To apply a binary operator, users have to choose the first layer concerned in the query and the operator. After that the second layer should be selected. A query constructed in this way takes all the features of a layer into consideration. If only one feature is concerned in a query, after layer selection, this feature can be selected on the map by tap gesture. In this way, all the other features in the same layer will be ignored.

In this way, the map is no longer only used to consult numerical attributes, but can be exploited during the query construction. An example of calculation of the union of two province features is shown in Fig.1.

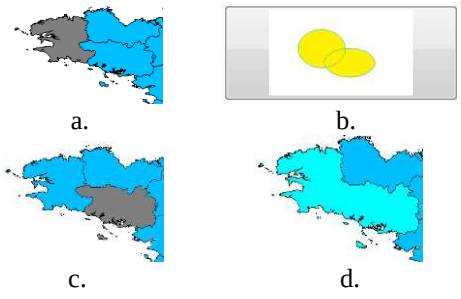

Fig. 1. Example of binary operator usage: a. select the first province b. select the union operator c. select the second province d. return the result

4 The sketch drawing based query method

To make it possible to construct complex spatial queries, we have proposed another query method, which is named the sketch drawing based query method (SDBQM). This method allows users to use a new visual query language to represent query concepts by drawing sketches.

4.1 Symbolic graphical objects

For most people, it may be natural to describe their abstract concepts by drawing pictures because the meaning of pictures may resemble those concepts better than words. In SDBQM, symbolic graphical objects (SGO) can be used to specify queries (Fig.2). Each layer can be represented by a SGO in the drawing area. The appearance of a SGO may be a point, a line or a polygon according to its layer. A SGO is defined as a 5-tuple

SGO = {LAYER, ALIAS, SHAPE, POSITION, PROPERTYSET}

- LAYER is the name of the layer which is represented by the SGO;
- ALIAS is the alias name used to identify the SGO;
- SHAPE is the geometry shape of the SGO;
- POSITION is the location of the SGO in the drawing area;
- PROPERTYSET includes all the attributes of the layer which is represented by the SGO.

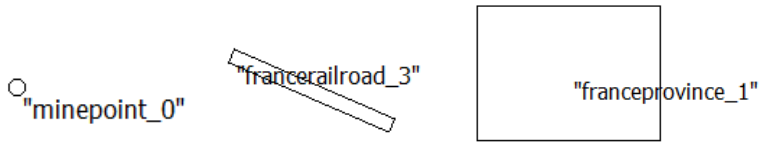

Fig. 2. SGO examples: Left: a point object; Middle: a line object; Right: a polygon object

4.2 Query operators

After drawing the sketch, query operators can be added to make a query about a specific SGO. There are two kinds of operators that can be selected: unary-SGO operators and binary-SGO operators. Each operator corresponds to a specific spatial query. After using a selection envelope to cover the SGO involved in the query, an operator menu can be called.

Unary-SGO operators

Unary-SGO operators (UOs) are used to called unary-parameter spatial functions (UFs), which are used to get attributes of a layer, such as buffers or boundaries. If a SGO is covered by the envelope, available UOs are displayed in the operator menu (Fig.3). After selecting one operator, the corresponding function will be called. The

geometry shape of the layer of the SGO is taken as parameter in the function. A new SGO is drawn to represent the result of the operator. By observing the appearance of the new SGO, users will know which operator has been selected.

Binary-SGO operators

Similarly Binary-SGO operators (BOs) are used to call binary-parameter spatial functions (BFs) which can be used to calculate the geometric relationships between two layers. To display BOs in the operator menu, two SGOs should be covered by the envelope. Once a BO is selected, the corresponding duo-parameter spatial function will be called. The geometry shapes of the two layers of the SGOs are taken as parameters. And a new SGO is drawn to represent the result of the operator.

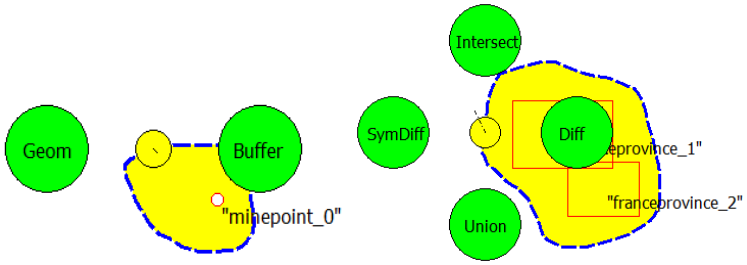

Fig. 3. SGO operators' example: Left: menu of UOs; Right: menu of BOs

4.3 Additional conditions

To give users the ability to express complex query concepts, it is possible to add additional conditions. Additional conditions are classified into two groups: spatial conditions and non-spatial conditions. Spatial conditions are used to describe topological relationships that features should satisfy, while non-spatial conditions give constraints about numerical attributes of features.

Spatial conditions

A spatial condition can be expressed by the topological relationship between two SGOs in the drawing area. Five relationships can be identified: DISJOINT, TOUCH, INTERSECT, EQUAL and COVER. Drawing SGOs in specific relationships to express spatial conditions affords an intuitive and comprehensible view of spatial queries. However, if each pair of relationship is translated into spatial conditions, ambiguity may appear. To avoid those ambiguities, a topological table is used. In the topological table each SGO is represented as a circle with its name. If the topological relation between two SGOs should be translated into a spatial condition, a connection line can be drawn between them. To remove a spatial condition, a cut line can be drawn to cancel the connection. In this way the meaning of the sketches can be explained more precisely. The example in Fig 4 has added two spatial conditions: franceprovince_0 has an intersection with franceprovince_1 and francerailroad_2 crosses franceprovince_0. The relationship between franceprovince_1 and fracerailroad_2 is ignored.

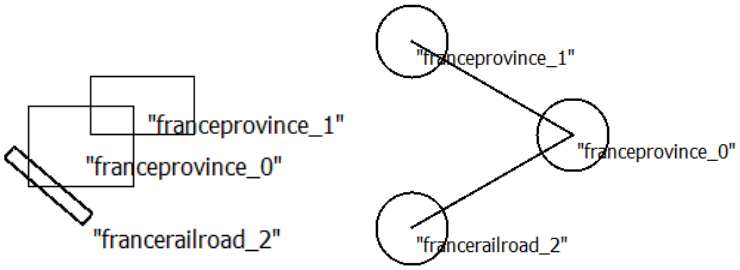

Fig. 4. Left: A sketch of three SGOs; Right: the corresponding topological table

Non-spatial conditions

In SDBQM, we have designed non-spatial condition objects (NCOs) and condition connector objects (CCOs) to add non-spatial conditions. A NCO is defined as 4-tuplet

$$NCO = \{LAYER, ATTRIBUTE, OPERATOR, VALUE\}$$

- LAYER is the name of the layer concerned in the condition;
- ATTRIBUTE is the attribute selected in LAYER;
- OPERATOR is used to decide which kind of constraint is set about ATTRIBUTE;
- VALUE is used to compare with ATTRIBUTE.

If more than one NCO is added, CCOs can be used to organize different NCOs in a nested structure. A CCO is defined as 3-tuplet

$$CCO = \{TYPE, NODE_{FIRST}, NODE_{SECOND}\}$$

- TYPE is the connection type. There are two types of CCO: AND or OR;
- $NODE_{FIRST}$ is the first object of the connection. It can be a NCO or a CCO;
- $NODE_{SECOND}$ is the second object of the connection. It can be a NCO or a CCO.

Three NCOs are organized in a nested structure in Fig.5 and two CCOs are used to connect them.

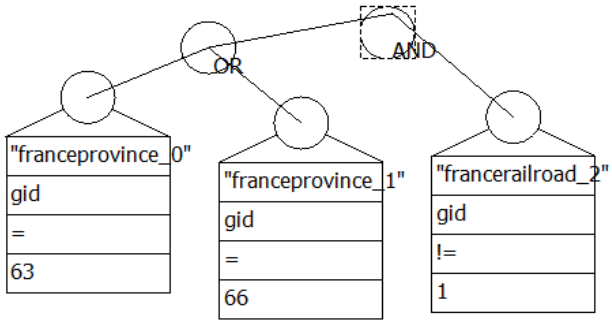

Fig. 5. Nested structure using conditions

4.4 Composed Query

After a query is executed, a new layer is generated and the query result is saved in this layer. Since the geometry types of features in the result layer may be different, it is impossible to use a classic geometry shape to represent the SGO of the new layer. Our solution of this problem is to use the shape of the result SGO which is generated in the last query. In this way, the appearance of the result SGO may be comprehensible and meaningful to recall what query has been done in the last time.

5 Touchable Interface

To realize the two query methods proposed in this paper, we have developed a touchable GIS interface. The interface consists of two parts: the basic interface (BI) and the sketch query interface (SQI). On BI, fundamental functions such as map manipulation and layer manipulation can be accomplished. On the right side of BI a toolbox is set. Three tags are used to display functions of different types. In the third tag spatial operators of MIBQM are provided. To apply SDBQM, SQI should be used. On SQI, there is a drawing area in which sketches can be drawn. Two condition-areas are set. One is used to set the topological table, and another is used to add NCOs.

On the touchable interface, users can apply various gestures for different tasks. In BI, gestures are used to manipulate the map. A drag gesture with one finger can translate the map and a pinch gesture with two fingers can scale the map. To rotate the map, users have to first press two fingers on the map to set their center as the rotation center, and then release one finger. The map can be rotated around the center by the remaining finger. In this way rotation and zoom manipulation can be separated. We have also designed a three-finger pan gesture to tilt the map in the horizontal or vertical direction. In BI, the data table can be called or hidden by a four-finger vertical gesture. In SQI, all the tasks such as drawing of SGOs, selection of operators and addition of conditions can be accomplished by gestures. To draw a SGO, users firstly have to select a layer in the layer list. If it is a point-shape layer, the SGO can be drawn by a tap gesture in the drawing area. Else the SGO of a line-shape or polygon-shape layer can be drawn by a drag gesture. After sketches are drawn, the drawing of a selection envelope can be started by pressing one finger in the drawing area. If the envelope crosses its start point and there are one or two SGOs covered by the envelope, the operator menu will be displayed. In the non-spatial condition area, a three-finger tap gesture can add an empty NCO. To switch from BI to SQI, users can drag five fingers from left to right in the map browsing window. Similarly a five-finger drag from right to left will hide SQI and recall BI.

6 Evaluation

We have performed an evaluation to examine the functionality of MIBQM. SDBQM will be evaluated in the future work. We asked 10 testers (8 male and 2 female) to accomplish 3 spatial query tasks by using SQL and MIBQM respectively. The aver-

age age of these testers is 26. All the testers are novices to GIS and only 2 of them are familiar with SQL. All three tasks are listed as follows.

1. Get the geometry shapes of all the features in the railway layer of France
2. Calculate the buffers of all the features in the mine field layer of France
3. Calculate the intersection of two provinces of France

Because the SQL console is not implemented in the touchable interface, so testers constructed SQL statements on OrbisGIS, which is an open source GIS. All the testers are divided into two groups. People in the first group tested SQL first and then MIBQM, while people in the second group did the inverse order. Before the evaluation, they were told how to accomplish these tasks and gave them some time to attempt the solution of each task. To compare the learning rate of these two query methods, each task has been repeated for three times. During the test, the performance time and the number of errors are measured. After the evaluation, each tester has answered a questionnaire to collect their comments.

6.1 Objective evaluation

From Table 1 we find out that MIBQM takes less time to fulfill all the tasks. When using SQL, testers spent most of the time in typing statements and consulting attribute values. However, MIBQM only necessitates a few of gestures. For SQL, the mean time for each task is 20s, 31s and 2:04min, and the standard deviations are 10s, 13s and 1:06min. For MIBQM, the mean time for each task is 7s, 21s and 20s, and the standard deviations are 5s, 10s and 11s. Performance improvements are 32%, 32% and 47% for SQL. For MIBQM improvements are 58%, 4% and 38%.

The table in Fig.6 shows that testers have made fewer errors for task1 and task2 when using MIBQM. Freshmen of SQL may find it hard to construct statements without grammar errors or false input of layer names and attribute values. However, for task3 MIBQM leads to more errors. Before feature selection, some users forgot to select the layer, so they found it impossible to select a feature. Each tap without a successful selection is considered as an error and it is why there are more errors for MIBQM. Some false selections have been made because some users wanted to select a feature when the size of map is not large enough. Some users confused the order of feature selection and the order of operator selection, so wrong results were obtained. Most of the errors with MIBQM are found and corrected by testers before running the query, while errors with SQL are found by the system after testers ran the query.

Time / Task	Test1SQL	Test2 SQL	Test3 SQL	Test1 MIBQM	Test2 MIBQM	Test3 MIBQM
Task1	0:25	0:20	0:17	0:12	0:06	0:05
Task2	0:37	0:33	0:25	0:24	0:18	0:23
Task3	2:55	1:45	1:33	0:26	0:18	0:16

Table 1. Average performance time

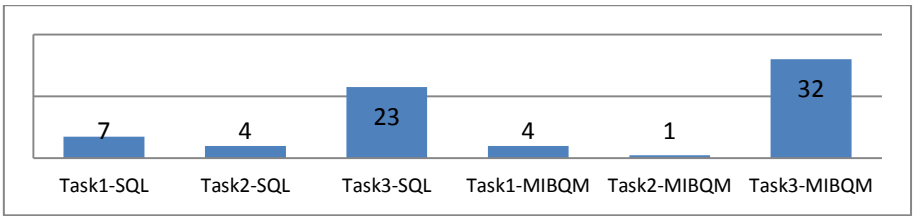

Fig. 6. Testers' number of errors

6.2 Subjective evaluation

In terms of usability, we found out that 80% testers think MIBQM is easy to use and 80% testers think features are easy to be selected. And in terms of satisfaction, all the testers agree that it is easier to add a condition by selecting a feature on the map than by inputting the primary key of the feature in the SQL statement. 80% of testers think that MIBQM performs better in the aspect of leading to fewer errors. All the testers prefer MIBQM to SQL.

Some problems of MIBQM are found during the evaluation. More user guidelines are expected to make the query procedures clearer, so that users will not be confused by the procedure orders. The feature selection method should be improved to avoid invalid selection. For the calculation of topological relationship, one tester thought it more reasonable to make operator selection before layer selection and suggested to use multi-features selection. So in the future work we will focus on these problems to improve the user experience of MIBQM.

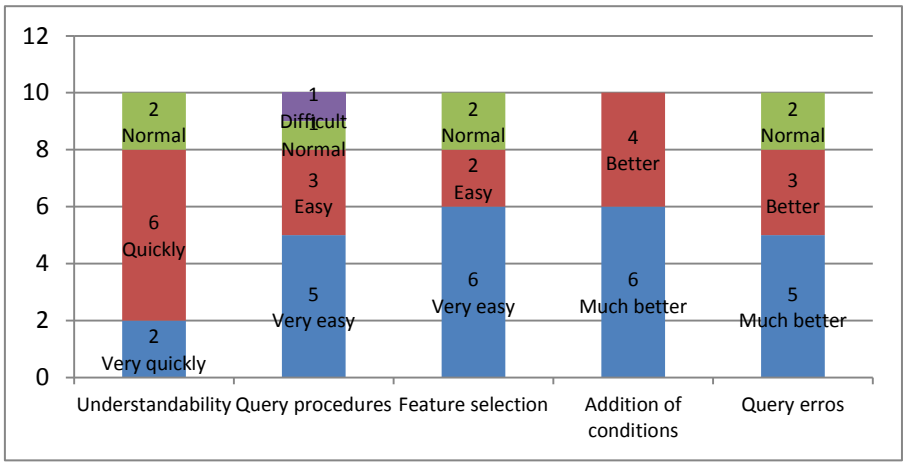

Fig. 7. Assessment of MIBQM

7 Conclusion

In this paper we have presented two spatial query methods which are realized on a touchable GIS interface. MIBQM allows users to construct spatial queries by select-

ing layers and query operators. If a query is made about a specific feature, users can directly select the feature on the map. All the other features in the same layer will be ignored. To make it possible to describe query concepts more precisely, additional conditions should be offered. SDBQM permits spatial query construction by drawing sketches of SGOs. If two SGOs are connected in the spatial relation table, a spatial condition will be added according to their topological relationship. Users can also draw NCOs to add non-spatial conditions. NCOs can be organized in a nested structure by adding CCOs. If a query is too complex, it can be separated to several simple queries. Layers generated in queries can also be represented by special SGOs to be reused. We have made an evaluation between SQL and MIBQM and we found that for novices to GIS, MIBQM seems easier to understand and apply. For simple spatial query tasks, using MIBQM takes less time and leads to fewer errors. In the future, we will compare the SQL and SDQM through constructing complex spatial queries. We also hope to improve MIBQM so that additional conditions can be added.

References

1. Denègre, J., Salgé, F.: Les systèmes d'information géographiques. In : Presses Universitaires de France, 2nd edition (2004)
2. ArcGIS, http://www.esri.com/software/arcgis/arcgis-for-desktop
3. OrbisGIS, http://www.orbisgis.org/
4. Kirby, K., Paner, M.: Graphic Map Algebra. In: Proceedings of the 4th International Symposium on Spatial Data Handling, K. Brassel and H. Kishimoto, eds. Zurich, Switzerland. 413-422 (1990)
5. Lanter, D., Essinger, R.: User-Centered Graphical User Interface Design for GIS. In: Technical Report 91-6, Santa Barbara, CA: National Center for Geographic Information and Analysis (1991)
6. ERDAS, Model Maker Tour Guide. Atlanta, GA: ERDAS, Inc (1993)
7. T. Catarci., M.F. Costabile., S. Levialdi., C. Batini.: Visual query systems for databases: a survey. In: Journal of Visual Languages and Computing , vol.8, pp.215-260 (1997)
8. Calcinelli, D., Mainguenaud, M.: Cigales, a Visual Query Language for a Geographical Information System: the User Interface. In: Journal of Visual Languages and Computing, vol.5, no.2, pp.113-132 (1994)
9. Sebillo, M., Tortora, G., Vitiello, G.: The Metaphor GIS Query Language. In: Journal of Visual Languages and Computing vol.11, pp.439-454 (2000)
10. Ferri, F., Rafanelli, M.: Resolution of Ambiguities in Query Interpretation for Geographical Pictorial Query Languages. In: Journal of Computing and Information Technology, vol.12, no.2, pp.119-126 (2004)
11. Egenhofer, M.J.: Query Processing in Spatial-Query-by-Sketch. In: Journal of Visual Languages and Computing, vol.8, no.4, pp.403-424 (1997)
12. Schoning, J., Raubal, M., Marsh, M., Hecht, B., Kruger, A., Rhos, M.: Improving Interaction with Virtual Globes through Spatial Thinking: Helping users Ask 'Why?'. In: Proceedings of the 13th annual ACM conference on Intelligent User Interfaces. ACM, USA (2008).