WPI-CS-TR-97-10 December 1997

SVIQUEL: A Spatial Visual Query and Exploration Language

by

Sudhir Kaushik
Elke Rundensteiner

Computer Science
Technical Report
Series

WORCESTER POLYTECHNIC INSTITUTE

Computer Science Department
100 Institute Road, Worcester, Massachusetts 01609-2280

SVIQUEL: A Spatial Visual Query and Exploration Language

Sudhir Kaushik   Elke A. Rundensteiner

Department of Computer Science
Worcester Polytechnic Institute, Worcester, MA 01609
sudhir@cs.wpi.edu, rundenst@cs.wpi.edu

Abstract

The need to analyze and query spatial data is becoming increasingly important with the advent of applications such as Geographic Information Systems, Image Databases and Remote Sensing. The focus of our research is to support spatial data analysis by developing a direct manipulation environment to visually query as well as browse spatial data and to review the visual results for trend analysis. In this paper, we present a visual query language (SVIQUEL) which allows us to specify the relative spatial position (both topology and direction) between objects using direct manipulation. This query language builds upon the notion of dynamic query filters and significantly extends them to support integrated querying of both topological and directional types of spatial data. In order to facilitate continuous querying as required by a direct manipulation environment, we designed an integrated neighborhood model for both kinds of spatial relationships (topology and direction). Our spatial query palette SVIQUEL allows us to query over any of the continuous sets of neighboring values. SVIQUEL is complimented by a Spatial Query Disambiguation diagram (SQUAD) which gives qualitative visual representations of the quantitative query. This increases the utility of the system for spatial browsing of data with no particular query in mind. Mapping functions between the quantitative SVIQUEL and the qualitative SQUAD have been developed. The resulting tight coupling between SVIQUEL and SQUAD allows the users to work with either qualitative query specifications or at a quantitative level of detail depending on his particular needs as well as to freely switch between the two while working in a continuous data exploration mode.

1 Introduction

The exploration of large information spaces remains a challenging task especially with the growth of the world wide web and other such huge repositories of information [AS94]. Many users are not able to cope with the flood of information [KS86] [Bor96]. Visual Information Seeking (VIS) addresses this problem by recognizing the enormous capacity of human visual information processing [AS94]. By presenting information visually and allowing dynamic user interaction through direct manipulation paradigms, it is possible to traverse larger information spaces in a shorter time [Shn92]. The key concept is to support browsing, visual query composition, graphical display of the results and continuous reformulation of goals [Shn92]. VIS principles include dynamic query filters, visualization of the results and tight coupling to preserve display invariants and support progressive refinement. The Dynamic Homefinder, a Real Estate Information Exploration System based on VIS principles [WS92], allows users to search for a house that meets their criteria by manipulating sliders for different input parameters (such as number of rooms, price etc) and to get a visual display of the results. An SQL interface for this application would require users to know of the syntax of the language, and also be aware of the contents of the database before specifying queries. On the other hand, a direct manipulation environment allows users to explore the database even without knowing what exactly they are looking for.

Our goal is to apply the concept of visual information seeking to systems with special-purpose data types such as spatial and temporal data. Our previous work, focussed on designing a direct manipulation environment for one dimensional temporal data, called TVQL [HR95], allowed a user to visually specify temporal queries over video data giving an instantaneous visualization of the results of the query. User interface studies of the TVQL show that the users could more accurately specify and more quickly adjust queries using TVQL than using a forms-based approach [HR97] indicating the promise of our approach. This positive feedback now caused us to extend our previous work to spatial data types.

In this current paper, we apply this concept of direct manipulation to the two dimensional (2D) domain of spatial data. Direct manipulation is very useful for spatial database systems as most users of Geographic Information Systems and Image Databases may not be exactly sure as to what they are looking for in an image and may require several iterations before they get the desired result (such as in the above real estate application). In a typical real estate application, a user may have to go through several iterations to understand the trends in the underlying real estate data before he realizes what is available (or what he can afford) in a given region .

We now extend the VIS paradigm to spatial data allowing the user to visually query over the relative positions of two sets of 2D objects. Our Spatial Visual Query and Exploration Language (SVIQUEL) allows the user to query over the relative spatial position of two sets of objects capturing both the topological and directional part of the relationship. The query filters, based on the dynamic query filters used in TVQL [HR95], are tailored for spatial analysis - allowing users to pose queries as well as to browse the data in a spatially continuous manner.

In order to use sliders to exploit the spatial continuity inherent in spatial data, we have developed a neighborhood model that incorporates both direction and topology and thereby defines neighbors between relative spatial positions. This neighborhood model is based on the set of primitive spatial relations that we have identified in our application that integrate topology and direction. Our SVIQUEL interface now allows the user to specify any simple spatial relationship as well as any legal combination of primitives (a neighborhood query) as part of the spatial query.

The SVIQUEL is augmented with a spatial query disambiguation diagram (SQUAD) to give a visual feedback of the query being specified. SVIQUEL allows the user to specify relative spatial position in precise numeric values while SQUAD allows the user to specify the same at a qualitative level of detail. We have devised mappings between the quantitative SVIQUEL and qualitative SQUAD to maintain consistency and synchronization between the two representations allowing users to work either at a quantitative level with the SVIQUEL or at a qualitative level with the SQUAD, giving him the flexibility to switch to the other mode whenever the need for it is felt.

In summary, the main contributions of this work are:

- An integrated framework for spatial data analysis incorporating both direction and topology based on the VIS paradigm.
- A neighborhood model for an integrated relation of both direction and topology as a formal basis for spatial querying.
- A visual query interface for two-dimensional spatial data over this integration of direction and topology (SVIQUEL).
- A qualitative query disambiguation diagram (SQUAD) for a quantitative SVIQUEL query.
- Mapping functions between the quantitative values in SVIQUEL to the qualitative values in SQUAD.

- A working implementation of the proposed system to serve as a proof of concept of ideas. This system has been implemented using the AWT library of JDK1.1.3 on a Unix platform and is designed to be run as an applet.

Section 2 gives a brief background on dynamic query filters and introduces various spatial query types. Section 3 gives an overview of the framework of the system. The SVIQUEL interface is introduced in Section 4 and the underlying formal model of neighborhood when both direction and topology are combined is presented. Section 5 introduces SQUAD as a tool to give visual qualitative representations of the query. Sections 6 and 7 define the mapping functions between the quantitative SVIQUEL and qualitative SQUAD. The dependencies between the SVIQUEL sliders are discussed in Section 8. Section 9 reviews related work and Section 10 concludes the paper.

2 Background

2.1 Dynamic Query Filters

Our spatial query filters are based on double-thumbed filters [All] as shown in Figure 1. The double thumbed slider provides a direct manipulation interface for selecting a continuous range of values from a continuous domain. In the double thumbed dynamic query filter, each thumb has an arrow, pointing towards the range being specified. The thumb arrow is filled to indicate that the endpoint of a range is included or empty to indicate that it is excluded. Selected ranges between the two filter thumbs are filled. The text above the slider indicates the exact value.

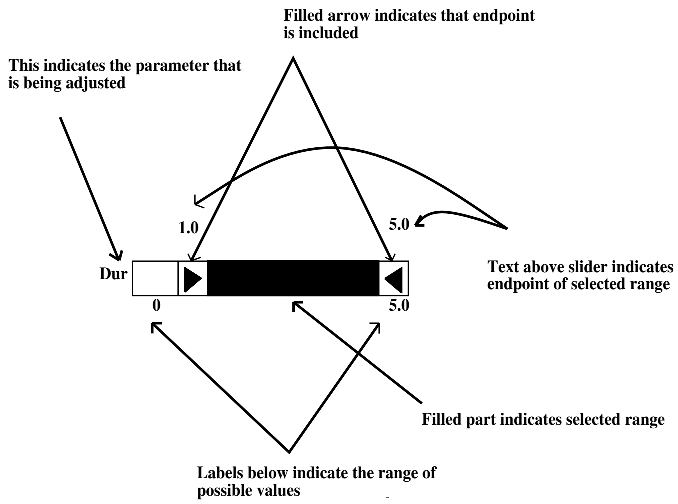


Figure 1: Dynamic Query Filter

Dynamic query filters are used in the Dynamic Homefinder application [WS92] to specify a range of values for various parameters of the house such as its cost, number of rooms and distance from a reference point. However, the Dynamic Homefinder application does not allow the user to specify the relative spatial position of two objects. So, if the user wants a house that is to the north of and touching a park, there is no easy way of specifying that in this system. One of the goals of our work is thus to enhance the dynamic sliders for such continuous spatial queries.

2.2 Spatial Query Types

Spatial queries are often processed qualitatively with the use of spatial relations. Spatial relations can be broadly classified into three basic categories [TP95]:

- Topological Relations
- Directional Relations
- Distance Relations

2.2.1 Topological Relations

Topological relations are those spatial relations which are preserved under groups of continuous transformations such as translation and rotation. In other words, if both the objects are translated or rotated in the same manner, then their topological relationship still remains the same.

Egenhofer's 4-intersection model [EF95] provides us with a means of classifying the various topologies into eight distinct categories: *Disjoint, Meet, Overlaps, Equal, Contains, Inside, Covers and Covered By.* This model considers the intersection values of the objects' boundaries and interiors. If the 4-intersection values of two pairs of point sets have different contents, then their topological relations are different as well, however the reverse cannot be stated. In order to arrive at a one to one correspondence between a value and a relation, Egenhofer further refined this 4-intersection model [EM95] by also considering the intersection between the object's exteriors (9-intersection model). This model gives a unique value for each topological relationship between two pairs of point sets.

2.2.2 Directional Relations

Directional relations are those spatial relations that deal with order in space between two objects. There are essentially two broad categories for directional relations that are commonly used in the literature [JTS96] [TP95].

- 4 way direction - (*North, South, East, West*)
- 8 way direction - (*North, South, East, West, North East, North West, South East, South West*)

The choice between these two models of directional relations depends on the level of granularity or precision at which these relations are to be viewed.

2.2.3 Distance Relations

Distance relations involve the concept of distance between two or more objects. They can be viewed at two levels:

- Qualitative level giving linguistic terms such as near, far and so on.
- Quantitative level giving the actual distance in a particular unit.

There have been a number of approaches to calculate distance between two regions [Her94]. Given two points in any N dimensional space, it is very straightforward to compute the Euclidean distance between them. This is not that obvious in the case of computing the distance between two regions which are two dimensional, as many different distance measures could be chosen to capture the distance between such complex object pairs. Some of the earlier work on distance [KSF$^+$96] between two objects views distance as the degree of similarity between the objects. So, the greater the distance, less similar are the objects.

On the other hand, we view distance as the actual spatial distance between the two regions. We restrict our spatial model at some level to the Minimum Bounding Rectangles (MBR) of complex two dimensional objects as commonly done in [TP95][PTSE95]. One possible approach to compute distance between two MBRs would be to consider the centers of the two objects to be their representative points and use the Euclidean distance between the two points as a measure of their distance between the two actual objects. This however fails in the case of large objects where the distance from one point on the object differs largely from the distance from another point. For example, if we were considering a house and a river as the two objects, since the river is very large compared to the house, the house may be far away from the center of the river but still close to the river. We thus assume a distance measure that essentially considers the shortest distance between the objects as the distance between the two objects. This measure reflects an intuitive measure of the actual distance between the two objects. For example, consider a user who is interested in finding the time it takes to go from Worcester to Boston, he generally refers to the time he takes to enter Boston once he has left Worcester. In other words, this means that he refers to the distance between Worcester and Boston as the shortest distance between the two cities.

2.2.4 Combining Spatial Relationships

There are a number of ways in which these kinds of relationships between two objects could be combined into more precise spatial relationship descriptions. Some of the existing approaches [JTS96] that have explicitly dealt with the integration of topological and directional relations are aimed at building a spatial inference engine. That is, they attempt to infer the spatial relationship between two objects, given a heterogeneous set of both topological and directional relationships. Hernandez [Her94] discusses the advantages of combining both topology and direction into one positional relationship and we propose to use these concepts for our direct manipulation 2D spatial environment.

3 Overview of Our Approach

Figure 2 presents our proposed system framework to applying and extending VIS to the problem of spatial data analysis [AS94] [HR95]. Users explore and analyze the spatial data through iterative specification of queries using our Spatial Visual Query Language (SVIQUEL) and reviewing the visualization of the results (SVIZ). The SVIQUEL interface is composed of dynamic query filters to allow rapid adjustment of query parameters via the use of sliders [AWS92]. This is in contrast to text-based query languages where query specification and modification is typically much more complicated and non-intuitive.

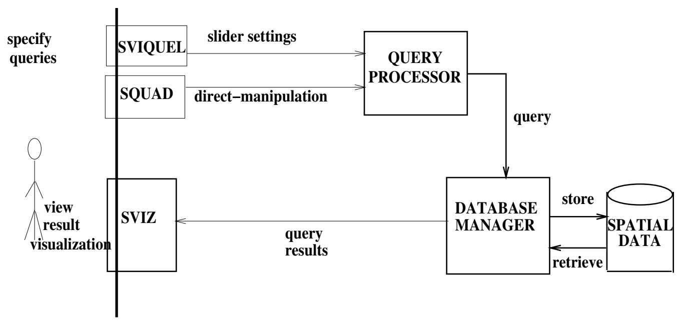


Figure 2: Overall Framework of the System

To the best of our knowledge, this is the first work on applying the VIS paradigm to spatial data types in a principled manner. We extend the VIS paradigm to exploit the characteristics inherent in spatial data such as the spatial continuity of the data, the possibility of sliding from a disjoint to a overlap position in one continuous motion, and the dependencies between the various spatial properties such as direction and topology [HR96a] [HR95]. In particular, we re-examine the basic spatial query types of both topology and direction to develop a spatial query model that consistently integrates those two spatial query classes - thus avoiding the combination of inappropriate spatial predicates and the specification of meaningless queries.

In order to exploit the continuity provided by the use of sliders, we need an underlying neighborhood model to support this incorporation of both classes of spatial queries. This is accomplished by developing a formal neighborhood model for a combination of the two spatial properties of topology and direction. As each of the sliders can specify a range of values, users could ask for a combination of two spatial relations in a query. Since, these sliders can only specify a continuous range of values, they can only specify those combinations that are “neighbors” of each other [Fre]. Our neighborhood model enables us to arrive at what these legal combinations are in the case when these two spatial properties are combined.

The four major components of the system include the Spatial Visual Query Interface (SVIQUEL), the Spatial Query Disambiguation diagram (SQUAD), the Spatial Query Processor and the Spatial Visualization of the results (SVIZ) as shown in Figure 3. The user specifies the query using a combination of SVIQUEL and SQUAD, the Spatial Query Processor processes the query and gives it to the Spatial Visualization component for visualization of the results. The SVIQUEL interface is tightly coupled to a Spatial Query Disambiguation diagram (SQUAD) which gives a qualitative visual representation of the quantitative range query specified. SQUAD presents a valuable aid in confirming the query specified in a visually intuitive manner, as it provides a mechanism to specify the relative positions of two objects - it does not however provide precise quantitative ways of manipulation as offered by SVIQUEL. Hence, these two tools are highly complimentary. For example, the specification of the exact distance between two objects would be accomplished by manipulating our SVIQUEL sliders while the fact that the two objects are disjoint from one another could be visually grasped by quickly scanning the SQUAD representation. The tight coupling between SVIQUEL and SQUAD is achieved by the mapping functions that we have established to translate between the quantitative spatial query in SVIQUEL to a rich qualitative visual presentation in SQUAD.

The queries expressed by SVIQUEL are mapped to range queries to be processed by the SVIQUEL query processor. Once processed against a set of spatial objects in the database, a graphical visualization of the results is presented to the user via our SVIZ. In this way, the visualization is immediately updated whenever users change a query filter. Users can visually scan the results to look for trends. If no trends are found, the users can adjust the sliders in SVIQUEL or directly manipulate the SQUAD to incrementally specify and fine-tune queries. In such an environment, users can gain a sense of causality between adjusting a query filter and the resulting changes presented in the SQUAD and the visualization of the results.

Let us now consider the close interaction between the SVIQUEL query interface, the SQUAD interface and SVIZ visualization. The user initially specifies the query using the sliders which are a part of SVIQUEL. The mapping from the quantitative SVIQUEL to the qualitative SQUAD (Line 1 in Figure 3) derives the SQUAD for the current

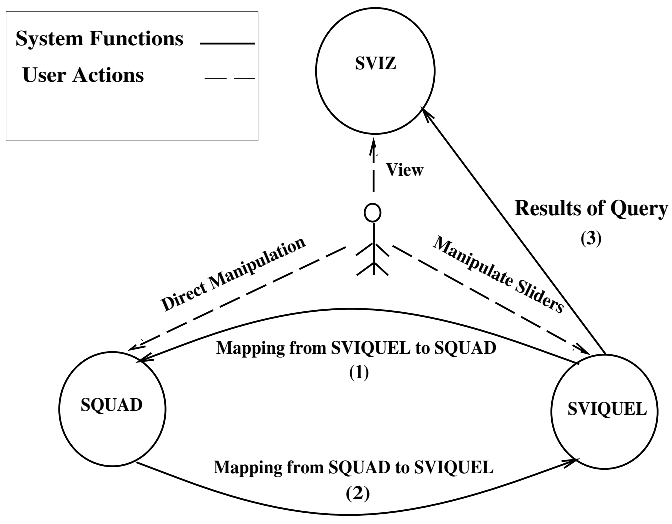


Figure 3: Interaction between SVIQUEL and SQUAD

setting of the sliders. The visualization also gets updated immediately to represent any changes in the output (Line 3 in Figure 3). One unique feature of our system is that we support for the user to directly manipulate the SQUAD to refine the current query using any direct manipulation technique, e.g., clicking on the SQUAD to select and deselect parts of the diagram. The mapping from the quantitative SQUAD to the qualitative SVIQUEL defines the new precise settings of the sliders (see Line 2 in Figure 3). In fact, as both of these interfaces are continuously synchronized with each other, depending on the task at hand (whether it requires precise numeric values for the spatial relationships or only approximate relative spatial positions of objects), the user can work on the interface that best meets their needs and can switch to the alternate interface at any time without losing context. The user could furthermore view the visualization to visually analyze the retrieved sets of data for trends, outliers or other relevant indicators and then directly adjust the sliders in order to be able to quickly scan over large sets of data. Thus, the user iteratively uses the SQUAD and the result visualization along with the sliders in SVIQUEL to incrementally specify the spatial queries until he gets the desired result as shown in Figure 3.

The following are the advantages of SVIQUEL over a standard text-based query interface:

Direct Manipulation Query Interface. SVIQUEL provides a direct manipulation interface where the user can specify queries via simple mouse manipulations. This type of interface relieves the user from having to repeatedly type in queries when the user is in the exploration mode.

Incremental Query Specification. Users can incrementally specify SVIQUEL queries by adjusting parameters of the previous query and seeing the corresponding visualization of the result, as they manipulate the spatial query filters. It allows the user to easily compare results. It is also easy to correct inadvertent errors in the specification of queries by just moving the filter to its previous position. The solution to the query could also be computed from the results of the previous query rather than computing it from scratch. In contrast, text based languages do not provide built-in support for such incremental query specification.

Power to Query and Browse. Text-based query languages typically support users who know what they are looking for and are in a position to specify a particular query in the desired syntax. In contrast, SVIQUEL provides users with support to both:

- specify a precise query when they know what they are looking for, or
- spatially browsing the data if no particular query in mind.

Browsing is supported by the power of direct manipulation to specify queries using buttons and sliders and use them to slide between spatial neighborhoods. An advantage of browsing is that it gives the user the chance to discover relationships he may not be aware of.

Visual Feedback. The spatial diagrams help the user to identify what query is being specified by the user. Text based query languages generally do not provide such feedback.

4 SVIQUEL Interface

The following are the goals in designing SVIQUEL:

- a visual interface to specify relative position spatial queries of any type such as topological or directional relationships in a precise yet simple manner.
- ability to quickly adjust spatial queries via direct manipulation and to incrementally specify spatial queries.
- power to both specify particular queries as well as browse over spatial relationships, with no particular query in mind.
- ability to compose complex queries such as a combination of spatial relationships in a simple manner.

Given two spatial objects A and B, the spatial (topological, directional or distance) relationship between the two objects could be inferred from the relationships between the extremities of each object, along both the X and Y dimension [1]. Fully constraining relative object placement gives complete information about how B is spatially placed with respect to A, such as whether B is to the north and overlaps with A. Our interface constrains the placement of one object B called the *primary* object with respect to the second object A that serves as the *reference* object [PTSE95]. To query over the relative spatial position, we design our language to specify any primitive spatial relationship as well as combinations of these primitive relationships as discussed below.

4.1 Primitive Spatial Relations (SPRIMs)

Given two spatial objects A and B, our SVIQUEL interface distinguishes between eight possible primitive topological relationships (disjoint, meet, overlap, covers, coveredBy, contains, inside and equal) and eight possible directional relationships (north, south, east, west, north east, north west, south east and south west) between the two objects and two special relationships of same-longitude and same-latitude. Since it is possible for two regions to have both kinds of relationships, each combination of a topology and a direction forms a *primitive* spatial relation (a SPRIM).

In our model, we have identified 45 meaningful primitives among the 64 possible ones (Figure 4), which capture both the topological and the directional relationship between the regions. The reason we have 45 spatial primitives

[1] One important assumption here is that we are approximating each object by its Minimum Bounding Rectangle (MBR) with its sides oriented along the X and Y axes, and the spatial relationship between the two objects is expressed in terms of its MBR

(SPRIMs) and not 64 is that for certain topologies like *inside*, *contains* or *equals* having a directional component is not very meaningful [Her94]. The SPRIMs as given in Figure 4 can be depicted in terms of the differences between

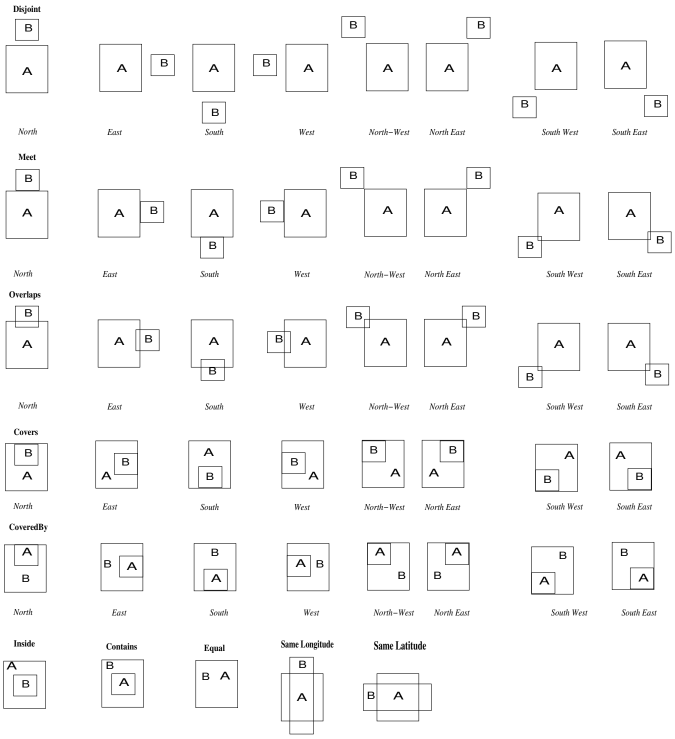


Figure 4: Spatial Primitives (SPRIMs) with Topology and Direction Combined

the left and right ends in the X dimension and top and bottom ends in the Y dimension of the two objects A and B. The left and right ends of an object are its extremities in the X dimension while the top and bottom ends are its extremities in the Y dimension. Consider two objects A and B with the extremities for each object along the X dimension given by Axl,Axr,Bxl,Bxr [2]. The spatial relationship between the two objects is defined by a conjunction of the eight pairwise relationships between the extremities in each dimension. In many cases, fewer than eight differences are sufficient to represent the relative spatial position of two objects:

((Axl θ Bxl) AND (Axl θ Bxr) AND (Axr θ Bxl) AND (Axr θ Bxr)) AND ((Ayt θ Byt) AND (Ayt θ Byb) AND (Ayb θ Byt) AND (Ayb θ Byb)) where $\theta = \{ <, >, = \}$.

Each of these eight expressions combined by the AND operator shows the relative spatial position of the corresponding extremity of the two objects. For example, Axl $<$ Bxl indicates that the left end of A, denoted by Axl, has to be to the left ("$<$") of the left end of B, denoted by Bxl. [3] [4] In our SVIQUEL interface, dTl,dTr,dTlr and dTrl represent these differences in the X dimension and dTt,dTb,dTtb and dTbt represent these differences in the Y dimension.

[2] Axl and Axr are the left and right extremities of A in the X dimension respectively and the same holds true for B also and along the Y dimension given by Ayt,Ayb,Byt,Byb. Ayt and Ayb are the top and bottom extremities of A in the Y dimension respectively and the same holds true for B also

[3] We assume that each object has a finite non-zero width and height and also Axl $<$ Axr, Ayt $<$ Ayb, Bxl $<$ Bxr, Byt $<$ Byb always.

[4] The sizes of A and B do not convey any information about the size of the actual objects that they represent

In order to make this clearer, let us consider a sample scenario of two objects, a house and a lake. Let the house be the primary object B and the lake be the reference object A. Suppose that the house is *disjoint* from the lake and to the *north*. This spatial relationship between the two objects could be expressed in terms of the eight differences, represented by the constants as introduced above.

- dTl = (positive or negative) // Left A - Left B
- dTlr = (negative) // Left A - Right B
- dTrl = (positive) // Right A - Left B
- dTr = (positive or negative) // Right A - Right B
- dTt =(negative) // Top A - Top B
- dTtb = (negative) // Top A - Bottom B
- dTbt =(negative) // Bottom A - Top B
- dTb = (negative) // Bottom A - Bottom B

The values dTt,dTbt,dTtb and dTb being negative implies that object B has to be above object A and so the topology is disjoint. Now, since dTlr is negative and dTrl is positive, it implies a portion of object B has to be exactly above object A and thus we can infer that B is to the north of A. So, the relative position of B with respect to A is *disjoint-north* (see the top leftmost primitive (SPRIM) in Figure 4).

4.2 The SVIQUEL Sliders

As outlined above, both the directional and topological content of a spatial relationship can be represented using these eight differences. In fact, different value combinations of these eight differences can uniquely specify all primitive spatial relationships (topological and directional) as shown in Figure 4. Each of these differences can assume a continuous range of values and therefore dynamic query sliders can be used to specify each of these eight differences. Hence, we use eight dynamic query sliders as part of our Spatial Visual Query and Exploration Language (SVIQUEL). These eight are split up as four along each dimension as each of the two 2D objects has two endpoints along each dimension (see Figure 5). Therefore, there are four endpoint relationships that each object has along each dimension. dTl,dTr,dTlr and dTrl are the relationships in the X dimension and dTt,dTb,dTtb and dTbt are the relationships along the Y dimension. Descriptive labels are provided alongside the filters so that the user is able to interpret the filters better. For example, the label (Left A, Left B) along the top-most slider indicates that the slider specifies a continuous range of values for the difference Left A - Left B. Figure 5 shows the interface of SVIQUEL for the query "Give me all cases where B is *disjoint-north* of A". Along the X dimension the slider (Left A - Right B) is set to negative values while (Right A - Left B) is set to positive values. Along the Y dimension, all sliders are set to negative values. The figure shows all the positions B can be in with respect to A in the upper right hand quadrant.

4.3 Integrated Neighborhood Concept for Topology and Direction

In addition to selecting one SPRIM among the 45 distinct possible spatial relations (SPRIMs), our SVIQUEL interface is designed to also allow the user to specify a combination of these primitives. Since our filters enable us to specify

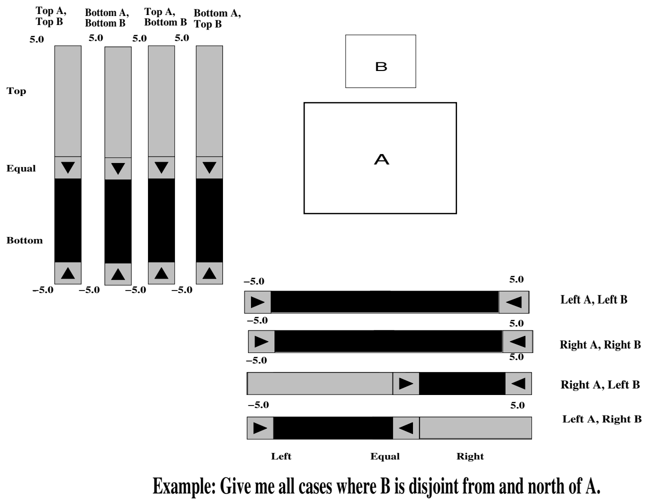


Figure 5: SVQL Interface for query "Find all cases where B is disjoint from A and to the north of A"

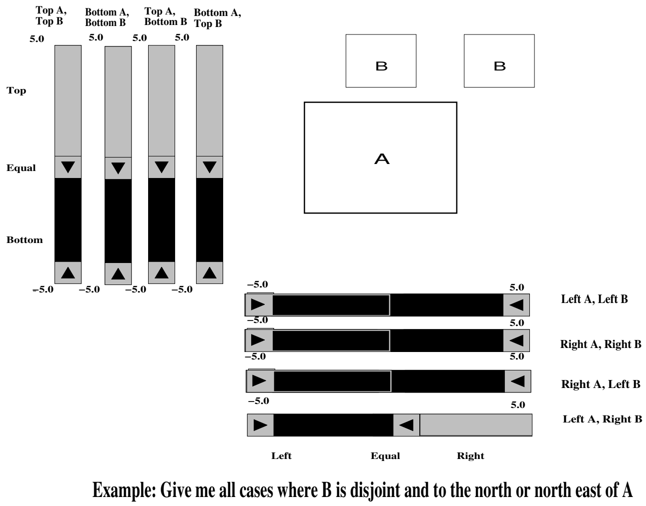


Figure 6: SVQL Interface for query "Find all cases where B is disjoint from A and to the north or north east of A"

a range of values for each of the sliders, we can also use them to specify combinations of spatial relations.[5] Let us consider the scenario where we have a number of houses in the vicinity of a lake and the user is interested in buying a house in the region. Suppose he asks the query, "Give me those houses which are to the *north* or the *north east* of the lake". In our example above, the user essentially has two spatial relations specified between the house and the lake, one is the *north* directional relation and the other is the *north east* directional relation. Since the filters specify a range of values, the user could ask for these two relations in a single query by proper adjustment of the appropriate filters. This way, the user is actually querying for combinations of spatial relations. Figure 6 shows the settings of the sliders for the configuration *disjoint-north.* If the user now adjusts the slider "Right A - Left B" to also include negative values, the configuration *disjoint-north east* is also included.

On the other hand, since the filters operate over continuous ranges of values, they can be only used to specify those combinations of relations which are "neighbors" of each other. The specification of continuous ranges of values by the sliders enables us to go from one spatial relation to the one closest to it by the simple move of the mouse and thus fosters rapid yet continuous exploration of the data set. Going to any arbitrary relation from the current one would require the skipping of some values in the middle which is not supported by our query paradigm.

This observation requires us to establish a precise definition for this intuitive concept of neighborhood viz the relations closest to a given relation form a neighborhood. Little work has been done in modeling conceptual neighborhoods among spatial relations. Freska [Fre] introduced the concept of neighborhood with respect to 1D temporal data. Freska defined two temporal relations to be conceptual neighbors of each other if a continuous change (e.g. shortening, lengthening, or moving the duration of the events) can be used to transform one relation into another without passing through an additional temporal primitive.

[5]For simplicity, we follow the common approach of modeling both the primary and the reference objects are modeled as two rectangles (the Minimum Bounding Rectangles of the two objects). The filters can be used to manipulate the left, right, top and bottom extremities of the rectangle.

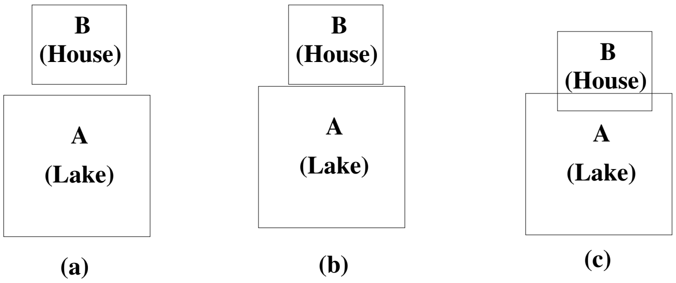
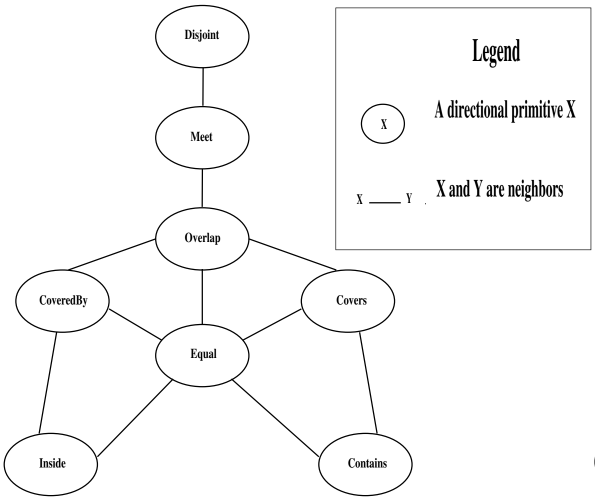


Figure 7: The relative spatial positions of the objects A and B for configurations (a)*disjoint-north*, (b)*meet-north* and (c)*overlap-north* respectively

We now consider to extend this concept of neighborhoods to the case of 2D spatial relations. Let us consider the same scenario of a number of houses in the vicinity of the lake first. Suppose the user specifies the query "Give me all houses that are *disjoint* from the park and to the *north*". The corresponding SPRIM is shown in Figure 7.a. Now suppose the user modifies his query to "Give me all houses which *meet* the park at the *north*" (see Figure 7.b). This just involves moving the bottom Y coordinate of the *house* object down until the top of the park object meets the bottom of the house object. This does not include any other SPRIM as part of the query and hence *disjoint-north* and *meet-north* are indeed neighbors. If the user however were to ask the query "Give me all houses that *overlap* the park from the north"(see Figure 7.c). This would involve moving the bottom of the house object down until it is below the top of the park object. This however passes through a stage where the top of the park object meets the bottom of the house object. So, the configurations *disjoint-north* and *overlap-north* are not neighbors as there is a configuration *meet north* that falls in between the two.



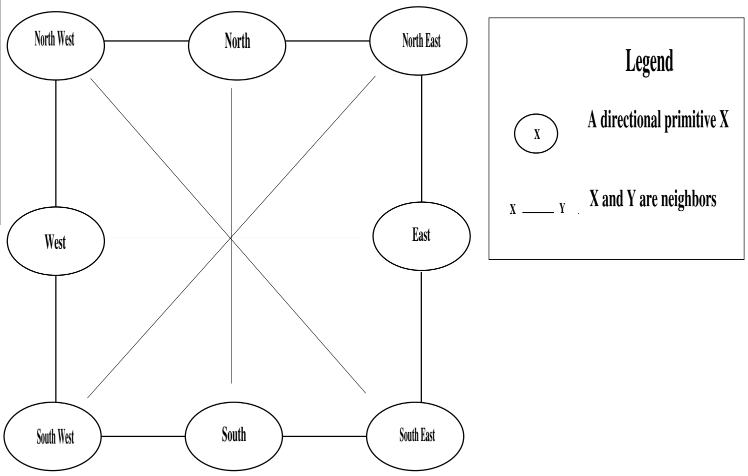


Figure 8: Neighborhood Model for Topological Relations Figure 9: Neighborhood Model for Directional Relations

As we have discussed earlier, each SPRIM in our model is a combination of both topology and direction. We need to find a way of combining the individual neighborhood models proposed in the literature for direction [Her94] (see Figure 9) and with those for topology [ET92] (see Figure 8) respectively in order to develop an integrated neighborhood model.

For topology, we borrow the concept of neighborhoods among topological spatial relations between a line and a region (see Figure 8) as introduced in [ET92]. This model considers the intersections between the interior, boundary and the exterior of the line and the region, and differentiates between the topologies based on these 9 intersection values. For direction, we use the concept of neighborhood as developed by Hernandez [Her94] (see Figurefig7). Given our goal of querying over both topology and direction, we propose to develop a neighborhood model for an aggregate relationship integrating both direction and topology. In short, our proposed neighborhood model is a natural extension of the two individual models. In the sense, when we ignore one of the two spatial components (direction or topology), our model defaults to the individual known model. The neighbors for a particular SPRIM can be obtained in our model using the knowledge of its neighbors with respect to topology and with respect to direction. So, the neighbors for a particular spatial primitive can be deduced as follows:

- Those relations which have the same topology but are neighbors with respect to direction.
- Those relations which have the same direction but are neighbors with respect to topology.

The underlying assumption here is that the user is allowed to go from one spatial primitive to the other only by moving along the two axes of our 2D space, one at a time, i.e., he is not allowed to move diagonally along the two axes. The chart showing the neighborhood relations of the SPRIMs in the *north*, *north east* and *north west*

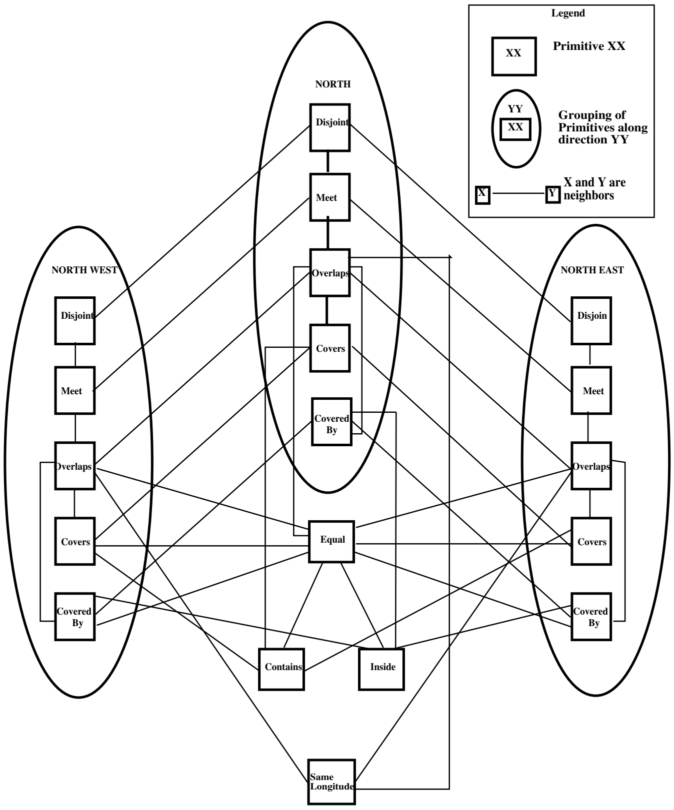


Figure 10: Our Neighborhood Model Combining Topology and Direction Relations.

directions are shown in Figure 10. [6] The other neighborhood relationships can also be derived in a similar manner

[6] The configuration *same-longitude* is the neighbor of *overlap-north* and *overlap-south* and also to all neighbors of these two configu-

and are omitted here for space reasons. The configurations *same-longitude* and *same-latitude* are special cases. This formal model of neighborhood enables us to specify all legal combinations of relative spatial positions that could be specified by SVIQUEL as well as incremental transitions between them. Thus, each SVIQUEL query corresponds to a set of neighboring SPRIMs and the interface allows to move from one SPRIM to another if and only if they form a neighborhood set. Let us consider one example where SVIQUEL is used to incrementally specify transitions between neighboring relations. For example, again considering the house scenario, suppose the query is "Give me cases where the house is *disjoint* and to the *north* of the lake". This would require the filter Top A - Bottom B to be set to a value less than zero as the topology is *disjoint*. Also, since the direction is *north*, the filter Left A - Left B should be less than or equal to zero, and Right A - Right B to be greater than or equal to zero. Figure 5 shows the SVIQUEL interface for this query. If now the user pulls the slider Right A - Right B to include values less than zero also, then the configuration *disjoint north east* is also included. Thus the user specifies a combination of two neighboring SPRIMs in his query (see Figure 6).

5 SQUAD: Spatial Query Disambiguation Diagram

Formal user studies of temporal queries in MMVIS have shown that a graphical depiction of a slider-based query for query disambiguation results in better performance and easier comprehension for the users [HR97]. Borrowing this idea of qualitative query disambiguation of MMVIS [HR96b] , we now propose to add a Spatial Query Disambiguation diagram as part of the SVIQUEL palette (SQUAD) to give visual *qualitative* representations of the query to the user. SQUAD represents a valuable aid for confirming the specified query to the users in a visually intuitive manner. This should increase the speed and the certainty with which the users modify and construct complex queries. Another advantage of SQUAD is that it increases the utility of the SVIQUEL interface for spatial browsing of the data with no particular query in mind. That is both SQUAD and the visualization of the results (SVIZ) are dynamically updated as users manipulate the sliders, thereby enabling the users to simply slide the filters front and back until an interesting result appears and then see the diagram to see what query was specified.

Our approach is to express SQUAD as the union of all the neighboring SPRIMs specified by the user as part of the SVIQUEL query. In other words, the SQUAD shows all the primitive positions the object B could occupy with respect to a reference object A with each possible relative position graphically depicted by one SPRIM primitive (see top right hand portion of Figure 6). In order to be able to arrive at the SQUAD, it is necessary to know the conditions that must be satisfied for a particular SPRIM to be displayed. As discussed in Section 4.1, each primitive is expressed in terms of the eight differences in the two dimensions. In order to determine if a particular primitive holds true, the required values for these eight differences for that primitive have to be determined.

Let us now consider the case of B being *disjoint-north* of A. The actual cases handled for the configuration *disjoint-north* are those in which there is at least some part of B that is exactly above the object A. The possible positions that B could occupy relative to object A are as shown in Figure 11. The whole object B however does not have to be above object A. So, the only constraints that exist for object B in the X dimension are that dTlr < 0 and dTrl > 0. Also, since the object B is disjoint from object A, all differences in the Y dimension dTt,dTb,dTtb and dTbt

rations. In the case of *same-latitude*, its neighbors are *overlap-east*, *overlap-west* and all the neighbors of these configurations.

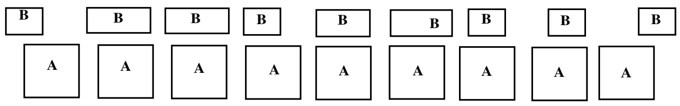


Figure 11: Actual Cases Handled for the Configuration *disjoint-north*.

are set to negative. So, the values for these eight differences for the primitive *disjoint-north* to be true and thus to be displayed are as shown in Table 1. The conditions for all other particular primitives to be a part of SQUAD can be determined similarly.

	dTl	dTr	dTlr	dTrl	dTt	dTb	dTtb	dTbt
Disjoint-North	X	X	<0	>0	<0	<0	<0	<0

Table 1: Conditions for the SPRIM *disjoint-north* Where X Denotes "irrelevant"

6 Mapping from SVIQUEL to SQUAD

As discussed in Section 4, the SVIQUEL interface consists of eight sliders that enable us to specify the relative spatial position between two objects. Given a particular setting of these eight sliders, our system needs to determine the corresponding SQUAD that visually depicts the same query as described by the sliders, though at a qualitative level instead of a quantitative level of detail. In other words, we need to develop an unambiguous mapping from the setting of the SVIQUEL sliders to the SQUAD representation (i.e., to a set of selected SPRIMs). This effectively corresponds to determining the maximal range query condition for each SPRIM.

SVIQUEL is quantitative, e.g., we have eight range queries each represented by its two endpoints over a continuous (numeric) domain, whereas the conditions in SQUAD are qualitative, i.e., they are set to '<','>0' or '0'. Hence, we need to map these numeric ranges to qualitative values. The mapping, as discussed below, helps us to arrive at the values for each of these differences. Below, we characterize the mapping mechanism for one slider and the same mapping principle can be applied to the other seven sliders as well.

Let dT' be one of the 8 sliders and dT'.l and dT'.r be the left and right thumbs of the slider dT'. Assuming the range of dT' slider is set to the continuous domain of $(min_{dT} \ldots max_{dT})$ where without loss of generality $min_{dT} < 0$ and $max_{dT} > 0$, then dT'.l and dT'.r can take on values in the range of $(min_{dT} \ldots max_{dT})$ with the constraint being $min_{dT} <= dT'.l <= dT'.r <= max_{dT}$. Now let the symbol dT denote the set of quantitative values that hold for slider dT', with the range of possible qualitative values being {'<0', '>0', '=0'}. Correspondingly, let dT.l and dT.r be symbols to denote the quantitative values of the left and right thumbs of slider dT'. Again, dT.l and dT.r can take on values from the set {'<0','>0', '=0'} with the only constraint being $dT.l <= dT.r$. The following table gives us the value of dT needed for SQUAD given the values of dT'.l and dT'.r (and hence dT.l and dT.r) from the corresponding SVIQUEL query.

In Table 2, X indicates that the condition is invalid as it violates the constraint that $dT.l <= dT.r$. The sets of qualitative values in the table indicate that any of the values in the set is in the allowed range. For example, the set

		dT.r		
	dT	< 0	=0	>0
dT.l	<0	{<0}	{<0,=0}	{<0,=0,>0}
	=0	X	{=0}	{0,>0}
	>0	X	X	{>0}

Table 2: Mapping Function from SVIQUEL to SQUAD.

{'<0','=0'} implies that the value dT could be either '<0' or '=0'. Given values of dT.l and dT.r for any slider dT', the values for the difference dT are determined using this mapping function in Table 2. Now, once the value of dT is determined for each of the eight sliders, it is possible to determine if a particular SPRIM is within the allowed range of the query. This is done by comparing whether the values of the eight differences imply the required conditions for a given primitive as shown in Table 1. We thus can use this mapping function to decide which SPRIMs are part of the user specified SVIQUEL query.

7 Mapping from SQUAD to SVIQUEL

SQUAD gives a visual qualitative representation of the query as specified by SVIQUEL. Using direct manipulation techniques, users can select and deselect the primitives shown in SQUAD in order to further refine the query. Alternatively, the user could choose to first work with the qualitative SQUAD environment to get the relative spatial positions that are desired and once the query specification (or the retrieved results) are satisfactory, the user could turn to the SVIQUEL interface to finish off the query specification by setting more precise numeric values (such as the amount of overlap between the two objects). In order to keep our query interfaces synchronized, we also need a mapping mechanism that adjusts the setting of SVIQUEL sliders (quantitative) in response to SQUAD changes (qualitative).

Let primitive P be a new SPRIM that is selected in SQUAD using a direct manipulation technique, e.g., using a mouse to click on the primitive. Let us assume all notations as introduced in Section 6 for the mapping from SVIQUEL to SQUAD such as dT', one of the eight sliders, and dT'.l and dT'.r the left and right thumbs. Now let the symbol dT denote the set of quantitative values that hold for this slider dT' and dT.l and dT.r be the qualitative values of the left and right thumbs of slider dT'. Table 3 gives us the values of dT'.l and dT'.r (and hence dT.l and dT.r) for a slider dT', given the value of dT for a primitive P. For each SPRIM P, we know its set of conditions which correspond to a set of qualitative values dT_i for $i = 1, \ldots, 8$. Table 1 gives the conditions that must hold true for the primitive *disjoint-north*.

For example, if a particular primitive P of SQUAD requires that the value of slider dT' be negative, i.e., dT = {'<0'}, then using the mapping function in Table 3, the left and right ends of slider dT' are both set to '< 0' as well. The important point here is that the mapping function only gives qualitative values for each end of a particular slider. So, in the above example, if the left end of the slider dT' had already been set to some numeric value with the same orientation, e.g., some negative numeric value, then no change is made to its setting as the mapping function just

dT	dT.l	dT.r
<0	<0	<0
=0	=0	=0
>0	>0	>0

Table 3: Mapping Function from SQUAD to SVIQUEL.

requires that this value is negative (as indicated by dT = {'<0'}). That is, the precise numeric setting of SVIQUEL overrides the vague qualitative setting of SQUAD. On the other hand, if the value of the left end had been set to a positive value, then it is now set to a negative value. In this case, as no precise numeric value is known, we assume the full range of possible negative values, which is equivalent to setting dT'.l to the lowest negative value, i.e dT'.l = min_{dT}. So, knowing the SQUAD primitive and the old values of a slider dT', the values of the left and right ends of the slider dT'.l and dT'.r can be found using this mapping function. In other words, the value of dT'.l (the same applies to dT'.r) can be found out using the following equation.

$$dT'.l_{new} = \begin{cases} dT'.l_{old} & \text{if dT.l} < 0 \text{ and } dT'.l_{old} < 0 \\ min_{dT} & \text{if dT.l} < 0 \text{ and } dT'.l_{old} \geq 0 \end{cases}$$

In the case of having more than one SPRIM selected as part of SQUAD, the mapping function is modified as follows: Let $P_1, \ldots, P_n$ be the selected SPRIMs. Let $dT.l_1, \ldots, dT.l_n$ be the values of the left end of a slider dT' for these primitives $P_1, \ldots, P_n$ using the notation as defined above. Then we set, $dT.l = min(dT.l_1, \ldots, dT.l_n)$. Similarly, let $dT.r_1, \ldots dT.r_n$ be the values of the right end of a slider dT' for these primitives $P_1, \ldots, P_n$ then we set $dT.r = max(dT.r_1, \ldots, dT.r_n)$. Once these values of dT.l and dT.r are found, the value of dT'.l and dT'.r could be found using the above equation.

This produces a setting of the sliders that satisfies all the SPRIMs that are part of the SQUAD. This is accomplished by arriving at the largest range of values the slider dT' could take on, which is equivalent to the lowest value that dT'.l could take on and the highest value that dT'.r could take on. Once the new settings of the sliders are known using the mapping from SVIQUEL to SQUAD, the SQUAD is redrawn so that the resulting SQUAD is consistent with the concept of neighborhood introduced in Section 4.3.

8 Constraints and Dependencies Between the SVIQUEL Filters

As discussed earlier, eight filters are used to specify the relative directional and topological placement of an object B with respect to a reference object A in SVIQUEL. Many times fewer than eight differences are sufficient to specify the relative spatial position of two objects. This is because specifying some of the differences puts some constraints on values that the other differences can take. Therefore, we introduce dependencies between the filters so that only legal combinations of values can be specified using our interface. There are two main advantages of creating such dependencies:

- it avoids the specification of illegal queries by the users
- it simplifies the process of query specification by auto setting the sliders for the users whenever possible

8.1 Dependencies Among the Two Dimensions

As we have seen, the filters are used to specify a continuous range of scalar values and therefore filters along the X dimension can be used to specify X values such as the relationship of the left and right X coordinates of object A with respect to the left and right coordinates of object B. The same holds true for the Y dimension also. On the other hand, a filter along the X dimension places no constraints on the Y values associated with those X values. These Y values have to be adjusted with the corresponding Y filters instead. This means that the X and Y dimension filters are essentially independent of each other and there exists no dependency between the filters in the two dimensions. We have thus reduced the two dimensional data dependencies to the treatment of two independent one dimensional query filters, the remaining discussion thus parallels work on one-dimensional slider dependencies [HR95].

8.2 Dependencies Between the Filters in One Dimension

Below, we consider the dependencies among the four filters in the X dimension, while the same also holds true for those in the Y dimension. In the spirit of a user-friendly direct manipulation environment, these four spatial queries are bound to one another so that the user cannot specify invalid queries. That is when one filter is changed, the other three are also changed automatically by our system to exclude invalid ranges, if necessary. The table in

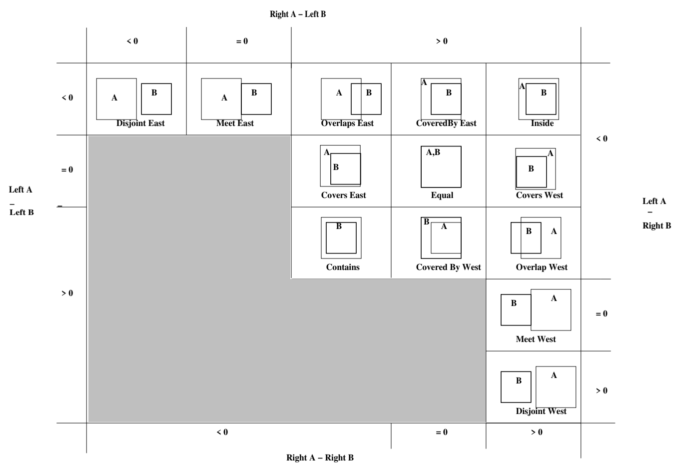


Figure 12: Relationship Between the Spatial Query Filters along X Dimension and SPRIMs

Figure 12 shows the relationships between the filters in the X dimension and the corresponding SPRIMs. The same holds true for the Y dimension. Based on the fact that both the objects A and B have non-zero width and height, the table identifies invalid combinations and sets constraints for the filters. In this way, the table indicates how the filters bind to one another. Consider the case when we are looking for objects that have the same bounding X value on the left (start at the same point in the X dimension), we are also constraining our search to find all objects in which Right A - Left B > 0 and Left A - Right B < 0. In other words, setting Left A - Left B $= 0$ is equivalent to selecting the second row of primitives (from *Covers-East* to *Covers-West*) and in this row, Right A - Left B > 0 and Left A - Right B < 0 are the only possible values for the relationships. Suppose the user sets the top filter for Left A - Left B to 0. Then the system adjusts the bottom filters so that Right A - Left B > 0 and Left A - Right B < 0, because these are the only valid ranges for them when Left A - Left B $= 0$. Hence, the second filter Right A -

Right B remains unchanged as it can have any value. Next assume the user slides the second filter to set Right A - Right B = 0, neither of the other filters gets updated because they are already set to exclude invalid values. There are however no constraints on Right A - Right B as it can have any value. This makes sense, as having two objects which start at the same point in the X dimension says nothing about its value on the right.

While the table in Figure 12 gives qualitative constraints, it says nothing about quantitative values. For example, if we are looking for all those B objects that overlap object A with a unit of 1 from the left, then we must make sure that the range specification for Right A - Left B includes some values greater than 1. This is due to the inherent condition that all objects have a non zero width and height. From the table in Figure 12, we have that when Left A - Left B = 1 unit, we must also have Right A - Left B > 0. However, we do not have any detailed information as to by how much Right A - Left B is greater than zero. Now we know that Width A = Right A - Left A and Width B = RightB - Left B and all objects have non-zero width and height, Right A - Left B should be greater than 1. We can treat all the relationships between the spatial query filters in a similar manner.

DQ Filter	Relationships to other DQ Filters	Implications for constraints
Left A – Left B = dTl	= dTrl – width A	dTrl.left > dTr.left (1) dTrl.right > dTr.right (2)
	= dTlr + width B	dTlr.left < dTl.left (3) dTlr.right < dTl.left (4)
Right A – Right B = dTr	= dTrl – width B	dTrl.left > dTr.left (5) dTrl.right > dTf.right (6)
	= dTlr +width A	dTlr.left < dTr.left (7) dTlr.right < dTr.right (8)
Right A – Left B = dTrl	= dTl + width A = dTr + width B	(1) AND (2) (5) AND (6)
	= dTlr + width A + width B	above AND (3) AND (4) AND (7) AND (8)
Left A – Right B = dTlr	= dTl – width B = dTr – width A	(3) AND (4) (7) AND (8)
	= dTrl – width A – width B	above AND (1) AND (2) AND (5) AND (6)

DQ	Move	dTl (Left A – Left B) Left Thumb	dTl (Left A – Left B) Right Thumb	dTr (Right A – Right B) Left Thumb	dTr (Right A – Right B) Right Thumb	dTrl (Right A – Left B) Left Thumb	dTrl (Right A – Left B) Right Thumb	dTlr (Left A – Right B) Left Thumb	dTlr (Left A – Right B) Right Thumb
dTl (Left A – Right A)	left , in					(1)			
	right, out						(2)		
	left, out							(3)	
	right,out								(4)
dTr (Right A – Right B)	left , in					(5)			
	right, out						(6)		
	left, out							(7)	
	right,out								(8)
dTrl (Right A – Left B)	left , in								
	right, out								
	left, out	(1)		(5)				(3),(7)	
	right,out		(2)		(6)				(4),(8)
dTlr (Left A – Right B)	left , in	(3)		(7)			(1),(5)		
	right, out		(4)		(8)			(2),(6)	
	left, out								
	right,out								

Figure 13: Algebraic relationships between Spatial Query Filters and Implications for Constraints Between Filters

Figure 14: Spatial Query Filter Interdependencies For Each Type of Filter Manipulation

The table in Figure 13 summarizes the filter relationships and indicates the implications of these relationships for constraining the interdependencies between the filters. These implications can now be used to dictate the quantitative constraints between the filters. Assuming that all A and B objects have non-zero width and height, the table indicates that adjusting the top two filters, the bottom two also get adjusted to prevent invalid ranges. Similarly, if the bottom two filters are adjusted, all three are checked for possible constraints. For example, from the table in Figure 13, when the filter Left A - Left B filter is manipulated, constraints 1 to 4 are maintained. However there are four different types of manipulations, and we can determine which constraints need to be checked for which type of manipulation. That is, each SVIQUEL slider has two filter thumbs and each manipulation (dragging a filter or toggling the fill on or off) results in moving a thumb in one of the two directions. Thus each filter is manipulated in one of the following ways:

- Left,In - Moving the left thumb in (increases lower bound of range)
- Right,Out - Moving the right thumb out (increases upper bound of range)
- Left,Out - Moving the left filter thumb out (decreases lower bound of range)
- Right,In - Moving the right filter thumb in (decreases upper bound of range)

Combining this information on slider manipulations with the constraints, we can define the spatial query filter interdependencies for each type of filter (see Table in Figure 14).

9 Related Work

Although a lot of work has been done on various spatial database issues, not much has been done in applying the direct manipulation paradigm to spatial environments. Visual query environments are based on the VIS paradigms introduced in [Shn92] which require a dynamic querying mechanism and a graphical display of results. We borrow their ideas of sliders and coupling of interfaces for feedback but add new features for spatial data types.

Most of the existing spatial query languages are SQL based, requiring users to refer to geometric data using textual queries [Ege94] [CW96]. Some of them map complex spatial data into the (simple) relational model, then requiring the users to understand the implementation details of spatial data. GEO-QUEL [BS77] and Query by Pictorial Example [CF80] are examples of query languages where an image depicting the relationship is drawn as a query to the system. The disadvantage of SQL type languages [Ege94] is that, users often think of spatial relationships in terms of images depicting the actual spatial positions, and the SQL type languages require the translation of this into a non-spatial language. Egenhofer proposed Spatial SQL [Ege94], which incorporates spatial operations and relationships into SQL. This, like the other text based approaches, requires the user to know what he is looking for when he specifies a query and does not allow for an incremental refinement of the query as typically needed in an analysis type of mode. Also, it requires the user to learn the language and its terminology before he can specify any query.

Our work focuses on applying the direct manipulation paradigm to a spatial environment and designing a visual query interface for the same. Cigales [WCL+94] and Spatial Query by Sketch [Ege] are examples of visual spatial query languages. Based on the Query by Example paradigm, they ask the user to specify the query with an actual drawing. They however require the user to be aware of what he is looking for and are not of much use to a user who is just browsing the spatial database in search of something interesting.

Theodoridis [TP95] gives a classification of all spatial relations into three main categories as topological, directional or distance relations. We combine the two types of spatial relationships of direction and topology into one composite relation and have built one query interface to query over this composite relation. Also, while a number of papers [JTS96] [SYLL95] focus on building an inference engine for a heterogeneous collection of directional and topological relations, no one besides [Her94] attempts to aggregate them into one composite relationship.

Hernandez [Her94] discusses the composition of topology and direction into one relation. We extend this work by giving a concrete definition to the concept of neighborhood when direction and topology are combined into one aggregate relation. Combining both direction and topology into one relation enables the user to specify relative position of two objects easily, and is a necessary basis for continuous direct-manipulation querying and exploration as required in our work.

Our work provides support for two kinds of users, those who know what they are looking for and those who are just browsing the database. Our SVIQUEL is based on our earlier work of building visual interfaces, in particular, the Temporal Visual Query Language [HR95] [HR96a] [HR97], designed to handle queries over one dimensional temporal data. We extend this work to two-dimensional spatial data and deal with the dependencies introduced between the two dimensions. We also look at an integration of topology and direction into one relation and derive a neighborhood model for the same. Lastly, we provide mechanisms for graphical disambiguation of the specified query.

10 Conclusions

In this paper, we have presented a framework for applying the direct manipulation paradigm to spatial data. In our work, we have defined a Spatial Visual Query and Exploration Language (SVIQUEL) for specifying relative spatial position queries. SVIQUEL enables users to specify any primitive spatial relation, with both direction and topology combined, as well as conceptual neighborhood combinations of them. Our SVIQUEL interface provides users with a visual paradigm for browsing the spatial data through direct manipulation and in a spatially continuous manner. SVIQUEL also introduces a qualitative query disambiguation diagram (SQUAD) which provides a rich qualitative description of the quantitative query specified in SVIQUEL. The mapping functions that we have developed ensure the consistency of the spatial queries by keeping the two interfaces synchronized. We have defined the concept of neighborhood in the context of spatial relations when both direction and topology are combined.

The primary contributions of this work include:

- a VIS approach to spatial data analysis
- a spatial query language for specifying relative position spatial queries (SVIQUEL)
- automated maintenance of interdependencies between the spatial query filters
- development of neighborhood concept with both topology and direction combined
- a method for visual qualitative representation of the query (SQUAD)
- mapping functions between the SVIQUEL(quantitative) and the SQUAD (qualitative) that ensures the consistency of the spatial query interfaces

Our future work includes the integration of spatial visualizations into our system as well as query processing techniques tuned for direct-manipulation queries.

Acknowledgments: We would like to thank Stacie Hibino for developing TVQL and for providing us with the base code of TVQL on which we based our implementation. Special thanks to the Database group at WPI whose feedback was so valuable to us at every stage of our research.

References

[All] J.F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26:832–843.

[AS94] C. Ahlberg and B. Shneiderman. Visual information seeking: Tight coupling of dynamic query filters with starfield displays. *Proceedings of CHI'94 Proceedings*, pages 313–317, April 1994.

[AWS92] C. Ahlberg, C. Williamson, and B. Shneiderman. Dynamic queries for information exploration: An implementation and evaluation. In *CHI Conference Proceedings*, pages 619–626. NY ACM Press, 1992.

[Bor96] C. L. Borgman. Why are online catalogs hard to use? *Journal of the American Society for Information Science*, 6(37):387–400, 1996.

[BS77] R. Berman and M. StoneBraker. Geo-Quel, a system for the manipulation and display of geometric data. *ACM Computer Graphics*, 11(2):186–191, 1977.

[CF80] N. S. Chang and K. S. Fu. Query by pictorial example. *IEEE Transactions on Software Engineering*, 6(6):519–524, November 1980.

[CW96] E. P. Chan and J. T. Wong. Querying and visualization of geometric data. *Proceedings of ACM GIS Workshop*, November 1996.

[EF95] M. Egenhofer and R. Franzosa. On the equivalence of topological relations. *International Journal of Geographic Information Systems*, 9:133–152, 1995.

[Ege] M. Egenhofer. Spatial query by sketch. Ongoing project at the University of Maine.

[Ege94] M. Egenhofer. Spatial SQL: A query and presentation language. *IEEE Transactions on Knowledge and Data Engineering*, 6:86–95, February 1994.

[EM95] M. Egenhofer and D. Mark. Modeling conceptual neighborhoods of topological line region relations. *International Journal of Geographic Information Systems*, 9:555–565, 1995.

[ET92] M. Egenhofer and K. Al. Taha. Reasoning about gradual changes in topological relationships. In *Theories and Methods of Spatio-Temporal Relationships*, pages 196–220, September 1992.

[Fre] C. Freska. Temporal reasoning about temporal intervals. In *Lecture Notes in Artificial Intelligence*, volume 54, pages 199–227.

[Her94] D. Hernandez. Qualitative representation of spatial knowledge. In *Lecture Notes in Artificial Intelligence*. Springer Verlag, February 1994.

[HR95] S. Hibino and E. A. Rundensteiner. A visual query language for identifying temporal trends in video data. *International Workshop on Multimedia Database Management Systems*, 1995.

[HR96a] S. Hibino and E. A. Rundensteiner. A visual multimedia query language for temporal analysis of video data. In *MultiMedia Database Systems: Design and Implementation Strategies*, pages 123–159. Kluwer Aca, 1996.

[HR96b] S. Hibino and E.A. Rundensteiner. MMVIS: Design and implementation of a multimedia visual information seeking environment. *ACM Multimedia*, pages 75–86, 1996.

[HR97] S. Hibino and E. A. Rundensteiner. User study evalution of direct-manipulation temporal interfaces. *ACM Multimedia*, November 1997.

[JTS96] Z. John, M. Tamer, and D. Szarfon. Spatial reasoning rules in multimedia management systems. Multimedia Technical Report TR96-05, Department of Computer Science, University of Alberta, March 1996.

[KS86] L. Koved and B. Shneiderman. Embedded menus: Selecting items in context. *Communications of the ACM*, pages 312–318, April 1986.

[KSF+96] F. Korn, N. Sidiropolous, C. Faloutos, E. Siegel, and Z. Protopapas. Fast nearest neighbor search in medical image databases. *International Conference on Very Large Data Bases*, pages 215–226, 1996.

[PTSE95] D. Papadias, Y. Theodoridis, T. Sellis, and M. Egenhofer. Topological relations in the world of minimum bounding rectangles: A study with R-trees. *Proceedings of SIGMOD*, 24, June 1995.

[Shn92] B. Shneiderman. *Designing the User Interface: Strategies for Effective Human Computer Interaction : Second Edition*. Addison Wesley Publ. Co, Reading, MA, 1992.

[SYLL95] A. P. Sistla, C. Yu, C. Liu, and K. Liu. Similarity based retrieval of pictures using indices on spatial relationships. *International Conference on Very Large Data Bases*, 21:619–629, 1995.

[TP95] Y. Theodoridis and D. Papadias. Range queries involving spatial relations: A performance analysis. In *Proceedings of the 2nd International Conference on Spatial Information Theory, COSIT*, Semmering, Austria, 1995. Springer-Verlag.

[WCL+94] A. B. Wansek, D. Calcinelli, B. Languou, C. Lecocq, and M. Mainguenaud. CIGALES: A visual query language for geographical information system: The user interface. *International Journal of Visual Languages and Computing*, 5:113–132, 1994.

[WS92] C. Williamson and B. Shneiderman. The Dynamic HomeFinder: Evaluating dynamic queries in a real-estate information exploration system. Technical Report CS-TR2819, Dept. of Computer Science,Univ. of Maryland, January 1992.