WPI-CS-TR-98-7 May 1998

SEE: A Spatial Exploration Environment Based on a Direct-Manipulation Paradigm

by

Sudhir Kaushik
Elke Rundensteiner

Computer Science
Technical Report
Series

WORCESTER POLYTECHNIC INSTITUTE

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

SEE: A Spatial Exploration Environment Based on a Direct-Manipulation Paradigm

Sudhir R. Kaushik and 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 provide effective tools for analyzing and querying spatial data is becoming increasingly important with the explosion of data in applications such as Geographic Information Systems, Image Databases, CAD and Remote Sensing. The SEE (Spatial Exploration System) is the first effort of applying direct-manipulation visual information seeking (VIS) techniques to spatial data analysis by visually querying as well as browsing spatial data and reviewing the visual results for trend analysis. The SEE system incorporates a visual query language (SVIQUEL) which allows users to specify the relative spatial position (both topology and direction) between objects using direct manipulation. The quantitative SVIQUEL sliders (S-sliders) are complemented by the qualitative Active-Picture-for-Querying (APIQ) which allows the user to specify qualitative relative position queries. APIQ provides qualitative visual representations of the quantitative query specified by the S-sliders. This increases the utility of the system for spatial browsing and spatial trend discovery with no particular query in mind. The queries are processed using a k-Bucket index structure specifically tuned for incremental processing of multi-dimensional range queries. We have also designed a map visualization which helps preserve the spatial context and a bar visualization that provides a qualitative abstraction of aggregates and enable the user to visualize the results of the spatial query as well as the non-spatial attributes of the underlying spatial objects. This system has been fully implemented as an applet using JDK1.1.4. Finally, we also evaluated our spatial exploration environment (SEE) technology with respect to its querying power and the ease of querying.

Keywords: Direct-manipulation, spatial visual query language, spatial trend discovery, multi-dimensional range queries.

1 Introduction

1.1 Background

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]. 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 concepts are to support browsing via direct-manipulation interfaces, visual query composition, graphical display of the results, continuous reformulation of goals and tight coupling to preserve display invariants and support progressive refinement [Shn92].

One example of such VIS technology is the Dynamic Homefinder, a Real Estate Information Exploration System [WS92], which allows users to search for a house that meets their criteria by manipulating sliders for different input parameters (such as the number of rooms and price) and to get a visual display of the results in the form of a map showing all the houses that satisfy the query. An SQL interface for this application would require users to learn the syntax of the language, and also be aware of the schema and content of the database before specifying queries. On the other hand, this direct manipulation environment allows users to explore the database even without knowing what exactly they are looking for.

Tioga-2 [ACSW96], Spotfire [Ahl96], VisDB [KK94] are also other examples of such visual exploration systems. These systems employ dynamic queries and interactive information visualization techniques for information retrieval. These systems however do not deal with special-purpose data types such as *spatial* and *temporal* data. The primary consideration for spatial data is that it is characterized by geometry and map display. Previous work that deal with spatial data such as, the Oracle-8 spatial cartridge [Ora] and SpatialWare from Mapinfo [Map], are typically restricted to simple more text-based (SQL) query languages for spatial data. Instead, direct-manipulation graphical methods for both input (query specification) and output (result validation) need to be developed and integrated in order to have the VIS paradigm applied in spatial database systems.

1.2 The Overall MMVIS Project

The overall goal of our research project is apply the concept of visual information seeking to systems with *special-purpose data types* such as *spatial* and *temporal* data, as to the best of our knowledge VIS has up to now only been applied to simple numeric data types [AS94]. Previous work in our research group focussed on designing a direct manipulation environment for one-dimensional temporal data, called MMVIS

(Multimedia Visual Information Seeking System) [HR97] [HR96]. MMVIS allowed a user to visually specify temporal queries over video data, giving an instantaneous visualization of the results of the query. In this paper, we extend this concept to the two-dimensional domain of spatial data.

The goal of MMVIS is to provide a new paradigm for video analysis - one in which users can temporally and continuously explore data in search of temporal relationships and trends. In VIS, users can browse a database of information through direct-manipulation of buttons and sliders. This new paradigm of video analysis is accomplished by extending existing work in VIS. Another system requirement of MMVIS is that it should be general enough to handle a variety of multimedia information. This is accomplished by using annotations to abstract spatio-temporal information from the original media and then analyzing the annotation collection. Formal user interface studies of MMVIS showed that users could more accurately specify and more quickly adjust queries using TVQL than using a forms-based approach [HR97] [HR98a], indicating the promise of our overall approach. This positive feedback led us to extend our previous work to spatial data types, which is the topic of this current paper.

1.3 Our Approach to a Spatial VIS System

The goal of the research presented in this current paper was to design and implement a spatial exploration environment called the SEE system employing all of the above VIS principles. SEE employs a Spatial Visual Query and Exploration Language (SVIQUEL) that 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 (S-sliders) of SVIQUEL, based on the dynamic query filters used in TVQL [HR96], are tailored for spatial analysis - allowing users to pose spatial queries as well as to browse the data in a spatially continuous manner. In order to use S-sliders to exploit the spatial continuity inherent in spatial data, we have developed a neighborhood model [Her94] [BE96] explained in [KR97] that incorporates both direction and topology and thereby defines neighbors between relative spatial positions. Our SVIQUEL interface 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 by simple mouse movements.

The S-sliders are augmented with an active-picture-for-querying (APIQ) which allows the user to *qualitatively* specify the relative position spatial query and to give a visual feedback of the quantitative query being specified by the S-sliders. We have devised mappings [KR97] between the quantitative S-sliders and qualitative APIQ to maintain consistency and synchronization between the two representations, allowing users to work in either mode and giving them the flexibility to switch to the other mode.

In this paper, we also present the visualization mechanism of SEE to visualize an abstraction of spatial object pairs that satisfy the specified SVIQUEL query in order to facilitate trend discovery by visually highlighting the frequency of different classes of spatial relationships. We employ a bar visualization to visualize these aggregate relationships between the pairs of spatial objects. We also provide mechanisms to display the spatial objects that are a part of the output with a map-like visualization to preserve their spatial context. These visualizations are tightly coupled to the SVIQUEL query interface and due to the incremental nature of the query processor, all interfaces get dynamically updated with every user manipulation to reflect the current query output. In addition to visualizing the output of the query, we also visually indicate the values that are assumed by the non-spatial attributes of the selected spatial objects. In a real estate application, as we shall consider in the rest of the paper, we also visualize the non-spatial attributes of a house such as the cost and size to indicate the values that it could take on.

We also discuss the k-Bucket index structure [HR98b] that we use to process the multi-dimensional range queries of SVIQUEL. This index structure allows for incremental processing of queries for every user query viz every S-slider movement and every APIQ manipulation. In other words, for every S-slider or APIQ manipulation, the query results are computed using the previous query results. This is tightly coupled to both the bar and map visualizations which get updated and represent the results of the current query.

The entire system is implemented using JDK1.1.4 [Kau98] as an extension of the MMVIS system [HR96]. The main packages used are java.awt, java.lang, java.io and java.util. There are 22 classes in all and about 9000 lines of code. This system has been implemented as an applet and can be run using any Java-enabled browser. We plan to make it available as a freeware shortly.

Lastly, we also present an informal evaluation of the system with respect to its ease of query specification and querying power in comparison to standard text-based approaches such as SQL-3 [Bun91].

1.4 Contributions

In summary, the main contributions of this work are:

- An integrated VIS framework for spatial data analysis incorporating both direction and topology queries into one uniform exploration environment,
- A visual query interface for two-dimensional spatial data over this integration of direction and topology (SVIQUEL) expanding VIS technology to more complex data types (space in this case) which has not been attempted previously.
- An active picture (APIQ) for query specification at the qualitative level and for providing a visual representation of the quantitative S-sliders query for query disambiguation.
- A visualization (SVIZ) of the results of the spatial query specified by the S-sliders and APIQ:
  - A map visualization to visualize the actual spatial positions of the spatial objects that satisfy the query and preserve the spatial context.
  - A bar visualization to visualize aggregate functions of the output spatial objects such as the frequency and provide a qualitative abstraction of the query results.

- Tight coupling between all three modules viz SVIQUEL, query processor and SVIZ to provide an integrated VIS framework for spatial data.
- A working implementation of the proposed system to serve as a proof of concept of the ideas. This system has been implemented using JDK1.1.4 on a Unix platform.
- An evaluation of the system with respect to its ease of query specification and querying power in comparison to standard text-based approaches such as SQL-3.

1.5 Organization of Remainder of Paper

Section 2 reviews related work. The SVIQUEL slider and APIQ interface is introduced in Section 3. Section 4 discusses the different visualization strategies used in the system. Section 5 describes the design and implementation of the SEE system. Section 6 presents an evaluation of the system. Section 7 concludes the paper with a list of contributions and a discussion of future work.

2 Related Work

Although a lot of work has been done on various spatial database issues, surprisingly little has been done thus far in applying the direct manipulation paradigm to spatial environments.

Visual query environments based on the VIS paradigms [Shn92] feature a dynamic querying mechanism and a graphical display of results are typically restricted to simple numeric domains [ACSW96] [DKR97]. 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 simple numeric input parameters (such as number of rooms and price) and to get a visual display of the results.

Tioga-2 [ACSW96], Spotfire [Ahl96], VisDB [KK94], DEVise [RLea97], Visage [DKR97], XMDV [War94] are all examples of visual exploration systems. These systems employ dynamic queries and interactive information visualization techniques for information retrieval. These systems however do not deal with special purpose data types such as *spatial* and *temporal* data.

Most existing spatial query languages are textual (SQL-based), requiring users to refer to geometric data using textual queries [Ege94] [CW96]. The *Oracle8 Spatial Cartridge* [Ora], SpatialWare [Map] and Spatial SQL [Ege94] use a text-based query language for spatial data. The disadvantage of these SQL type languages 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.

GEO-QUEL [BS77], Query by Pictorial Example [CF80] and Pictorial SQL [ea88] are examples of query languages where an image depicting the relationship is drawn as a query to the system. 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 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.

3 SVIQUEL Interface

3.1 Goals of the SVIQUEL Interface

The following are the goals in designing SVIQUEL:

- a visual interface to specify the *relative position* spatial queries between two sets of objects of both topological or directional relationships in a precise yet simple manner, both qualitatively and quantitatively.

- specifying quantitative queries using S-sliders and qualitative queries using APIQ.
- ability to quickly adjust and incrementally refine spatial queries via direct manipulation.
- power to both specify particular queries as well as to browse over spatial relationships, with no particular query in mind.
- support for composing complex queries such as a combination of spatial relationships with as much ease as possible.
- qualitative graphical disambiguation of the relative spatial position query.

The SVIQUEL interface enables the users to specify the relative spatial position queries between two sets of objects and returns as output all pairs of objects that meet the spatial predicate. Given two spatial objects A and B, the spatial (topological, directional or distance) relationship between the two objects can 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 SVIQUEL 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 SVIQUEL language to specify any primitive spatial relationship as well as combinations of these primitive relationships as discussed below.

[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 their MBRs.

3.2 Selection of Object Sets

SVIQUEL is based on the exploration of spatial relationships between object pairs and hence requires that the first user select the two subsets between which the relationship is to be specified. A separate subset query palette is provided for each subset. Each subset palette contains a list of spatial object types from which to choose from. Multiple selections are allowed and hence one or more object types could be a part of the subset. Selected items of a list are ORed and the results from each list are ANDed together. Figures 1 and 2 show two such subset query palettes with subset A set to all objects of type "House" and subset B set to all objects of type "Lake" where the "Total" refers to the number of objects of each type. Selecting "House" in subset A also allows the user to set the non-spatial attributes of the house such as its cost and size from the list as shown in Figure 3.

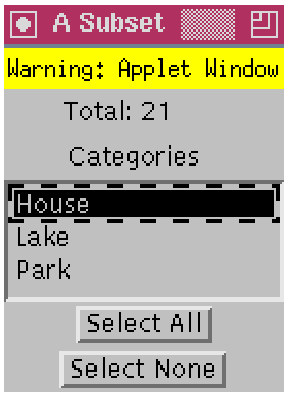

Figure 1: "House" selected in Subset A

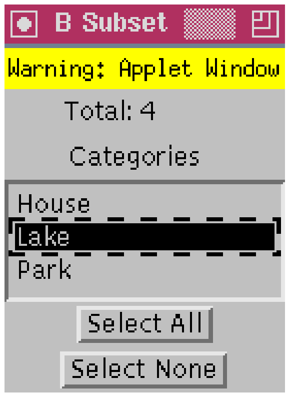

Figure 2: "Lake" selected in Subset B

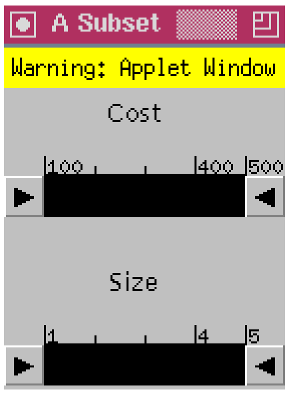

Figure 3: Cost and Size Palette

3.3 Semantics of Relative Spatial Queries in SEE

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)* [EF95] [EM95] and eight possible directional relationships *(north, south, east, west, north east, north west, south east and south west)* between the two objects [JTS96] [TP95] 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 (SPRIM) in our spatial model.

Egenhofer et al [PTSE95] extended Allen's 13 relations in 1D space to the 2D domain using the projection on the X and Y axis. In 2D, the number of pairwise disjoint relations is 169. Of these 169 configurations, 48 of them are of the topology *disjoint*, 40 of them of the topology *meet*, and so on [PTSE95]. We now propose to incorporate direction along with topology. Given this, it is possible to map each of these 169 con-

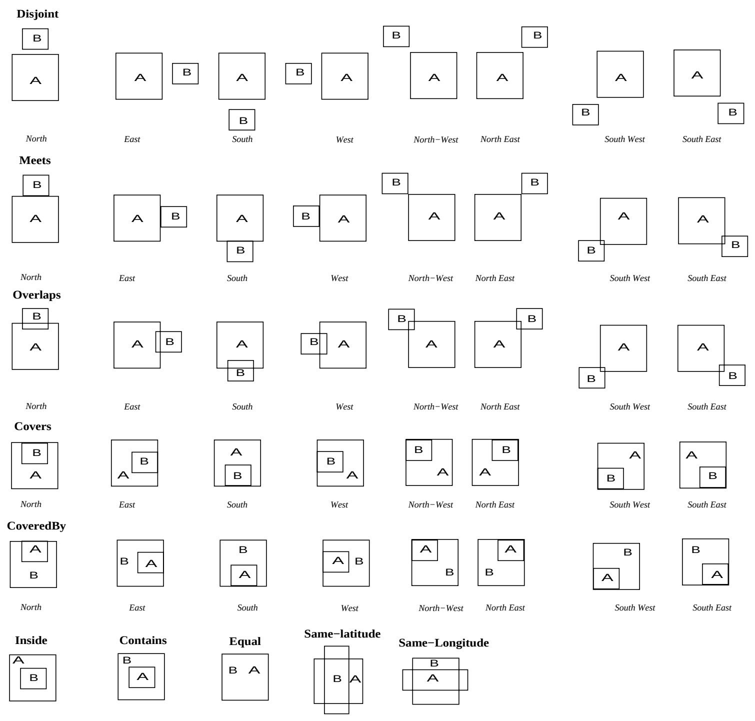

Figure 4: **Spatial Primitives (SPRIMs) with Topology and Direction Combined.**

figurations to one or more combinations of *direction* and *topology*. In other words, some of them would map to a configuration of *disjoint-north*, some of them to *disjoint-west*, and so on. In all in our spatial primitive model, we have 64 possible combinations of direction and topology to which these 169 configurations could be mapped, called SPRIMs (for spatial primitives). Our model achieves this mapping and thus effectively extends the 1D temporal relations to the 2D domain of direction and topology. 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 (SPRIMs) and not 64 is that for certain topologies like *inside*, *contains* or *equals* having a directional component is not meaningful [Her94].

The SPRIMs as given in Figure 4 can be depicted in terms of the differences between the left and right ends in the X dimension and the 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:

[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.

((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 $\theta = \{ <, >, = \}$. In many cases, fewer than eight differences are sufficient to represent the relative spatial position of two objects. [3] [4]

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 The SVIQUEL S-sliders

Each of the eight differences could be mapped to exactly one slider, four sliders in each of the dimension. In our SVIQUEL interface, dTl (LeftA, LeftB), dTr (RightA, RightB), dTlr (LeftA, RightB) and dTrl (RightA, LeftB) represent these differences in the X dimension and dTt (TopA, TopB), dTb (BottomA, BottomB), dTtb (TopA, BottomB) and dTbt (BottomA, TopA) represent these differences in the Y dimension. In

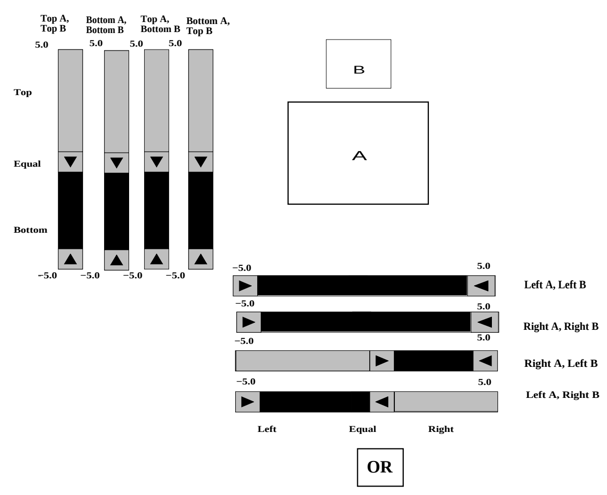

Figure 5: **S-slider Interface for Query "*Find all cases where B is disjoint and to the north of A*".**

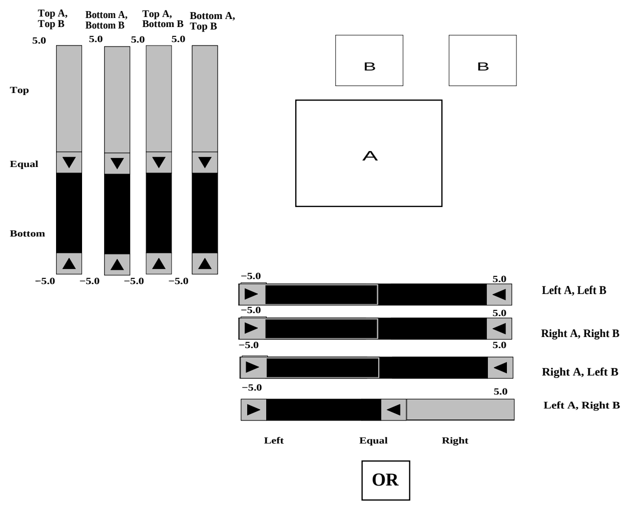

Figure 6: **S-slider Interface for Query "*Find all cases where B is disjoint and north, north east of A*".**

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, as introduced below.

[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

- 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 imply 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. 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* (as modeled by the top leftmost SPRIM in Figure 4).

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 assumes 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 (S-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 sliders so that the user is able to interpret the sliders 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 positions of the S-sliders 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. The sliders *(Left A - Right B)* and *(Right A - Right B)* are set to include all possible 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. In order to prevent the specification of illegal queries, constraints and dependencies are introduced between the sliders so that only legal combinations of values can be specified using our interface [KR97]. [KR97] discusses these constraints and dependencies in detail. There are two advantages of creating such dependencies:

- it avoids the specification of illegal queries by users, and
- it simplifies the process of query specification for the users whenever possible.

Neighborhood Queries in SVIQUEL.

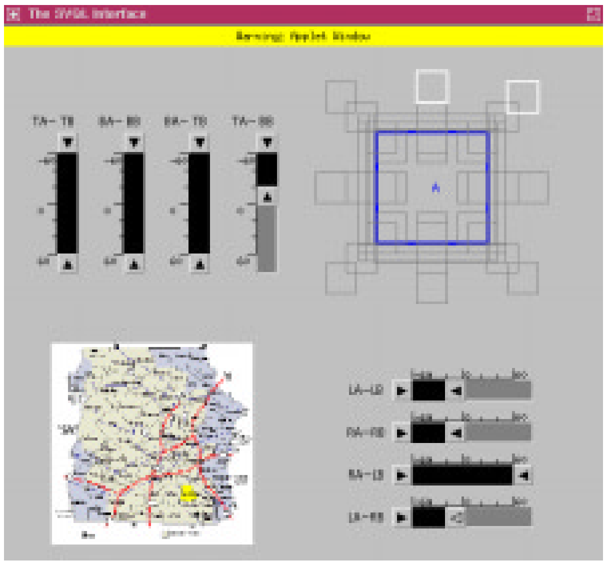

Figure 7: **SVIQUEL Interface for Query, "*Give me cases where B is disjoint and to the north or north east of A*".**

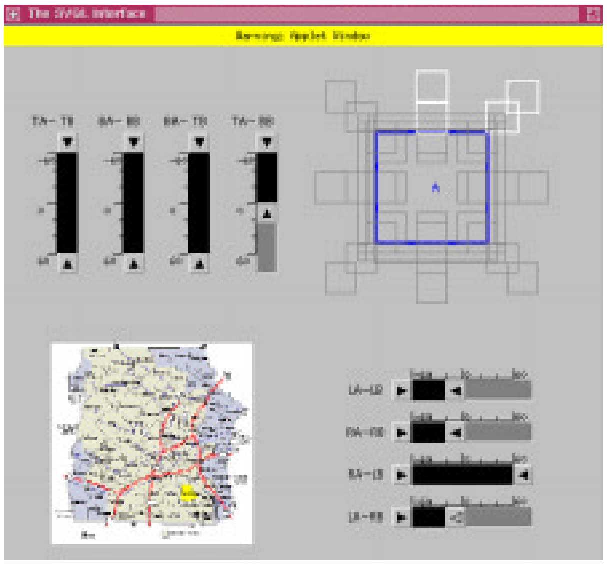

Figure 8: **SVIQUEL Interface for Query "*Give me cases where B is disjoint or meets at the north, north east of A*".**

SVIQUEL enables us to specify a combination of neighboring primitives as part of the query, consistent with the concept of neighborhood when both direction and topology are combined as discussed in [KR97]. The additional constraints imposed on the query are dealt with in Section 6.1. The top-left and bottom-right portions of Figure 7 shows the setting of the S-sliders for the neighborhood query "Give me all cases where B is *disjoint* and to the *north* or *north-east* of A". Now, if the user wants to include *meet-north* as part of the query, then moving the slider *Top A - Bottom B* to include the value of zero, includes the primitives *meet-north* and *disjoint-north east* as they are the neighbors of the primitives *disjoint-north* and *disjoint-north east*. Figure 8 shows the settings of the S-sliders for the defined query, "Give me all cases where B is *disjoint* or *meets* at the *north*, *north-east* of A". Thus the S-sliders can be used to specify relative position spatial queries involving one or more neighboring spatial primitives. Section 3.5 describes the APIQ and how it could be used to qualitatively specify the same set of queries.

3.5 APIQ: An Active Picture for Querying

3.5.1 Qualitative Query Representation using APIQ

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 [HR96], we now propose to add an Active Picture for Querying (APIQ) as part of the SVIQUEL palette to give visual *qualitative* representations of the query to the user. APIQ is designed as a visual aid for confirming the specified query to the users in a visually intuitive manner.

APIQ is the natural complement to the SVIQUEL query model as it corresponds to the union of all neighboring SPRIMs specified by the user in the SVIQUEL query (see Section 3.3 for a description of the SPRIMs). In other words, APIQ shows all the primitive positions object B could occupy with respect to a reference object A - each position graphically depicted by one SPRIM primitive (see top right hand portion of Figure 6). The actual conditions that must hold true for each primitive to be displayed is discussed in further detail in [Kau98].

Based on previous user studies [HR97], we anticipate that this should increase the speed and the certainty with which the users modify and construct complex queries. Another advantage of APIQ 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 APIQ and the visualization of the results (SVIZ) are dynamically updated as users manipulate the sliders, thereby enabling the users to simply move the S-sliders back and forth until an interesting result appears and then see from APIQ what query was specified.

3.5.2 Direct Manipulation in APIQ

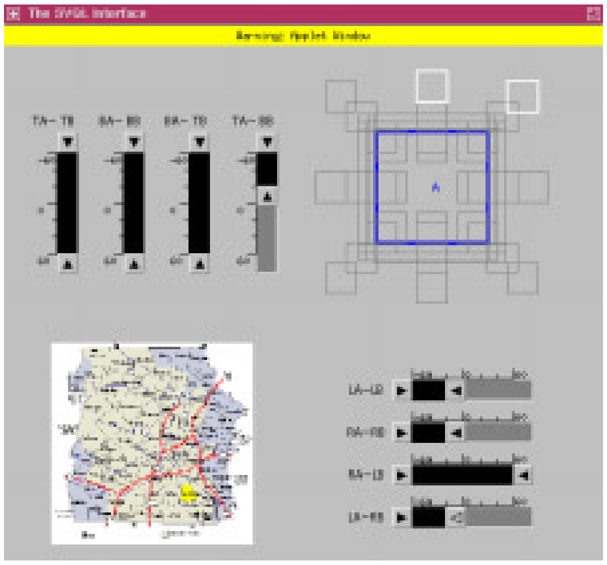

Figure 9: **APIQ for Query *"Find all cases where B is disjoint from A and to the north or north east of A"*.**

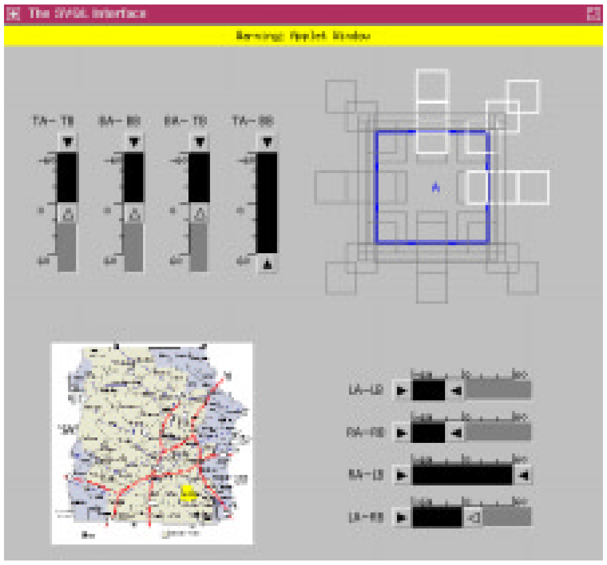

Figure 10: **APIQ for Query *"Find all cases where B is disjoint and to the north, north east or east of A"*.**

In addition to providing a rich qualitative representation of the query, the user could choose to work with the APIQ environment to first specify a qualitative relative position spatial query and then use the S-sliders to specify precise quantitative numeric values. In our system, the user selects and de-selects the primitives which are a part of the query using mouse clicks. For example, given the APIQ depicted in Figure 9, if the user wishes to specify the primitive *disjoint-east* as part of the query, then a mouse click on that primitive results in including that primitive as part of the query. The resulting APIQ and S-sliders, consistent with the integrated neighborhood model described in [KR97], are shown in Figure 10. The tight coupling between the S-sliders and APIQ achieved by our mapping functions between the two gives the new setting of the S-sliders as a result of the direct manipulation of APIQ [Kau98].

4 SVIZ: Spatial Visualization of Results

4.1 Goals of SVIZ

When designing a database query system using a data visualization method, it is important to understand the effects a given visualization method has on the user's perception and understanding of the results from different types of queries [Wis94]. Important design criteria that should be considered [Wis94] include :

- Compact data display,
- Smooth and fast redrawing,
- Easily distinguishable separate data points ,
- Easily distinguishable marked subsets, and
- Subsets connected in the data space also connected in the visualization space.

In this section, we present the spatial visualization of the results, referred to as SVIZ, which satisfies the above criteria defined for a dynamic query system. Most of the existing GIS and spatial database systems, such as MapInfo [Map] and ArcView [ESR], use a map-like visualization for the underlying spatial data. Such a display of results that distributes spatial relations over the 2D space could be extremely densely populated. In addition to providing a map-like visualization, we augment SViz with a more abstract visualization for summarizing the selected subsets and highlighting the presence or absence of spatial relations between them. Alongside this aggregate visualization, we also provide qualitative visual representations of the values of the non-spatial attributes of these subsets. The major goals of SViz are:

- provide the context for the underlying spatial data as much as possible (e.g., use of icons to visually represent descriptive content of the underlying spatial object types),
- abstract and summarize information about the spatial positions of the spatial objects and also about the number of occurences of each spatial object type,
- aggregate and highlight the strength of the SVIQUEL specified spatial relationships between the subset members in addition to the map visualization which highlights the spatial positions of the underlying spatial objects.

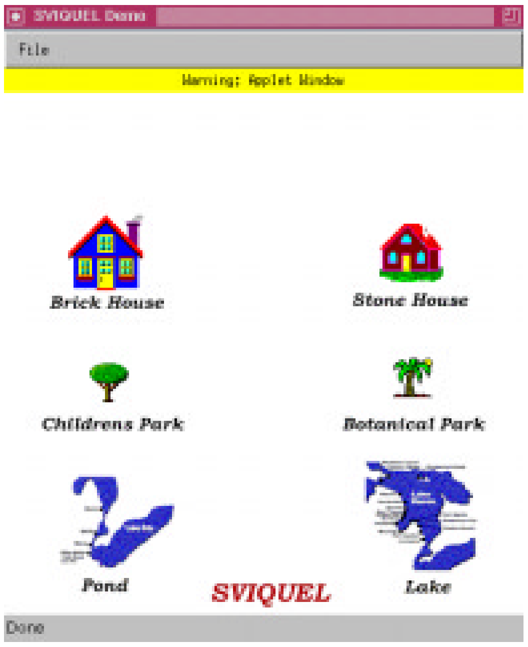

Figure 11: **Main SEE Window for Visualization.**

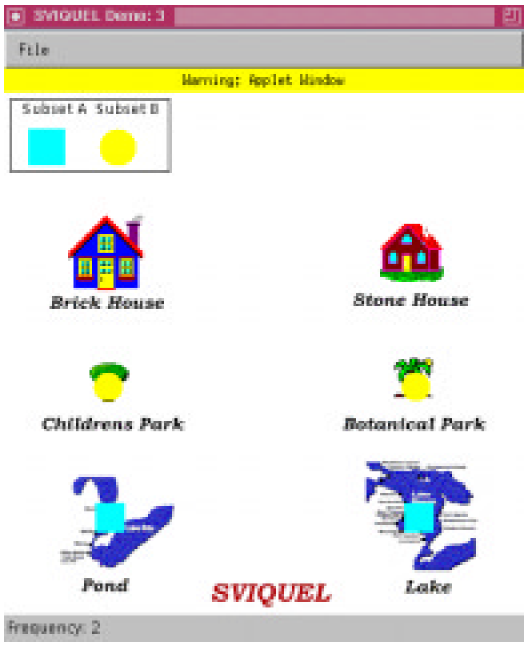

Figure 12: **Visualization of Selected Subsets.**

4.2 The Main SEE Window

Figure 11 represents the main SEE window for visualization. In SViz, the default display includes icons representing the different types of spatial objects in the database, where a separate icon is displayed for each type of spatial object in the database. In our real estate application, the number of spatial object types was small enough to display all icons on the screen at once (brick house, stone house, pond, lake, children's park and botanical park). The actual placement of the icons is decided by the designer of the application. However in cases, where the number of spatial object types is large, our solution may have to be extended to select descriptive objects to display or build a hierarchy of objects for grouping and examining them at different levels of abstraction [HR95] [HR96].

4.3 Visualization of the Selected Object Subsets

The subset selection highlighters are designed to support users in comparing summaries of occurences of the underlying spatial objects within the context of the main SEE window. This is accomplished by having overlays to visually indicate the subsets that are being selected/de-selected. Members of subset A are highlighted by squares and members of subset B are highlighted by circles. The relative sizes of these overlays are indicative of the number of occurences of each spatial object type in the underlying spatial database.

Figure 12 shows an example where the subset A is set to the spatial object type *lake* and the subset

B is set to the spatial object type *park*. Each of the types of parks and lakes can be individually selected and de-selected using the mouse. The objects in subset A are highlighted in *cyan* and those of subset B are highlighted in *yellow*.

4.4 Visualization of Non-Spatial Attributes

In addition to selecting the subsets, SVIQUEL also allows us to constrain the non-spatial attributes of the underlying spatial object types. For example, in a real estate application, SVIQUEL allows us to specify the non-spatial attributes of the spatial object type, house such as its cost and size (See Figure 3). Both these attributes can assume a range of numeric values and therefore dynamic query filters are used to specify these attributes.

The main SEE window also provides a visualization of the values of these non-spatial attributes. Two bar-like indicators at the top right portion of the window give the user a visual indication of the values of these attributes. The indicator is calibrated to increase from left to right. In other words, if the bar is filled on the left portion, it means that the attribute can take on low values, whereas if it is filled at the right portion, it means it can take on high values too. This is just a qualitative visual representation of all the possible values of the non-spatial attributes. The background color of the region in which the bar falls also indicates the subset that the bar is referring to. In other words, if the cost bar has a background color of *cyan*, it means that the bar is indicative of the cost of objects in subset A. If the cost bar has a background color of *yellow*, it means that the bar is indicative of the cost of objects in subset B.

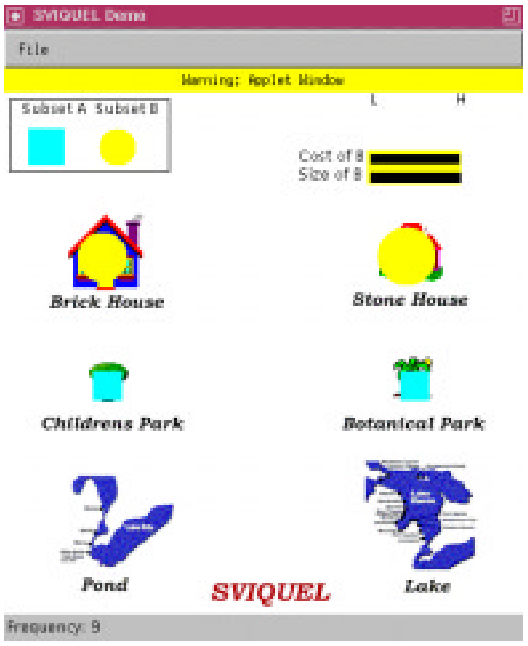

Figure 13: **Visualization of Non-Spatial Attributes.**

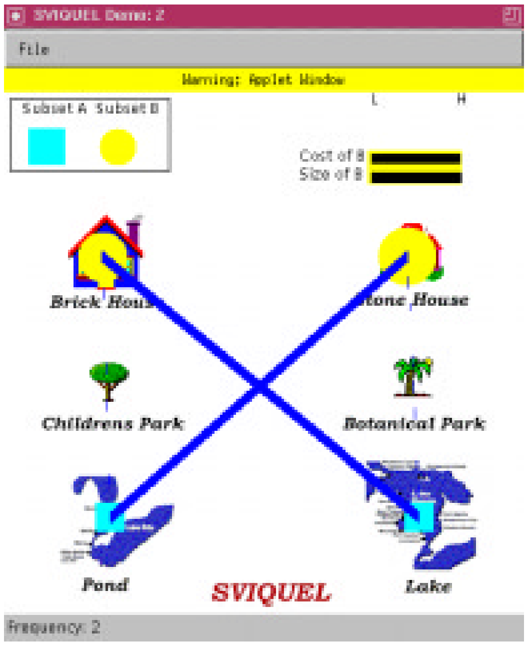

Figure 14: **Bar Visualization of Spatial Relationships.**

Figure 13 shows an example of the main SEE window with the qualitative indicators for the cost and size of a house. In this example, the spatial object *park* is selected as subset A and the spatial object *house* is selected as subset B. The cost and size of the houses have been set to include only low values. As a result, the cost and size indicator bars in the upper right corner of the main window are filled only at the left end. This means that only low values are accepted for these non-spatial attributes for items in that subset, in this case subset A.

4.5 Visualization of Spatial Relationships

Once the users have specified the A and B subsets, they can use SVIQUEL to specify a spatial query. We use a connector between two subset icons to indicate the existence of pairs that meet the spatial predicate with the thickness of the connector showing the strength of the spatial relationship. As they manipulate the S-sliders or the APIQ, users can see the connectors appear and dissappear, grow and shrink between the subset icons, thereby indicating the existence and the strength of the spatial relationship. The relative frequency that the spatial relationship occurs is denoted by the base width of the connector. For example, assume that the user wanted to look for all houses that are *disjoint* and *north* of a lake. The user would first use the subset selection palette to select the two subsets and then use the S-sliders and the APIQ to specify the query *disjoint north*. If there are houses that are *disjoint north* of a lake, then a bar would appear between the two icons indicating the existence of a pair of objects that are disjoint. The thickness of the bar would indicate the number of houses that meet that criteria. If none of the houses meet that criteria, then no bar will be drawn between the two icons.

Figure 14 shows the example of a bar visualization for a *disjoint north* query between houses (subset A) and lakes (subset B). Thick bars between the house icons and the lake icons indicate that there are a number of houses that are disjoint and north of a lake. Thus, it gives a qualitative representation of the number of spatial object pairs that meet a particular predicate.

4.6 Map Visualization

While the visualization of the spatial relationships gives a qualitative indication of the strength of the spatial relationship, it gives no idea about the actual spatial objects which satisfy the query nor about their relative spatial positions. In other words, it does not help capture the spatial positions of each of the spatial objects that satisfy the query. A map visualization (Figure 15) helps to look at all the spatial objects that satisfy the query and also where they are located in space. The main advantage of using a map is that it provides a context for the users to view the underlying spatial objects. It is also possible to combine the bar visualization and the map visualization. But for a large application, the visualization gets too crowded and less useful. So, we abstract the qualitative information about the strengths of the relationships between the

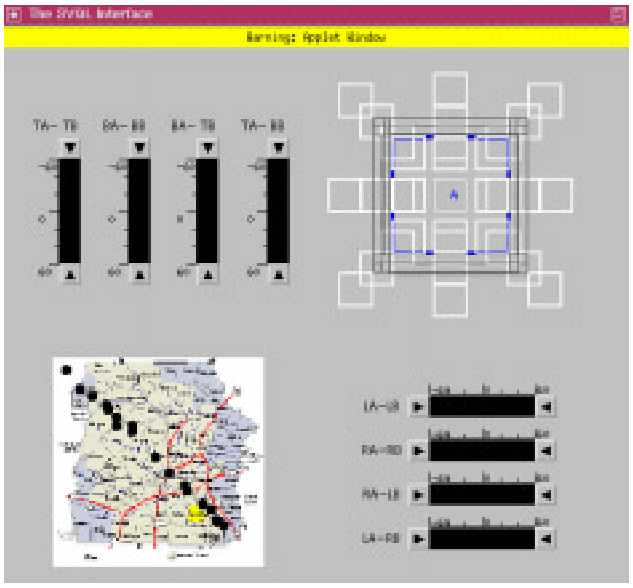

Figure 15: **Map Visualization of Spatial Objects Selected by Query.**

pairs of objects from the actual spatial positions of these objects.

Figure 15 shows an example of the map visualization with the spatial object type *lake* selected in subset A and *park* selected in subset B. The visualization shows the spatial objects that satisfy the spatial predicate which allows for all possible relative spatial positions between the lake and the park. The objects that belong to subset A and subset B are represented by *cyan* circles and *yellow* squares, respectively.

5 Implementation of the SEE System

5.1 System Design

Figure 16 shows the overall SEE system design. This design is an extension of the design of the MMVIS system [HR95] [HR96], a VIS system for video data. The major components of the system include:

- Query Processor Module (Section 5.2)
- Spatial VIS Module (Section 4)
- S-Slider and APIQ Query Interface Module (Section 3.4)
- Subset Selection Query Interface Module (Section 3.2)
- Spatial Database and Database Manager (Section 5.2)

At start-up, the user selects two subsets over which the relative spatial query is to be specified. The subset selection component provides the user with two subset selection palettes to select the two subsets of interest.

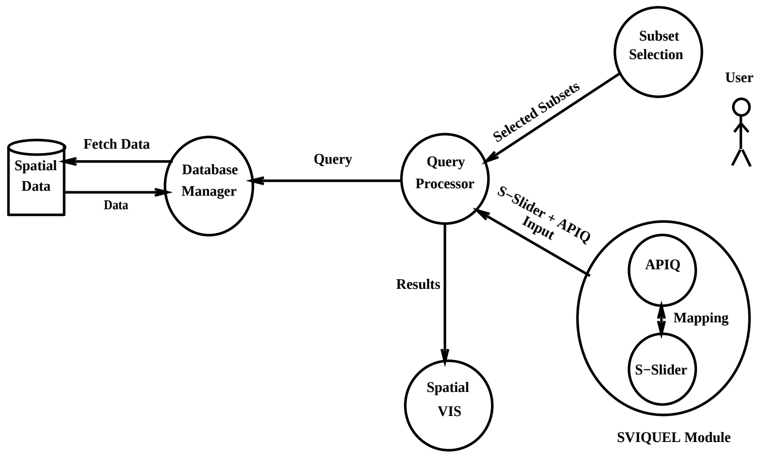

Figure 16: **Overall SEE System Design.**

Then the user refines the query using the S-sliders and the APIQ to set relative spatial relationships between the two subsets of objects. The query processor takes as input, the subsets and the query from the query interface component and computes the result of the query while interacting with the database manager. The results are then sent by the query processor to the spatial visualization component to generate the visualization of the spatial results. The diagram in Figure 17 shows the overall class hierarchy for the SEE system. The classes are divided into different categories based on their functionality as shown in Figure

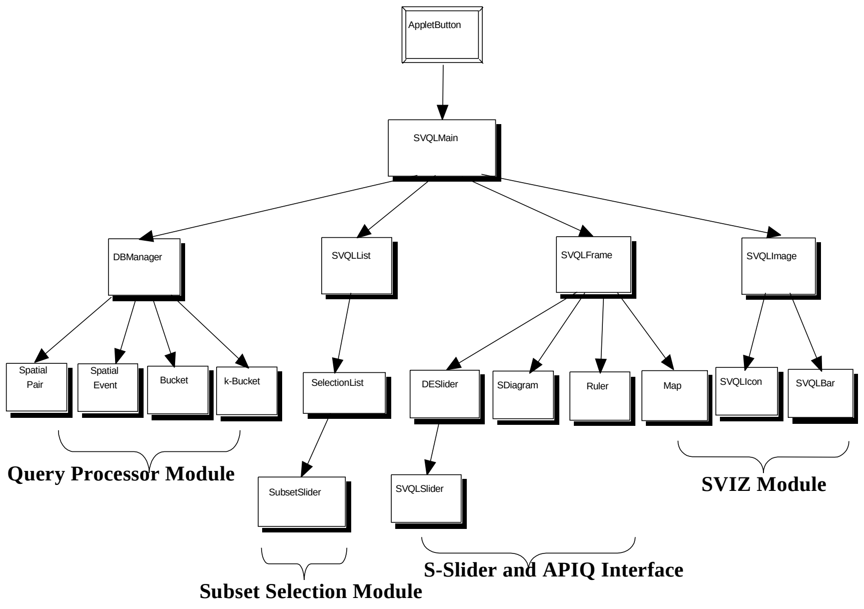

Figure 17: **Overall Class Hierarchy.**

17. The SEE system is implemented as an applet using JDK1.1.4 and can be run using any Java enabled

browser. There are 29 classes in all and the whole system is about 9000 lines of code. The Java packages primarily used are: java.awt, java.io, java.util and java.lang.

5.2 Strategy Used by Query Processor

The problem of processing dynamic queries in SVIQUEL can be characterized as a multidimensional range query problem in which queries are incrementally specified. More specifically, given a set of sliders, we have:

- each slider represents one attribute of the data set,
- each item in the data set can be placed into one and only slot of each DQ filter,
- a query corresponds to a direct manipulation adjusting and selecting valid attribute ranges for each DQ filter,
- a query is processed incrementally as each filter is adjusted, rather than waiting until all valid ranges have been set,
- an item in the data set is in the solution set iff each value of each attribute of the data item is in the selected range of the corresponding DQ filter

We use the k-Bucket index structure [HR98b] to process these multidimensional range queries of SVIQUEL. This indes structure is ideally suited to the incremental processing of our SVIQUEL queries. This is discussed in further detail in [Kau98].

6 Evaluation of SVIQUEL

We evaluate the expressive power of SVIQUEL to specify relative spatial position queries and also compare it to standard text-based query languages, in particular, the spatial operators of SQL-3 [Bun91].

6.1 Queries Expressed by SVIQUEL

Unlike SQL, we did not strive to have a complete language, but the primary goal was to have a direct-manipulation spatial exploration environment capable of specifying relative position spatial queries. The SVIQUEL interface enables us to specify a wide range of spatial queries of both topological and directional type. We introduce below a characterization of the queries supported by the system by mapping them to SQL query semantics.

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 the house is the primary object (represented by object B) and the lake is the reference object (represented by object A).

The queries that can be asked using SVIQUEL fall into three main categories as outlined below:

- **Case 1:** The first class of queries involves configurations having the same topological relationships between B and A but different directional relationships, where the directional relations are neighbors along the same directional dimension. For example, "Give me all houses that are *disjoint* from the lake and either to the *north* or *north-east* of the lake." The corresponding SVIQUEL interface for this

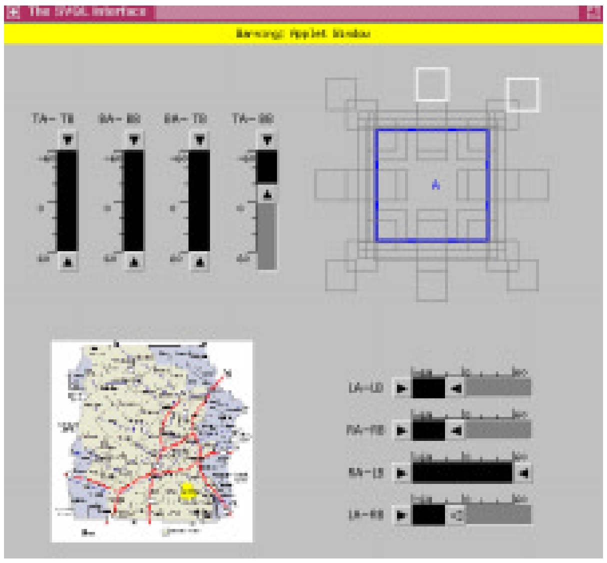

Figure 18: **SVIQUEL Interface for Query *"Give me all houses that are disjoint and to the north or north-east of lakes"***

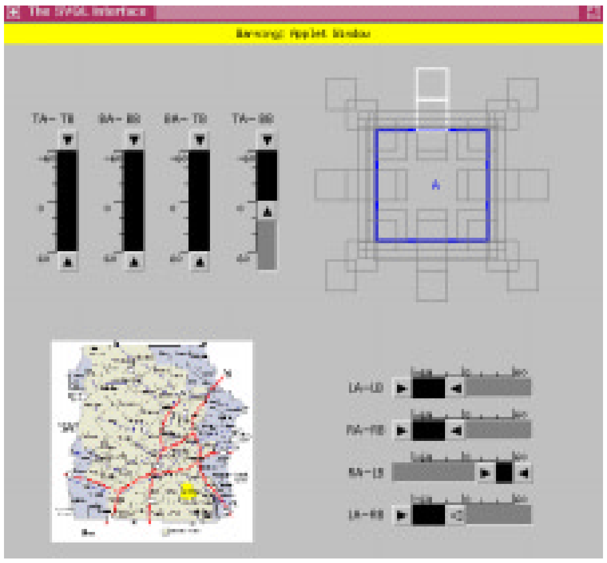

Figure 19: **SVIQUEL Interface for Query *"Give me all houses that are disjoint or meet the lake at the north"***

query is given in Figure 18. The same query when specified in SQL3 [Bun91] is shown below:

SELECT HOUSES.NAME, LAKES.NAME

FROM HOUSES, LAKES

WHERE **DISJOINT_FROM**(HOUSES.BOUNDARY,LAKES.BOUNDARY)

AND **NORTH_OF**(HOUSES.BOUNDARY,LAKES.BOUNDARY)

OR (**NORTH_OF**(HOUSES.BOUNDARY,LAKES.BOUNDARY)

AND **EAST_OF**(HOUSES.BOUNDARY,LAKES.BOUNDARY)).

- **Case 2:** The second class of queries involves configurations having the same directional relationship between B and A but different topological relationships, where the topologies are neighbors. For example, "Give me all houses that are either *disjoint* or *meet* the lake from the *north* of the lake" is a valid query. The corresponding SVIQUEL interface for this query is given in Figure 19. The same query in SQL3 is shown below:

SELECT HOUSES.NAME, LAKES.NAME

FROM HOUSES, LAKES

WHERE **DISJOINT_FROM**(HOUSES.BOUNDARY,LAKES.BOUNDARY)

OR **MEET_FROM**(HOUSES.BOUNDARY,LAKES.BOUNDARY)

AND **NORTH_OF**(HOUSES.BOUNDARY,LAKES.BOUNDARY).

- **Case 3:** The third class is essentially a combination of the above two kinds of queries. As we have seen in case one, one could specify queries involving a disjunction between configurations with the same value along the topology component and neighboring values along the direction component. This could be extended along with case two, namely to include neighboring values along both the direction and the topology component and the query implies a disjunction between the configurations having all possible combinations of the selected direction and topology values. For example, "Give me all houses which are either *disjoint* or *meet* the lake from the *north* or the *north-east*." The corresponding

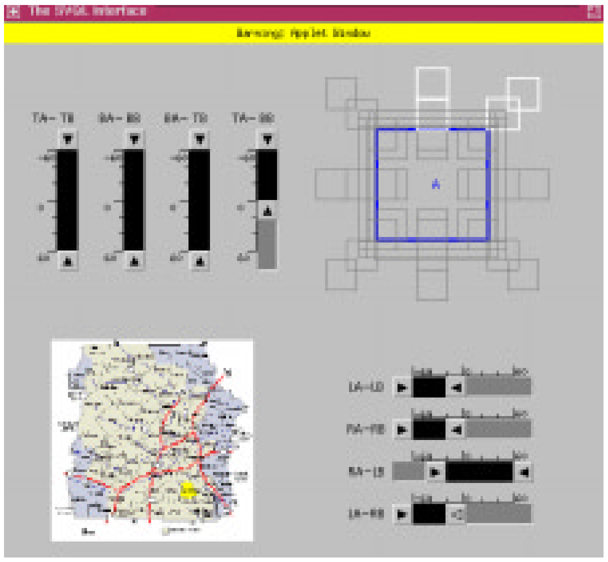

Figure 20: **SVIQUEL Interface for Query "*Give me all houses that are disjoint or meet the lake at the north or north-east*".**

SVIQUEL interface for this query is given in Figure 20. The same query in SQL3 is shown below:

SELECT HOUSES.NAME, LAKES.NAME

FROM HOUSES, LAKES

WHERE (**DISJOINT_FROM**(HOUSES.BOUNDARY,LAKES.BOUNDARY))

OR (**MEET_FROM**(HOUSES.BOUNDARY,LAKES.BOUNDARY))

AND ((**NORTH_OF**(HOUSES.BOUNDARY,LAKES.BOUNDARY))

OR ((**NORTH_OF**(HOUSES.BOUNDARY,LAKES.BOUNDARY))

AND (**EAST_OF**(HOUSES.BOUNDARY,LAKES.BOUNDARY)))

6.2 Comparison of SVIQUEL and other Text-Based Query Languages

SVIQUEL is a visual query language for specifying spatial relationship queries between two sets of objects and offers several advantages over textual spatial languages. While on the one hand SVIQUEL provides features not supported by textual languages, e.g., Spatial SQL by Egenhofer [Ege94], on the other hand, it has more limitations than such a general textual language in terms of its expressive power. We discuss below some of the advantages and disadvantages of SVIQUEL in comparison to text-based spatial languages such as Oracle8 SQL [Ora] and SpatialWare SQL [Map].

6.2.1 Advantages of SVIQUEL

We discuss the advantages of these unique SVIQUEL features over any text-based query language below:

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 and Refinement. Users can incrementally specify SVIQUEL queries by adjusting parameters of the previous query and see 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 easily computed from the results of the previous query rather than computing it from scratch. In contrast, the SQL languages does not provide built-in support for such incremental query specification. While users can specify a series of text-based queries by saving text or by re-typing, this is equivalent to simply dragging one of the filters in our SEE environment.

Power to Query and Browse. The text-based languages typically supports 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, and
- spatially browse the data with no 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 also gives the user the chance to discover relationships he may not be aware of.

Visual Feedback. The spatial visualizations help the user to identify what query is being specified by the user. The SQL languages does not provide such feedback.

6.2.2 Limitations of SVIQUEL

In order to preserve the above VIS advantages, SVIQUEL is restricted in its querying power.

Comparison of only two objects at a time. SVIQUEL, in its current design, is limited in the sense that it allows the user to only specify relationships between two sets of objects at a time similar to how spatial predicates in SQL-3 are binary relations. That is, if we had three objects X, Y and Z, we could only examine pairwise relationships between two sets of objects or pairs such as A = X and B = (Y or Z) by applying the OR button and working with several SVIQUEL palettes.

Querying power is limited. While SVIQUEL is very powerful in expressing queries involving a combination of direction and topology, it is not easy to specify queries involving only topology and only direction. For example, the query "Give me all houses that are *disjoint* from the lake" is tedious to specify in our system, requiring four disjunctions to be specified opening up four SVIQUEL palettes. This is because these configurations are not neighbors under the neighborhood model for a combination of topology and direction.

Inability to mix relational and relative queries. SVIQUEL does not allow the queries to have both comparative evaluation about spatial relations themselves. So, for example, we cannot have queries such as "Give me all houses that have the same relationship as the red houses have with the lake".

7 Conclusions

In summary, the main contributions of this work are:

- An integrated framework for spatial data analysis applying the visual paradigm to all aspects of the spatial exploration environment from query specfication to the result visualization.
- A visual query interface for two-dimensional spatial data over an integrated query model of direction and topology (SVIQUEL) expanding VIS technology to more complex data types (spatial in this case) which has not been attempted previously.
- An active picture (APIQ) for query specification at the qualitative level and for providing a visual representation of the quantitative S-sliders query for query disambiguation.
- A visualization of the results of the spatial query specified by the S-sliders and APIQ. Two types of visualizations are provided, a qualitative abstraction of the output (bar visualization) and a map visualization to preserve the spatial context.
- A complete design of the system along with a query processing strategy that exploits the incremental nature of the query interface.

- Tight integration of all query interfaces and the result visualization via incremental mapping functions to provide an integrated spatial VIS environment.
- A working implementation of the complete system in Java using JDK1.1.4 which can be found at the following URL ("http://metal.wpi.edu/mmvis/mmvis.html").
- A preliminary evaluation of the system with respect to its ease of query specification and querying power in comparison to standard text-based approaches like SQL-3.

In the future, we plan to incorporate SVIQUEL into MMVIS as a spatio-temporal environment as well as investigate the extension of SVIQUEL to more than two dimensions.

References

[ACSW96] A. Aiken, J. Chen, M. Stonebraker, and A. Woodruff. Tioga-2: A direct manipulation database visualization environment. *IEEE Int. Conf. on Data Eng.*, pages 208–217, 1996.

[Ahl96] C. Ahlberg. Spotfire: An information exploration environment. *Proceedings of SIGMOD*, 25:25–29, 1996.

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

[BE96] H. Bruns and M. Egenhofer. Similarity of spatial scenes. *International Symposium on Spatial Data Handling*, pages 31–42, Aug 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.

[Bun91] M.S. Bundock. SQL-SX: Spatially extended SQL - becoming a reality. *EGIS*, Apr 1991.

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

[CW96] E. P. Chan and J. T. Wong. Querying and visualization of geometric data. *Proc. of ACM GIS Workshop*, pages 129–138, Nov 1996.

[DKR97] M. Derthick, J. Kolojejchick, and S.F. Roth. An interactive visualization environment for data exploration. *Proceedings of Knowledge Discovery in Databases*, pages 2–9, Aug 1997.

[ea88] N.Roussopoulos et all. An efficient pictorial database system for PSQL. *IEEE Transactions on Software Engineering*, 14:639–651, May 1988.

[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, "http://www.ncgia.maine.edu/ abl/SQBS/sqbswel.htm".

[Ege94] M. Egenhofer. Spatial SQL: A query and presentation language. *IEEE Trans. on Knowledge and Data Eng.*, 6:86–95, Feb 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.

[ESR] ESRI ("http://www.esri.com"). *ArcView.*

[Her94] D. Hernandez. Qualitative representation of spatial knowledge. In *Lecture Notes in Artificial Intelligence*, volume 804. Springer Verlag, Feb 1994.

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

[HR96] S. Hibino and E.A. Rundensteiner. MMVIS: Design and implementation of a multimedia VIS environment. *ACM Multimedia*, pages 75–86, 1996.

[HR97] S. Hibino and E. A. Rundensteiner. User study evaluation of direct-manipulation temporal interfaces. *ACM Multimedia*, pages 99–109, Nov 1997.

[HR98a] S. Hibino and E. A. Rundensteiner. Comparing MMVIS to a timeline for temporal trend analysis of video data. *Advanced Visual Interfaces*, May 1998.

[HR98b] S. Hibino and E. A. Rundensteiner. Processing incremental multidim. range queries in a direct manipulation paradigms. *IEEE Int. Conf. on Data Eng.*, pages 458–465, Feb 1998.

[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, Mar 1996.

[Kau98] S. Kaushik. Direct manipulation spatial exploration using SVIQUEL. Master's thesis, Worcester Polytechnic Institute, 1998.

[KK94] D. Kheim and H. Kreigel. VisDB: Database exploration using multidimensional visualization. *Computer Graphics and Applications*, 1994.

[KR97] S. Kaushik and E. A. Rundensteiner. SVIQUEL: A spatial visual query and exploration language. Technical Report CS-TR-97-10, Comp. Science Dept, Worcester Polytechnic Institute, 1997.

[Map] MapInfo Corporation. *SpatialWare - http://www.mapinfo.com/spatialware/spatial20.html.*

[Ora] Oracle Corporation (”http://www.oracle.com”). *Oracle 8.0.*

[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.

[RLea97] R. Ramkrishnan, M. Livny, and K. Beyer et all. DEVise: Integrated querying and visual exploration of large datasets. *Proceedings of SIGMOD*, pages 517–521, May 1997.

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

[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.

[War94] M.O. Ward. XmdvTool: Integrating multiple methods for visualizing multivariate data. *Proc. of Visualization*, 1994.

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

[Wis94] E. Wistrand. Visualization methods for dynamic queries databases. Master's thesis, Dept. of Computing Science, Gteborg University and Chalmers University of technology, 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, Jan 1992.