A Visual Tool for Querying Geographic Databases*

Andreas D. Blaser and Max J. Egenhofer

National Center for Geographic Information and Analysis

Department of Spatial Information Science and Engineering

University of Maine

Orono, ME 04469-5711, USA

{abl,max}@spatial.maine.edu

Abstract

To support users in querying geographic databases we have developed a system that lets people sketch what they are looking for. It closes the gap between user and information system, because the translation of a user's question into a processable query statement is delegated to the information system so that a user can focus on the actual query rather than spending time with its formulation. This system paper highlights a set of interaction methods and sketch interpretation algorithms that are necessary for pen-based querying of geographic information systems. They are part of a comprehensive prototype implementation of Spatial-Query-by-Sketch, which provides feature-based and relation-based spatial similarity retrieval.

1. Introduction

Geographic information systems are still difficult to use (Egenhofer and Kuhn 1999). Many systems provide only simple text-based interfaces so that a user can query words, substrings, or combinations using basic logical operators. With other approaches users are guided through a maze of dialog boxes and pull-down menus. While such user interfaces may be appropriate for queries that contain only simple textual constraints, such as "Retrieve all five-story buildings in Bangor," they fall short if constraints about spatial configurations are used. For example, the enumeration of the spatial constraints becomes tedious if a user wants to find configurations in which the library, the gym, and the student union have a similar configuration as on the University of Maine campus. Since *what you can specify is what you get* (Shneiderman 1991), alternative methods to formulate queries have been considered.

Several approaches have used the query-by-example metaphor, which allows users to specify a sample of what they are looking for. Some focus on such image properties as color, shape, or hue (Barber et al. 1994; Corridoni et al. 1996), others consider the spatial location of homogenous image areas (Smith and Chang 1999), while a third group favors object-oriented query models that use templates of abstract geometric elements (Del Bimbo et al. 1992; Calcinelli and Mainguenaud 1994; Di Loreto et al. 1996; Haarslev 1996; Haarslev and Möller 1997), sometimes in combination with icons (Meyer 1994; Lee and Chin 1995; Bonhomme *et al.*1999).

We pursue an approach that is based on freehand sketching. Our method is different from other research initiatives, because the user freely specifies each object's properties, particularly its shape, type, location, orientation, and spatial relation to other objects. This approach simplifies the query formulation, because it shifts the responsibility of a correct query interpretation from the user to the system. Compared with other freehand sketches (Landay and Myers 1995; Chok and Marriott 1996; Citrin and Gross 1996; Gross 1996), we base our approach on an analysis of sketched objects and their spatial relations. This paper focuses on the design issues learnt from the prototype implementation of Spatial-Query-by-Sketch and discusses the interpretation and analysis of freehand sketches as an interface for querying geographic databases.

* This research was partially support by grants from the Air Force Research Laboratory under grant number F30602-95-1-0042, the National Science Foundation under NSF grant IRI-9613646, and the National Imagery and Mapping Agency under grant number NMA202-97-1-1023. Max Egenhofer's research is further supported by the National Science Foundation under NSF grants SBR-9700465, BDI-9723873, EIA-9876707, and IIS-9970123; the National Institute of Environmental Health Sciences, NIH, under grant number 1 R 01 ES09816-01; and Lockheed Martin M&DS.

Spatial-Query-by-Sketch has been implemented in C++ and C using Microsoft's MFC classes for the user interface. It runs on Windows 95/98/NT platforms and under Virtual PC, the Windows emulation for Mac OS. The prototype requires at least a 100MHz processor and 24MB of RAM (and approximately twice the amount under Virtual PC). The compiled code of the prototype and a user's guide can be downloaded from http://www.spatial.maine.edu/~abl/SQBS.

The remainder of this system paper describes implementation aspects of the user interface of Spatial-Query-by-Sketch. After a brief summary of the sketching processing (Section 2), we analyze the opportunities provided from sketch-based spatial querying (Section 3). Section 4 describes how Spatial-Query-by-Sketch translates line strokes into meaningful objects that are ready for query processing. Section 5 introduces intermediate views appropriate for editing sketches. Section 6 shows how Spatial-Query-by-Sketch presents the query results to a user. Conclusions and future work are presented in Section 7.

2. Spatial-Query-by-Sketch

Spatial-Query-by-Sketch (Egenhofer 1996b) is a sketch-based user interface to query spatial information within a GIS environment. Similar to the Electronic Cocktail Napkin (Gross 1996), Spatial-Query-by-Sketch associates sketching with freehand drawing, rather than with the construction of geometric figures or with the composition of icons or symbols. It aims at an implementation of the original meaning of the electronic paper metaphor (Kuhn 1993). Although it is extensible to multi-modal spatial querying with voice (Egenhofer 1996a), we focus on the graphical interactions of drawing a spatial query.

2.1 Query Formulation and Parsing

A user formulates a spatial query by drawing a sketch with an electronic pen on a touch-sensitive LCD monitor. The sketch parser keeps track of inconsistencies among objects or their relations. Sketching deficiencies, such as incomplete or incorrect geometric figures, are resolved during this initial phase of interpretation. The user is asked for additional input only if an automatic resolution of constraints is impossible, keeping user interventions minimal.

2.2 Query Processing and Relaxation

The sketch parser creates a canonical representation of the sketched query, the *digital sketch*. It is a meaningful interpretation of the line strokes the user drew, consisting of spatial objects and their binary spatial relations (Egenhofer 1997). For each object pair, the digital sketch captures their topological, directional, and metric relations. During query processing, the digital sketch is analyzed and its relevant parts are transformed into a spatial query that can be processed against a database. The user may review the database query and edit it if desired.

The query processor generates a new query with increasingly more relaxed constraints if no spatial scenes can be found that fulfill all query constraints. Relaxing a sketch may involve accepting relations that deviate from those sketched, or eliminating sketched objects. Relaxation iteration stops when the query found matches in the database or when another relaxation would not make sense any more.

2.3 Result Ranking and Presentation

To the user, the result of a sketched query is a set of spatial scenes that are similar to the sketch. Since some scenes will be better matches than others, it is necessary to sort the query results from highest to lowest similarity with respect to the sketched query. This similarity is calculated based on similarity models for each of the three types of spatial relations, the geometry of objects, and a factor that reflects any difference between the number of objects in the sketched query and in the query result. The system generates for each scene a similarity value with respect to the user's sketch. Spatial-Query-by-Sketch presents the retrieved results as footprints of the best matches next to the sketched query. The user can browse and navigate through the results in decreasing order. New queries can be created on top of previously retrieved results and the original query can be incrementally adjusted if necessary.

3. Pen-Based Interaction in Spatial-Query-by-Sketch

The design of the user interface of Spatial-Query-by-Sketch is based on a sketchpad metaphor, mimicking the principal functionalities of a piece of paper and a pen. It enhances the analog sketching behavior with non-destructive editing (e.g., reposition a sketched object without the need to delete and redraw it), multiple views of the sketch (i.e., the graphical view of the actual sketch vs. an interpreted view of meaningful objects vs. a diagrammatic representation suitable for database query processing), sketch analysis tools, and polymorphous characteristics of the input device (e.g., drawing and editing with the same device).

A critical aspect for a smooth interaction is that the sketch pen can be used for two purposes: (1) the drawing of the query scene and (2) the interaction with the elements of the sketch to accommodate such operations as selecting drawn objects, repositioning or erasing objects, and changing the user's view over the drawing. Today's pen technology supports the selection of multiple interaction modes. The pen used in the implementation of Spatial-Query-by-Sketch has a rocker switch with two positions: *forward* changes the mode of the currently selected tool, while the *backward* position initiates a context dependent menu. The pen tip introduces another two states (*pressed* and *not pressed*), which yields a total of 2*3 pen modes and enables the selection of graphical gestures during sketching and editing (Table 1).

Rocker switch	Pen tip	Action
Neutral	pressed	Draw, zoom, or rectangular selection
Neutral	not pressed	Move cursor
Forward	pressed	Draw straight line, container selection, move selection, pan
Forward	not pressed	None.
Backward	pressed	Activate context-dependent menu
Backward	not pressed	None.

Table 1: Pen modes.

Pen-based interaction provides a more direct zooming method than the common tool-based zooming in today's user interfaces. With the rocker switch in the neutral position and the pen tip pressed, a pen movement towards the center of the drawing area makes Spatial-Query-by-Sketch zoom gradually out (Figure 1a), while the reverse gesture zooms in (Figure 1b). Without changing the tool, users can also pan the entire sketch into any direction they want. For this purpose, they press the rocker switch forward while the pen is in zoom mode, and drag the sketch into the desired direction.

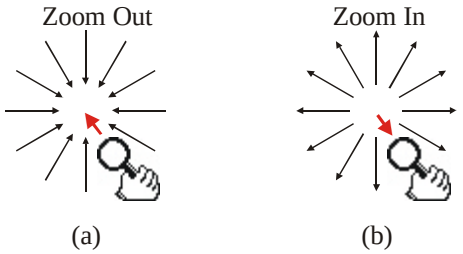


Figure 1: Pen-based zooming gestures: (a) zooming out and (b) zooming in.

Sketched elements can be selected the same way as objects are drawn; that is, the user has to draw a container gesture over the portion of the sketch (involving single or multiple line strokes or objects) that he or she wants to select (Figure 2). This method is less constrained than selecting sketched elements with a bounding rectangle, because it eliminates the requirement of convex polygons, allowing allows users to specify arbitrarily-shaped areas. Conventional methods, such as selecting drawn elements by pointing, are possible as well. This allows a user to pick the appropriate method according to the actual task and his or her preference.

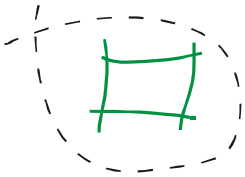


Figure 2: Container gesture to select an object.

A cross-out gesture eliminates objects from a sketch. It consists of two intersecting pen strokes that are approximately perpendicular and of similar length. The targeted object is identified based on spatial and temporal proximity: The key is the location of the gesture (i.e., the intersection of the two pen strokes, Figure 3); however, if more than one sketched object occupies that space, the most recently drawn object is identified and deleted.

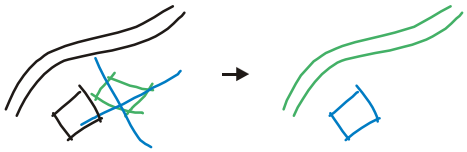


Figure 3: Cross-out gesture to erase a drawn object from a sketch.

4. Transforming Line Strokes into Objects

The sketch interpretation has two subtasks: (1) parsing the sketch and identifying what strokes to aggregate into objects, and (2) the actual aggregation and simplification of entire objects.

4.1 Sketch Parsing

The number of points in a freehand line stroke depends on the user's drawing speed, the type of the input device, and the computer's performance. While in drawing mode, Spatial-Query-by-Sketch records the pen's location, for a typically drawn line stroke, approximately every 1/10 of a second. This process yields a high-resolution line that captures intended direction changes as well as small, unintended changes in the line's direction. For query processing, however, such sketched lines need to be simplified. Spatial-Query-by-Sketch uses the Douglas-Peucker Algorithm (Douglas and Peucker 1973) to filter significant breakpoints. This method eliminates unnecessary breakpoints based on how far these points are from a generalized line shape.

After a stroke has been preprocessed, it must be aggregated to an object. For each new stroke there are three cases, distinguishing to what object the new stroke may belong:

- the current object,
- an object other than the current one, or
- no other previously drawn object (i.e., the line stroke is the first of a new object).

The criteria for these distinctions are based on the location of the new stroke in relation to other objects and on the time difference (δt) between the previously drawn stroke and the current one. Each measure relies on thresholds that can be adjusted according to personal preferences. Figure 4 shows a sample distribution of the connectivity function for the time δt between two strokes. If δt is less than 1.2 seconds then the stroke will be associated to the previously drawn object. After that the connection value decreases linearly as a function of time.

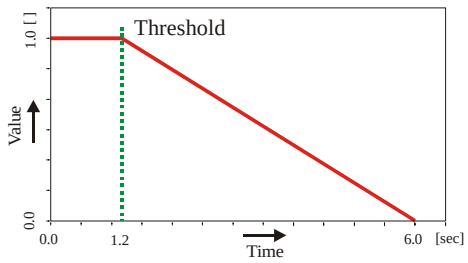


Figure 4: Relation of the time interval δt between two strokes and the connectivity value.

The model for the spatial sequencing of strokes is based on two factors:

- The closeness between the boundary points of the new stroke and the boundary points of a previously drawn object. If the distance between these points is smaller than or equal to a set threshold, the gap between the points is closed and the latest line stroke is aggregated to the object.
- One of the line stroke's boundary points is close to the edge of another object. Again, a distance threshold is employed to determine if the stroke belongs to the object and whether to close an eventual gap, or whether to eliminate an overshoot.

Such automated sketch editing makes use of a buffer zone around a line stroke and its boundaries (Figure 5). Any line stroke drawn later whose boundary falls within the buffer zone gets connected to the first stroke (lines C and D in Figure 5), while lines stay disconnected as long as they intersect the buffer zone without starting or ending inside the buffer zone (lines A and B in Figure 5).

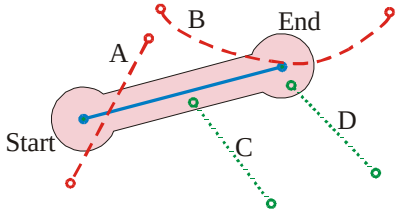


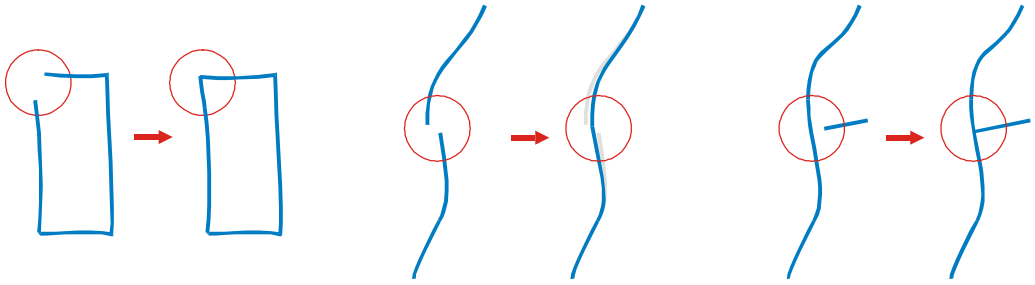
Figure 5: Spatial sequencing of line strokes.

Spatial-Query-by-Sketch gives users feedback about the sequencing of strokes by dynamically coloring the drawn objects. The currently selected object is always blue and the previously drawn object green, while all other objects are displayed in black. This method highlights how strokes are aggregated to objects and helps a user to quickly identify strokes that have been associated with the wrong object.

4.2 Object Processing

Object processing transforms line strokes into the geometric figures suitable for query processing. This task involves clean-up operations of the line strokes and the distinction of regions, lines, and symbols.

Sketching often results in incomplete or inaccurately drawn geometric configurations. Spatial-Query-by-Sketch detects such sketch deficiencies and corrects them automatically. The three most common corrections are closing polygons, continuing interrupted lines, and connecting overshoots (i.e., short intersections) and undershoots (i.e., small gaps) (Figure 6).

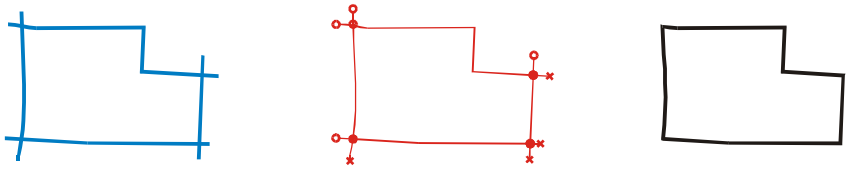


(a) (b) (c)

Figure 6: Three cases of object completion: (a) completing a closed polygon, (b) continuing an interrupted line stroke, and (c) connecting an undershoot to a base line.

The second clean-up operation, segmentation, breaks objects into sets of non-intersecting line segments. During this process, single strokes can be combined or divided (Figure 7). The segmentation of an object is a central task, because subsequent processes rely on the set of segments rather than on original strokes.

The set of segments of an object is assumed to delineate the outline of a 2-dimensional object (region) if the segments form a closed loop. A region may have more than one such loop, for instance if the object has holes. Spatial-Query-by-Sketch attempts to find any closed areas within an object. If one or more areas are detected then they are sorted according to their size. If no loops are found, Spatial-Query-by-Sketch rejects the hypothesis that the object is a region.



(a) (b) (c)

Figure 7: Segmentation: (a) original object, (b) segmented object, and (c) outline used for processing the sketched query.

Each non-region object is examined as to whether any line segments are approximately parallel. If so, Spatial-Query-by-Sketch substitutes any pair of parallel segments with a centerline (Figure 8).

(a) (b)

Figure 8: Centerline extraction: (a) sketched object and (b) its interpretation with centerlines.

Sketched symbols are repeated patterns, which play an important role in semantics of graphics (Bertin 1983). Typically, sketched symbols consist of a few line strokes. Although there is no standard for the use of sketched symbols, an analysis of people's sketches found a small set that people frequently use (Blaser 1998). Currently Spatial-Query-by-Sketch detects two symbols: dashed lines and hatched areas. Dashes are completed to form a continuous line, while the hatches identify the interior of a region, even if the region's boundary has wide gaps.

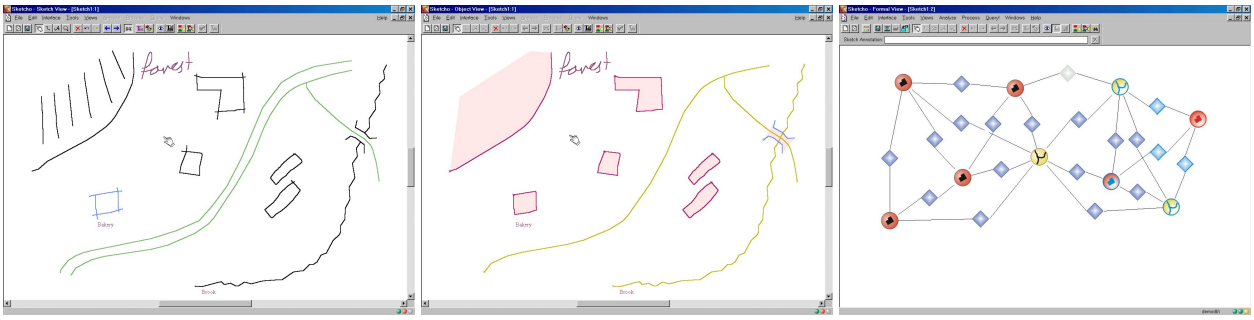
5. Levels of Abstraction

Spatial-Query-by-Sketch allows a user to display a sketch at three different levels of abstraction: (1) the original sketch with the user's line strokes; (2) the interpreted object view, which displays how Spatial-Query-by-Sketch translated the line strokes into objects; and (3) a diagrammatic view, which captures the spatial relations among identified objects that are considered for query processing.

The *sketch view* accommodates the sketching environment of the user interface. It is the place where a sketch is initially created and edited (Figure 9a).

The *object view* displays how the system has aggregated strokes into objects. This view shows the simplified and interpreted outline of each object. It distinguishes different object types (lines, regions, and symbols) through the use of different colors (Figure 9b). The object view allows a user to evaluate the interpreted sketch and modify it if desired.

The *diagram view* is the most abstract view of a sketch (Figure 9c), as it displays objects and binary spatial relations symbolically. Spatial relations are generated and computed according to the configuration of the digital sketch, while the set of objects is given by the sketched query. This view allows a user to examine and edit certain aspects of a sketch better than in the sketch view or the object view. For instance, objects or relations can be selected or un-selected for query processing and binary spatial relations can be created, deleted, and modified (particularly if they were not drawn precisely in the sketch query).



(a) (b) (c)

Figure 9: Different views of a sketch: (a) sketch, (b) object view, and (c) diagram view.

6. Using Spatial-Query-by-Sketch

Querying a spatial scene is a simple task. The users draw a sketch of the spatial scene that they have in their mind. Upon completion of the drawing and any optional editing in the object view or the diagram view, Spatial-Query-by-Sketch compares all scenes in a selected database with the sketched query and computes for each sketch-scene pair a scene similarity value. These values are based on various characteristics of objects and their spatial relations (Egenhofer 1997). The result of a query is presented in the result browser (Figure 10).

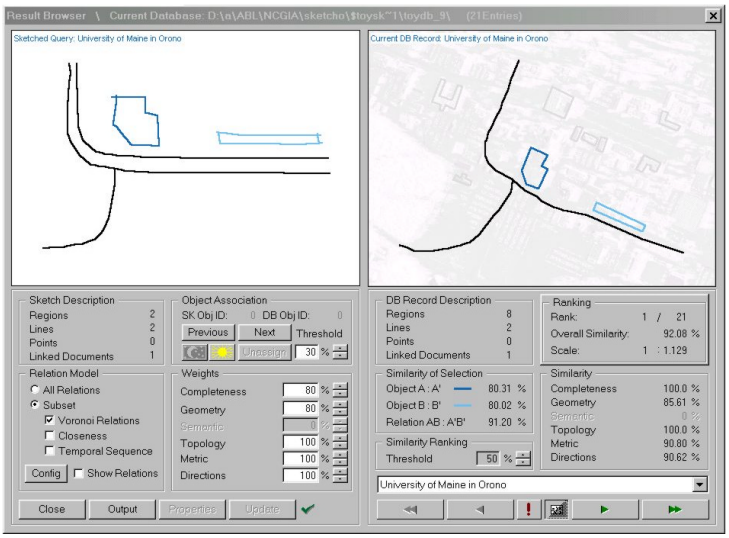


Figure 10: Browser dialog of Spatial-Query-by-Sketch.

The top left shows the sketched query, while the top right displays the retrieved results, one by one. Results are ranked—most similar scenes first, least similar scene (up to a user-defined threshold) last. The user can browse through the results (bottom-right buttons) or change parameters for the computation of the scene similarity (bottom center).

7. Conclusions and Future Work

The Spatial-Query-by-Sketch prototype provides an exciting test bed for GIS user interface design and for spatial query processing. Parsing and preprocessing a sketch is sophisticated and fast, so that a user can focus entirely on the query formulation. The quality of the current query processing mechanisms has been tested informally, indicating a good match of the similarity retrieval with people's own judgments.

A major focus of future extensions is the incorporation of non-geometric aspects into sketched spatial queries. While Spatial-Query-by-Sketch can already parse text annotations and distinguish them from sketched geometry, we want to include verbal annotations and voice-based query formulations as well. Such a Sketch-and-Talk system (Egenhofer 1996a) would enable true multi-modal spatial querying.

8. References

R. Barber, M. Flickner, J. Hafner, D. Lee, W. Niblack, D. Petkovic, J. Ashley, T. McConnell, J. Ho, J. Jang, D. Berkowitz, P. Yanker, M. Vo, D. Haas, D. Lassig, S. Tate, A. Chang, P. van Houten, J. Chang, T. Peterson, D. Lutrell, M. Snedden, P. Faust, C. Matteucci, M. Rayner, R. Peters, W. Beck, and J. Witsett (1994) Ultimedia Manager: Query by Image Content and its Applications. *COMPCON 94*, San Francisco, CA, pp. 424-429.

B. Benderson, J. Jollan, K. Perlin, J. Meyer, D. Bacon, and G. Furnas (1996) Pad++: A Zoomable Graphical Sketchpad for Exploring Alternate Interface Physics. *Journal of Visual Languages and Computing* 7(1): 3-31.

J. Bertin (1983) *Semiology of Graphics*, The University of Wisconsin Press, Madison, WI.

A. Blaser (1998) *Geo-Spatial Sketches*. National Center of Geographic Information and Analysis, University of Maine, Orono, Technical Report TR98-1.

A. Blaser, M. Sester, and M. Egenhofer (2000) Visualization in an Early Stage of the Problem Solving Process in GIS. *Computer & Geosciences* (in press).

C. Bonhomme, C. Trepied, M.-A. Aufaure, and R. Laurini (1999) A Visual Language for Querying Spatio-Temporal Databases. in: C. Bauzer-Medeiros (ed.), *ACM GIS '99*, pp. 34-39, Kansas City, MO.

D. Calcinelli and M. Mainguenaud (1994) Cigales, a Visual Query Language for a Geographical Information System: the User Interface. *Journal of Visual Languages and Computing* 5(2): 113-132.

S. Chok and K. Marriott (1996) Automatic Construction of User Interfaces for Pen-Based Computers. in: *AVI '96*, Gubbio, Italy, pp. 254-257.

W. Citrin and M. Gross (1996) Distributed Architectures for Pen-Based Input and Diagram Recognition. in: *AVI '96* Gubbio, Italy, Gubbio, Italy, pp. 132-140.

J. Corridoni, A. Del Bimbo, S. De Magistris, and E. Vicario (1996a) A Visual Language for Color-Based Painting Retrieval. in: M. Burnett and W. Citrin (Eds.), *VL '96: IEEE Symposium on Visual Languages*. Boulder, CO, pp. 68-75.

A. Del Bimbo, M. Campanai, and P. Nesi (1992) 3-D Visual Query Language for Image Databases. *Journal of Visual Languages and Computing* 3: 257-271.

F. Di Loreto, F. Ferri, F. Massari, and M. Rafanelli (1996) A Pictorial Query Language for Geographic Databases (PQL). in: *AVI '96*, Gubbio, Italy, pp. 233-244.

D. Douglas and T. Peucker (1973) Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or its Caricature. *Canadian Cartographer* 10(2): 112-122.

M. Egenhofer (1996a) Multi-Modal Spatial Querying. in: M.-J. Kraak and M. Molenaar (Eds.), *Seventh International Symposium on Spatial Data Handling*, Delft, The Netherlands, pp. 785-799.

M. Egenhofer (1996b) Spatial-Query-by-Sketch. in: M. Burnett and W. Citrin (Eds.), *VL '96: IEEE Symposium on Visual Languages*. Boulder, CO, pp. 60-67.

M. Egenhofer (1997) Query Processing in Spatial-Query-by-Sketch. *Journal of Visual Languages and Computing* 8(4): 403-424.

M. Egenhofer and W. Kuhn (1999) Interacting with GIS. in: P. Longley, M. Goodchild, D. Maguire, and D. Rhind (Eds.), *Geographical Information Systems: Principles, Techniques, Applications, and Management*. pp. 401-412, Wiley, New York.

M. Gross (1996) The Electronic Cocktail Napkin—Computer Support for Working with Diagrams. *Design Studies* 17(1): 53-69.

V. Haarslev (1996) Using Description Logic for Reasoning about Diagrammatical Notations. in: L. Padgham, E. Franconi, M. Gehrke, D. McGuinness, and P. Patel-Schneider (Eds.), *International Workshop on Description Logics*, Cambridge, MA, 1996, pp. 124-128.

V. Haarslev and R. Möller (1997) SBox: A Qualitative Spatial Reasoner. in: L. Ironi (Ed.), *11th IEEE Symposium on Qualitative Reasoning*, Cortona, Italy, pp. 105-113.

W. Kuhn (1993) Metaphors Create Theories for Users. in: A. Frank and I. Campari (Eds.), *Spatial Information Theory, COSIT '93*, Marciana Marina, Italy. Lecture Notes in Computer Science, Vol. 716, pp. 366-376, Springer-Verlag, New York.

J. Landay and B. Myers (1995) Interactive Sketching for the Early Stages of User Interface Design. in: *CHI '95: Human Factors in Computing Systems*, Denver, CO, pp. 43-50.

Y.C. Lee and F. Lin (1995) An Iconic Query Language for Topological Relationships in GIS, *International Journal of Geographical Information Systems* 9(1): 25-46.

B. Meyer (1993) Beyond Icons—Towards New Metaphors for Visual Query Languages for Spatial Information Systems. in: *International Workshop on Interfaces to Database Systems*, Glasgow, Scottland, pp. 113-135.

B. Shneiderman (1991) Visual User Interfaces for Information Exploration. in: *54th Annual Meeting of the American Society for Information Science*, Washington, DC, pp. 379-384.

J. Smith and S.-F. Chang (1999) Integrated Spatial and Feature Image Query. *Multimedia Systems* 7: 129-140.

C. Williamson and B. Shneiderman (1992) The Dynamic HomeFinder: Evaluating Dynamic Queries in a Real-Estate Information Exploration System. in: *ACM SIGIR '92*, pp. 338-346.