EFFICIENT SPATIAL QUERIES WITH SKETCHES

Matthias Kopczynski

Institute of Cartography and Geoinformatics, University of Hannover, Germany - Matthias.Kopczynski@ikg.uni-hannover.de

KEY WORDS: Databases, GIS, Identification, Interpretation, Matching, Pattern.

ABSTRACT:

Hand drawn sketches can be used as an aid in spatial queries for identifying spatial relationships. This paper is developing a framework describing the computational methods for solving these queries when a spatial reference data set is given. It proposes a suitable representation of the spatial data and how constraints in the data set can be modeled for the purpose of sketch interpretation. Topologic relationships between spatial objects are the most important source of information which is applied in this context. The discrete relaxation algorithm is used to find a set of objects which is consistent with the constraints from the relationship graph. Finally a sketchpad application, developed for experiments with the algorithm, is presented.

1. INTRODUCTION

In the last few year there was a lot of progress in the field of information retrieval, especially on textual information in the well known large internet search engines (Brin and Page, 1998). These approaches are using unstructured text data and are unspecific regarding the context. It is hard to make semantic decisions based on pure textual data. Much easier to handle is structured data. In this case the meaning of a data field is known and based on this knowledge it is possible to make very specific decisions about the importance of a piece of data in terms of a user query. All of these approaches are naturally domain specific. An example for this is the geospatial query language Spatial SQL of Egenhofer (1994).

The query approach is having great impact on the design of user interfaces. In a spatial context it is often necessary to find a location where the user knows something about, e.g. an associated name or a description of the environment. The best representation of the query is depending on the task to solve and which tools are available to create it.

The most simple way to do this in the real world is to point on a map. In the computer world this is a very common way to specify geographic locations. A map server is presenting a map where a position has to be selected with a pointing device. But this is not always the best solution because the user already needs to know a lot about the location. He must select the appropriate map sheet and a suitable scale. Additionally it is cumbersome to specify more than a few positions and it is getting completely impossible if only some constraints on the relations between real world objects are known.

One way to deal with this is to draw a sketch which is emphasizing the use of relations between objects instead of absolute positions.

2. SKETCH BASED QUERIES

One of the first using sketches with computers was Sutherland (1963) with his famous *SKETCHPAD*, which was used to *create* sketches. Later sketches started to be a tool for computer aided design (Herot, 1976) is an easy input method for constructing exact line drawings from inexact input. The sketched lines are used as prototypes for the exact geometry by applying drawing behavior recognition for removing the inexactness in this process. This is still a mature field in the scientific community due to its great industrial relevance. In current research sketches are used as patterns for querying databases, where the sketch in as an example

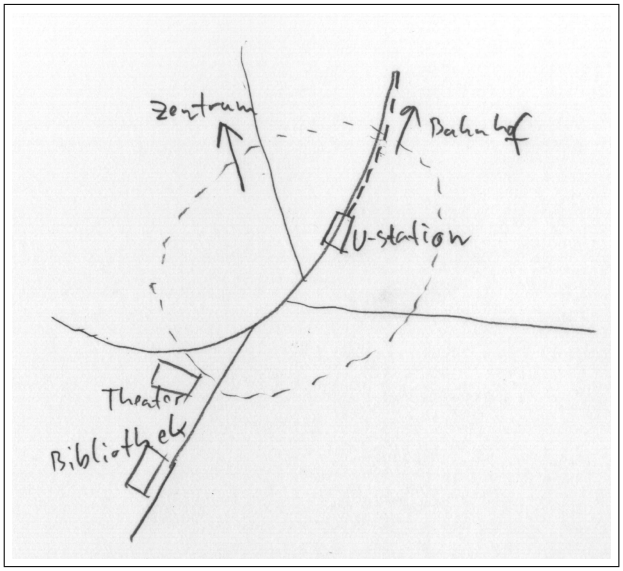


Figure 1. A hand drawn sketch

for objects to retrieve from an existing data set. Objects of interest could be images (Flickner *et al.*, 1995), CAD models (Pu *et al.*, 2005) or prepared spatial sketches (Blaser, 2000). The most recent work in the geospatial context is bringing together sketch queries with mobile applications (Caduff and Egenhofer, 2005).

2.1 Spatial Sketches

This paper will address the use of sketches for retrieving arbitrary spatial locations in contrast to using them for CAD drawing support or model retrieval. In this case its contents is described by the way humans are drawing way-finding sketches (see Tappe and Habel, 1998). Blaser (2000) has given a more specific list of typical objects and relations in a sketch. A spatial sketch is roughly drawn like a map with symbols giving the spatial position of an object, but without the exactness and only containing the relevant objects. The geometric inexactness on the one hand comes from the difficulty of exact drawing and from the lack of knowledge about the accurate positions. Figure 1 is a hand drawn example where this kind of inexactness is directly visible from the drawing style. An important role is playing the free available space on the drawing media which is limited in most cases. If the user is short of space the symbols are moved and distorted until they fit into the remaining space. Therefore a quick sketch and mainly using topology to express the situation in mind.

The drawn objects are typical for the proximity of the location and usually contain parts of the transportation network (roads and railways) since they are in permanent use. Also important are building and outstanding topographic objects when they are visible. But sometimes also invisible objects like administrative boundaries and well known regions are utilized because they allow to give a very short description on locations that might be difficult to describe by enumerating all the topographic objects of the boundary. Fortunately todays topographic datasets and navigation datasets are containing these features and should be useful for the automated sketch locating process. Difficulties arise when working with large scales and when personal knowledge is used. An example is when two people who know about a event in the past and using it (its location) as a symbol in the sketch. Computers only "know" what is in their data sets and thus can not help us in solving such queries.

2.2 Abstraction Levels

The first step towards an automated interpretation process is to choose an acceptable level of abstraction for allowed sketches. There are three levels to choose from:

1. The sketch is composed with a pen on a canvas by drawing single line strokes. No additional information is given. Text and symbols are implicit by the position of their line strokes.
2. Objects are already represented by symbols which are placed on the canvas. Text is given as an annotation to the objects.
3. The sketch is represented by topologic relations of classified objects. The position on a canvas is irrelevant since it does not provide additional information[1].

If one really wants to find a location from a set of raw lines drawn with a pen on a canvas, which is the lowest level of abstraction, the first step would be to detect the meaning of drawn symbols and recognize annotating texts. This approach is important when the sketch is a tool for editing CAD models where lots of research is happening and would also be a great benefit when searching for sketched locations. This interpretation step needs a lot of expert knowledge and the application of automated learning since every user has a different background and uses different symbols and letters. The result of this step is a set of symbols in the plane with textual annotations and is identical with the second level of abstraction. This level already contains semantic information from the choice of symbols and it is clear what is an individual entity in the sketch, unlike in the purely geometric representation where an object could be a combination of primives. Additionally connecting text with objects provides further information about the specific object.

This paper is discussing the interpretation process assuming a sketch represented in the second level because the first step is a research topic on its own and does have little direct implications on the sketch based location retrieval process. While the first two level are visualization oriented and more object centric, the third is containing relationships *between* the objects which must be extracted from the geometric data. Processing of the third level then leads to the location(s) of the sketch and is still a great challenge since it has to be done in real time and with a high degree of reliability.

[1] However, this position can still be useful as an attribute for further processing.

3. EXTRACTING TOPOLOGY

The computer must have knowledge about the real world by using *reference data*. Since the sketch itself is similar to a map the map data can be handled like sketch data, which is just more accurate and complete. This way the same processes can be applied on both the reference data and the sketch until we get a comparable representation (the third level of abstraction). Since the (usually large) reference data does not change it can be preprocessed and no time for redundant geometric calculations is consumed at query time.

For data on the third abstraction level the extraction of topology is needed on both the reference data set and the sketch. The most important tool for this is the dimensionally extended 9-intersection (Clementini *et al.*, 1993) which is based on the work of Egenhofer and Franzosa (1991). Five basic topological relations are defined by this method: *touch*, *in*, *cross*, *overlap* and *disjoint*, which are mutually exclusive and cover the whole space.

Every pair of spatial objects, is described by exactly exactly one of this relations. Following from this it is necessary to do the geometrical calculations for every pair of objects, leading in principle to an algorithm with $O(n^2)$ storage and runtime. But most of the features in a geographic dataset are relatively small compared to the covered region and therefore the *disjoint* relationship is by far the most dominant. This is true for all data sets which are efficient with R-trees (Guttman, 1984) and makes use of the fact that only few overlapping regions exist. The *disjoint* relationship can be assumed the standard case and there is no need to make it explicit. The other four relations are then used to build a relationship graph of all the objects. Note that the graph is only connected when there are no objects inside holes of other objects. However, this is allowed for the matching algorithm described in section 5. but will lead to situations where some regions are not reachable. When using an R-tree for indexing the source data, storage and processing time is far better than in the worst case and is expected to be near $O(n \log n)$.

The resulting structure is containing all the topological neighborhood information and it is possible to use it for different query strategies.

4. DISTRIBUTION BASED APPROACHES

Since the reference data and the sketch data is converted to a graph structure it appears to be possible to apply general graph matching algorithms to our data. Unfortunately this is leading to inefficient algorithms since finding subgraph isomorphisms is a problem with exponential runtime. In reality the sketch is providing more than just the structural information. Every node has type and name information. The edges themselves are typed by the relation they represent. Is it possible to just ignore the graph structure and rely on the additional information?

The idea is to count the number of object types, relation types and names in a region around a central object in the reference data and to compare these frequencies with the sketch. The frequencies can be compiled into two histograms which are easy to compare via the Kolmogoroff-Smirnoff distance (Kopczynski and Sester, 2005). For each object in the reference we get a value for the similarity of the histograms and we can choose the best 10 as potential solutions.

The great benefit of this technique is that only one evaluation of the similarity function per object pair is needed. Since the sketch should be small in nearly all cases, the runtime is only depending linear on the size of the reference database.

On the other hand the results of these queries are not optimal in a great number of cases. There are several reasons for this behavior. At first the sketch is not a complete transcription of the location that should be queried. Only known and important features are included. Additionally the sketched region does not need to be a perfect circle. It is not clear which nodes of the reference should be included for the comparison. The second reason is more fundamentally linked to the kind of data we are using. The objects are spread on the surface of the earth more or less at random, following a certain probability distribution. The entropy of this distribution is a measure for the contained information and how many different places can be distinguished with this method. It shows up that these number is generally small compared to the possible number of places and it is very likely to have a good similarity value for a place which is not very similar from a human point of view. What can be learned from this is that graph structure is vital for a correct interpretation.

5. RELAXATION BASED APPROACH

When using graph structure and additional information in combination it is possible to find such an efficient algorithm by using the graph structure as a basis for deciding which additional information has to be compared. This method relies on "true" information about the sketched area. Can we trust our user?

From Section 2.1 we now that only important and known objects are drawn and the sketch is containing all the knowledge about the location. It is the only source for knowledge about what the user intended with this query. as a consequence we indeed have to trust the information we get and this is especially true for the *existence* of all the sketched objects. Every object in the sketch must be somewhere in the reference. More difficult is the handling of relationships between the objects. As mentioned above, topology should be the first thing to consider, because this is the way we reason about our environment.

But there is a difference of how we are *thinking* about our environment and how we are are *drawing* it. Problematic in particular is the *touch* relation. Frequently one of the lines is a small amount too short or too long. While the first case would mimic a *disjoint* relation, it will become a *crosses* relation in the second. This problems must be addressed in the topology extraction step! If care is taken about this implications, it still is true, that topology is preserved between the sketch and the reference. Thus our task is to solve for the topologic constraints between the drawn objects to get a valid interpretation.

This section will give a framework for modeling the constraints which are describing the relation between sketch and reference objects. A boolean matrix is combining these constraints and an iterative matching algorithm is used to find a solution for this matrix which satisfies all constraints. Finally a set of predicates specific for the sketch matching is defined.

5.1 General Model of the Constraints

For the set of n objects in the reference data we can then write

$$R = \{r_1, r_2, r_3, \ldots, r_n\}$$

The set of m sketched objects shall be

$$S = \{s_1, s_2, s_3, \ldots, s_m, \}$$

Then we can define binary *relation* predicates $p^{r,s}(i,j)$ for each pair (r_i, r_j) or (s_i, s_j), containing the topologic information. The predicates are modeling the relationship between objects in *one* dataset. Based on this we can also define *assignment* predicates $p^a(i,j)$ between reference objects and sketch objects (r_i, s_j) of *different* data sets for testing if two objects are compatible with each other. These predicates can also take other properties than topology into account.

5.2 Defining the Matching Problem

The aim of the algorithm is to find at least one matching reference object for each of the sketch objects. Whether two objects are matching is determined by the boolean matrix $\underline{\mathbf{A}}$ with dimensions n, m. The value of its components are $a_{i,j} = 0$ (false) if no assignment from the sketch object to the reference object is possible and $a_{i,j} = 1$ (true) if there is a potential matching between them. The matching algorithm is supposed to return a matrix $\underline{\mathbf{A}}$ with a consistent assignment. An assignment is consistent in terms of the given predicates if all the assignment predicates are true:

$$a_{i,j} \stackrel{!}{=} \underbrace{\bigwedge_{\forall k} p_k^a(i,j)}_{c_p(i,j)} \qquad \forall i, \forall j$$

Before evaluating any of the predicates it is impossible to make decisions about assigning objects S to objects from R. More exactly all assignments could be true. The matrix $\underline{\mathbf{A}}$ does not contain true propositions in general but only what is known in a certain state of the matching process. Thus it is possible to write

$$\underline{\mathbf{A}}_0 = \underline{\mathbf{1}}$$

which means "nothing certain is known about the current problem".

5.3 Matching Algorithm

The crucial question after the problem definition is how to get a consistent assignment matrix which satisfies all the assignment predicates. This paper is using the discrete relaxation algorithm introduced by Waltz (1975).

At first it seems obvious to evaluate all predicates for each element of the assignment matrix. And indeed this gives the right answer when we are checking for consistent object types and names. But in case of the existence of dependencies inside the data set, expressed by the existence of predicates $p^{r,s}(i,j)$, this only solves a part of the problem. An assignment might be eliminated which was used in an earlier evaluation being used in yet another earlier evaluation, etc.

Following Waltz the evaluation must be repeated as long as there are changes in the assignment. The process will stop in at most $\max(n,m)$ steps and always gives the same result independent from the order of evaluations. The update rule can be written as

$$\underline{\mathbf{A}}_{i+1} = \begin{bmatrix} a_{1,1}^i \wedge c_p^i(1,1) & \cdots & a_{1,n}^i \wedge c_p^i(1,n) \\ \vdots & & \vdots \\ a_{m,1}^i \wedge c_p^i(m,1) & \cdots & a_{m,n}^i \wedge c_p^i(m,n) \end{bmatrix}$$

The worst case runtime of this algorithm is

$$O(\max(n,m) \cdot m \cdot n \cdot q)$$

Where q is a factor determined by the predicates $p^{r,s}$. m can always considered to be small compared to n and has the effect of a constant because the sketches will be more or less the same size.

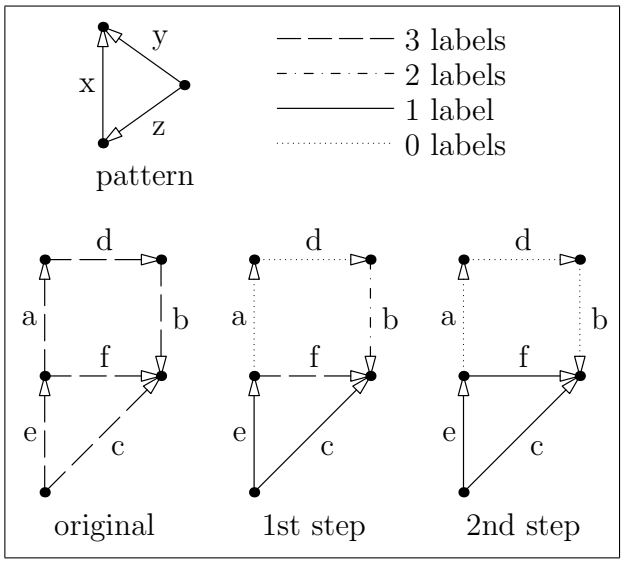


Figure 2. Assignment of a simple pattern to a reference graph with two iterations of the algorithm.

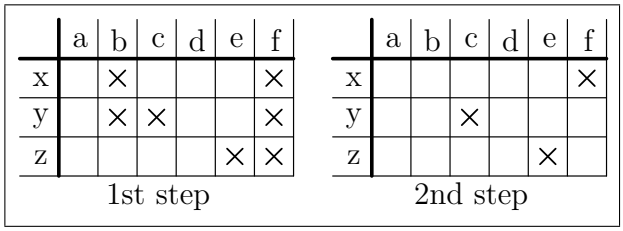


Figure 3. Elimination of labels. Possible assignments are marked with a cross

Figure 2 is showing how this works on two small graphs with oriented edges. The only predicate used for this example checks if both nodes of an edge from the pattern can be assigned to an edge of the reference where connected edges have the same orientation as in the pattern. After the second iteration no further changes of the labeling is possible.

The tables in Figure 3 are showing in detail what is happening with the assignments between the iterations (the table before the first iteration would have marks in all the cells). The algorithm looks into every marked cell and checks for the predicate. If the predicate is false, the cell is cleared and will affect the predicate evaluation of the next iteration. After the second step we have reached a unique labeling as a result.

5.4 Specialized Set of Predicates

The algorithm of the last section can be applied to any set of predicates for the given data sets. From the properties of our data sets we can now derive the problem specific predicates.

Dependencies *inside* the data sets are

$$\begin{aligned} cp_1^{r,s}(i,j) &= \text{within}(i,j) \\ p_2^{r,s}(i,j) &= \text{overlaps}(i,j) \\ p_3^{r,s}(i,j) &= \text{crosses}(i,j) \\ p_4^{r,s}(i,j) &= \text{touches}(i,j) \end{aligned}$$

Since it is not possible to distinguish between the touches and crosses relationship in hand drawings, the predicates p_3^s and p_4^s could be joined to

$$p_3^s(i,j) = \text{crosses}(i,j) \vee \text{touches}(i,j)$$

as it was done in some experiments. However, the correct way to handle this would be the topology extraction process!

Dependencies *between* the data sets are

$$\begin{aligned} cp_1^a(i,j) &= (\text{typeof}(r_i) \sqsubseteq \text{typeof}(s_j)) \\ p_2^a(i,j) &= (\text{nameof}(s_j) \hat{=} \text{nameof}(r_j)) \\ p_3^a(i,j) &= (\dim(s_j) = \dim(r_i)) \end{aligned}$$

The first predicate ensures that the object type from the reference must be more more specific or equal to the object type from the sketch. The second one is true when the name of the sketch object is part of the reference objects name and the last is ensuring that only objects of equal topologic dimension are assigned. This predicates do not have data set internal dependencies which are present in the fourth predicate:

$$\begin{aligned} cp_4^a(i,j) = \bigcup_{n_s \in N_s(j)} T(n_s, N_r(i)) \; &\geq |N_s(j)| \\ N_r(i) &= \{r_k | \neg \,\text{disjoint}(i,k), \forall r_k \in R\} \\ N_s(j) &= \{s_k | \neg \,\text{disjoint}(j,k), \forall s_k \in S\} \\ T(a,B) &= \{b | \,\text{topeq}(b,a), \forall b \in B\} \end{aligned}$$

$$\begin{aligned} \text{topeq}(a,b) = &(\text{within}(r_i,a) \wedge \text{within}(s_j,b)) \quad \vee \\ &(\text{overlaps}(r_i,a) \wedge \text{overlaps}(s_j,b)) \quad \vee \\ &(\text{crosses}(r_i,a) \wedge \text{crosses}(s_j,b)) \quad \vee \\ &(\text{touches}(r_i,a) \wedge \text{touches}(s_j,b)) \end{aligned}$$

Two objects r_i and s_j are topological consistent when for every object in the set of non disjoint neighbors of s_j ($N_s(j)$) a neighbor (from $N_r(j)$) of r_i exists which has the same topologic type. Additionally no neighbor is allowed to be used twice. This uses the predicate $\text{topeq}(a,b)$ which is true, when two objects $a \in R, b \in S$ have the same topologic relationship to the objects r_i and s_j. The predicate p_4^a uses the marriage theorem of Hall (Sun, 2001) for the test of topological consistency. Due to that the union set of all possible assignments for each of the neigbours around s_j must have at least as many elements as we have neighbors of s_j. It is easy to see that looking for the existence of *any* consistent labeling is computationally much less expensive than finding *one* labeling.

5.5 Improving the Predicate Set

Although the proposed set of predicates is doing a good job in respect to the problem definition, there is some room for improvements. It is required to have a complete object in the reference data base for each of the sketched objects. This does not fit well in cases where for instance a sketched area resolves to more than one object in the reference. A solution would be to decompose the reference data into atomic cells in the plane and insert a predicate that can test for a composition of cells which is satisfying the boundary conditions.

The developed framework does ignore all geometric information for good reasons. The topology is determining the overall context in the sketch. But sometimes geometric properties are used for refining the overall conditions. An example would be a drawn building *near* a junction, while keeping a greater distance to another junction. Which junction is right can not be decided from topology if there are no further specifications and a new predicate for *near* must be introduced. This is problematic although near is defined by geometry but depends on the context and a (uncertain) threshold must be chosen.

The challenge of geometric conditions is to find discrete predicates for a continuous phenomenon while topologic conditions are discrete by their nature. Another approach could be to turn the predicates into probabilities, which allow to say *more or less* rather than *yes* or *no*. This requires to change the algorithm to continuous relaxation (Hummel and Zucker, 1983) which is computationally more complex and do not generate simple results since one must decide at the end whether a probability of 0.8 is enough to support the assignment of two objects.

6. IMPLEMENTATION AND EXAMPLES

The described algorithm and the predicates have been used to develop a sketch retrieval application which can be used to preprocess reference data sets, draw sketches and solve the sketched constraints. The results can be viewed in a technically useful way or more intuitive on a map. It is implemented in Java and uses JTS (Java Topology Suite) from Vivid Solutions for all spatial computations.

6.1 Drawing a Sketch

The sketching interface is very simple and supports the drawing of points, lines and areas which can carry two attributes: object type and name. Available object types are depending on the used data set and are hierarchically organized. This way an object can be specified to be a local road, a road in general or just an object from the transportation network.

Currently only deleting an object is implemented as an editing gesture, but in the future more gestures tailored for using pen based input devices would be needed.

6.2 Interpretation and Visualization

For experimental purposes it is useful to see what is happening inside the interpretation process and consequently the sketching application allows for visualizing the current matching state for each step in the interpretation process.

Before the interpretation can be started, a preprocessed data set must be selected and the interpretation dialog is offering to step through each of the processing steps:

- Checking for topologic dimension
- Checking for object types and names
- One iteration for evaluating p_4^a on the assignment matrix.

The evaluation stops when the assignment matrix did not change in the last iteration. The visualization of the assignment matrix and the map display is updated during the processing and the progress can be watched.

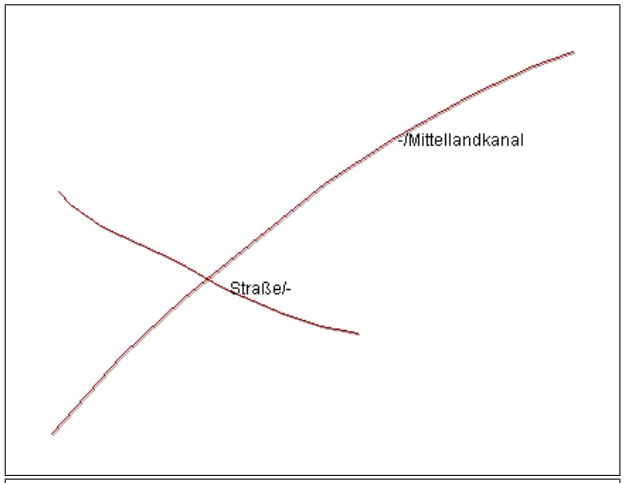
In the assignment matrix view all relevant information of the objects of both data sets are shown and possible assignments between two objects are indicated by a green label in the cell of the intersecting rows and columns. When the algorithm has finished, the remaining green labels are signaling the correct assignment between the sketch objects and the reference objects.

The matrix view is as good tool for discovering what is going on inside the algorithm. But for quick checking the correctness of the result it is more comfortable to have a map visualizing the assignments by highlighting. This exactly what the map view of the sketching application is doing.

6.3 Examples

The sketching application is used here to show some examples. As reference data a navigation data set for the region covering the city of Hannover and surrounding places is used. The complete transportation network, administrative areas, most of the topographic features and points of interests are included. No performance measures are carried out so far but the interpretation step never requires more than a second for the complete solution.

The first scene in Figure 4 is showing a sketch for *"finding all roads crossing the Mittellandkanal"* and the result how it is shown in the assignment matrix view.



	Topolo...	Type	Name	0	1	SourceCount
Type					road	
TargetCount				7	57	
Name				Mittellandk...		
34335		water_centre_line	Mittellandkanal ...			1
34336		water_centre_line	Mittellandkanal ...			1
34337		water_centre_line	Mittellandkanal ...			1
34347		water_centre_line	Mittellandkanal ...			1
34349		water_centre_line	Mittellandkanal ...			1
34351		water_centre_line	Mittellandkanal ...			1
34354		water_centre_line	Mittellandkanal ...			1
Topology						
TargetAssi...						
0		road_00	...			0
1		road_00	...			0
2		road_00	...			0
3		road_00	...			0
4		road_00	...			0
5		road_00	...			0
6		road_00	...			1
7		road_00	...			1
8		road_00	...			0
9		road_00	...			0
10		road_00	...			0
11		road_00	...			0
12		road_00	...			1
13		road_00	...			0
14		road_00	...			0
15		road_00	...			1
16		road_00	...			0
17		road_00	...			0
18		road_00	...			0
19		road_00	...			0

Figure 4. All roads crossing the Mittellandkanal (an important channel through northern Germany). Result viewed in the assignment matrix

The other scene in Figure 5 is querying for a situation where a road is intersecting the park "Eilenriede", which is crossed by two other (unspecified) lines which themselves are crossing a railway. The result of the matching process is showing how the yellow area is crossed by several lines in red (the selected roads) and the railway on the bottom (in blue) which is also partly selected.

7. CONCLUSIONS

This paper presented a framework for identifying locations by using sketches of the queried area. This method does not require

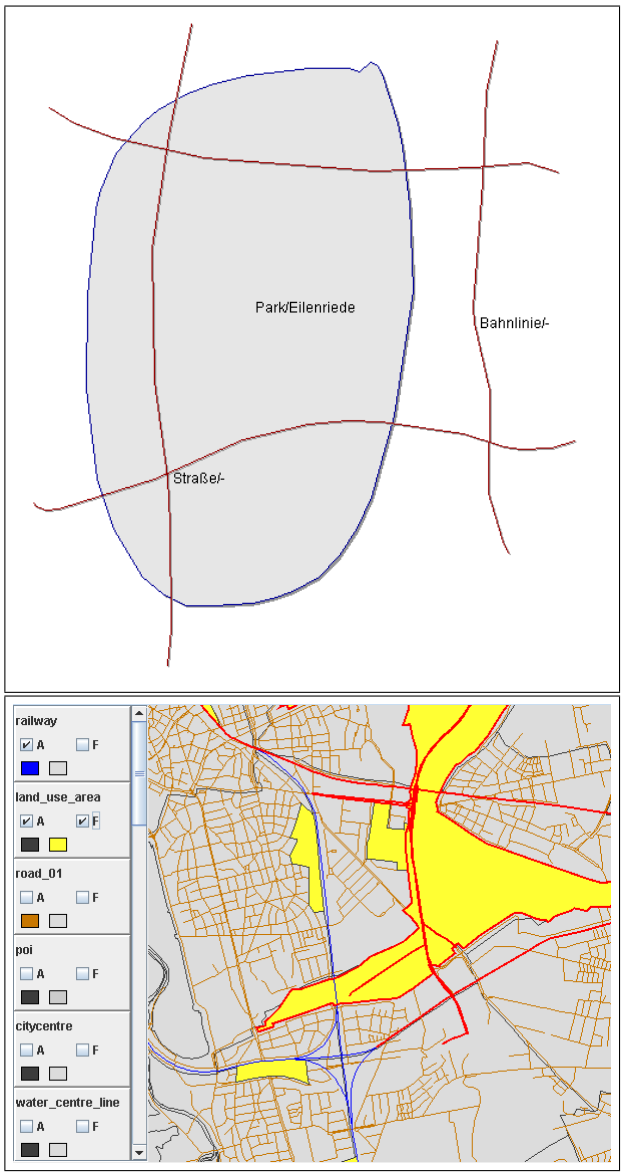


Figure 5. Querying a situation which involves an area and an indirectly connected line

prepared scenes and can find any set of objects as long as they satisfy the constraints from the sketch and the reference data set contains the requested objects and relations. For this the application of topologic constraints in combination with some object inherent information seems to be powerful enough for solving a large range of spatial queries. Nevertheless there is still room for more flexible combination of topologic constraints and introducing geometric constraints for refinement.

The proposed method is also computational efficient and is scalable for the use of much larger data sets than used in the testbed. However an extension to data sets exceeding memory size (topologic predicates do not consume much memory) will need to deal with the resulting complications. This especially true for mobile services where this technique would have many benefits. Switching to a client - server architecture will be necessary to proof this in the future. Nevertheless the current implementation is demonstrating the power of the proposed method.

REFERENCES

Blaser, A. D, 2000, May). *Sketching Spatial Queries*. Ph. D. thesis, SIE Department of Spatial Information Science and Engineering.

Brin, S and L Page, 1998. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems* 30(1-7), 107–117.

Caduff, D and M. J Egenhofer, 2005, December). Geo-mobile queries: Sketch-based queries in mobile gis-environments. In K.-J Li and C Vangenot (Eds.), *W2GIS: 5th International Workshop on Web and Wireless Geographical Information Systems, Lausanne, Schweiz*, Lecture Notes in Computer Science, pp. 143–154.

Clementini, E, P. D Felice, and P van Oosterom, 1993, June 23-25). A small set of formal topological relationships suitable for end-user interaction. In D Abel and B. C Ooi (Eds.), *Advances in Spatial Databases, Third Internatuional Symposium, SSD '93*, pp. 277–295. Springer-Verlag.

Egenhofer, M, 1994. Spatial sql: A query and presentation language. *IEEE Transactions on Knowledge and Data Engineering* 6(1), 86–95.

Egenhofer, M. J and R. D Franzosa, 1991. Point-set topological spatial relations. *International Journal for Geographical Information Systems* 5(2), 161–174.

Flickner, M, H Sawhney, W Niblack, J Ashley, Q Huang, B Dom, M Gorkani, J Hafner, D Lee, D Petkovic, D Steele, and P Yanker, 1995, September). Query by image and video content: The qbic system. *Computer* 28(9), 23–32.

Guttman, A, 1984. R-trees: a dynamic index structure for spatial searching. In *Proceedings of the 1984 ACM SIGMOD international conference on Management of data*, Volume 14 of *ACM SIGMOD*, pp. 47–57. ACM Press, New York, NY, USA.

Herot, C. F, 1976. Sketch recognition for computer-aided design. In *Proceedings of the ACM/SIGRAPH workshop on User-oriented design of interactive grap*, International Conference on Computer Graphics and Interactive Techniques, pp. 31–35. ACM Press, New York, NY, USA.

Hummel, R. A and S. W Zucker, 1983. On the foundations of relaxation labeling processes. In *IEEE Transactions on Pattern Analysis and Machine Intelligence, 1983, PAMI-5*, pp. 267–287.

Kopczynski, M and M Sester, 2005. Graph based methods for localisation by a sketch. In *Proceedings of 22nd International Cartographic Conference, 9.-16. July 2005, La Coruna/Spain*.

Pu, J, K Lou, and K Ramani, 2005. A 2d sketch-based user interface for 3d cad model retrieval. *Computer-Aided Design & Applications* 2(1-4), 1–9.

Sun, Z.-W, 2001. Hall's theorem revisited. In *Proceedings of the American Mathematical Society*, Volume 129, pp. 3129–3131.

Sutherland, I. E, 1963, January). *SKETCHPAD - A Man-Machine Graphical Communication System*. Phd. theses, Massachusetts Institute of Technology.

Tappe, H and C Habel, 1998. Verbalisation of dynamic sketch maps: Layers of representation and their interaction. Citeseer.

Waltz, D, 1975. *Understanding line drawings of scenes with shadows*, Chapter 2, pp. 19–91. McGraw-Hill computer science series. McGraw-Hill. ISBN: 0-07-071048-1.