# 6

# A Visual Database System for Spatial and Non-spatial Data Management

*Theodore Dalamagas, Timos Sellis, Loukas Sinos*
*Computer Science Division*
*National Technical University of Athens*
*Zographou, Greece, 15773*
*E-mail: {dalamag, timos, lsinos}@dbnet.ece.ntua.gr*

**Abstract**

A symbolic image is an array representing a set of objects and a set of direction relations among them. Symbolic images can be the source for retrieving direction relations via the Pictorial Query-by-Example (PQBE) language, a visual language which generalises from the example given by the user and is based on skeleton images which are by themselves symbolic images.

This paper presents an implementation of the PQBE language. The PQBE language is enriched with SQL-like predicates for non-spatial attributes and symbolic images are combined with a relational database scheme in a prototype visual database system, which integrates spatial and non-spatial management.

**Keywords**

spatial management, spatial query language, direction relations, spatial constraints, non-spatial constraints, visual database system

## 1 INTRODUCTION

Spatial data management requires the handling of *spatial relations* among *spatial objects*. Spatial relations include *direction relations* that describe order in space (e.g. north, southwest), *topological relations* that describe neighborhood, inclusion and incidence (e.g. inside, overlap) and *distance relations* (e.g. far, near) [Pullar and Egenhofer, 1988]. The management of direction relations and their representation is crucial for user interfaces, when the visual component is of great importance [Mark, 1992].

A *symbolic image* is an array representing a set of objects and a set of direction relations among them. Symbolic images and related structures such as the *2D strings* [Chang et al., 1987] (one dimensional encodings of symbolic images), *symbolic arrays* [Glasgow and Papadias, 1987] (hierarchic sym-

Visual Database Systems 4 Y. Ioannidis & W. Klas (Eds.)

| NorthWest | North | NorthEast |
|---|---|---|
| West | Y | East |
| SouthWest | South | SouthEast |

**Table 1** Primitive direction relations

bolic images) and *symbolic spatial indexes* [Papadias and Sellis, 1993] (a set of symbolic images where the symbols correspond to direction and topological representative points), have been proposed for the visual representation of spatial information, especially for the two-dimensional space.

Using symbolic images, nine binary, pairwise disjoint, direction relations can be represented: {*NorthWest, North, NorthEast, West, SamePosition, East, SouthWest, South, SouthEast*}. These relations are known as *primitive direction relations* and correspond to the highest resolution that can be acquired using one symbol per object in symbolic images. Exactly one of the previous relations holds true between any pair of objects in a symbolic image. Table 1 presents the primitive direction relations among objects in a symbolic image. The symbol $Y$ denotes the *reference object* and the other symbols refer to the direction relation depending on the position of the *primary object* in the symbolic image. The term *primary object* refers to the object which is to be located, while the term *reference object* refers to the object in relation to which the primary object is located.

Symbolic images can be the source for retrieving direction relations via the *Pictorial Query-by-Example (PQBE)* language [Papadias and Sellis, 1995], a visual language which generalises from the example given by the user and is based on *skeleton images* which are by themselves symbolic images.

In this paper, we present an implementation of such a language. The PQBE language is enriched with SQL-like predicates for non-spatial attributes. Symbolic images are combined with a relational database scheme so that an integrated visual database system is constructed which is able to deal with spatial and non-spatial attributes of objects.

The next section presents an overview of the PQBE language and its capabilities by providing several examples. In the third section the PQBE language is extended in order to handle non-spatial object attributes. A prototype information system is presented in the fourth section and finally, in the last section, directions of further work are discussed.

## 2 THE PICTORIAL QUERY-BY-EXAMPLE LANGUAGE

The Pictorial Query-by-example (PQBE) language aims at retrieving direction relations from symbolic images. In Figure 1 a simple example of such a retrieval is presented.

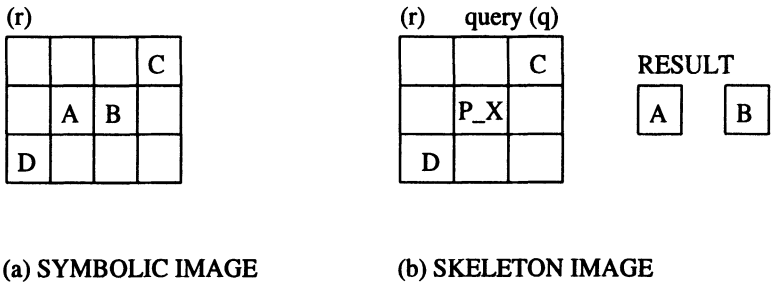


**Figure 1** A simple retrieval using the PQBE language

The query $q$ can be stated as follows:
*"Retrieve and print all objects which are northeast of object D and southwest of object C in the symbolic image (r)."*
$A$ and $B$ are the objects that satisfy this spatial constraint.

The following subsections present the capabilities of the PQBE language in more detail. All the examples are with respect to the symbolic images of Figure 2.

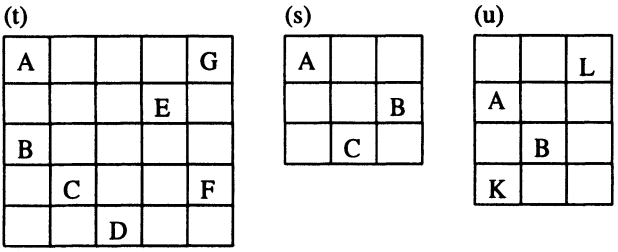


**Figure 2** Symbolic images for the following examples

### 2.1 Queries involving one skeleton image

This class of queries involves the retrieval of subimages which contain primary objects which satisfy some spatial constraints with respect to the reference

object constants or variables of the skeleton image. Variables denoting objects and images are preceded by the special character "_", while constants appear without qualification. Primary objects are preceded with the special character "$P$", which makes the value of this object to be retrieved and displayed. Hence, direction retrieval with the PQBE language includes the following constructive elements: ***skeleton images***, ***symbolic images***, ***object variables***, ***object constants***, ***image variables***, and ***special characters***.

In the example of Figure 1:

- $A, B, C, D$ are the object constants of the symbolic image $r$.
- $D, C$ are the object constants of the skeleton image $q$.
- $X$ is an object variable.
- $P$ (in $P\_X$) is a special character: retrieve and display the values of $X$.

The result of a query is the set of all symbolic subimages which satisfy the spatial conditions imposed by the skeleton images. For each one of the following example queries, we include the skeleton image, the imposed spatial constraints and the resulting subimage with respect to the symbolic images of Figure 2.

The skeleton image $q1$ of Figure 3a will retrieve all symbolic subimages $I$ of symbolic image $t$ which contain an object $X$ (primary object variable), where $X$ is *SouthEast* of $A$ (reference object constant) and *NorthEast* of $D$ (another reference object constant), where $A$ is *NorthWest* of D, in $t$. The subimages of $t$ which satisfy the previous sets are also illustrated in Figure 3b.

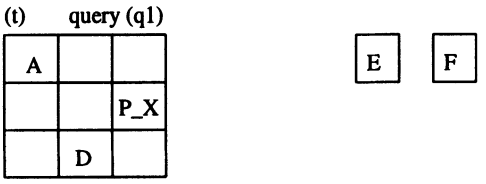


**Figure 3** Query involving one primary object and two reference object constants

Similarly, the skeleton image $q2$ of Figure 4a will retrieve all the subimages $I$ of symbolic image $u$ which contain an object $X$, where $X$ is *NorthEast* of $A$ (reference object constant) and *NorthEast* of some object $Y$ (reference object variable), where $Y$ is *SouthEast* of $A$, in $u$.

If primary objects are not preceded by the character $P$, the skeleton image takes the form of a boolean query. Such an example is presented in Figure 4b.

All previous queries result in subimages containing a single object. However,

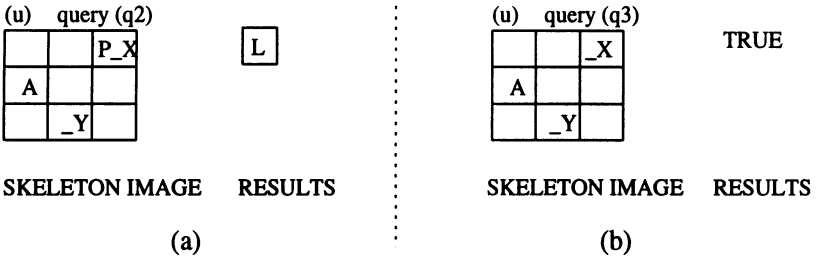


**Figure 4** (a) Query involving one primary object, one reference object variable and one reference object constant. (b) Query involving two reference object variables and one reference object constant.

we can also have subimages as results of queries, which contain configurations of two or more objects. The skeleton image $q4$ of Figure 5a, for example, will retrieve all subimages $I$ of $t$ which contain two objects in such configuration that the one is *North* of the other in $t$.

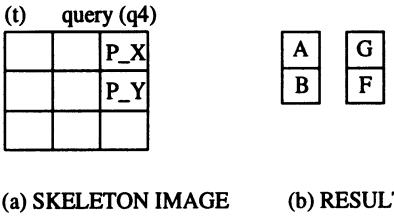


**Figure 5** Query involving retrieval of object configurations

Unlike the previous queries which refer to *image constants* (i.e. the skeleton images are applied to a specific symbolic image), the query of Figure 6a refers to an *image variable*. The skeleton image $q5$ will retrieve all subimages $I$ which contain an object $X$ which is *SouthWest* of $B$ in some image $J$ (image variable).

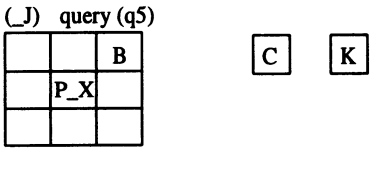


**Figure 6** Query involving image variables

## 2.2 Queries involving multiple skeleton images

This section describes queries which involve more than one skeleton images. By using multiple skeleton images we can express *intersection*, *join* and *union* operations.

The skeleton images of Figure 7a will retrieve all subimages $I$ of $t$ which contain objects which are *NorthWest* of $D$ and *SouthWest* of $E$ in $t$, i.e. the result is the intersection of two subimages.

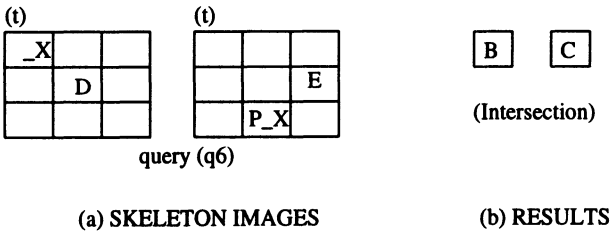


**Figure 7** Query involving intersection

Join is expressed by a chain of common objects connecting two or more skeleton images. The skeleton images of Figure 8a will retrieve all subimages $I$ of $u$ containing an object $Y$ which is *NorthWest* of an object $X$ in image $u$, where $X$ is *South* of A in $t$. This query corresponds to the join operation, where $X$ is the common object connecting the images.

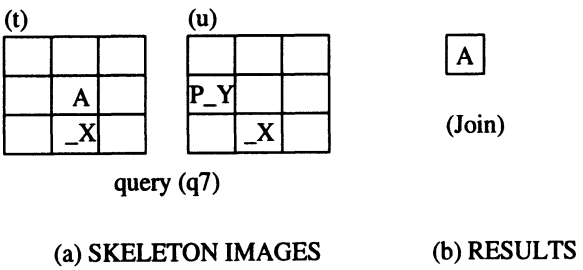


**Figure 8** Query involving join

Finally, the skeleton images of Figure 9a will retrieve all subimages $I$ of $t$ containing objects which are *NorthWest* of $C$ in $t$ or *NorthEast* of $K$ in $u$, i.e. the result is the union of two subimages.

In comparison to verbal QBE [Zloof, 1977], PQBE provides a more intuitive and easy to use interface because of its close correspondence with the structure of 2D space. Unlike verbal languages it does not presume knowledge of keywords for spatial relations on behalf of the user, nor familiarity

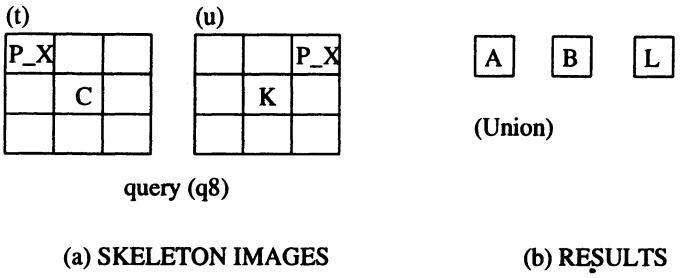


**Figure 9** Query involving union

with database languages and complex spatial conditions are expressed in a straightforward manner. Furthermore, several features can be incorporated to increase the expressive power of the PQBE, like update operations and negation usage.

[Chang et al., 1988] developed a similar pictorial language for retrieving object configurations stored in 2D strings. In addition to the retrieval of object configurations, PQBE has the ability to express negation, independent sub-conditions in the form of multiple skeleton images, union, intersection and join operations, image and relation retrieval, and to perform inference. Visualseek [Smith and Chang, 1996], a fully automated content-based image query system, uses a similar approach for inserting spatial queries, but its expressiveness is limited compared to the PQBE language. For a complete description of the PQBE language one can refer to [Papadias and Sellis, 1995].

## 3 EXTENDING THE PICTORIAL QUERY-BY-EXAMPLE LANGUAGE

Spatial data management with PQBE excludes object characteristics which are expressed with non-spatial attributes. In the following, PQBE is extended with predicates for non-spatial attributes, using SQL-like syntax. The extension aims at managing the spatial constraints (based on direction relations among objects) and the non-spatial constraints (based on object characteristics) in an integrated way. An example of a retrieval based on spatial and non-spatial attributes using the extended PQBE language is depicted in Figure 10.

The query $q$ of Figure 10 can be stated as follows:
*"Retrieve and print out all objects (cities) which are northeast of object D and southwest of object C with population more than 1000000."*

Without the non-spatial constraints the result would be the objects $A$ (the city of Athens) and $B$ (the mountain of Olympus). By applying the non-spatial constraints, the object B is eliminated because it is not a city.

Every predicate for non-spatial constraints has one of the following forms:

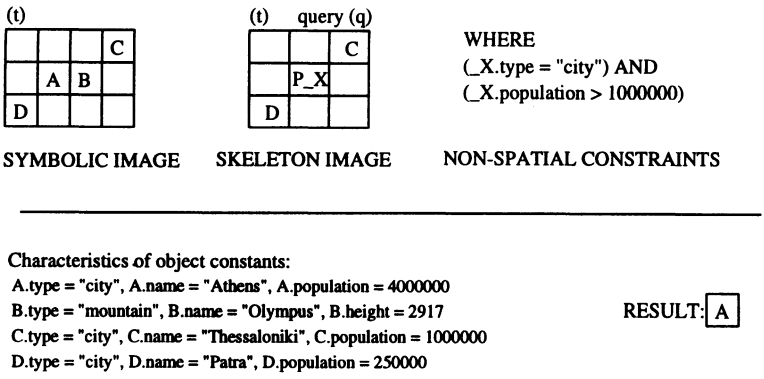


**Figure 10** A simple retrieval using the extended PQBE language

- *Type predicate*:
  <object>.type = (<object>.type OR constantValue)
- *Attribute predicate*:
  <object>.attribute <operator> (<object>.attribute OR constantValue)

where

- <object> can be an object constant or an object variable
- <operator> can be one of the known operators like $>, \geq, <, \leq, =, \neq$.

The presence of an attribute predicate assumes the presence of its corresponding type predicate, since every attribute belongs to a specific type. For example, in Figure 10 the attribute predicate (_X.population > 1000000) can not be used without the constraint imposed by the type predicate (_X.type = "city"). The set of the type predicate together with its corresponding attribute predicates (if any), forms a non-spatial constraint for the object variable it refers to. These attribute predicates may be joined with AND or OR operators. However, the type predicate is always followed by an AND operator if its attribute predicates are to follow. In Figure 10, for example, the predicate (_X.population > 1000000) and (_X.type = "city") imposes a non-spatial constraint for the object variable $X$. Notice that the type predicate is joined with an AND operator with its corresponding attribute predicate, otherwise the constraint is not consistent.

In Figure 11, another example is presented. The query will retrieve all symbolic subimages $I$ of symbolic image $w$ which contain two objects (a city and a lake), where the city is *North* of the lake, its population is less than 250000, and both objects are *SouthWest* of the mountain of Olympus.

After presenting the above extensions to the PQBE language, we move to present a prototype that has been developed.

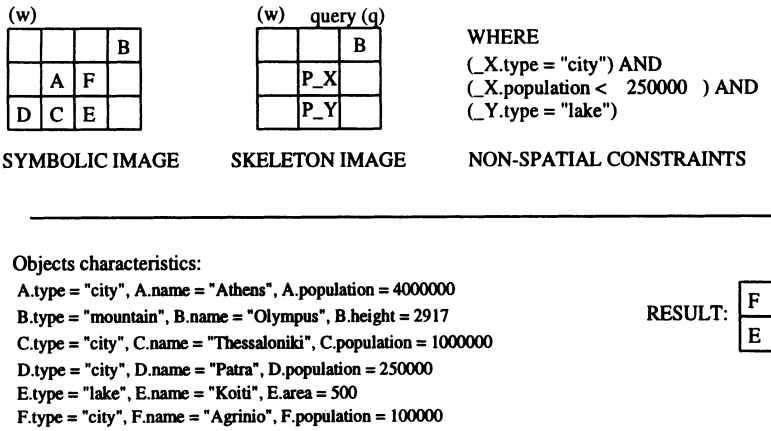


**Figure 11** A retrieval using the extended PQBE language: two type predicates and one attribute predicate

## 4 A PROTOTYPE VISUAL DATABASE SYSTEM

The PQBE language and its extensions have been exploited for the implementation of a visual database system [Sinos, 1997] which gives the user the opportunity to search for objects in digital images, based on their spatial configuration and their particular characteristics, as well as to retrieve images containing objects with special spatial configuration. The system has been implemented for the Windows 95© platform, using the Microsoft Visual C++© programming language and the Microsoft Access© database.

The system consists of the following basic subsystems (see Figure 12): Image managing and data entry subsystem, Query managing subsystem, Display subsystem, and Database subsystem.

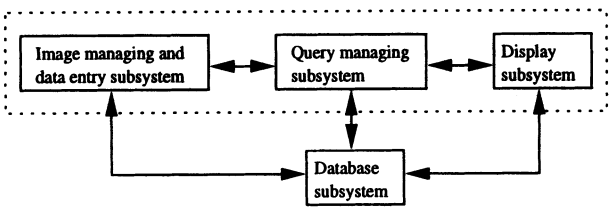


**Figure 12** System architecture

Each one of the above subsystems is discussed in more detail in the following sections.

## 4.1 Image managing and data entry subsystem

The image managing and data entry subsystem deals with the following functions:

- Image loading
  The subsystem can load digital images in various formats. These images may represent for example geographical maps or satellite images. Every object (e.g. a city) in an image is represented by a point, whose coordinates determine the position of the object in the image.
- Object insertion and definition
  The process of object insertion can be accomplished manually. The user is able to define objects that represent spatial entities for a specific image. The user may also specify the type of every inserted object and its particular characteristics. New types with new attributes may be constructed dynamically. All information concerning the objects is stored in a database (see database subsystem). An example of object insertion in a map of Cyprus is presented in Figure 13.

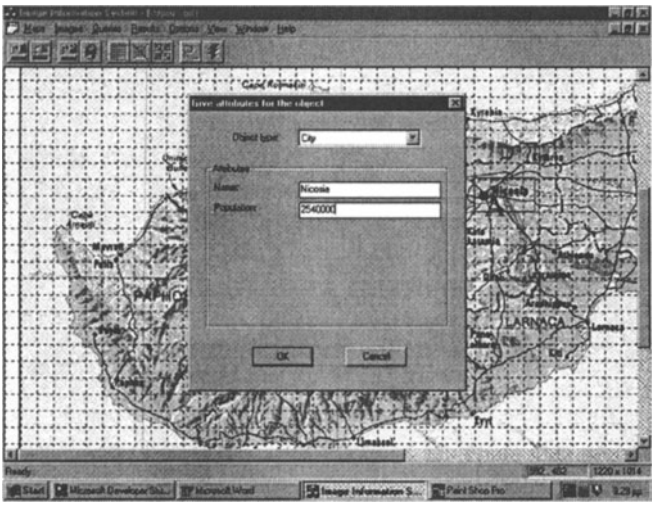


**Figure 13** Object insertion: A user has already inserted the object $A$ which represents the city of Nicosia in Cyprus.

- Symbolic image construction
  Having the definition of objects in a digital image, the corresponding symbolic image which represents the direction relations of objects is constructed.

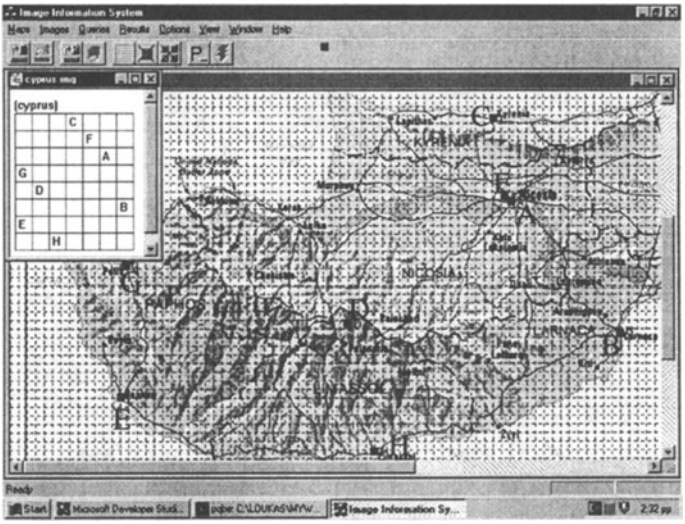


**Figure 14** Symbolic image construction: A user has inserted many objects in the map of Cyprus and a symbolic image has been constructed, based on their spatial configuration

An example of a symbolic image construction with objects from a map of Cyprus is depicted in Figure 14. One can see a grid that is displayed on top of the map. That grid, whose cells' dimensions are denoted by $(\delta x, \delta y)$, is necessary to define the accuracy of the direction relations. If $(\delta x, \delta y) = (0, 0)$, an object $A$ with coordinates (10,40) is northwest of an object $B$(15,15), while if $(\delta x, \delta y) = (10, 10)$, $A$ is north of $B$. The user can change the $(\delta x, \delta y)$ values anytime.

*2D strings* [Chang et al., 1987] are used for storing all symbolic images. 2D strings are derived after taking the vertical and horizontal projections of the objects in the symbolic image. An example of 2D string construction is depicted in Figure 15.

## 4.2 Query managing subsystem

The query managing subsystem processes queries which are given by the users. Spatial and non-spatial constraints are handled by two separate procedures. The implementation algorithms are based on the "select first, join later" principle so that select operations are performed earlier than join operations for better performance.

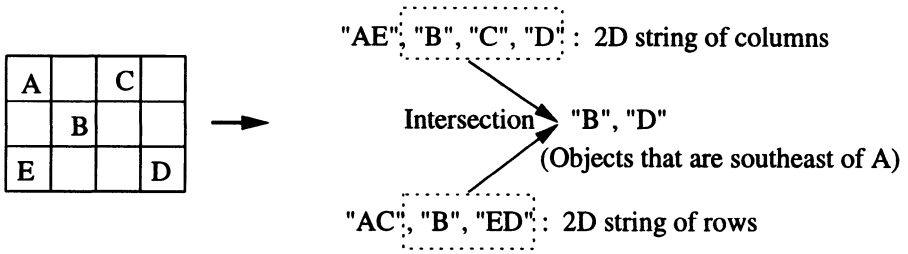


**Figure 15** 2D strings

| Object elements | Possible results |
|---|---|
| Object constant and object constant | TRUE or FALSE |
| Object constant and object variable | NULL or a set of object constants |
| Object variable and object variable | NULL or pairs of object constants |

**Table 2** Basic binary constraints that can be expressed with the PQBE language

## Spatial constraints

Every skeleton image is decomposed in skeleton subimages which express only binary constraints. Binary constraints can be classified into three categories:

1. Spatial constraint between an object constant and another object constant.
2. Spatial constraint between an object constant and an object variable.
3. Spatial constraint between an object variable and another object variable.

As one can see in Table 2, the three categories differ from each other in the output that they have as a result.

In Figure 16, a skeleton image has been decomposed into skeleton subimages which correspond to all of its binary spatial constraints. One can also see the output results for each one of these subimages. Having these results, a three-step join operation is applied in order to detect the symbolic subimages that satisfy the spatial constraints imposed by the query for the corresponding symbolic image $t$ (see Figure 17):

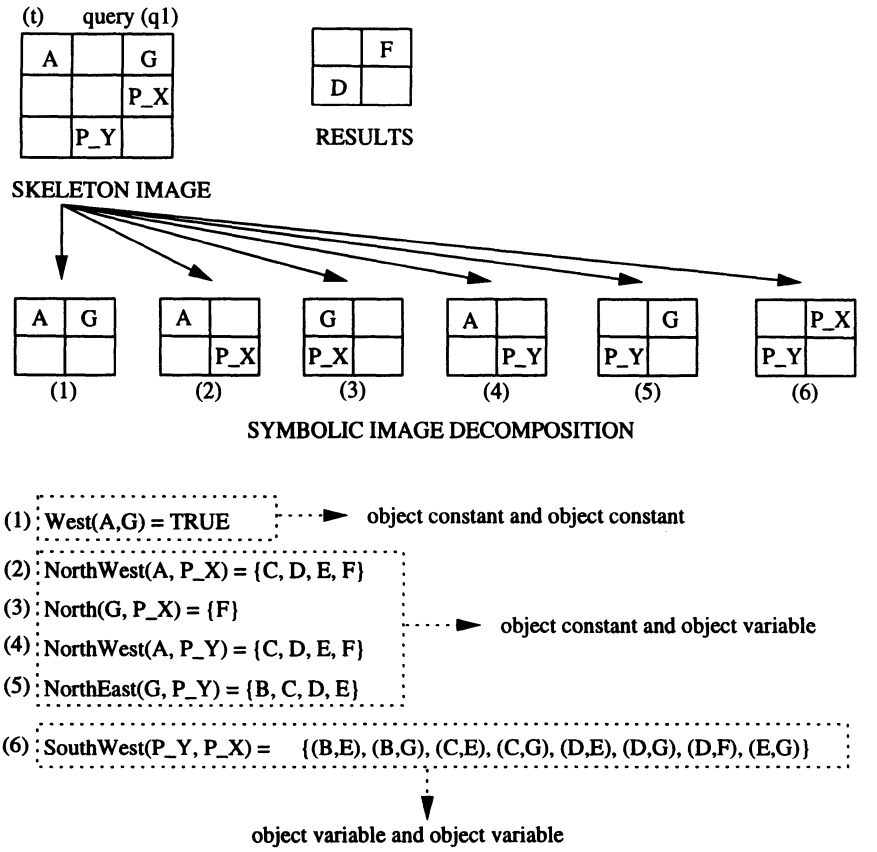


**Figure 16** Skeleton image decomposition

1. Before any other action, all binary spatial constraints that include only two object constants are checked if they hold true or not. If at least one such constraint does not hold true, then the query returns null as a result. In the example of Figure 16 one can observe that the constraint $West(A, G)$ holds in $t$.
2. The operation proceeds by determining the permissible values for each object variable, taking into consideration only the binary spatial constraints that include one object constant and one object variable. In the example of Figure 16 the permissible values for the object variable $X$ are those which result from the intersection of the two sets $\{C, D, E, F\}$ and $\{F\}$, thus $F$. Similarly, the permissible values for the object variable $Y$ are: $\{C, D, E\}$.
3. Finally, the permissible values for each object variable are joined with the pairs of objects that derive from the spatial binary constraints that include only two object variables. In the example of Figure 16 only the pair $(D, F)$ is retrieved, since $D \in \{C, D, E\}$ and $F \in \{F\}$.

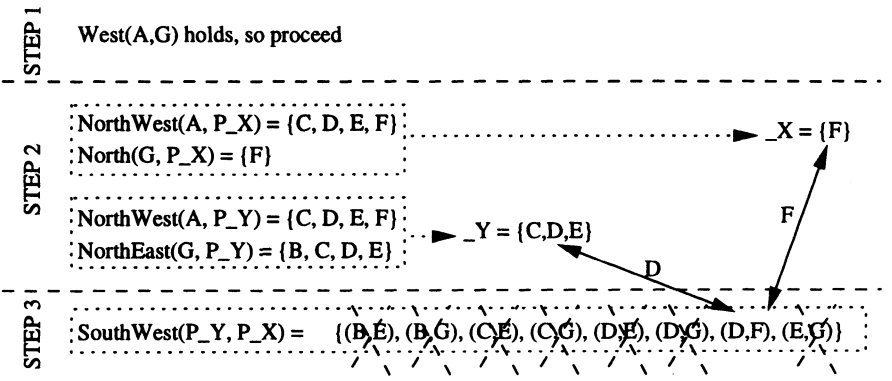


**Figure 17** Query processing

A detailed description of the algorithms that are used for processing more complicated queries can be found in [Dalamagas, 1995].

## Non-spatial constraints

The non-spatial constraints can be taken into account during the three-step operation of Figure 17. However, the earlier these constraints are applied, the more effective is the query processing in terms of performance. Depending on the kind of predicates which are used, three different ways to consider them are suggested:

1. Predicates with only object constants or object constants and constant values, e.g. A.attr1 = 1000, A.attr = B.attr, are taken into account in the first step of query processing (see Figure 17). If at least one such predicate does not hold, then the whole query returns null as a result.
2. Predicates with one object variable and one object constant or a constant value, e.g. _X.type = "city", _X.attr = A.attr, are taken into account in the second step (see Figure 17). From all the permissible values of the object variable, those that do not satisfy the constraint imposed by the corresponding predicate are eliminated and are not used in the third step.
3. Predicates with only object variables (e.g. $\_X$.type = $\_Y$.type) are taken into account in the last step. For example in Figure 17, given a predicate of the form ($\_X$.type = $\_Y$.type), all pairs of objects that do not satisfy the constraint imposed by the predicate are eliminated before the join with the result of the second step.

## 4.3 Display subsystem

The display subsystem presents the results of the query given by the user. The objects that satisfy the constraints imposed by the query are highlighted on the image, so that they can be easily detected. Moreover, the results are presented with the form of symbolic subimages. Figure 18 depicts the four main windows of the implemented system: the query window, the map window, the results window and the symbolic images window. Moreover, a dialogbox displays information concerning an object which has been selected by the user.

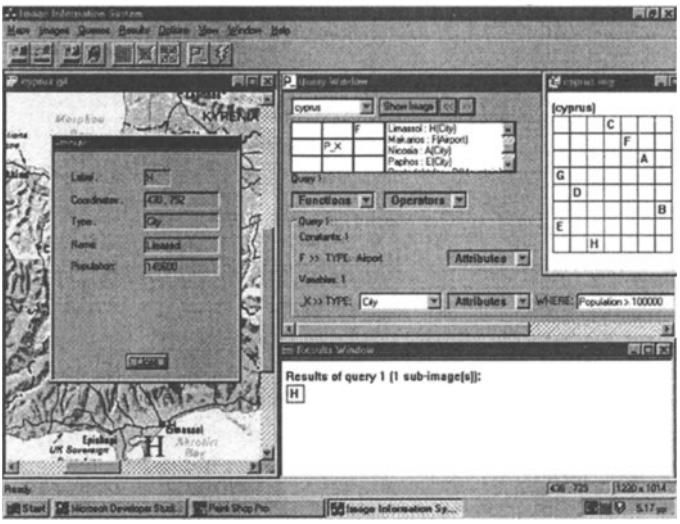


**Figure 18** Query state and display of results: The above query has retrieved an object $H$ (the city of Limassol) which is southwest of the object $F$ (airport) and its population is greater than 100000.

## 4.4 Database subsystem

All information concerning the objects is stored in a relational database. The schema of the database consists of the following entities:

- *Image:* It represents the digital image where the objects are inserted.
- *Symbolic image:* It represents the symbolic image which is created from a digital image with objects.
- *Spatial entities:* They represent the types of the objects that are inserted in a digital image, for example city, airport, mountain, etc.

All spatial entities have some common attributes, like a label, an identifier of the digital image and the symbolic image in which they are located and their coordinates. Moreover, each spatial entity has its individual attributes, for example the entity "city" has attributes like "name" and "population". The user can modify the schema of the database dynamically, that is to create new spatial entities with new attributes.

## 5 CONCLUSIONS AND FURTHER WORK

In this paper we presented a visual database system for managing spatial and non-spatial data. This system gives the user the opportunity to search for objects in digital images, like maps or satellite images, based on their spatial configuration and their particular characteristics and retrieve images containing objects with specific spatial configuration.

Symbolic images have been proposed for the visual representation of spatial information. They can be exploited effectively to represent direction relations among objects. Moreover, they can be the source for retrieving direction relations via the Pictorial Query-By-Example language (PQBE), a visual language which generalises from the example given by the user and is based on skeleton images which are by themselves symbolic images. Our system exploits the PQBE language for managing the direction relations among objects.

Because the PQBE language excludes object characteristics or quantitative information, it was enriched with SQL-like predicates for handling non-spatial data, like object characteristics. The extension aims at managing the spatial constraints, which are based on the direction relations among objects, and the non-spatial constraints, which are based on object characteristics, in an integrated way.

Many issues need to be further considered in detail for the presented visual spatial information system.

- *Automatic object insertion*
  In the current implementation, the user can manually define objects which represent spatial entities for a specific image. Image recognition routines may be used in order to detect automatically objects, based on the symbols that describe them. Given for example a digital map, a city may be represented by a black point, an airport by an aeroplane symbol, etc. Based on a sample database of such symbols, these routines can detect objects, by searching for their symbols. These objects will be the basis for constructing the corresponding symbolic images.
- *Query processing*
  The query processing is based on the decomposition of the skeleton images in skeleton subimages which express only binary constraints. Currently, all these subimages are taken into account when processing the query. However, many of them are redundant due to the fact that they can be inferred

from others. For example, if an object variable $X$ is north of $B$ and $B$ is north of $A$, then only the *North(X,B)* relation is necessary for identifying the values that satisfy the previous spatial constraints. The retrieved objects will also satisfy the relation *North(X,A)* because $A$ is south of $B$. Such inferences have been classified in [Papadias and Sellis, 1995] and can be used to reduce the query processing overhead.

- *PQBE implementation*
  Although the PQBE has the ability to handle negation and to perform update operations, the current implementation does not support this feature. Such an addition will extend the expressive power of the language and the overall usefulness of the system.
- *User evaluation*
  The implemented visual database system is based on a multiple-window interface that includes query windows, map windows and result windows. Such an environment should be evaluated by performing user tests for its effectiveness.

## REFERENCES

[Chang et al., 1987] Chang, S. K., Shi, Q. Y., and Yan, C. W. (1987). Iconic indexing by 2D strings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(3):413–428.

[Chang et al., 1988] Chang, S. K., Yan, C. W., Arndt, T., and Dimitroff, D. (1988). An intelligent image database system. *IEEE Transactions on Software Engineering*, pages 681–688.

[Dalamagas, 1995] Dalamagas, T. (1995). Spatial information retrieval using the Pictorial Query by Example language. Diploma thesis, National Techn. Univ. of Athens, (in Greek).

[Glasgow and Papadias, 1987] Glasgow, J. and Papadias, D. (1987). Computational imagery. *Cognitive Science*, 16:355–394.

[Mark, 1992] Mark, D. (1992). Counter-intuitive geographic Facts: Clues for spatial reasoning at geographic scales. In Frank, A., Campari, I., and Formentini, U., editors, *International Conference GIS - From Space to Territory: Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, Pisa, Italy. Springer Verlag.

[Papadias and Sellis, 1993] Papadias, D. and Sellis, T. (1993). The semantics of relations in 2D space using representation points: spatial indexes. In Frank, A. and Campari, I., editors, *European Conference on Spatial Information Theory*, Elba, Italy. Springer Verlag LNCS.

[Papadias and Sellis, 1995] Papadias, D. and Sellis, T. (1995). A pictorial query by example language. *Journal of Visual Language and Computing*, 6(1):53–72.

[Pullar and Egenhofer, 1988] Pullar, D. and Egenhofer, M. (1988). Toward formal definitions of topological relations among spatial objects. In *Pro-*

*ceedings of the Third International Symposium on Spatial Data Handling*, Syndey, Australia.

[Sinos, 1997] Sinos, L. (1997). An information system for spatial and image retrieval. Diploma thesis, National Techn. Univ. of Athens, (in Greek).

[Smith and Chang, 1996] Smith, J. and Chang, S.-F. (1996). VisualSEEk: a fully automated content-based image query system. In *Proceedings of the Fourth ACM International Multimedia Conference Multimedia 96*, pages 87–98, Boston MA, USA.

[Zloof, 1977] Zloof, M. (1977). Query-by-example: A data base language. *IBM Systems Journal*, 4:324–343.

## BIOGRAPHIES

The authors are members of the Knowledge and Database Systems (KDBS) Laboratory, Department of Electrical and Computer Engineering (ECE), National Technical University of Athens (NTUA), Athens, Greece.

Theodore Dalamagas received his Diploma degree in Electrical and Computer Engineering in 1995 from the NTUA. During 1996, he worked as a software engineer in the KDBS Lab. In 1997 he received the M.Sc. degree from the University of Glasgow. He is currently doing his Phd in the department of ECE of the NTUA. His research interests include spatial information retrieval, text information retrieval and on-line analytical processing techniques for database systems.

Timos Sellis received his Diploma degree in Electrical Engineering in 1982 from the NTUA. In 1983 he received the M.Sc. degree from Harvard University and in 1986 the Ph.D. degree from the University of California at Berkeley, where he was a member of the INGRES group, both in Computer Science. In 1986, he joined the department of Computer Science of the University of Maryland, College Park as an Assistant Professor, and became an Associate Professor in 1992. Between 1992 and 1996 he was an Associate Professor at the Computer Science Division of the NTUA, where he is currently a Full Professor. His research interests include extended relational database systems, active database systems, and spatial, image and multimedia database systems. He has published over 80 articles in refereed journals and international conferences in the above areas.

Loukas Sinos received his Diploma degree in Electrical and Computer Engineering in 1997 from the NTUA. He is currently working as a software engineer in the KDBS Lab. His research interests include spatial and image databases.