Control and Cybernetics

vol. **33** (2004) No. 1

A fuzzy shape database to support conceptual design

by

Jinglan Zhang, Binh Pham and Phoebe Chen

Faculty of Information Technology
Queensland University of Technology
GPO Box 2434 Brisbane QLD 4001 Australia
e-mail: (jinglan.zhang, b.pham, p.chen)@qut.edu.au

Abstract: At the conceptual stage of design, designers only have vague ideas of initial shapes which they gradually refine. These imprecise shapes may be specified by a set of fuzzy shape descriptors which represent the intent of a designer. It is also desirable to be able to save them in a database for future reference or for use as initial shapes for new designs. Most research on fuzzy databases has been focused on theoretical aspects while a fuzzy database is rarely seen in practice, especially in the design area. This paper aims to construct a fuzzy shape database to support shape design by integrating fuzzy data processing and fuzzy querying functions into a conventional database. A possibility-based framework is used for a fuzzy relational database model.

Keywords: fuzzy shape representation, indexing and retrieval of fuzzy shapes, shape database, fuzzy database, conceptual design.

1. Introduction

Design is generally defined as a process of creating a description of an artificial object that satisfies certain constraints. Conceptual design is the process in witch tentative design alternatives are produced. This process is generally achieved by sketching and prototyping. Conceptual design is intrinsically imprecise with imprecision coming from both designers' thinking and practical problems. At this stage, designers have only vague ideas of initial shapes that they gradually refine. Therefore, appropriate tools should satisfy the requirements of fuzziness in shape description, and allow designers to work at a higher level without having to consider precise details so that their creativity is not hindered. Existing CAD systems based on rigid, precise geometric representation such as vertices, edges and surfaces lack these imprecise properties. In order to support the top-down shape design process, it is desirable to be able tc represent the intent of a designer using a set of descriptive words that we call shape descriptors. The fuzzy set approach that was introduced by Zadeh (Zadeh, 1965, 1999) is particularly suitable for handling the imprecise shape perception process of humans and hence is chosen for the shape representation. In addition to its perceptual aspect, a shape also needs to be physically modelled in a CAD (Computer-Aided Design) system so designers can communicate their designs to manufacturers, therefore an underlying geometric shape representation is needed. We have constructed a fuzzy shape specification system to bridge these two aspects of shape (Pham and Zhang, 2000), where each set of shape descriptors corresponds to a set of shapes that looks similar yet slightly different. We call this set of shapes a fuzzy shape.

In many cases, successful designs may be stored in a database and retrieved later to be used as initial shapes for new designs. The database that performs fuzzy shape management is called a fuzzy shape database. Most research on fuzzy databases has been focused on theoretical aspects while fuzzy databases are rarely seen in practice, especially in the design area. This paper aims to construct a fuzzy shape database to support shape design by integrating fuzzy data processing and fuzzy querying functions into a conventional relational database. We concentrate on fuzzy shape representation, indexing, retrieving and querying issues.

In a fuzzy database, the fuzzy set approach is used to represent and manipulate imprecise or uncertain information. Fuzzy relational database is an extension of conventional relational database in the sense that it allows fuzzy attribute values or fuzzy relations to be represented in a relational model. A key characteristic of fuzzy database is that the domain values need not be atomic.

There are two main aspects in the application of the fuzzy set approach to the database area:

- Building fuzzy front-end querying systems for regular, crisp databases, such as the well-known SQLf and FQUERY systems (Kacprzyk and Zadrożny, 1997; Petry and Bosc, 1996). This kind of systems is used to make the query more flexible. Fuzziness happens only at the query level.
- Building a full Database Management System (DBMS) that can facilitate the manipulation of imprecision and vagueness represented by fuzzy sets. In such a system, fuzziness happens in the database itself including attribute data, relationship and entity as well as external querying (Petry and Bosc, 1996; Pons et al., 1997; Vila et al., 1995; Yazici et al., 1999).

Our work belongs to the second class because all shape descriptors are descriptive terms associated with fuzzy sets and all parameters hold fuzzy set values. Yet in practice, we map this application to the first class by treating the attributes with fuzzy set values in a similar way as multiple-valued attributes are usually treated in a conventional relational database. Add-in functions are employed to perform fuzzy updating, querying and retrieving.

Current fuzzy database representation approaches can be categorised into three frameworks according to the way fuzzy data is represented: similarity-based (Buckles and Petry, 1982; Petry and Bosc, 1996), fuzzy-relation-based

(Baldwin, 1983; Baldwin et al., 1995) and possibility-based (Bosc and Galibourg, 1989; Petry and Bosc, 1996; Prade and Testemale, 1987; Umano, 1982). The *similarity-based* framework associates a domain similarity relation with each attribute instead of just an identity relation. This method relies on the pre-partition of attribute's domain and the pre-definition of the corresponding similarity relation. The *fuzzy-relation-based* framework uses weighted tuples to represent fuzzy relations in which each tuple is associated with a degree of truth while the values of individual attributes are crisp. As this framework represents the imprecise information through a membership value associated with the overall tuple, it is not very expressible. In practice, it is often used along with other frameworks such as the possibility-based framework (Baldwin et al., 1995; Umano, 1983). The *possibility-based* framework uses possibility distribution to represent imprecise information including linguistic terms. In this framework, the relation is an ordinary relation yet available imprecise information about the value of an attribute for a tuple is represented by a possibility distribution. Hence, this representation is more flexible and expressive than the similarity-based framework and the fuzzy-relation-based framework. As the possibility-based fuzzy database model associates imprecise information directly with data items, it satisfies the need for storing fuzzy shapes whose parameters are represented as possibility distributions. We will therefore focus on this framework in this paper.

This paper is organized as follows. In Section 2, we introduce the geometric and perceptual representation of fuzzy shapes. A fuzzy shape specification system that provides the original data for the fuzzy shape database is briefly outlined. The shape descriptions that are based on the vague perception in the design process are then discussed. In Section 3, an overview of the database structure is outlined. Section 4 presents the conceptual modelling of this fuzzy shape database. Section 5 gives the shape indexing methods that index shapes in several different ways. For quick and crisp searching of shape, a shape identification number is automatically generated and will be used as the primary index. For general shape retrieval, the combination of shape name and version is used as an alternative index. For flexible and vague shape retrieval, the combination of a set of shape descriptors is employed as another candidate index of shapes. Since the first indexing method is very common in database design and the second indexing method is conventional in engineering databases, we do not need to discuss these techniques in this paper. Instead, we will focus on the third shape indexing approach. In Section 6, a knowledge-base assisted shape querying method is discussed. A graphical user interface is constructed for natural-language-like query input and a graphical three-dimensional shape display window is employed for supporting query output. The SQL querying mechanism of a commercial database management system is employed as the underlying querying engine. In Section 7, the possibility-based fuzzy shape retrieval method is presented. A hierarchical shape matching approach that aims to speed up the fuzzy shape retrieval process is discussed. Finally, Section 8 provides the implementation and test results. Conclusions and future work are given in the last section.

2. Shape representation

Many representational schemes exist for modeling 3D shapes: Constructive Solid Geometry (CSG), Boundary Representation (B-Rep), sweeping, cellular decomposition (Bronsvoort, 1990; Mntyl, 1988), NURBS (Piegl and Tiller, 1997), and parametric and feature-based modeling (Shah and Mntyl, 1995). These modeling methods are based on precise representations of geometric objects, such as vertices, edges, surfaces as well as exact topology relationships between them. These methods are good at supporting the detail design process where all precise design details must be represented. However, they are not suitable for supporting conceptual design which is inherently qualitative and uncertain.

Qualitative shape models such as *geons* have been used for object recognition in the computer vision area (Biederman, 1987; Dickinson et al., 1993; Dickinson, 1994) because they are suitable for dealing with imprecise data or lack of data. However, qualitative models cannot provide sufficient quantitative information that is required for shape specification and manipulation in CAD/CAM (Computer-Aided Design and Manufacturing) systems. Deformable superquadrics are composed of basic super-ellipsoids and their deformations. Basic shape parameters and deformation parameters provide precise quantitative information about a shape. Since the linguistic descriptions of a shape can be linked to shape parameters directly, deformable superquadrics also provide qualitative information. The properties of being both quantitative and qualitative makes superquadrics the ideal candidate for bridging qualitative conceptual shape design and detail design. These properties also make it possible for us to introduce fuzzy set approach into a CAD system, where fuzzy sets are used to represent design intents formulated by descriptive terms and shape parameters are used to construct a solid model. We have therefore chosen to use superquadrics as the basic representation for 3D shapes, where a shape is represented by a set of shape descriptors and a set of shape parameters. The descriptors are the semantic features of a shape such as *roundness or bendiness*. The parameters are the geometric elements of a shape that are used to define a shape geometrically. The following subsections explain these shape parameters and descriptors in more detail.

2.1. Geometric representation: deformable superquadrics

Superquadrics may be expressed by an implicit equation:

$$\left(\left|\frac{x}{a_1}\right|^{2/\varepsilon_2} + \left|\frac{x}{a_2}\right|^{2/\varepsilon_2}\right)^{\varepsilon_2/\varepsilon_1} + \left|\frac{z}{a_3}\right|^{2/\varepsilon_1} = 1 \tag{1}$$

where, ε_1, ε_2 are shape parameters which control the shape roundness and squareness along the north-south direction and the west-east direction, respectively. For example, if $a_1 = a_2 = a_3 = 1$, the shape is a cube when $\varepsilon_1 = \varepsilon_2 \to 0$, and the shape is a sphere when $\varepsilon_1 = \varepsilon_2 = 1$. When ε_1 and ε_2 change from nearly 0 to 1, the shape change continuously from a cube to a sphere. a_1,a_2 and a_3 are scalar parameters which represent the length of a shape along the x, y and z axes, respectively. Since the shape description of an object does not relate to its size, we let $a_1 = 1$ and a_2, a_3 represent the ratio of a_2/a_1 and a_2/a_1. Thus a_2 and a_3 control the relative dimension of the two cross sections of a superquadric shape. For example, when $\varepsilon_1 = \varepsilon_2 = 1$, if $a_2 = a_3 = 1$, the shape is a sphere, otherwise it is an ellipsoid.

Several deformation parameters, k_x, k_y, k_v and t, are employed to control the deformation quality tapering, bending and twisting. The parameters k_x and k_y control the tapering property. When $k_x = k_y = 0$, there is no tapering, while when $k_x = k_y = 1$, the shape is extremely sharp (one end is reduced to a point). The parameter k_v controls the bending property. When $k_v = 0$, the shape is not bent at all while when $k_v > 0$, the shape can change from slightly bent to extremely bent. The parameter t controls the twisting property. When $t = 0$, shape is not twisted at all while when $t > 0$, shape can change from slightly twisted to extremely twisted. With the incorporation of the four deformation parameters into the original superquadrics, a deformable superquadric shape can be represented by eight shape parameters:

$$\{\varepsilon_1, \varepsilon_2, a_2, a_3, k_x, k_y, k_v, t\}.$$

For further information on deformable superquadrics, please refer to Barr (1981, 1984, 1992), Pham and Zhang (2000).

The small number of parameters makes deformable superquadrics easy to control. However, it is still difficult for designers to create desired shapes using these parameters directly because this requires a good understanding of the underlying mathematical model which general users do not possess. In addition, some of the parameters have not direct engineering meaning and the relationship between these parameters and shapes is non-linear, hence the process of generating shapes by trial-and-error is tedious. On the other hand, designers often have some vague ideas of geometric shapes in mind and wish to obtain such a shape quickly. An effective supporting tool should allow them to express their intent in a natural and semantic way. Therefore, we proposed to use a set of shape descriptors to represent shape and link them to shape parameters through a fuzzy shape specification system (Pham and Zhang, 2000). We discuss this perceptual representation of shape in Section 2.2.

2.2. Possibility-based perceptual representation: fuzzy shape descriptors

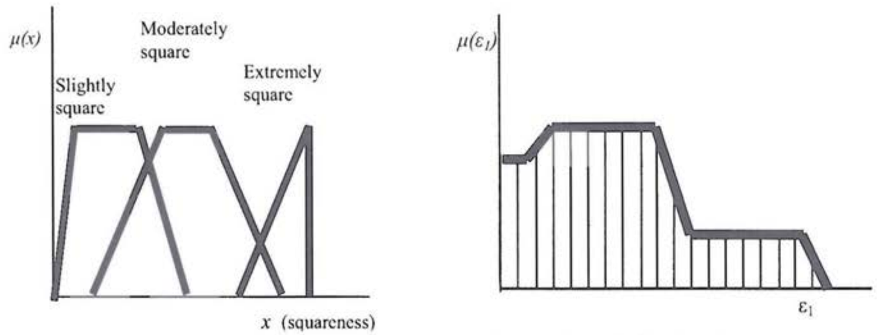
Most of the shapes used in design are geometric shapes which can be defined by mathematical equations and their combinations. However, it is more convenient and intuitive for designers to represent shapes using common shape descriptors, especially in the conceptual design stage where rough expression is sufficient. Since the human ability for shape recognition is only approximate, it is not necessary to depend on exact quantitative details to make judgements. Furthermore, such judgments are slow and error prone. Instead, in a previous paper (Pham and Zhang, 2000), we use fuzzy concepts to describe the global characteristics of shapes because they are more natural. To do so, we first collected a set of words, such as *round, square, cylindrical, ellipsoidal, bent, sharp, twisted* and *pinched* as well as a set of linguistic hedges, such as *extremely, very, moderately* etc. to represent the description of a shape. A set of shape descriptors, including *roundness, squareness, bevel-ness, pinchness, flatness, taperness, bendness, twistness,* and *shearness*, were used to describe a shape.

A fuzzy set, which is characterized by a membership function, is a mapping from the universe of discourse (or reference set) to the interval $[0, 1]$ (Kruse et al., 1994). In other words, a fuzzy set is a set that allows its members to have partial degrees of membership. There are a large variety of different interpretations of fuzzy sets in the literature such as possibility functions, similarity-based functions, or preference functions (Ruspini et al., 1998).

A possibility distribution is a mapping from the universe of discourse (or reference set) to the unit interval $[0, 1]$, where at least one element has the grade 1 (Kruse et al., 1994). Possibility can be interpreted physically to represent preference: the most feasible ones are usually preferred. It can also be interpreted in an epistemic way to represent the consistency of a datum with the available information (Ruspini et al., 1998).

Possibility theory is related to but independent of fuzzy sets because it can be derived with or without reference to fuzzy sets. Normal fuzzy sets and possibility distributions may have the same representation but with different interpretations. The grade of membership expresses the extent to which a well-known value in the universe belongs to an ill-defined set. The degree of possibility refers to the strength of a value to be the actual value given an ill-defined set. When a fuzzy set is used to represent the uncertain information about the value of a single-valued variable, the degree attached to a value means the possibility that this value is the actual value of this variable. The fuzzy set is then interpreted as a possibility distribution, which expresses the degrees of plausibility or preference of the possible values of the ill known variable.

Zadeh (1978) suggested using a fuzzy set to convey the meaning of a concept such as *tall* or *old*. The use of fuzzy sets to represent the value of a variable induces a possibility distribution. In imprecise conceptual shape design, a fuzzy set can be used to convey the meaning of a linguistic concept such as *extremely* *square*. Thus, the value of the variable of a shape can be represented by a possibility distribution induced by this description in a predefined universe of discourse. In this context, a fuzzy set is equated with a possibility distribution. For example, *squareness* can be *extremely square, moderately square or slightly square* in the universe of discourse of $[0, 3]$ and corresponding possibility distributions are shown in Fig. 1a. The grade of membership function represents the degree of a value being in a fuzzy set, but also the degree of possibility with which the variable takes this value. The value of each shape parameter is also a possibility distribution (Fig. 1b) which is inferred from several shape descriptors through a fuzzy shape specification system, where the relationship between shape descriptors and shape parameters is represented by a set of rules. For example, if *squareness* is *extremely* square then ε_1 is *nearly zero*. An experimentation of shape description was carried out to investigate how people perceive and describe shape characteristics. The resultant data was used to construct fuzzy membership functions for these shape descriptors and the inference rules that link these descriptors to the values of the shape parameters. More details on this shape specification system may be found in Pham and Zhang (2000).



a) Membership functions for a shape descriptor squareness

b) Membership function for a shape parameter ε_1

Figure 1. Membership functions for shape descriptors and shape parameters

3. System architecture

To efficiently support 3D shape storage and retrieval, the fuzzy shape database consists of the following components:

1. A Graphical User Interface (GUI) that is used to input a user's query. The commonly used shape descriptive terms are displayed in the GUI and

users need only to click on the buttons to formulate simple or composite shape queries.

2. A translator that is used to generate valid, standard queries. This translator is required because initial queries need to be intuitive and easy to understand for the user but they cannot be used to retrieve shape directly since not all descriptors used in the GUI level are stored in the database. For compactness and uniqueness, a set of standard shape descriptors is chosen to code the shape and stored in the database. The intuitive queries must be translated into standard queries before the querying process is performed.
3. A fuzzy predicate evaluation module which handles fuzzy data manipulation. The descriptive terms inputted by users are associated with their granule meaning through a fuzzy predicate library. When a new fuzzy shape needs to be inserted into the database, a fuzzy data closeness measure, which represents the similarity of two shapes, is used to check data redundancy. The possibility and certainty degrees are employed to perform hierarchical data retrieving.
4. An existing relational database which provides the basis for the implementation of this fuzzy database. It provides the basic DBMS resources and some SQL program tools for user's application. All information about a shape, including shape identifier, name, version, perceptual descriptors and geometric parameters as well as membership functions associated with fuzzy attributes, are stored in the shape database.
5. A rendering software which is used to display the 3D shapes retrieved from the shape database. As a fuzzy shape represents a set of shapes that have similar properties, a multi-viewed window is needed to display several typical shapes of a fuzzy shape.

Fig. 2 shows the basic structure of the fuzzy shape database.

4. Conceptual modelling of the fuzzy shape database

The Entity-Relationship (ER) model has been widely accepted for conceptual database design because of its ease of use and that the entities and relationships are natural concepts in the real world (McFadden et al., 1999). Therefore, the *Fuzzy Entity-Relationship* model (Chen, 1998), which incorporates fuzzy set theory to the basic ER concepts, is used to represent the conceptual structure of the fuzzy shape database. Since fuzzy shape descriptors and parameters are represented by possibility distributions, we chose to use a possibility-based fuzzy database model. In database design, the fuzziness can happen at three levels (Chen, 1999; Petry and Bosc, 1996). The first level is the fuzzy conceptual model which results in fuzzy semantic objects including fuzzy entity sets, fuzzy relationship sets, and fuzzy attribute sets. Fuzziness in this level may arise during database design and may be solved by designers before database implementation. The second level concerns the occurrences of entities and re-

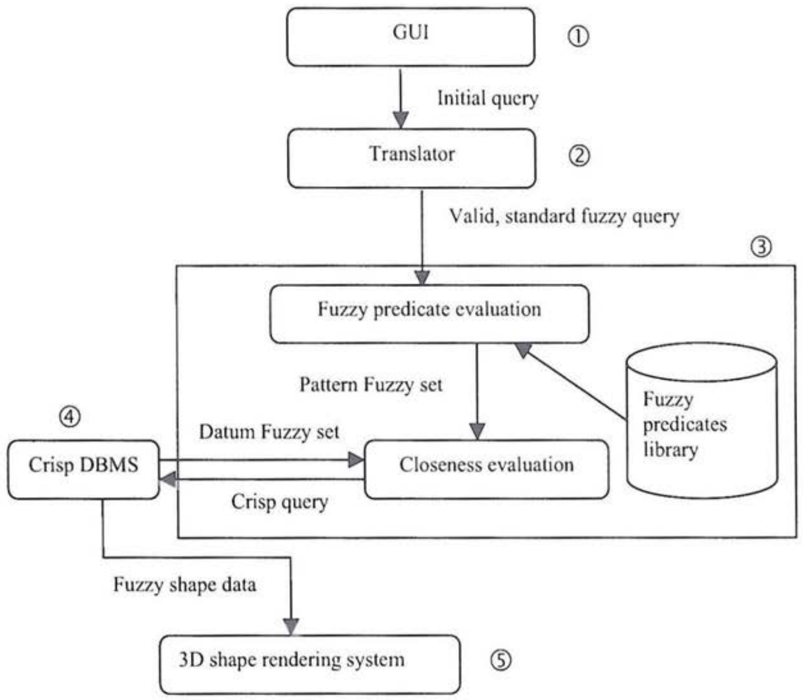


Figure 2. Fuzzy Database Structure

lationships. The third level deals with fuzzy attribute values. The fuzzy shape database has only fuzzy attribute values and all entities and relationships are ordinary. Hence, we consider the conceptual modeling process of the fuzzy relational shape database to be similar to that of a conventional relational database because we can consider the fuzzy attributes to be ordinary except for the fuzzy representation of their values.

4.1. Schema design

A shape is described by a set of shape descriptors and represented by a set of shape parameters. It also has a name, version and an underlying identifier. All shape descriptors are fuzzy terms and corresponding certainty levels. Each fuzzy term has a label and an associated fuzzy set represented by a possibility distribution. All parameters also possess fuzzy set values. Although shape parameters can be derived from shape descriptors, we store them explicitly in the database because they are used as design solutions. As a shape has many descriptors and parameters, the detailed E-R model is very big, hence we present here only an abstract E-R model in which Di and Pi represent fuzzy descriptors and parameters respectively (Fig.3).

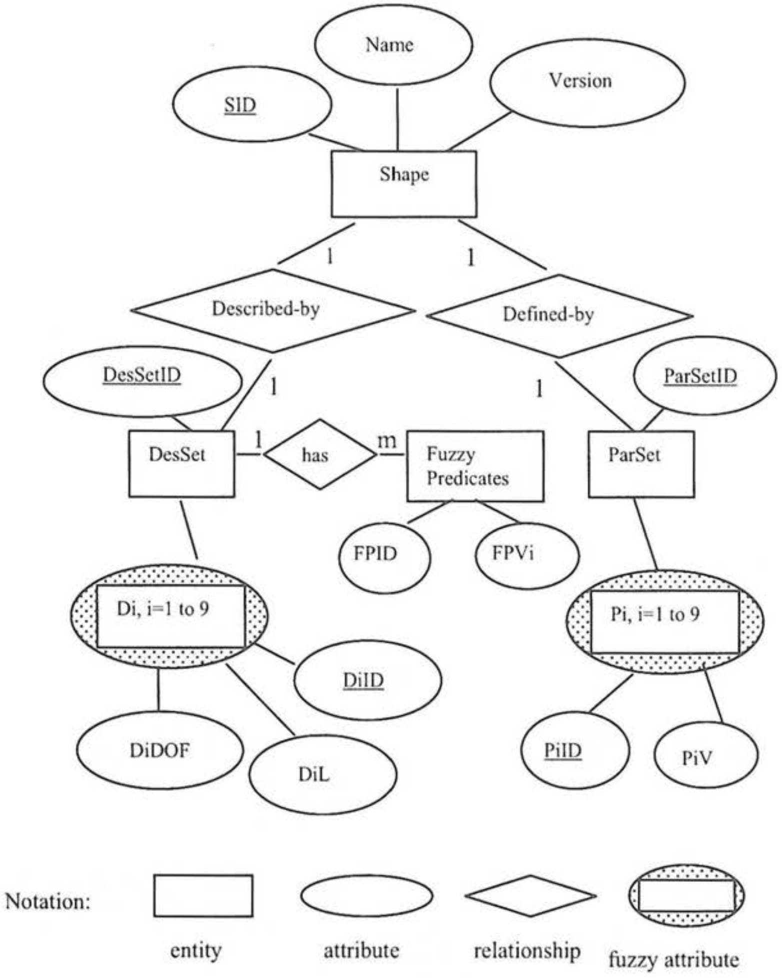
In Fig. 3, a shape has four attributes: the shape identifier SID, the name and corresponding version number of a shape as well as the identifier of the set of descriptors. The version of a shape, is introduced here to facilitate the gradual refining process of the same design object. It is also useful when a designer prefers to assign the same name to a design object but has different design solutions for it. DesSet stands for Description Set which has DesSetID as identifier and a list of fuzzy descriptors. ParSet is a Parameter Set which has ParSetID as identifier and a list of fuzzy parameters. Each descriptor or parameter is an attribute of the shape with a value expressed as a possibility distribution. Such an attribute is called a fuzzy attribute. For each descriptor Di, DiID is its identifier, DiLabel is the label of a fuzzy predicate, and DiDOF is the preferred threshold. For each parameter Pi, PiID is the identifier of a parameter and PiValue is the corresponding possibility distribution. The Fuzzy Predicates are used to represent the distribution parameters of fuzzy shape descriptive terms such as extremely bent. Each fuzzy predicate has its identifier FPID and corresponding fuzzy distribution parameters FPValue. We represent DesSet in a separate table to avoid the table Shape being too big. Since ParSet is dependent on DesSet, it is also saved in a separate table.

4.2. Data types

A key aspect of a fuzzy relational database is that the domain values need not be atomic. In a possibility-based approach, the available information about the value of a single-valued attribute A for a tuple t is represented by a possibility distribution $\Pi_{A(t)}$. The universe of discourse of this distribution is $D\bigcup\{e\}$, where D is the domain of attribute A and e is an extra element which means the attribute does not apply to tuple t. The possibility distribution is a set of the possible and mutually exclusive values of $A(t)$ in $D\bigcup\{e\}$ with possibilities between 0 and 1.

The possibility distribution representation of data provides a unified framework to deal with precise and imprecise values. A precise value can be represented as a possibility distribution with the membership grade 1 for one crisp element and the membership grades zero for all other elements. Since the storage space required for an imprecise value is much more than a precise value, we still use different data types in the fuzzy shape database for different attributes depending on the real values they may take. For example, attributes that might have fuzzy values are represented by possibility distributions no matter whether their real values are crisp or fuzzy, while attributes that can only take precise values are represented by the precise data type.

Since the shape database is limited to the specific application domain, only



(Note: the fuzzy attribute notation *is inspired by the fuzzy attribute notation in fuzzy ExIFO model [Yazici and George, 1999].)*

Figure 3. Abstract fuzzy E-R model of the shape database

three types of data are involved. The first type is the *precise value* such as shape identifier $ID = 20$. The second type is the *fuzzy set value* without label such as shape parameter $p1 = \{1/0.2, 1.2/0.8, 1.6/1.0, 1.8/0.4\}$. The third type is the *mixed fuzzy notion and fuzzy set value* such as the descriptor *roundness = extremely round with membership grade over 0.9*, where *extremely round* is the label of a membership function. The third type is essentially the same as the second, i.e., fuzzy set values represented by possibility distributions. We treated it as a separate data type in practice for efficiency.

5. Fuzzy shape indexing

Database indexing is the study of data structures to allow for efficient search and retrieval of a collection of data. The choice of an index depends on the nature of the data and the expected query types. The shape indexing and retrieval approaches can enhance the efficiency of geometric searching and have been explored mainly in image processing area (Lu, 1999). However, only few attempts have been made in CAD systems (Cybenko et al., 1997; Kriegel, 1993; Kriegel et al., 2001; M. Hardwick et al., 2000; McWherter et al., 2001). Existing shape indexing approaches are based on precise shape representation and cannot facilitate fuzzy shape indexing and retrieval. We therefore aim to develop a shape indexing approach based on the proposed fuzzy shape representation.

In a conventional database, the data item is self-indexed. For example, the condition "$sid = 10$" can find the tuple whose attribute *sid* is 10 if this tuple exists. Therefore, it is quite easy to find a desired tuple which satisfies the search condition by Boolean matching. However, a fuzzy value cannot index itself because the index must be suitable for all elements in the fuzzy set. For example, the fuzzy predicate *"roundness = extremely round"* cannot index a shape which is *extremely round.* A Degree of Fulfilment (DOF) of a user to the predicate *"roundness = extremely round"* must be used. If the datum is represented as a fuzzy set in the database, only one DOF is not enough to retrieve a tuple. In this case, two degrees, possibility degree and necessity degree can be used along with the predicate to retrieve a tuple. Hence, the indexing task in fuzzy database is a severe problem. Usually an additional attribute is employed as identifier (Petry and Bosc, 1996).

Yazici and Cibiceli (1999) utilized the multi-level grid file approach to develop an access structure for similarity-based fuzzy databases. In this approach, the domain of an attribute with fuzzy values is partitioned into several regions related to predefined fuzzy predicates. Each predicate has an identification code. Then, the region of each predicate is further divided into three parts: left, middle and right. This partition process can be performed recursively until the desired granule level is reached. Each sub-region has a unique local code, which is a bit pattern. The code of each sub-region is the concatenation of its parents' code and its own local code. An additional bit is used to identify if an item is a fuzzy or crisp value. The original value is then associated with this bit pattern. Take the attribute of human height for example. Three predicates are used: short, middle and tall. Their bit codes are assumed to be 00, 01, 11. Then, the short predicate is divided into three parts: short left, short middle and short right with the local code 00, 01 and 11. Then, the index for short middle is 0001. This approach can index both crisp and fuzzy values. Hence, it is suitable for indexing a database which may have both crisp and fuzzy values for the same attribute such as the height of a person (heterogeneous attribute). However, fuzzy values need to be first defuzzified into crisp values in order to obtain a suitable code. That is to say, a fuzzy set is not indexed by its fuzzy value but by the crisp value of its defuzzification result. This makes the fuzzy values lose their fuzziness very early and may lead to the loss of the effectiveness of indexing. This indexing approach relies on the pre-partition of the attribute domain.

Bosc proposed some indexing principles for a possibility-based fuzzy database, where the value of an attribute is a fuzzy set represented by a possibility distribution (Bosc and Galibourg, 1989). This approach works as follows:

- The indexing of a tuple is not by the fuzzy values themselves but by their *support* or *core*. The support of a fuzzy set is the crisp set of all elements in the universe of discourse with nonzero membership grades and the core of a fuzzy set is the crisp set of all elements in the universe of discourse with membership grade(s) one (1). For example, if a fuzzy set is represented as a trapezoid and the vertices (a, b, c, d) are from left to right, the support bound is $[a, d]$ and the core bound is $[b, c]$. The four bound values, including the lower and upper bounds of the support and core of the possibility distribution, are used to index the fuzzy set.
- A two-step searching is employed for fuzzy selection. Firstly, a subset of initial relations is determined by comparing the support and core bounds of the datum set and the condition set; secondly, a measure indicating the minimum degree to which a datum possibly or necessarily satisfies a condition is computed over the reduced relation and compared to a predefined threshold. Thus, a fuzzy selection is converted to a Boolean multi-key searching.

Since the calculation of possibility and necessity degree requires that the fuzzy sets of interest have bounded supports, the Bosc's indexing structure is suitable for databases where the values for the same attribute are of the same type (homogeneous) and the supports of all fuzzy sets are closed intervals. As the Bosc's indexing approach uses only the supports of fuzzy sets and the possibility and/or necessity degrees for indexing, it does not rely on the pre-partition of the attribute domain.

In the fuzzy shape database, the values of all shape descriptors are fuzzy real numbers within finite domains, hence they are homogeneous. The "homogeneity" restriction of the Bosc's indexing approach is trivial in this case. On the other hand, it is difficult to pre-partition the fuzzy attribute domain into a certain granule level. Therefore, the Bosc's indexing approach is chosen for fuzzy shape indexing and retrieval according to shape descriptors. However, it is used as the secondary index because the searching process based on fuzzy sets is much slower than searching a single value. An integer number that is automatically generated by the database management system is employed as the primary index for efficiency.

5.1. Primary index

Since searching through shape descriptors that are represented by fuzzy set is relatively slow, we provide the shape ID which is automatically generated by the conventional DBMS as the primary index. Frequent users may search a shape using this number. We also provide the combination of name and version as an alternative index of shape. This is the conventional indexing method used in current CAD systems. These two indexing methods are implemented by crisp searching and relevant techniques that can be found in many database textbooks. Since we aim to introduce the fuzzy set approach into CAD systems in order to support the conceptual design, the semantic searching of shape through fuzzy descriptors is essential. Hence an indexing method based on fuzzy descriptors is discussed in the next section.

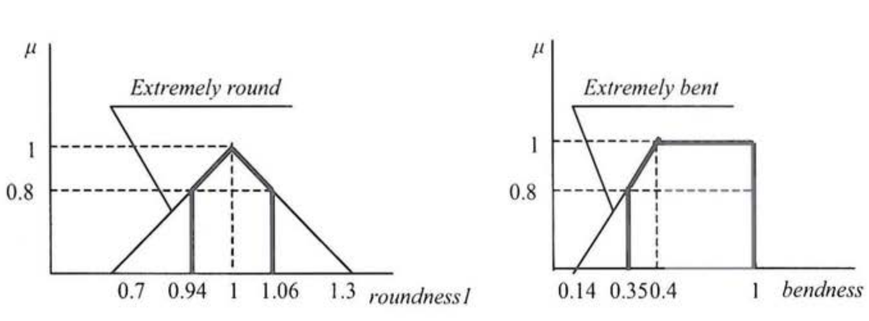
5.2. Fuzzy descriptor index

The shape parameters (P) are derived from shape descriptors (D) and corresponding Degrees of Fulfilment to predefined membership functions (*DOF*) as well as the Fuzzy Inference System (*FIS*) in a shape specification system. Hence the shape parameters can be denoted as $P = F(D, DOF, FIS)$. Once the Fuzzy Inference System is generated, we consider it as fixed. The shape parameters are determined only by the shape descriptors and corresponding thresholds. So, it can be denoted as $P = F(D, DOF)$. For the same set of descriptors and *DOF*s, the set of fuzzy parameters will be the same. Therefore, the descriptors and *DOF*s can be used to index a fuzzy shape. When retrieving a shape, the descriptors and corresponding *DOF* levels can identify the appropriate tuples in the fuzzy shape database. Since the shape descriptors people use are numerous and not all of them can be applied to index a shape, a primary set of shape descriptors is selected for indexing. Other descriptors will be mapped to these basic descriptors or their combinations through a translator.

The primary set of shape descriptors includes:

- *round-ness, square-ness, bevel-ness, pinch-ness* and *flat-ness* in north-south direction and east-west direction of superquadrics.
- *taper-ness* (along X and Y axis)
- *bend-ness* (along Z axis)
- *twist-ness* (around Z axis)
- *shear-ness* (along X axis)

Although the expressive ability of these primary shape descriptors is still very limited, they can describe all the shapes currently representable in our superquadrics-based 3D shape modelling system. The perceptual shape description is the media for fuzzy shape representation, specification, storage and retrieval. For example, we assume that in the conceptual design stage, the designer intends to have a shape which is ellipsoidal and extremely bent with satisfaction degree 0.8. This specification implies that the descriptor *roundness1* (roundness in east-west direction of a superquadric shape) has a fuzzy value *extremely round with satisfaction degree* 0.8 and the descriptor *bendness* has a fuzzy value *extremely bent with satisfaction degree* 0.8. There will be a set of crisp shapes that can fulfill this specification with different certainty levels. We call this set of crisp shapes a fuzzy shape. This fuzzy shape can be indexed and retrieved according to the shape descriptors and their linguistic values. We assume that the membership function of the fuzzy predicate *extremely round* is a triangle characterized by points $\{(0.7, 0), (1, 1), (1.3, 0)\}$ in the universe of $[0, 3]$. The membership function of the fuzzy predicate *extremely bent* is a piece-wise linear function characterized by points $\{(0.14, 0), (0.4, 1), (1, 1), (1, 0)\}$ in the universe of $[0, 1]$. Then the membership function of the descriptors *roundness1* and *bendness* are derived from the two fuzzy predicates and the satisfaction degree (bold lines in Fig. 4). The supports of the descriptors *roundness1* and *bendness* are $[0.94, 1.06]$ and $[0.35, 1]$ respectively. These lower and upper bound values of supports are used as indices of the corresponding shape descriptors. They can be used for filtering out many irrelevant tuples in the shape database during shape retrieval which will be discussed in a later section.



a) Membership function for roundness1. b) Membership function for bendness.

Figure 4. Support of shape descriptors

In addition to the primary shape descriptors which are used for shape indexing, we also employ a set of secondary shape descriptors at the user interface level. A list of commonly used descriptive terms, including 3D geometric primitives, geometric descriptors and linguistic hedges, is extracted from several references. The secondary shape descriptors, which are currently used in the system are geometric shape descriptors such as *cubical, conical, spherical, cylindrical* and *ellipsoidal.* The secondary shape descriptors are mapped to the combinations of primary shape descriptors through a translator.

6. Fuzzy shape queries

Querying and retrieving of data from a database is an important activity. Fuzzy querying allows users to formulate queries using linguistic words, hence it is more flexible than crisp querying. In addition, fuzzy queries produce naturally ranked results whereas conventional queries bring back only undifferentiated tuples. Fuzzy queries may also provide reasonable answers where crisp queries fail to find solutions. When querying the shape database, a user can formulate a query by searching the shape name and version. Frequent users may also query shapes using the shape ID, which is automatically generated by the computer. However, it is sometimes desirable to search shapes using natural-language-like shape descriptors, especially for occasional users. It is also desirable to allow users to express preferences and thus make the querying results more feasible. Using vague predicates represented by fuzzy sets to perform a query is one approach to achieve the above targets.

6.1. Query requirement analysis

Query requirement analysis is based on the task the query will complete and the type of users. The aim of querying a shape database is to obtain the appropriate shape. People usually perceive a shape by commonly used regular shape names, by pictures or by shape description. For example, for a cubical shape, people will describe it as *cube, cuboid, cubic, cubical,* or *square.* Hence, the system should allow querying by shape descriptions. If the shape number of this cube in the shape database is 10 and the user knows this number, s/he may just ask "list out the 10^{th} shape". The query system should respect the diversity of querying, therefore multiple query options should be provided.

The users can be classified into primary users who use the system regularly and secondary users who use the system only casually. The query system should allow primary users to input their query as quickly as possible. This is achieved by querying the database using shape ID or name. For novice or casual users, the system should provide effective guidance. The Query-By-Example method and intelligent query assistant are usually employed to guide the query process. In the shape database, the underlying shape descriptors represent the geometric characteristics of shapes along different directions based on the shape representation approach. Although they are represented by words and can be understood by professional users, it is still hard for general users to understand

and formulate shape queries using these descriptors. Hence, a Graphical User Interface is utilised to help users to formulate query. Commonly used shape descriptors are displayed in the GUI and they will be translated into underlying basic shape descriptors through a translator. In addition, the support messages, such as on-line help and error messages should be provided.

6.2. Categories of queries

The shapes are classified into two classes: fuzzy shapes and crisp shapes. The main difference between these two kinds of shapes is the data associated with the shape parameters. For a crisp shape, each parameter has only one value whereas for a fuzzy shape, each parameter has a fuzzy set value. In the case of fuzzy query on crisp shapes, each tuple will be assigned a Degree of Fulfilment (DOF) to the fuzzy condition. In the case of fuzzy query on fuzzy shapes, the similar method as fuzzy query on crisp shapes can be employed but the calculation method for DOF is different and two DOFs are needed. In the latter case, we use the possibility and necessity degrees to measure the extent to which a datum satisfies a condition.

In a fuzzy database, the crisp data and the fuzzy data can be represented uniformly by fuzzy sets, and a crisp value is only a special case of a fuzzy value where the membership grade is one for a crisp element and zero for all others (Bosc and Galibourg, 1989). Since the fuzzy shape database is mainly used for storing and retrieving initial fuzzy shapes that have fuzzy set values, hereafter we consider fuzzy queries on fuzzy data only. The possibility/necessity measures will be used to represent the upper and lower bounds of the satisfaction degree of a fuzzy datum with respect to a fuzzy condition.

The categories of fuzzy queries can be further classified into the following classes: *simple query* and *combined query*. A simple query refers to a query by a single condition. For example, the user inputs a single shape descriptor such as *extremely round* and a series of shapes will be retrieved from the database and will be displayed on the screen in multiple views. A combined query refers to a query composed of multiple descriptions. For example, a user inputs a combination of shape descriptions, such as *extremely round* and *slightly bevel*, and a series of shapes will be retrieved from the database and will be displayed on the screen.

Shape description combination can be classified into *feasible combination* and *infeasible combination*. Feasible combination means that two descriptors on the two sides of an AND operator can be used to describe the same shape at the same time. For example, the descriptors *extremely cylindrical* and *slightly bent* can exist at the same time because they describe a shape that is a slightly bent cylinder. Infeasible combination means that two descriptors on the two sides of an AND operator cannot be used to describe the same shape at the same time. For example, a *cylindrical* shape cannot be *pyramidal*. The feasible combination can be passed to the inference engine for deriving the result values.

The infeasible combinations will be checked out by the system according to a constraint table and the user will be asked to reformulate another query.

6.3. Query input and generation

A conventional database query language, such as SQL, requires of users to understand the logical database schema, relational algebra and the query language itself to formulate a query. The popular QBE (Query-by-Example) avoids using a special query language yet still makes the DBMS users to work on the table level. Therefore, it is very difficult for novice users to communicate their needs to DBMS. Integrating artificial intelligence techniques and database techniques in order to support novice users to access database has become increasingly popular (Wu et al., 1996). Since the underlying shape representation is difficult to understand for general shape designers or users, it is extremely difficult for them to query the shape database through SQL or QBE directly. It is desirable to provide a mechanism to allow users to perform a query according to commonly used shape descriptors, hence it is necessary to construct a friendly and intelligent database query front-end. By intelligent query, we mainly mean the following:

a) Allowing end-users to formulate database queries using natural-language-like descriptive terms with fuzzy internal meaning. Although a completely natural language interface is most desirable, automatic interpretation of a natural language such as English is very difficult because of its complexity and ambiguity. In addition, it is impossible for us to obtain the fuzzy meaning for all words even if they are limited to shape description, hence only a selected set of shape descriptors such as *very round* are allowed in the menu-based GUI (Graphical User Interface).

b) Having knowledge and reasoning ability for formulating conventional queries according to general inquiries. A set of primary shape descriptors is selected and associated with corresponding fuzzy sets. A set of secondary shape descriptors is also used in shape query and they are mapped to the combinations of the primary shape descriptors through a translator.

c) Providing end-users with substantial guidance in query formulation by performing validation checking and automatically applying some constraints as users select their query from GUI.

General intelligent query systems also generate cooperative responses for ill-formed queries and provide natural language explanation of the query result (Wu et al., 1996). A fuzzy query itself is a kind of intelligent query because it decreases the chances for the null answer and can provide ordered weighted answers naturally, so we will not design a separate cooperative answering system. As the query result is a fuzzy 3D shape that is displayed on the screen as response to a query, whether the query result is correct or not is intuitive. Hence we do not need to translate the query results back to natural language-like explanations.

When constructing GUI, we constrain the user's query to a restricted linguistic domain, that is, the commonly used shape description words. The users can input their query in the query window through menu selection. All available shape descriptors and corresponding linguistic hedges are listed in the query window. The default logic operator is AND. Users need only to click on the check box to formulate query.

Once the query is formulated, a constraint table (Table 1) of description combinations is employed to perform validity checking for combined queries through a validation-checking module. The valid general query is passed to the translator and translated into a combination of internal (primary) descriptors according to the mapping rules that are derived from the relationship table of user descriptors and internal descriptors (Table 2). For example, *IF shape IS cylindrical THEN shape in north-south direction IS extremely square and shape in east-west direction IS extremely round.*

Table 1. Constrainst of shape description combinations

	Square	Round	Cylindrical	Ellipsoidal	Conical	Tapered	Bent
Square	Y	Y	N	N	N	Y	Y
Round		Y	N	Y	N	Y	Y
Cylindrical			Y	N	N	Y	Y
Ellipsoidal				Y	N	Y	Y
Conical					Y	Y	Y
Tapered						Y	Y
Bent							Y

Note: Y - means that combination is accepted, N - that it is not.

Table 2. User descriptors-interal descriptors relationship

	Sq1	Rd1	Sq2	Rd2	Ob1	Ob2	Tp1	Tp2	Bt
Square	T		T		T	T			
Round		T		T	T	T			
Cylindrical	T			T	T				
Ellipsoidal		T		T	T				
Conical	T			T	T		T	T	
Tapered							T	(And/or)T	
Bent									T

Note: T for *True* means that for a commonly used shape descriptive term columns correspond to internal shape descriptors, the corresponding internal shape descriptor while rows to commonly used shape descriptors is true.

The main difference between this translator and a conventional knowledge base is that in a conventional knowledge base, the facts and rules are fixed and users perform different queries. Whereas in this translator, the rules and queries are fixed but the facts are changed according to the user's input. The output of this translator is a standard fuzzy query that is then passed to the fuzzy processing module to perform shape retrieval. The relevant techniques for fuzzy shape retrieval will be discussed in the next section.

7. Shape retrieval

Shape retrieval methods can be classified into two classes: content-based retrieval and annotation-based retrieval. Content-based retrieval concentrates on lower level shape features such as geometric parameters. Annotation-based retrieval is based on higher level semantic features such as perceptual descriptions, therefore it can support direct, natural queries. Annotation-based retrieval is complementary to content-based retrieval and depends on the conceptual data available. In a fuzzy shape specification system, the conceptual data such as shape descriptions is acquired in the design process and can be stored in the database. They are very important information for fuzzy shape retrieval. By using these annotations (or symbols), the user can query the database using higher level descriptors rather than the detailed shape parameters. The annotations used in shape database include the feature-based descriptive terms such as very round, extremely sharp, slightly bent etc. These terms will be used to help fuzzy shape retrieval. Unlike the annotation-based retrieval in conventional database which is based on alphabetic matching, the annotation-based retrieval in fuzzy database, is performed on the granule meanings of each annotation.

The fuzzy sets of geometric parameters are stored in the shape database but general users usually do not query the shape database by shape parameters. Instead, they query the shape database by shape descriptors. Shape descriptors and corresponding DOFs (thresholds) are also stored in the database but we cannot query them in a crisp way because every shape descriptor has a granule meaning. Hence, querying the shape database using shape descriptors should be based on their granule meaning. We propose to use a fuzzy predicate and a threshold (or DOF) to define a fuzzy set which is related to a shape descriptor. The DOF functions as a linguistic hedge that modifies a predefined fuzzy predicate in order to obtain a fuzzy datum set. The fuzzy set associated with a predicate used in fuzzy query acts as the condition set. The possibility and necessity degree proposed by Prade and Testemale (1984) can be used to perform shape retrieval. The query form *attribute* = *value* in relational database can be extended to *attribute* = *value with degree* θ, where θ can be possibility or necessity degree.

Given two fuzzy sets in the same universe of discourse, they can be compared according to the possibility and necessity measures. This comparison leads to two types of degrees: the possibility degree and the necessity degree, meaning to what extent two fuzzy sets possibly and necessarily match. The possibility degree Π represents the extent of the intersection between the pattern set and the datum set. It is the maximum membership value of the intersection set. The necessity degree N represents the extent of semantic matching of a pattern set for a given datum set. It is the minimum membership value of the union of the pattern set and the complement of the datum set. The interval defined by $[N, \Pi]$ represents the lower and upper bounds of the degree of matching between such pattern and datum sets. Since what is necessary must be possible, the possibility degree is always not less than the necessity degree. The proof of this property and the detailed formulas for calculating the possibility and necessity degrees can be found in Bosc and Galibourg (1989), Prade and Testemale (1984). Fig. 5 shows a pattern and a datum fuzzy set as well as the corresponding possibility and necessity matching degrees.

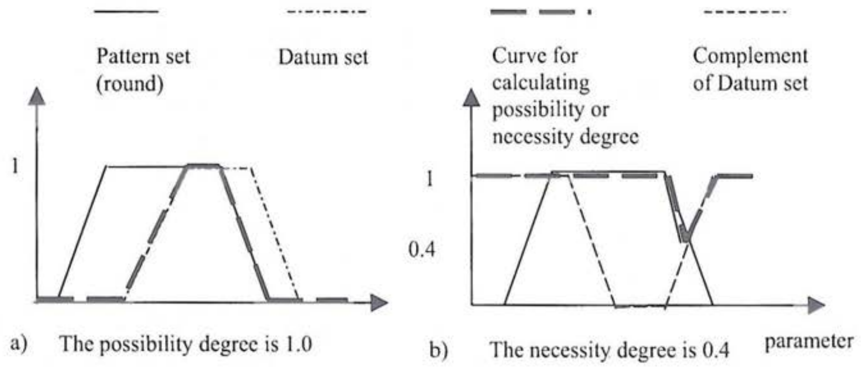


Figure 5. Possibility degree and necessity degree

A simple fuzzy query to a fuzzy database involves only one fuzzy condition and is performed by calculating the necessity and possibility degrees of a datum set to the pattern set. Compound condition involves the disjunction, conjunction, or negation of simple conditions. When the attribute values are logically independent (i.e., the value of one attribute does not rely on or is controlled by the values of other attributes), the overall necessity and possibility degrees of a tuple are the combination of elemental possibility and necessity degrees using min/max operator. The detailed equations that express the decomposability properties of possibility and necessity degrees can be found in Bosc and Galibourg (1989), Petry and Bosc (1996).

Our initial test on shape retrieval using possibility/necessity degrees shows that the possibility degree is too optimistic while the necessity degree is too pessimistic. For example, if the cores of the possibility distributions of two fuzzy sets have one common point, then the possibility degree is 1 even though the two compared sets are very different from each other. On the other hand, the necessity degree for a triangle fuzzy set A to lie in A is only 0.5. This makes the meaning of the entailment relation counter-intuitive and causes difficulties in shape retrieval. Hence, we use the possibility degree as a filter and compose another entailment measure called certainty degree for shape retrieval.

Given two fuzzy sets A and B in the same universe of discourse, the certainty degree to which B lies in A is the ratio of the cardinality of the intersection of these two fuzzy sets to the cardinality of B. The cardinality (or power) of a fuzzy set is the sigma-count (sum) of the membership degrees of all elements (Yager and Filev, 1994). We can see that the certainty degree is reflexive (the certainty degree to which A lies in A is 1.0) but not symmetric (the certainty degree to which A lies in B is not the same as that of B lies in A). The certainty degree is 1.0 if two fuzzy sets have exactly the same representation and 0.0 if their supports have no common region. In other cases, the certainty degree varies between 0.0 and 1.0. The overall certainty degree of a tuple over composite domains is the combination of the certainty degrees of elemental domains. In fuzzy retrieval, given a fuzzy condition and a fuzzy datum, if their certainty degree is higher than a predefined threshold, the datum satisfies the condition and will be retrieved otherwise it will be discarded.

Since the perception-based shape retrieval can support natural language querying, we use a set of primary shape descriptors for shape indexing and retrieving. The shape searching process is the matching process of fuzzy granule meanings of primary descriptors that are listed in the shape indexing section. We use a hierarchical matching process of fuzzy set to perform fuzzy shape retrieval. This hierarchical matching process comprises of the following steps:

1. Support-Core Matching. If the supports of two fuzzy sets are not intersected, they are not matched. This filtering process can cut off many irrelevant tuples.
2. Possibility Degree Matching. The possibility degree is calculated for the retained tuples from step 1). If this degree is lower than a predefined threshold, the matching process can stop.
3. Certainty Degree Matching. The certainty degree is calculated for all tuples retained from step 2).
4. ANDing and ORing: If the compared tuples have more than one fuzzy attribute, the conjunction and disjunction operation is performed to obtain the overall possibility and certainty degrees for all retained tuples. The final matching degrees are saved in the fields in the corresponding tables. This approach is the same as that used in fuzzy-relation-based framework in the sense that a matching degree is attached to each tuple.
5. Boolean selection based on the matching degree attached to each tuple.

As the large number of possible valuations of incomplete information, complex relational operations such as join operations applied to imprecise attributes are infeasible (Bosc et al., 2000). Hence, in the shape retrieval process, we perform only the selection/projection operations and join operations applied to precise attributes such as shape identity number. Actually we need to retrieve shape ID according to some criteria applied to some shape descriptors. For example, the possibility matching may look like this:

SELECT shape.shpID
FROM shape, Des1, Des2, Des3
WHERE (((Des1.poss)> 0.9*) AND ((Des2.poss)>* 0.9*) AND ((Des3.poss)>* 0.9*));*

where *poss* is the possibility degree to which a fuzzy datum satisfies a fuzzy condition. Once the shape ID is obtained, all shape data can be retrieved through this primary key. These data can be passed to a 3D shape rendering system to display the fuzzy shape.

8. Implementation and test results

In the actual implementation, we did not use a special data structure for imprecise data representation. Instead, we cast the fuzzy data modelling in a conventional crisp data modelling frame, that is, any fuzzy attribute with a fuzzy set value is represented as a set of ordinary attributes whose values range over the underlying distribution domain.

Since the domains of different attributes may be different but all normal membership functions have the same value range $[0, 1]$, we represent each fuzzy set using a family of α-cuts, so that all fuzzy sets have uniform representation. The α-cut of a fuzzy set is a crisp set of elements, for which membership grades are equal to or greater than α. Since a fuzzy set can be uniquely represented by a family of α-cuts (Klir and Yuan, 1995), we predefine a set of α values and each α-cut is characterised by two endpoints of the fuzzy set at this α-level. In some cases, the two endpoints at one α-level may converge to one point. Since the set of α levels are the same for all fuzzy sets, we can save it in a separate table and do not need to save it for each fuzzy set. Hence, all fuzzy sets can be represented by a set of α-level endpoints in the same representation scheme and have the same precision determined by the number of α-levels used. For example, if we partition the interval $[0, 1]$ into five equal intervals, the set of α level values may be $\alpha = \{0.0001, 0.2, 0.4, 0.6, 0.8, 1.0\}$. The support and core of a fuzzy set (when $\alpha \to 0$ and $\alpha = 1$ respectively) are also naturally represented in this representation scheme. Hence, this representation scheme is also consistent with the support-core indexing scheme for fuzzy set proposed in Bosc and Galibourg (1989) and discussed in Section 5. The family of α-cuts can represent any shape of fuzzy sets within a finite domain.

The development of a real Fuzzy Relational Management System is generally expensive and time consuming. However, using an existing system, we can concentrate on the conceptual and logical design of a database and need not spend much time on physical design (Bosc and Galibourg, 1989). Therefore, an existing popular commercial Relational Database Management System is employed as the basis of implementation in order to take advantage of some known access methods already in use. Implementing a fuzzy relational database in a conventional database allows us to incorporate fuzzy functions into existing relational databases, an effective way of developing a fuzzy database (Bosc and Galibourg, 1989).

The fuzzy shape database is mapped into a conventional relational database by storing the fuzzy set values in separate tables. The crisp and fuzzy shapes can be stored in one database because they can be represented uniformly using possibility distribution (currently we got only fuzzy shapes). Microsoft ACCESS is chosen as the implementation environment because of its popularity, easy availability and its friendly user interface as well as its convenience for communicating with other software packages.

Fuzzy sets provide the imprecise values of single-valued attributes in the shape database, so they are processed in a 'compact' form and users do not need to access their elements separately. Since fuzzy sets take complex data structures, we can also treat them as objects which have data and associated methods by building a top level on the current relational database systems to simulate the object features in an object-relational model.

A GUI is constructed within Microsoft ACCESS to provide a user-friendly interface to accept query input. A set of commonly used shape descriptors are displayed on the screen and are grouped into different option groups. Users need only to click appropriate shape descriptors. Once query conditions and an overall degree of fulfilment of all conditions are selected, users click *Get It* button and the system will perform querying and display the searched shapes on the screen.

Rules for query generation are stored in a knowledge base which along with an inference engine is called a translator. The initial query generated in query GUI is used as the input of the translator. The standard query is generated by the built-in backtracking facility of a PROLOG programming language, then it is sent back to the database to perform query through the fuzzy processing module. A hierarchical matching process based on possibility/certainty degree is employed to perform shape searching. Since the selection results are fuzzy shapes which have fuzzy set as parameter values, only typical crisp shapes are displayed on the screen in multiple views. Typical shapes can be obtained by defuzzifying shape parameters using typical defuzzification approaches such as minimum of maximum, maximum of maximum, centroid of area etc. More details on these approaches may be found in Berkan and Trubatch (1997) and many other books.

EXAMPLE 1 *A series of tests have been performed on this prototype system. Here we use an example to explain the usage of this shape database. To make the meaning clear and save space, we list out only the attribute values that need to be compared in this example. All fuzzy sets take piece-wise linear membership functions and are represented by a set of α-cut endpoints from left to right at the α levels* $\alpha = [0.0001, 0.2, 0.4, 0.6, 0.8, 1.0]$.

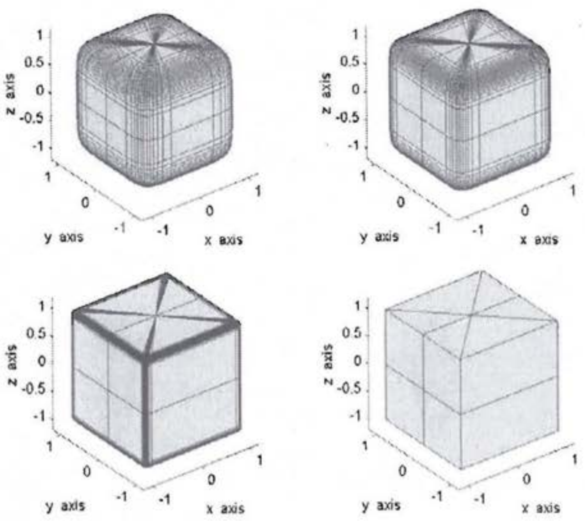


Figure 6. Typical shape elements of an extremely square fuzzy shape (Shp1)

We assume three fuzzy shapes are generated using the descriptive terms *extremely square with threshold 0.6*, *ellipsoidal with threshold 0.4*, and *ellipsoidal and extremely bent with threshold 0.8*. Typical shape elements of the three fuzzy shapes are shown in Figs. 6, 7, 8. The membership function of the fuzzy predicate *extremely round* is a triangle characterized by points $\{(0.7, 0), (1, 1), (1.3, 0)\}$. The membership function of the fuzzy predicate *extremely bent* is a piece-wise linear function characterized by points {(0.14, 0), (0.4, 1),(1, 1),(1, 0)}. The corresponding fuzzy set values of the descriptors Des1 *(roundness1)* and Des2 *(bendness)* derived from the above two fuzzy predicates are shown in Fig. 4 and in the form of family of α-cuts in Table 3.

Table 3 Fuzzy set values for shape descriptors

shpID	Des1 (bendness)	Des2 (roundness1)
Shp1	0 (not bent at all)	Φ (empty set)
Shp2	0 (not bent at all)	[0.82, 0.82, 0.82, 0.88, 0.94, 1, 1, 1.06, 1.12, 1.18, 1.18, 1.18]
Shp3	[0.35, 0.35, 0.35, 0.35, 0.35, 0.4, 0.4, 1, 1, 1, 1, 1]	[0.94, 0.94, 0.94, 0.94, 0.94, 1, 1, 1.06, 1.06, 1.06, 1.06, 1.06]

The fuzzy shape retrieval process is as follows.

1. Inputting descriptive words: *ellipsoidal, extremely bent* with threshold 0.6.

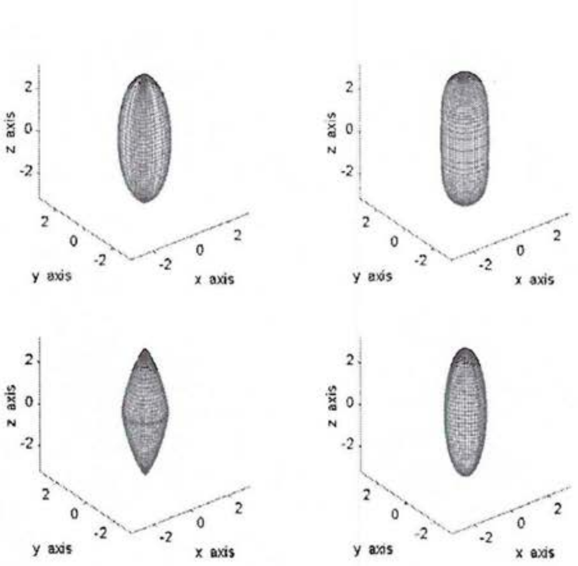


Figure 7. Typical shape elements of an ellipsoidal fuzzy shape (Shp2)

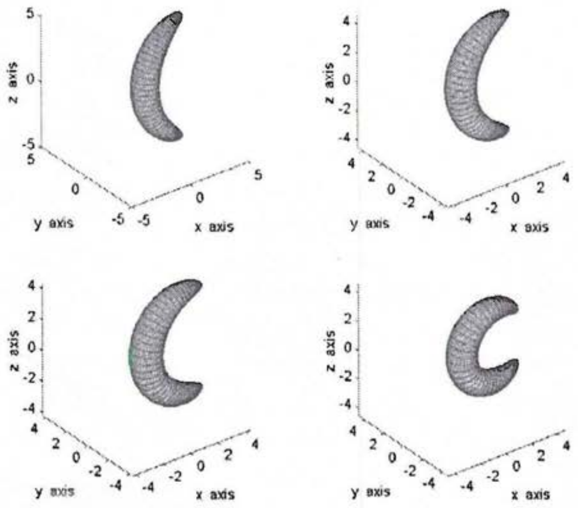


Figure 8. Typical shape elements of an ellipsoidal and extremely bent fuzzy shape (Shp3)

2. Translating input words into standard words.
   The word *ellipsoidal* is translated into *extremely round* in two directions, keeping the descriptive term *extremely bent* because it is a standard word.
3. Evaluating the input fuzzy words according to the threshold and the predefined fuzzy predicates.
   We use the descriptive words and corresponding thresholds, *extremely bent/0.6, extremely round/0.6* and the predefined membership function for the predicates *extremely bent* and *extremely round*, to derive pattern sets which are represented by family of α-cuts $P1 = [0.3, 0.3, 0.3, 0.3, 0.35, 0.4, 1, 1, 1, 1, 1, 1]$, and $P2 = [0.88, 0.88, 0.88, 0.88, 0.94, 1, 1, 1.06, 1.12, 1.12, 1.12, 1.12]$.
4. Retrieving data sets which are also represented by family of α-cuts from database. We obtain two fuzzy data sets for shape Shp3: $D1 = [0.35, 0.35, 0.35, 0.35, 0.35, 0.4, 0.4, 1, 1, 1, 1, 1]$, and $D2 = [0.94, 0.94, 0.94, 0.94, 0.94, 1, 1, 1.06, 1.06, 1.06, 1.06, 1.06]$.
5. Comparing the supports and cores of corresponding data set and pattern set.
   We first compare the supports of the data sets and the cores of the corresponding pattern sets. If all supports of the data sets fall in the cores of the pattern sets, then the data sets definitely lie in the pattern sets and the corresponding tuple can be retrieved. The core of the pattern set $P2$ is $(1, 1)$ and the support of the datum set $D2$ is $(0.94, 1.06)$, so the support of the datum set is not within the core of the pattern set and further calculation is needed. Then we compare the support of the pattern set $P2, (0.88, 1.12)$, and that of the datum set $D2, (0.94, 1.06)$. We can see that they intersect. We can also see that the support of the pattern set $P1$ and the datum set $D1$ also overlap. Hence, the tuple containing the attribute value $(D1, D2)$ possibly satisfy the condition $(P1, P2)$ and further calculations are needed. We can easily see that not all supports of the fuzzy sets in Shp1 and Shp2 intersect with those of the corresponding pattern sets, so that these two shapes cannot satisfy the query conditions and are discarded in this step.
6. Calculating the possibility and certainty degree.
   If the possibility degree is lower than a required threshold then the certainty degree need not be calculated, otherwise calculate the certainty degree. The possibility degree of $P1$ and $D1$ is $poss1 = poss(P1, D1) = 1.0$, the corresponding certainty degree of these two sets is $sd1 = sd(P1, D1) = 1.0$. Accordingly, we can obtain the possibility degree of $P2$ and $D2$, $poss2 = 1.0$ and the corresponding certainty degree $sd2 = 1.0$. Once the possibility and certainty degrees are obtained, we store them in intermediate tables for Boolean selection.
7. Performing ANDing or ORing operation.
   Since the attribute values of shape descriptors are independent, we can use the decomposability properties of possibility and certainty degrees.

Once we calculate all possibility and certainty degrees for all elemental querying conditions, we perform the conjunction or disjunction operation using min/max method. Take the ANDing operation for example, the overall possibility degree of a tuple is the minimum of the component possibility degrees. In this example, the component possibility degrees are all 1.0, so the overall possibility degree of this shape, the minimum of them, is 1.0 ($P = min(poss1, poss2) = min(1.0, 1.0) = 1.0$). The component certainty degrees are 1.0 and 1.0, the minimum of these two, 1.0, is the overall certainty degree ($SD = min(sd1, sd2) = (1.0, 1.0) = 1.0$).

8. Performing Boolean selection.
   Assume the thresholds for the overall possibility and certainty degrees are 0.9 and 0.6. Since the overall possibility and certainty degrees, 1.0 and 1.0, are higher than the required thresholds, the tuple containing the attribute value ($D1, D2$) are retrieved, otherwise this tuple will be discarded.
9. Displaying shape.
   Once we get all possibility and certainty degrees of all possible matching records, a Boolean selection is used to cull out some shapes for which the possibility and certainty degrees are lower than predefined thresholds. In this example, Shp3 is retrieved and the other two shapes are discarded. Fig. 8 shows the fuzzy shape Shp3 which is a set of *extremely bent* shapes. The upper-left shape with the membership grade 0.8 means the possibility of this shape to be considered as *extremely bent*. As the shape becomes increasingly bent, the membership grade increases. The last shape has membership grade 1 because everybody will consider it to be an *extremely bent* shape.

9. Conclusions and future work

A possibility-based fuzzy shape database has been constructed within a conventional relational database. A fuzzy shape is represented by a set of shape descriptors and shape parameters. It is indexed and retrieved by fuzzy shape descriptors. A graphical user interface is constructed to provide human consistent and natural-language-like queries. A 3D shape rendering system is developed to display the retrieved shapes.

This fuzzy shape database provides a perceptual shape indexing and retrieval mechanism, hence the users can query this database using higher level shape descriptions in a natural way. The system is only a prototype, but we have seen the power of combining the fuzzy set approach and visual display in supporting the fuzzy shape querying. It also provides evidence that a fuzzy database with fuzzy attribute values can be implemented in a well-developed commercial relational database management system in an effective and economical way. The main contribution of this paper lies in the method we employed to index and retrieve 3D shapes using descriptive terms. The perceptual shape indexing and retrieval mechanisms allow users to query the shape database using semantic shape descriptions. They also form an important aspect of introduction of the fuzzy set approach into CAD systems for supporting conceptual design when designers have only vague ideas and need fast prototyping.

Future work will involve improving and refining this prototype shape database and extending it to facilitate the management of composite fuzzy shapes. A composite shape consists of primitive shape elements (simple shapes) and other composite shapes. Therefore, composite shapes have more complex structural, physical, technical and other information than simple shapes. It is inefficient to represent such complex shapes in a conventional relational data model. Thus, a more powerful model which can reflect the hierarchical and modular nature of composite shapes and a more efficient manipulation mechanism, which can bring together all information about a composite shape, are needed. Since the object-oriented data model organizes and manipulates abstract concepts and real things in terms of objects with collections of data and operations on the data, it can support complex data structure and reflect the design process. Hence, it is a promising candidate for handling complex shapes in CAD systems. Therefore, a commercial environment with object features will be employed to manage composite fuzzy shapes, and relevant issues for handling fuzzy information will be explored.

Acknowledgements. The authors would like to thank the anonymous referees for their detailed comments on the earlier version of this paper.

References

BALDWIN, J.F. (1983) *A Fuzzy Relational Inference Language.* Pergamon Press.

BALDWIN, J.F., MARTIN, T.P. AND PILSWORTH, B.W. (1995) *Fril- Fuzzy and Evidential Reasoning in Artificial Intelligence.* University of Bristol, UK.

BARR, A.H. (1981) Superquadrics and Angle-preserving Transformations. *IEEE Computer Graphics and Appl.* **1**, 11–23.

BARR, A.H. (1984) Global and Local Deformations of Solid Primitives. *Computer Graphics* **18** (3), 21–30.

BARR, A.H. (1992) Rigid Physically Based Superquadrics. *Graphics Gems III,* 137–148.

BERKAN, R.C. and TRUBATCH, S.L. (1997) *Fuzzy System Design Principles.* IEEE Press, NY.

BIEDERMAN, I. (1987) Recognition-by-Components: A Theory of Human Image Understanding. *Psychological Review* **2**, 115–147.

BOSC, P., DUVAL, L. and PIVERT, O. (2000) Value-Based and Representation-Based Querying of Possibilistic Databases. In: D. Bordogna and G. Pasi, eds., *Recent Issues on Fuzzy Databases.* Physica-Verlag, Heidelberg-New York.

BOSC, P. and GALIBOURG, M. (1989) Indexing Principles for a Fuzzy Data Base. *Information Systems* **14** (6), 493–499.

BRONSVOORT, W.F. (1990) *Direct Display Algorithms for Solid Modeling.* Delft University Press.

BUCKLES, B. and PETRY, F. (1982) Fuzzy Databases and Their Applications. *J. Fuzzy Sets and Systems* **7**, 213–226.

CHEN, G. (1998) *Fuzzy Logic in Data Modeling: Semantics, Constraints, and Database Design.* Kluwer Academic Publishers, USA.

CHEN, G. (1999) Data Models for Dealing with Linguistic and Imprecise Information. In: L. A. Zadeh and J. Kacprzyk, eds., *Computing with Words in Information/Intelligent Systems 2: Applications.* Physica-Verlag, Heidelberg, 325–344.

CYBENKO, G., BHASIN, A. and COHEN, K.D. (1997) Pattern recognition of 3D CAD objects: Towards an electronic yellow pages of mechanical parts. *Smart Engineering Systems Design* **1**, 1–13.

DICKINSON, S., BIEDERMAN, I., PENTLAND, A., EKLUNDH, J.-O., BERGEVIN, R. and MUNCK-FAIWOOD, R. (1993) The Use of Geons for Generic 3-D Object Recognition. *Proc. International Joint Conference on Artificial Intelligence (IJCAI),* Chambery, France, 1693–1699.

DICKINSON, S.J. (1994) Integrating Qualitative and Quantitative Shape Recovery. *International Journal of Computer Vision* **13** (3), 311–330.

HARDWICK, M., MORRIS, K.C., SPOONER, D.L., RANDO, T. and DENNO, P. (2000) Lessons learned developing protocols for the industrial virtual enterprise. *Computer Aided Design* **32** (2), 159–166.

KACPRZYK, J. and ZADROŻNY, S. (1997) Flexible Querying Using Fuzzy Logic: An implementation for Microsoft ACCESS. In: T. Andreasen, H. Christiansen and H.L. Larsen, eds., *Flexible Query Answering Systems.* Kluwer Academic Publishers, 247–275.

KLIR, G. and YUAN, B. (1995) *Fuzzy Sets and Fuzzy Logic.* Prentice Hall, New Jersey.

KRIEGEL, H.-P. (1993) Handling Geometric Objects with Free Form Curves in Spatial Databases. *2nd ACM Symposium on Solid Modeling and Applications*, Montreal, Canada.

KRIEGEL, H.-P., MÜLLER, A., PÖTKE, M. and SEIDL, T. (2001) DIVE: Database Integration for Virtual Engineering. *17th Int. Conf. on Data Engineering (ICDE 2001)*, Heidelberg, Germany.

KRUSE, R., GEBHARDT, J. and KLAWONN, F. (1994) *Foundations of Fuzzy Systems.* John Wiley & Sons Ltd., England.

LU, G. (1999) *Multimedia database management systems.* Artech House, Boston-London.

MÄNTYLÄ, M. (1988) *An Introduction to Solid Modeling.* Computer Science Press, Inc., U.S.A.

MCFADDEN, F.R., HOFFER, J.A. and PRESCOTT, M.B. (1999) *Modern database management.* Addison-Wesley, Reading-Harlow.

McWHERTER, D., PEABODY, M., SHOKOUFANDEH A.C. and REGLI, W. (2001) Database techniques for archival of solid models. *6th ACM Symposium on Solid Modeling and Applications.* Ann Arbor, Michigan.

PETRY, F.E. and BOSC, P. (1996) *Fuzzy Databases: Principles and Applications.* Kluwer Academic Publishers, USA.

PHAM, B. and ZHANG, J. (2000) A Fuzzy Shape Specification System to Support Design for Aesthetics. In: L. Reznik, ed., *Soft Computing in Measurement and Information Acquisition.* Physica-Verlag, Heidelberg, in print.

PIEGL, L.A. and TILLER, W. (1997) *The NURBS Book.* Springer, Berlin-New York.

PONS, O., MEDINA, J.M., CUBERO, J.C. and VILA, M.A. (1997) A Fuzzy Deductive Relational Database. In: T. Andreasen, H. Christiansen, and H. L. Larsen, eds., *Flexible Query Answering Systems.* Kluwer Academic Publishers, 79–101.

PRADE, H. and TESTEMALE, C. (1984) Generalizing Database Relational Algebra for the Treatment of Incomplete/Uncertain Information and Vague Queries. *Information Sciences* **34**, 115–143.

PRADE, H. and TESTEMALE, C. (1987) Representation of Soft Constraints and Fuzzy Attribute Values by Means of Possibility Distributions in Databases. In: J. Bezdek, ed., *Analysis of Fuzzy Information - vol. 2: Artificial Intelligence and Decision Systems.* CRC Press, 213–229.

RUSPINI, E.H., BONISSONE, P.P. and PEDRYCZ, W. (1998) *Handbook of fuzzy computation.* Institute of Physics Pub., Bristol-Philadelphia.

SHAH, J. J. and MÄNTYLÄ, M. (1995) *Parametric and Feature-Based CAD/CAM.* John Wiley & Sons, Inc., NY.

UMANO, M. (1982) Freedom-0: A Fuzzy Database System. In: Dubois D., Prade H. and R. Yager, eds., *Fuzzy Sets for Intelligent Systems.* Morgan Kaufmann Publishers, Inc., San Mateo, CA, 667–675.

UMANO, M. (1983) Retrieval from Fuzzy Database by Fuzzy Relational Algebra. In: E. Sanchez and M. M. Gupta, eds., *Fuzzy Information, Knowledge Representation and Decision Analysis.* Pergmon Press, 1–6.

VILA, M.A., CUBERO, J.C., MEDINA, J.M. and PONS, O. (1995) Logic and Fuzzy Relational Databases: A New Language and a New Definition. In: P. Bosc and J. Kacprzyk, eds., *Fuzziness in Database Management Systems.* Physica-Verlag, Heidelberg, 114–138.

WU, X., ICHIKAWA, T. and CERCONE, N. (1996) *Knowledge-Base Assisted Database Retrieval Systems.* World Scientific, River Edge, NJ.

YAGER, R.R. and FILEV, D.P. (1994) *Essentials of Fuzzy Modeling and Control.* J. Wiley, New York.

YAZICI, A., BUCKLES, B.P. and PETRY, F.E. (1999) Handling Complex and Uncertainty Information in the ExIFO and NF2 Data Models. *IEEE Transactions on Fuzzy Systems* **7** (6), 659–676.

YAZICI, A. and CIBICELI, D. (1999) An access structure for similarity-based fuzzy databases. *Information Sciences* **115**, 137–163.

YAZICI, A. and GEORGE, R. (1999) *Fuzzy Database Modeling.* Physica-Verlag, New York-Heidelberg.

ZADEH, L.A. (1965) Fuzzy Sets. *Information and Control* **8**, 338–353.

ZADEH, L.A. (1978) Fuzzy Sets as a Basis for a Theory of Possibility. *Fuzzy Sets and Systems* **1**, 3–28.

ZADEH, L.A. (1999) Fuzzy Logic=Computing with Words. In: L.A. Zadeh and J. Kacprzyk, eds., *Studies in Fuzziness and Soft Computing: Computing with Words in Information/Intelligent Systems 1: Foundations.* Physica-Verlag, Heidelberg.