Multimedia Tools and Applications, 24, 125–153, 2004

A Logic for SVG Documents Query and Retrieval

EUGENIO DI SCIASCIO disciascio@poliba.it
Politecnico di Bari, Via Re David, 200-70125 Bari, Italy

FRANCESCO M. DONINI donini@unitus.it
Università della Tuscia, Via S. Carlo, 32-01100 Viterbo, Italy

MARINA MONGIELLO mongiello@poliba.it
Politecnico di Bari, Via Re David, 200-70125 Bari, Italy

Abstract. We propose a knowledge representation approach to the semantic retrieval by content of graphics described in Scalable Vector Graphics (*SVG*), the novel XML based W3C approved standard language for describing two-dimensional graphics.

The approach is based on a description logic devised for the semantic indexing and retrieval of complex objects. We provide a syntax to describe basic shapes, complex objects as compositions of basic ones, and transformations. An extensional semantics, which is compositional, is introduced for defining retrieval, classification, and subsumption services. Algorithms exploiting reasoning services, which are sound with respect to the semantics, are also described.

Using our logical approach as a formal specification, we implemented a prototype system. A set of experiments, carried out on a testbed of *SVG* documents to assess the retrieval capabilities of the system, is presented.

Keywords: Scalable Vector Graphics, retrieval, knowledge representation, spatial similarity

1. Introduction

Scalable Vector Graphics (*SVG*) language [34] is a W3C approved standard for describing two-dimensional graphics at the level of graphical objects rather than individual points, with descriptions based on XML. The inherent well structured, syntactically clear building blocks, i.e., basic shapes and transformations, call for applications able to support structured, semantically clear, content-based indexing and retrieval of *SVG* documents.

However, to the best of authors' knowledge there are no proposals—neither theoretical, nor applicative—that try to solve Content-Based Image Retrieval (CBIR) when images are represented in a language for vector graphics. In contrast, there are a huge number of proposals for CBIR when images are represented by matrices of pixels, and several proposals also when images are represented in some compressed form. But obviously, reverting a vector graphics to a pixel matrix only to apply CBIR methods is unreasonable: it introduces an improper discretization, segmentation problems, and an image processing overhead which are necessary only for digital representation of *natural* pictures, not artificial ones. Therefore, there is now a growing number of documents containing *SVG* images, without suitable tools for query and retrieval of such images.

We propose a knowledge representation approach that is based on a logical, declarative language. The language, devised along the lines of Description Logics, is endowed of a syntax whose objects are basic shapes, composition of basic shapes to create complex objects and transformations. An extensional semantics is also proposed, in terms of sets of retrievable documents. The semantics is compositional, i.e., adding new details to the query may only reduce the cardinality of the retrieved set. Syntax and semantics constitute a Semantic Data Model, in which the various characteristic features of each component are given an explicit notation through a geometric transformation. The semantics allows one to define hierarchical descriptions, based on set containment between interpretation of descriptions. Retrieval corresponds in our setting to the selection of interpretations satisfying a description. Notice that SVG basic elements can be composed to form complex objects and scenes, with elements modified with reference to their prototypical ones, and spatially composed. In our setting we devised exact and approximate algorithms for composite elements recognition and ranked retrieval, which are correct with respect to the semantics.

We remark that our formalism has a general applicability to any language for vector graphics (e.g., the one of CorelDraw). We focused on *SVG* just because its open-source definition, and its adoption as a standard by well-renowned firms like Adobe, Microsoft, as well as the independent W3 consortium.

The proposed logical framework is particularly suitable for a query by sketch approach; a user starts a query by drawing, with the aid of a graphical interface, a few elements that he/she considers the most prominent for the search, and as long as there are too many items in the retrieved set he/she can iteratively add new details to reduce the scope of the search. Notice that this process has resemblance with classical relevance feedback [10, 13, 17, 32], where retrieval effectiveness is also improved by incorporating the user in the query-retrieval loop. Depending on the initial query the system retrieves a set of documents that the user can mark either as relevant or irrelevant. The system, based on the user preferences, refines the initial query retrieving a new set of documents that should be closer to the user's information need. In our setting the user in the query retrieval loop refines his/her search by adding new details, but he/she can be sure that at least elements explicitly included in the query will be present in the retrieved set.

To manage and query the *SVG* graphical documents we built a knowledge based system, whose definition and query language follows the proposed syntax and semantics. Indexing and retrieval are carried out as a semantic indexing in which classes and objects are described through the logical language. The logical approach provides semantical properties and formal tools to conduct retrieval, recognition and classification in an effective and efficient manner. In this perspective retrieval amounts to finding the most specific descriptions, i.e., *SVG* documents, that can be classified under the query description. Besides, complex services such as reasoning about queries, e.g., containment and emptiness can be exploited.

The rest of the paper is organized as follows. In the next section we briefly describe *SVG* basics. Then we present our logical language in terms of syntax and semantics and available reasoning services. We then propose some of the algorithms used to compute similarities in the indexing and retrieval phases. The prototype system implementing the logic is described in Section 4, and experiments carried out to validate the approach are presented in Section 5.

Although the proposed approach is the first one, to the best of our knowledge, to deal specifically with *SVG* documents retrieval, we obviously build on previous work on content based retrieval, which is briefly recalled in Section 6. Last section draws the conclusions.

2. Scalable vector graphics

We provide here a brief description of the *SVG* syntax. The complete standard is available in [34].

We limit our analysis to elements we actually use in our approach, obviously the standard has several other characteristics we do not report here about.

SVG works with three types of graphical objects: vectorial graphical shapes (such as lines and shapes), images and text. The *image* element indicates that the contents of a complete raster file are to be rendered within a given rectangle. In this paper we do not consider this aspect, as raster images can be dealt with using typical CBIR techniques. So our objects are only vectorial shapes and text.

Objects can be grouped and modified in the context of previously interpreted ones. Sketches drawn with *SVG* can be dynamic and interactive. Scripting languages that use the *SVG* Document Object Model (DOM) can provide several other sophisticated applications. As far as graphical objects are concerned, *SVG* provides basic geometric shapes, such as rectangle, circle, etc. Standard symbols can be defined: the user can create and reuse her/his symbols, modify its size and orientations. Graphical effects such as filters, and illumination effects such as shadows and lights can be applied on client side while drawing the object.

A *fragment* of a *SVG* document contains several *SVG* elements. A fragment can be either an empty element, or contain only basic shapes, or can be a complex collection of containers and graphical objects. A fragment of a *SVG* document can be a *SVG* document or can be part of a XML document.

The structure of a document is composed by the svg and the g element. The svg element contains the *SVG* fragment between a pair of syntactic parenthesis. The attributes of a svg element are: resource-name, a standard attribute for the namespace definition, version number of the *SVG* language, currently it is 1.0, the x coordinate of the top left angle of the rectangle in which the image should be drawn, the implicit value is 0, the y coordinate of the same point, the width and the height of the rectangle in which the image will be drawn. The g element is a container for a set of graphical objects that can be named using the id attribute. Objects can be drawn in the *SVG viewport*, a limited window of a hypothetical infinite canvas. The *SVG* viewport is a window whose size depends on a negotiation between the *SVG* fragment and the parent document. The viewport definition determines also the initial coordinate system. Generally, the coordinate system has its origin in the top left vertex of the viewport, the x and y axis orientations are respectively towards the right and the bottom sides of the viewport. A new coordinate system is obtained through transformations that can be performed using the *transform* attribute or defining a new viewbox.

Basic shapes. Basic shapes are rectangle, circle, ellipse, lines, polylines, polygons.

Rect draws a rectangle aligned with the axis of the current user-defined coordinates system. Attributes of the rectangle are x and y coordinates, width and height length, r_x and

r_y lengths that are the lengths of the radius along the corresponding axis for the ellipse used to adjust rectangles with round vertexes.

Circle draws a circle depending on a point and a radius. The attributes are c_x and c_y coordinates of the center and the length of the radius.

Ellipse defines a shape whose axis are aligned with the current coordinate system. The attributes are c_x and c_y coordinates of the center, r_x and r_y lengths of the radius along the corresponding axis.

Line is a segment joining two points. The attributes are x_1 and y_1 coordinates of the starting point, x_2 and y_2 coordinates of the second point. A line can not be filled in with a color.

Polyline defines a set of line segments. A polyline can be an opened or a closed one. The attributes are the list of points in which each pair defines a vertex of the polyline; the coordinates are separated by a comma while the pairs are separated by a blank.

Polygon defines a closed shape composed by a set of lines. The attributes are a list of points in which each pair represents a vertex of the polygon. The pairs are separated by a blank; the coordinates inside each pair are separated by a comma. All the coordinates are represented in the current system and their number must be even.

Basic shapes can be filled in with a color, described as 3-term RGB tuple, using the attribute *fill* to fill the body of the shape and *stroke* to fill the perimeter of the shapes with color. The filling can be defined as a single color or with gradient (linear radial), or a user-defined color or a texture.

Transformations. Transformations allow to manipulate the images and modify the shapes. A point is modified by changing its position, objects are modified by changing the position of single points. The modified object is obtained by connecting the single points. Mathematical transformation are represented using a matrix of transformation. Since the product is always defined for square matrices, the matrices of transformation are properly modified in order to obtain square matrices. Transformation matrices define the mathematical mapping from one coordinate system into another with a 3×3 matrix, using the equation $[x'\ y'\ 1] = [x\ y\ 1] * matrix$. Clearly, several transformations can be composed into a single one by matrix multiplication. The Current Transformation Matrix (CTM) defines the mapping from the user coordinate system into the viewport coordinate system. A transformation is performed using the attribute *transform* that can be applied to a single container and inherited by all the elements contained in it; the attribute *transform* is followed by a list of possible transformations.

Translation. Position of objects changes without further distortions. To apply such type of transformation, x and y coordinates are increased of a constant value. The syntax of a translation is the following: $translate(\langle t_x \rangle [\langle t_y \rangle])$ (the t_y value is assumed 0 if not provided).

Scaling. Scaling is adopted for modifying the dimensions and the shape of an object. A scale along the origin of the coordinate system is determined by two scale factors: along the x and y axis respectively. The object is enlarged if both parameters are greater than one in absolute value; to reduce the dimensions of an object both the factors must be smaller than 1. A transformation is symmetric if it modifies only the dimensions of the objects and

does not produce further changes. Negative values of the parameters produce a reflection of the object. The syntax of a scale transformation is the following: *scale*($\langle s_x \rangle[\langle s_y \rangle]$).

Rotation. A rotation changes the orientation of an object and is defined using a point and an angle. Besides it can be positive or negative (respectively for a counterclockwise and a clockwise rotation). The syntax is the following: *rotate*($\langle$*rotate-angle*$\rangle[\langle c_x \rangle \langle c_y \rangle]$). *rotate-angle* value is the rotation degree with respect to a given point. Rotation is performed with respect to the point whose coordinates are c_x and c_y whether these values are specified, otherwise it applied with respect to the origin.

Shearing. A shearing modifies the shape contour changing only one coordinate. In *SVG*, shearings along the two axis have been distinguished and are named *skewings*. Such an operation is performed by multiplying the coordinates to the tangent of a predefined angle. The syntax is given by the following expression: *skewX*($\langle$*skew-angle*$\rangle$).

Composition of multiple transformations. Transformations on a single object can be multiple ones and are obtained as the product of the CTM matrices.

A simple example of a *SVG* file description is in figure 1, and the corresponding graphic output is in figure 2.

```
<?xml version="1.0" standalone="no"?>
 <!DOCTYPE svg PUBLIC
"-//W3C//DTD SVG 20010904//EN" "C:\svg\svg10.dtd">
 <svg xmlns="http://www.w3.org/2000/svg">
    <g id="BJT1" stroke="rgb(15,15,15)"
    stroke-width="1" transform="translate(170,215)">
        <line y1="-10" y2="10"/>
        <line y1="-5" x2="20" y2="-15"/>
        <line y1="5" x2="20" y2="15"/>
        <polygon fill="rgb(15,15,15)"
        points="20,15 15,8 12,15"/>
        <line x1="-20"/>
    </g>
</svg>
```

Figure 1. A simple *SVG* file description.

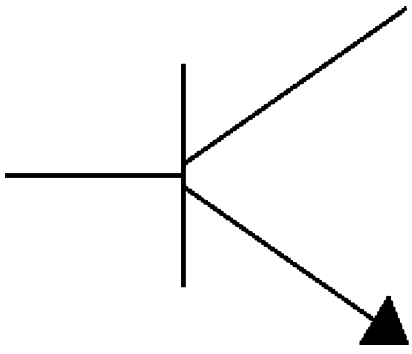

Figure 2. Output of the *SVG* description in figure 1.

3. A description logic for SVG retrieval

To represent graphic objects in the *SVG* standard, basic features are included in the object definition as attributes. This means that documents must be processed to extract their features: shapes can be recognized only by interpreting the syntax of the *SVG* description. Unfortunately, even this is not always sufficient. There are cases when "there is more the picture than meets the eye", as user-defined shapes can be, through transformations, made completely different from their prototypical basic shape, to visually resemble e.g., other basic shapes. For instance, a closed polyline might visually resemble a rectangle.

We cope with this situation as described in the following sections, by defining a proper semantics for *SVG* documents. We start with a brief recall of few notions from Description Logics, then we define an abstract syntax for *SVG* fragments, and then we set a semantics for such a syntax.

3.1. Description logics

Description Logics (DLs) [7, 18, 38] are a family of logic formalisms for Knowledge Representation (KR). As any KR formalism, they are equipped with a syntax to express pieces of knowledge, a semantics (which for DL is usually model-theoretic), and a set of reasoning services that infer implicit knowledge from asserted expressions. In DL, the basic syntax elements are: *concept* names and *role* names.

Intuitively, concepts stand for sets of objects, and roles link objects in different concepts. Formally, concepts are interpreted as subsets of a domain of interpretation Δ, and roles as binary relations (subsets of $\Delta \times \Delta$). Basic elements can be combined using *constructors* to form concept and role *expressions*, and each DL has its distinguished set of constructors.

Every known DL allows one to form a *conjunction* of concepts, usually denoted as $\sqcap$. Usually, roles can be combined with concepts using *existential role quantification*, and *universal role quantification*. However, we do not use here usual role constructors as we need to express compositions of shapes according to transformations.

Concept expressions can be used in *inclusion assertions*, and *definitions*, which impose restrictions on possible interpretations according to the knowledge elicited for a given domain. A set of such inclusions is called a TBox. Individuals can be asserted to belong to a concept using *membership assertions* in an ABox.

Description logics are equipped with *reasoning services*: logical problems whose solution can make explicit information which was implicit in the assertions. The main reasoning services we are interested in are *classification*, *subsumption*, *retrieval*, and *recognition*. We describe how we use these services later on, after introducing syntax and semantics of our logic.

3.2. A logical language

We describe here the formalism for syntax and semantics of our language to manage *SVG* documents, which is adapted from a previous work of the authors [16]. In our language

concepts and roles are used to define shapes and geometric transformations of basic shapes to build complex objects.

Syntax. We abstract from the syntax of *SVG* , and define our language through basic shapes, position of shapes, and geometric transformations.

Basic shapes we consider are those proposed by *SVG* standard, described through their attributes and transformations applied in the *SVG* language. Basic shapes, as defined in the *SVG* standard are circle, rectangle, ellipse, line, polyline, polygon. In our formal setting basic shapes are denoted with the letter B, and have an edge contour $e(B)$ characterizing them. We assume that $e(B)$ is described in a space whose origin coincides with the centroid of B.

The possible transformations are those defined in *SVG* . We globally denote a transformation as τ. Transformations can be composed in sequences $\tau_1 \circ \cdots \circ \tau_n$, and they form a mathematical group. To make the syntax uniform, we consider also the pose of a basic shape as a transformation. For example, for a circle, specifying the position of the center and the length of the radius can be considered as a translation-scaling of a hypothetical circle with unit radius and center in the origin. This means that in our abstract syntax it is sufficient to consider as basic shapes: one unit circle (from which every circle and ellipse can be obtained) a unit square (for every rectangle) and a unit horizontal segment. Polylines and polygons, instead, represent infinitely basic shapes according to the vertices they contain.

The basic building block of our syntax is a *basic shape component* $B[c, \tau]$, which represents a basic shape B with color c and edge contour $\tau(e(B))$. With $\tau(e(B))$ we denote the pointwise transformation τ of the whole contour of B.

Composite shape descriptions are conjunctions of basic shape components denoted as

$$C = B_1[c_1, \tau_1] \sqcap \cdots \sqcap B_n[c_n, \tau_n]$$

For example, the *SVG* fragment of figure 1 is abstractly represented as

$$\texttt{line}[\tau_1] \sqcap \texttt{line}[\tau_2] \sqcap \texttt{line}[\tau_3] \sqcap \texttt{line}[\tau_4] \sqcap \texttt{polygon}[0, 0\ 1, 1.71\ 2, 0][\tau_5]$$

where `line` is the unit horizontal segment, and the vertices of the polygon describe a prototypical equilaterus triangle. We depict this composition as in figure 3.

Obviously end users of our system do not have to actually define descriptions using the syntax, which is just an *internal* representation. The system can maintain it while the user sets the proper attributes of the *SVG* definition for the new object and its transformations with relative parameters, with the help of a GUI.

We do not present here the syntax for applying transformations to whole groups, and for naming and reusing groups, which is anyway only a simple addition to the outlined framework.

Semantics. The semantics of our language is defined as an interpretation of the syntactic elements of the language on a domain. Figure 4 explains through an example the semantics of

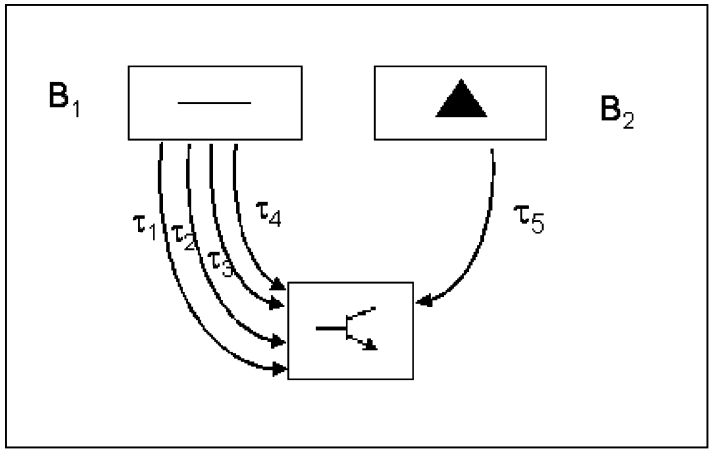

Figure 3. A composition of basic shapes with transformations.

the language. Only for this section, we do not use color specifications, to simplify notation. We use colors later on, when we reconsider all features for similarity computation.

The domain Δ is a set of *SVG* fragments and the interpretation of an object composed by basic shapes is a subset of the domain Δ, in which the basic shapes have the proper features. We consider an extensional semantics, in which syntactic expressions are interpreted as subsets of a domain. For our setting, the domain of interpretation is a set of *SVG* fragments Δ, and shapes and components are interpreted as subsets of Δ. Hence, also a collection of *SVG* fragments is a domain of interpretation, and a complex shape C is a subset of such a domain—the fragments to be retrieved from the collection when C is viewed as a query.

Formally, an interpretation is a pair $(\mathcal{I}, \Delta)$, where Δ is a set of *SVG* fragments, and $\mathcal{I}$ is a mapping from shapes and components to subsets of Δ. We identify each fragment F with the set of shapes $\{s_1, \ldots, s_n\}$ it is composed by. Each shape s comes with its own edge contour $e(s)$. A *SVG* fragment $F \in \Delta$ belongs to the interpretation of a basic shape component $B[\tau]^{\mathcal{I}}$ if F contains a shape whose contour matches $\tau(e(B))$. Note that by resorting on contours, we also take into account resemblances between different basic shapes, e.g., `polygon`[0,0 1,0 1,1 0,1] is in the interpretation of the unit square. The definition can be extended to approximate recognition as follows. Recall that the *characteristic function* f_S of a set S is a function whose value is either 1 or 0; $f_S(x) = 1$ if $x \in S$, $f_S(x) = 0$ otherwise. We consider now the characteristic function of the set $B[\tau]^{\mathcal{I}}$.

Let F be a *SVG* fragment; if F belongs to $B[\tau]^{\mathcal{I}}$, then the characteristic function computed on F has value 1, otherwise it has value 0. To keep the number of symbols low, we overload the expression $B[\tau]^{\mathcal{I}}$ also to denote the characteristic function (with an argument (F) to distinguish it from the set).

$$B[\tau]^{\mathcal{I}}(F) = \begin{cases} 1 & \text{if } \exists s \in F : e(s) = \tau(e(B)) \\ 0 & \text{otherwise} \end{cases}$$

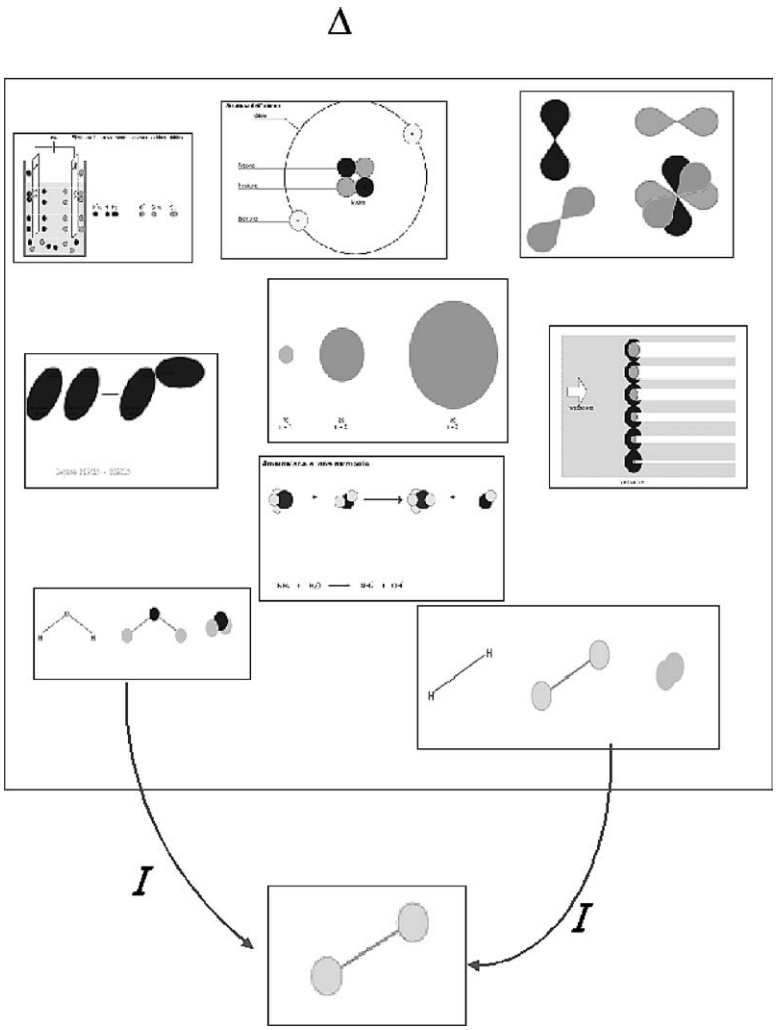

Figure 4. An example picturing the semantics of our language.

Now we reformulate this function in order to make it return a real number in the range [0, 1]—as usual in fuzzy logic [40]. Let $sim(\cdot, \cdot)$ be a similarity measure from pairs of contours into the range [0, 1] of real numbers (where 1 is perfect matching). We use $sim(\cdot, \cdot)$ instead of equality to compare shapes. Then, the characteristic function for the approximate recognition in an image F of a basic component, is:

$$B[\tau]^{\mathcal{I}}(F) = \max_{s \in F}\{sim(e(s), \tau(e(B)))\}$$

Note that *sim* depends on transformations, since we are looking for shapes in F whose contour matches $e(B)$, with reference to the position and size specified by τ. The interpretation

of basic shapes, instead, includes a transformation invariant recognition. We define the interpretation of a basic shape in the approximate recognition as the function

$$B^{\mathcal{I}}(F) = \max_{\tau} \max_{s \in F} \{sim(e(s), \tau(e(B)))\}$$

The maximization over all possible transformations $\max_\tau$ can be effectively computed by using a similarity measure sim_{ss} that is invariant with reference to transformations (see Section 3.4). In this way, a basic shape B can be used as a query to retrieve all fragments from Δ which are in $B^{\mathcal{I}}$.

Composite shape descriptions are interpreted as sets of *SVG* fragments that contain all components of the composite shape. Components can be anywhere in the fragment description, as long as they are in the described arrangement relative to each other.

Let C be a composite shape description $B_1[\tau_1] \sqcap \cdots \sqcap B_n[\tau_n]$. For approximate matching the interpretation of a composite shape is:

$$C^{\mathcal{I}}(F) = \max_{\tau} \left\{ \min_{i=1}^{n} \{B_i[(\tau \circ \tau_i)]^{\mathcal{I}}(F)\} \right\} \tag{1}$$

Observe that we require all components of C to be transformed using the same transformation τ. This preserves the arrangement of the components relative to each other—given by each τ_i—while allowing $C^{\mathcal{I}}$ to include every document containing a group of shapes in the right arrangement, wholly displaced by τ. The measure by which a query belongs to the interpretation of a composite shape description is dominated by the least similar shape component (the one with the minimum similarity). Hence, if a basic shape component is very dissimilar from every shape in F, this brings near to 0 also the measure of $C^{\mathcal{I}}(F)$. This is more strict than, e.g., Gudivada and Raghavan's [23] or El-Kwae and Kabuka's [20] approaches, in which a non-appearing component can decrease the similarity value of $C^{\mathcal{I}}(F)$, but F can be still above a threshold.

Although this requirement may seem a strict one, it captures the way details are used to refine a query: the "dominant" shapes are used first, and, if the retrieved set is still too large, the user adds details to restrict the results. In this refinement process, it should not happen that other images that match only some new details, "pop up" *enlarging* the set of results that the user was trying to restrict. We formalize this refinement process through the following definition.

Definition 1 (Downward refinement). Let C be a composite shape description, and let D be a refinement of C, that is $D \doteq C \sqcap B'[\tau']$. For every interpretation $\mathcal{I}$, if elements are interpreted as in (1), then for every document F it holds $D^{\mathcal{I}}(F) \leq C^{\mathcal{I}}(F)$.

The above property makes our language fully *compositional*. Namely, let C be a composite shape description; we can consider the meaning of C—when used as a query—as the set of documents that can be potentially retrieved using C. At least, this will be the meaning perceived by an end user of a system. Downward refinement ensures that the meaning of C can be obtained by starting with one component, and then progressively adding other components in any order. We remark that for other frameworks cited above [20, 23] this property

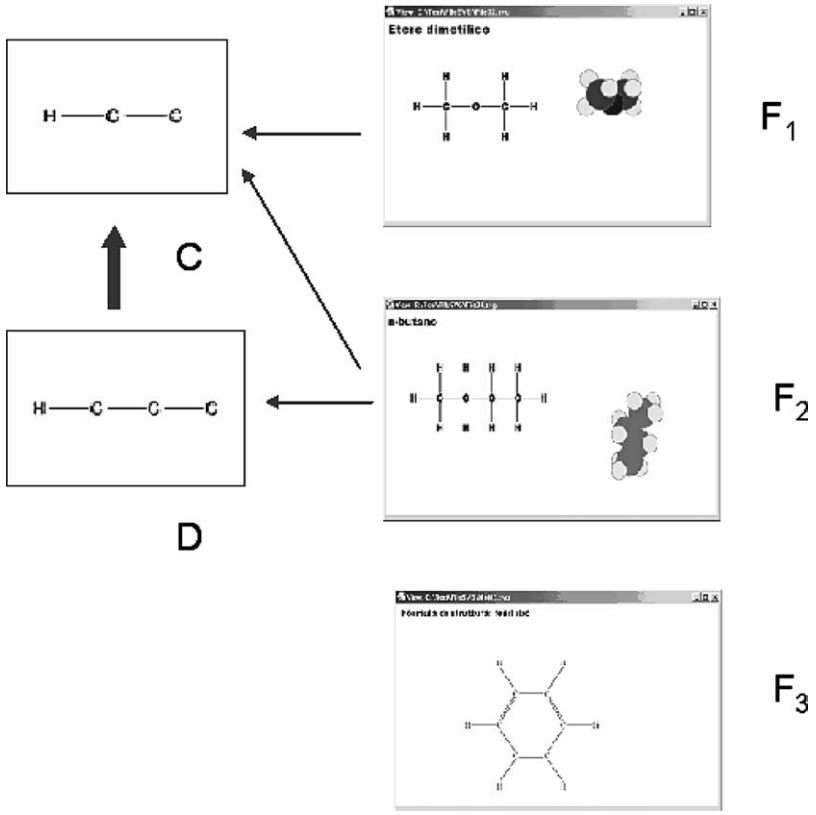

Figure 5. Downward refinement: The thin arrows denote non-zero similarity in approximate recognition. The thick arrow denotes a refinement.

does not hold. We illustrate the problem in figure 5. Starting with shape description C, we may retrieve (among many others) the two fragments F_1, F_2, for which both $C^{\mathcal{I}}(F_1)$ and $C^{\mathcal{I}}(F_2)$ are above a threshold t, while another image F_3 is not in the set because $C^{\mathcal{I}}(F_3) < t$. In order to be more selective, we try adding details, and we obtain the shape description D. Using D, we may still retrieve F_2, and discard F_1. However, observe that F_3 now partially matches the new details of D. If Downward refinement holds, $D^{\mathcal{I}}(F_3) \leq C^{\mathcal{I}}(F_3) < t$, and F_3 cannot "pop up". In contrast, if Downward refinement does not hold (as in Gudivada and Raghavan's approach) it can be $D^{\mathcal{I}}(F_3) > t > C^{\mathcal{I}}(F_3)$ because matched details in D raise the similarity sum weighted over all components. In this case, the meaning of a sketch cannot be defined in terms of its components.

Downward refinement is a property linking syntax to semantics. Thanks to the extensional semantics, it can be extended to an even more meaningful semantic relation, namely, subsumption. We borrow this definition from Description Logics [18], and its fuzzy extensions [35, 39].

Definition 2 (Subsumption). A description C subsumes a description D if for every interpretation $\mathcal{I}$, $D^{\mathcal{I}} \subseteq C^{\mathcal{I}}$. If (1) is used, C subsumes D if for every interpretation $\mathcal{I}$ and fragment $F \in \Delta$, it is $D^{\mathcal{I}}(F) \leq C^{\mathcal{I}}(F)$.

Subsumption takes into account the fact that a description might contain a syntactic variant of another, without both the user and the system explicitly knowing this fact. The notion of subsumption extends downward refinement. It enables also a hierarchy of shape descriptions, in which a description D is below another C if D is subsumed by C. When C and D are used as queries, the subsumption hierarchy makes easy to detect *query containment*. Containment can be used to speed up retrieval: all fragments retrieved using D as a query can be immediately retrieved also when C is used as a query, without recomputing similarities. While query containment is important in standard databases [37], it becomes paramount in any setting in which the recognition of features can be computationally demanding.

3.3. Reasoning services

Based on our formalism we can set up several reasoning services:

1. recognition: Given an *SVG* fragment F and a description D, decide if D is recognized in F.
2. retrieval: given a collection of *SVG* documents and a shape description D, retrieve all documents in which D can be recognized.
3. classification: given a fragment F and a collection of descriptions $D_1, \ldots, D_n$, find which descriptions can be recognized in F. In practice, F is classified by finding the *most specific* descriptions (with reference to subsumption) it satisfies. Observe that classification is a way of "preprocessing" recognition.
4. description subsumption (and classification): given a (new) description D and a collection of descriptions $D_1, \ldots, D_n$, decide whether D subsumes (or is subsumed by) each D_i, for $i = 1, \ldots, n$.

Services 1–2 are standard in a content-based retrieval system, while services 3–4 are less obvious. Their utility can be explained as follows: given a query (shape description) D, if there exists a collection of queries (descriptions) $D_1, \ldots, D_n$ and all fragments in the collection were already classified with reference to $D_1, \ldots, D_n$, then it may suffice to classify (i.e., Service 4) D with reference to $D_1, \ldots, D_n$ to find (most of) the fragments satisfying D. This is the usual way in which classification in Description Logics—which amounts to a semantic indexing—can help query answering [30]. This process obviously leads to a drastic reduction in query answering time. The problem of computing subsumption between descriptions is reduced to recognition and solved by the algorithm for recognition given in the next section. Subsumption in this simple logic relies on the composition of basic shapes. Intuitively, to actually decide if D is subsumed by C, we check if the description associated with D, considered as a fragment, would be retrieved when using C as a query.

3.4. *Computing similarities*

In this section we describe the algorithms adopted to actually compute similarity as formally outlined in the preceding section. The objective is then to determine a ranked approximate recognition between a composite shape $C = B_1[c_1, \tau_1] \sqcap \cdots \sqcap B_n[c_n, \tau_n]$, and a *SVG* fragment F composed by shapes $\{s_1, \ldots, s_m\}$. We assume $n \leq m$, since all components of C must appear in F. We now reconsider color of shapes.

Let $sim(s, B[c, \tau])$ be a similarity measure that considers a shape s with its color $c(s)$ and a component $B[c, \tau]$ into the range [0, 1] of real numbers (where 1 is perfect matching). Note that here we reconsider the color we skipped for simplicity in the previous section.

It can be proved [16] that formula (1)—i.e., the measure of how well C is recognized in F—can be computed also according to the following formula:

$$\max_{j:\{1..n\}\to\{1..m\}} \max_{\tau} \min_{i=1}^{n} \left\{ sim\big(s_{j(i)}, B_i[c_i, (\tau \circ \tau_i)]\big) \right\} \tag{2}$$

where $j : \{1..n\} \to \{1..m\}$ denotes an injective mapping from indexes $1, \ldots, n$ of components of C into indexes $1, \ldots, m$ of basic shapes of F. That is, if $B_i[c_i, \tau_i]$ is a component of C, then j maps it into shape $s_{j(i)}$ of F. The maximization over τ is needed to recognize fragments in which the group C is translated, rotated, scaled, anywhere.

This formula suggests that from all the groups of shapes in a fragment that might resemble the components, we should select the groups that present the higher similarity. Since the maximization over all possible τ is still unfeasible [9, 12], we adopt some heuristics to evaluate the above formula. First of all, we decompose $sim(s, B[c, \tau])$ as the following sum of weighted contributions.

$$\begin{aligned} sim(s, B[c, \tau]) = {} & sim_{\text{shape}}(e(s), e(B)) \cdot \alpha \\ & + sim_{\text{color}}(c(s), c) \cdot \beta \\ & + sim_{\text{spatial}}(e(s), \tau(e(B)) \cdot \gamma \\ & + sim_{\text{trans}f}(e(s), \tau(e(B)) \cdot \delta \\ & + sim_{\text{position}}(e(s), \tau(e(B)) \cdot \eta \end{aligned} \tag{3}$$

where coefficients $\alpha, \beta, \gamma, \delta, \eta$ weight the relevance each feature has in the overall similarity computation. Obviously, we impose that $\alpha + \beta + \gamma + \delta + \eta = 1$, and that all coefficients are greater than 0. For each feature we compute a similarity measure as explained in the following. Then, we assign to all similarities of a feature the worst similarity in the group. This yields a pessimistic estimate of Formula (2); however this estimate keeps the semantics fully compositional. We now explain how the summands are obtained.

First of all note that for similarity computation of each term we use the function $\Phi(x, fx, fy)$. The role of this function is to change a distance x (in which 0 corresponds to perfect matching) to a similarity measure (in which the value 1 corresponds to perfect matching), and to "smooth" the changes of the quantity x, depending on two parameters

fx, fy, as follows:

$$\Phi(x, fx, fy) = \begin{cases} fy + (1 - fy) \cdot \cos\left(\dfrac{\pi x}{2 \cdot fx}\right) & \text{if } 0 \leq x < fx; \\ fy \cdot \left[1 - \dfrac{\arctan\left[\frac{\pi \cdot (x - fx) \cdot (1 - fy)}{fx \cdot fy}\right]}{\pi}\right] & \text{if } x > fx \end{cases}$$

where $fx > 0$ and $0 < fy < 1$.

Shape similarity. Similarity of the shapes (translation-rotation-scale invariant) is denoted by $sim_{\text{shape}}(e(s), e(B))$. Its computation depends only on geometrical features of the shapes, hence it is independent of the pose.

Color Similarity. Also color similarity is independent of the pose. The value of $sim_{\text{color}}(c(s), c)$ measures the similarity in terms of color appearance as the Euclidean distance between RGB components of k-th component of D and the region $s_{j(k)}$. Although more effective color models could be applied we revert to RGB simple color model, which allows a straightforward computation from the *SVG* document. Then, the smoothing function is applied to obtain a similarity measure.

The other contributions depend on the pose—hence on the transformation τ—and try to evaluate how the pose of each shape in the selected group is similar to the pose specified by the corresponding component in the query.

Transformations similarity. The measure $sim_{\text{trans}f}$ is used to compare shapes that do not belong to the same syntactic object but are comparable since they have been modified by a transformation. For example: a skewing transformation applied on a rectangle that becomes comparable with a parallelogram. *SVG* assigns each object the CTM matrix, obtained as a product matrix over all the transformations of the elements and of its parent nodes. For each shape a set of meaningful point of the shape perimeter is extracted to determine the shape contour (a set of x and y coordinates) and the CTM is applied to the this set returning another set of points.

Spatial similarity. The feature $sim_{\text{spatial}}(e(s), \tau(e(B))$ measures the relative spatial positions, i.e., how coincident are the centroids of s and of $\tau(e(B))$. Since this measure is at the core of our semantic retrieval, we analyze it in detail. Notice that several algorithms have been proposed to cope with this issue and we report on them in Section 6; our algorithm extends previous ones [20, 22, 23] in that it allows the presence of multiple instances of an object and is correct with reference to the semantics.

For a given component—say, component 1—we compute all angles under which the other components are seen from 1. Formally, let $\alpha_{\widehat{i1h}}$ be the counter-clockwise-oriented angle with vertex in the centroid of component 1, and formed by the lines linking this centroid with the centroids of components i and h. There are $n(n - 1)/2$ such angles.

Then, we compute the correspondent angles for shape $s_{j(1)}$, namely, angles $\beta_{j(i)\widehat{j(1)}j(h)}$ with vertex in the centroid of $s_{j(1)}$, formed by the lines linking this centroid with the centroids of shapes $s_{j(i)}$ and $s_{j(h)}$ respectively. A pictorial representation of the angles is given in figure 6.

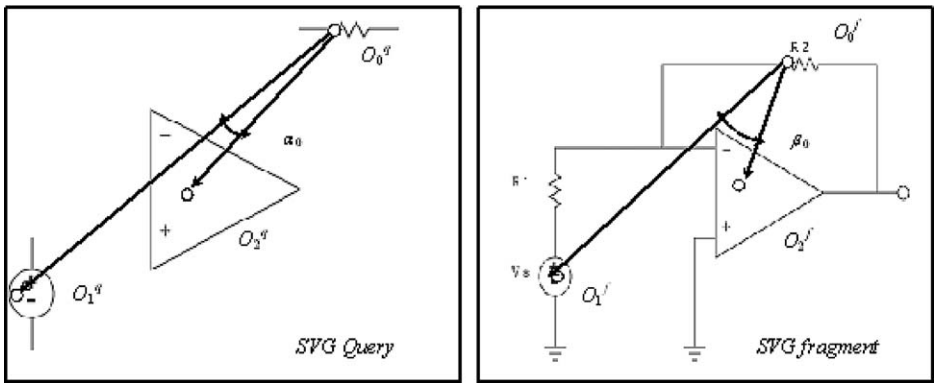

Figure 6. Representation of angles used for computing spatial similarity of component 1 and region $r_{j(1)}$.

Then we let the difference $\Delta_{\text{spatial}}(e(s_{j(1)}), \tau_1(e(B_1))$ be the maximal absolute difference between correspondent angles:

$$\Delta_{\text{spatial}}\big(e\big(s_{j(1)}\big), \tau_1(e(B_1)) = \max_{i,h=2,\ldots,n,i\neq h} \big\{\big|\alpha_{\widehat{i1h}} - \beta_{j(i)\widehat{j(1)}j(h)}\big|\big\}$$

We compute an analogous measure for components $2, \ldots, n$, and then we select the maximum of such differences:

$$\Delta_{\text{spatial}}[j] = \max_{i=1}^{n} \big\{\Delta_{\text{spatial}}\big(e\big(s_{j(i)}\big), \tau_i(e(B_i))\big\} \tag{4}$$

where the argument j highlights the fact that this measure depends on the mapping j in Formula (2). Finally, we transform this maximal difference—for which perfect matching yields 0—into a minimal similarity—perfect matching yields 1—with the help of the function Φ previously described. This minimal similarity is then assigned to every $sim_{\text{spatial}}(e(s_{j(i)}), \tau_i(e(B_i))$, for $i = 1, \ldots, n$.

If there exists a transformation bringing components into shapes exactly, every difference is 0, so Formula (4) yields 0, and so sim_{spatial} raises to 1 for every component. The more an arrangement is scattered with reference to the query arrangement, the higher its maximum difference.

Figure 6 pictures a *SVG* query and a database *SVG* fragment. In the first step the algorithm considers the object O_0^q in the query i.e., the resistor and the pivot vector with origin in the resistor and vertex in the next object, O_1^q, the voltage source; it computes the angles α_0 obtained considering the pivot vector and the vector with origin in the next object O_1^q; in the same manner it computes the angle α_1, considering the pivot vector and the vector with vertex in object O_3^q. In the next step, the algorithm considers a different pivot vector with origin in O_0^q and vertex in the next object O_2^q and extracts the remaining angle α_1.

The corresponding angles are extracted for the database fragment; in figure 6 they are noted as β_i. The algorithm provides a scale-rotation-translation invariant measure of similarity. For a given object, the algorithm computes the maximum error between

corresponding angles. Similarity measure is obtained as a function of the maximum errors for all the groups of objects.

Position similarity. Finally, $sim_{\text{position}}(e(s), \tau(e(B))$ is computed as the Euclidean distance between the centroid of a shape in the query and the corresponding shape in a database fragment. The centroid is assumed as the center of the shape for circles and ellipses, for a rectangle it is the intersection between the axis relative to the base and to the height. For polygons and polylines it is computed as the arithmetic mean of the vertices coordinates.

It can be proved that the similarity measure obtained in this way from Formula (4), although it is a pessimistic approximation of Formula (2), is monotonic with reference to the selectivity of details.

4. Knowledge based system for SVG retrieval

Using our logical language as a formal specification, we implemented a prototype system to evaluate the effectiveness of the approach. The knowledge based system supports the following functionalities: new *SVG* document insertion; query and retrieval. Documents to be indexed can be either created by scratch using the system, or simply loaded into the knowledge base.

New graphics are created by composing basic shapes with the aid of a graphical interface. At conceptual level, such functionalities are obtained managing a hierarchical graph to represent and organize *SVG* basic shapes and documents. As usual in DLs an insertion and a query are conceptually similar, as they both consist of a description insertion; the position is determined considering the descriptions the new one is subsumed by; in a query all descriptions tied to the query or below it in the hierarchy are retrieved, therefore we limit our description to a new insertion. Predefined basic shapes belong to the higher level of the hierarchy. More complex shapes are obtained by combining such elementary shapes and/or by applying transformations to basic shapes.

An *SVG* fragment is linked to a node N if it contains the object or the basic shape corresponding to the node. Notice that, as pointed out in the previous section, basic shapes can be subject to transformations that make them quite similar to other basic shapes, the simplest example being a closed polyline with five vertices identical to a rectangle. So the system will obviously carry out a preliminary check on shape similarity not only considering *element* type description, but also the actual geometric appearance which we consider meaning of the description. This implies that a given *SVG* description can be subsumed by (i.e., classified) various basic elements. Figure 7 shows the structure of the hierarchy with reference to an actual example.

New insertion/query. A new description is inserted in the knowledge base as a new node. The insertion is carried out through a search process in the hierarchy to find the exact position where the new description has to be inserted. The position is determined considering the descriptions that the new one is subsumed by. Once the position has been found, the shapes or fragments that are recognized in the new description are linked to it. Basic shapes have no parents, so they are at the top of the hierarchy. Complex objects are linked to the basic shapes they contain. *SVG* documents are linked to the basic shapes or to the node containing a group of shapes whose configuration is similar to those present in the document.

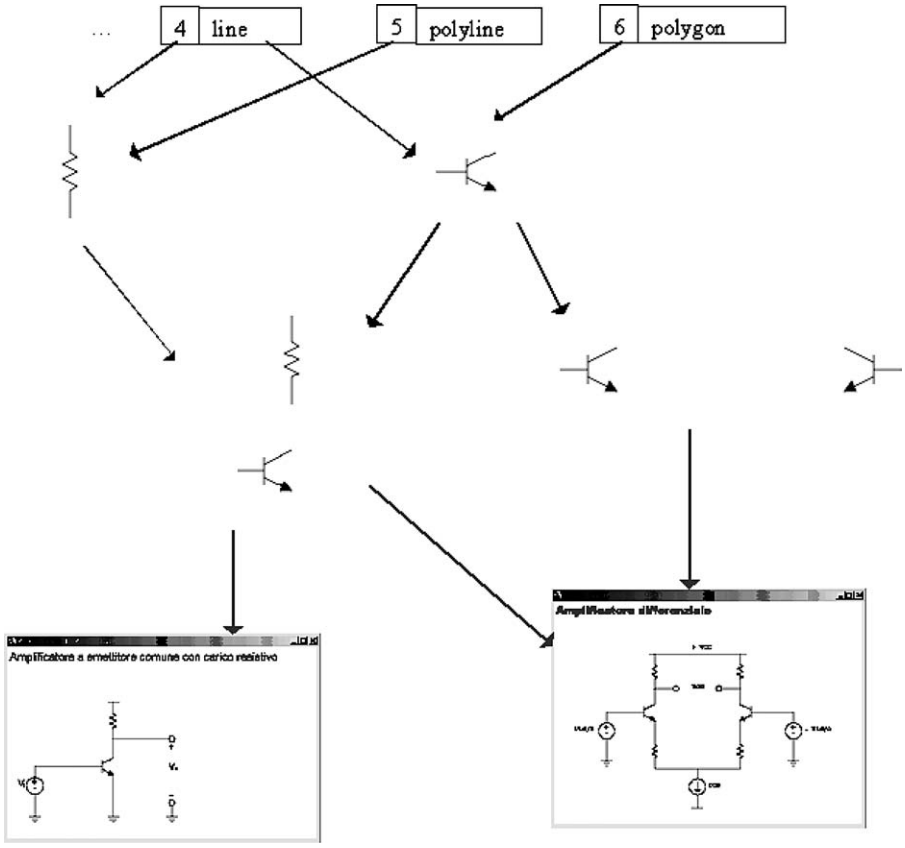

Figure 7. Conceptual hierarchy of shapes in the knowledge base.

The first step of the algorithm searches in the top level of the graph the parent nodes N_i of the new object. The set $G = \{N_0, \ldots, N_{g-1}\}$ is filled with those nodes corresponding to the basic shape recognized in the new object. In the next steps the algorithm performs a depth-first search in the graph for each node $N_i \in G$. Given the set C of child nodes containing objects of O, the elements $C_i \in C$ replace their parents in G only if all parent nodes belong to G. At the end of the search for all the nodes in G, the set G will contain the direct ancestors of the new node.

The algorithm determines the set H_O of graphical documents that might contain the new object O. Given a node N_i, the set of documents linked to the node N_i or to a derived node is obtained as:

$$X_{N_i} = \left[\bigcup_{j=1}^{n_D} X_{D_j} \right] \cup I_{N_i}$$

where I_{N_i} is the set of documents linked to N_i and D_j is a node derived by N_i. Given the set $G = \{N_0, \ldots, N_{G-1}\}$ of parents of O, the set of documents to link to the new shape is:

$$H_O = \bigcup_{i=0}^{n_G-1} X_{N_i}$$

H_O is the set of documents containing the basic shapes of O. The set $T_O \subseteq H_O$ contains the documents in H_O that actually contain O. Given the set of documents linked to the nodes $N_i \in G$:

$$M_O = \bigcup_{i=0}^{n_G-1} I_{N_i}$$

is determined.

The links to documents in the set $T_O \cap M_O$ are moved to the new node N and links to documents in $[T_O - (T_O \cap M_O)]$ are copied in N instead of being moved.

The behavior of the system obviously depends both on the values given to the coefficients $\alpha, \beta, \gamma, \delta, \eta$ in Formula (4), which weight the relevance features have in the similarity computation, and on the values given to fx, fy in the smoothing function Φ. Table 1 contains the parameters set for the similarity measure, and the f_x and f_y values for the Φ function.

Table 1. Configuration parameters.

Parameter	Value
α (Shape similarity)	0.30
γ (Spatial similarity)	0.30
β (Color similarity)	0.1
η (Position similarity)	0.1
δ (Transformation similarity)	0.2
Shape similarity sensitivity f_x	60
Shape similarity sensitivity f_y	0.5
Spatial similarity sensitivity f_x	20
Spatial similarity sensitivity f_y	0.2
Color similarity sensitivity f_x	10
Color similarity sensitivity f_y	0.1
Position similarity sensitivity f_x	100
Position similarity sensitivity f_y	0.4
Transformation similarity sensitivity f_x	0.5
Transformation similarity sensitivity f_y	0.5
Global shape threshold	20%
Global position threshold	200

Distinguishing aspects of our approach include the extensional semantics, which is compositional and enforces the downward refinement property. We remark here that compositional means that retrieval of a document takes place only if at least all objects in the query are present in it. Obviously the retrieved documents can have far more objects, but the approach requires all the objects in the query be somewhere present. This tends to increase precision, also at the expenses of recall, thus we may accept false negatives, but prevent false positives.

Figure 8 shows that our system supports subsumption as reasoning service. Consider for example the query n. 4 that requires the system to retrieve all the documents containing chemical structures with at least a molecule of hydrogen and two of carbon. All the documents the system considers relevant are shown below. Refining the query as in query n.11 that is subsumed by query n.4 since it contains 1 molecule of carbon, 1 of hydrogen and 1 of oxygen, the system discards all previous results except for the document containing 1 molecule of oxygen.

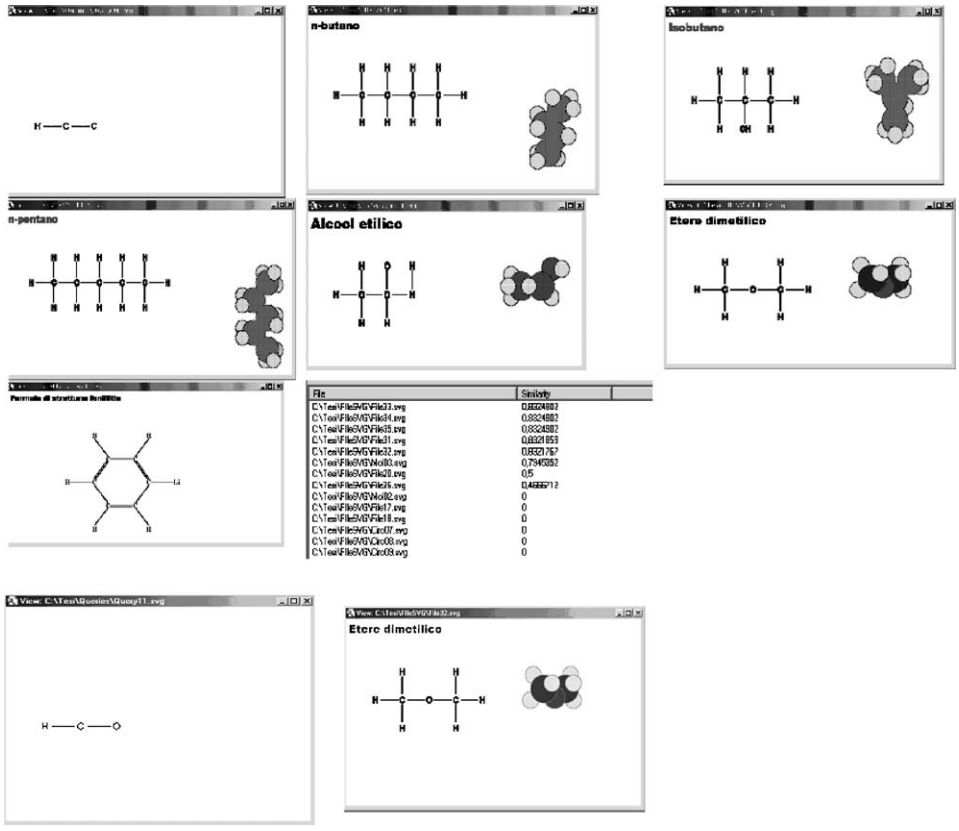

Figure 8. A subsumption example.

It is noteworthy that our system can also deal with text, which in the *SVG* standard is treated as an element. In this case the similarity value is endowed of another term. We currently use as measure a simple Edit-distance based algorithm. Since it is not at the core of our approach we did not use this term in the experiments.

5. Experiments

In order to assess the performance of the proposed approach and of the system implementing it, we have carried out a set of experiments on a test dataset of *SVG* documents. The experimental framework is largely based on the one proposed in [23], which relies on a comparison of the system performances versus the judgement of human experts.

The test data set consists of a collection of 48 *SVG* documents picturing chemical structures, electronic circuits and general subject graphics not concerning a specific theme; a sample of them is shown in figures 9 and 10. We selected from the test data set 20 documents to be used as queries.

We separately asked two volunteers to classify in decreasing order, according to their judgment, the 48 documents based on their similarity to each *SVG* graphic of the selected query set. The volunteers had never used the system and they were only briefly instructed

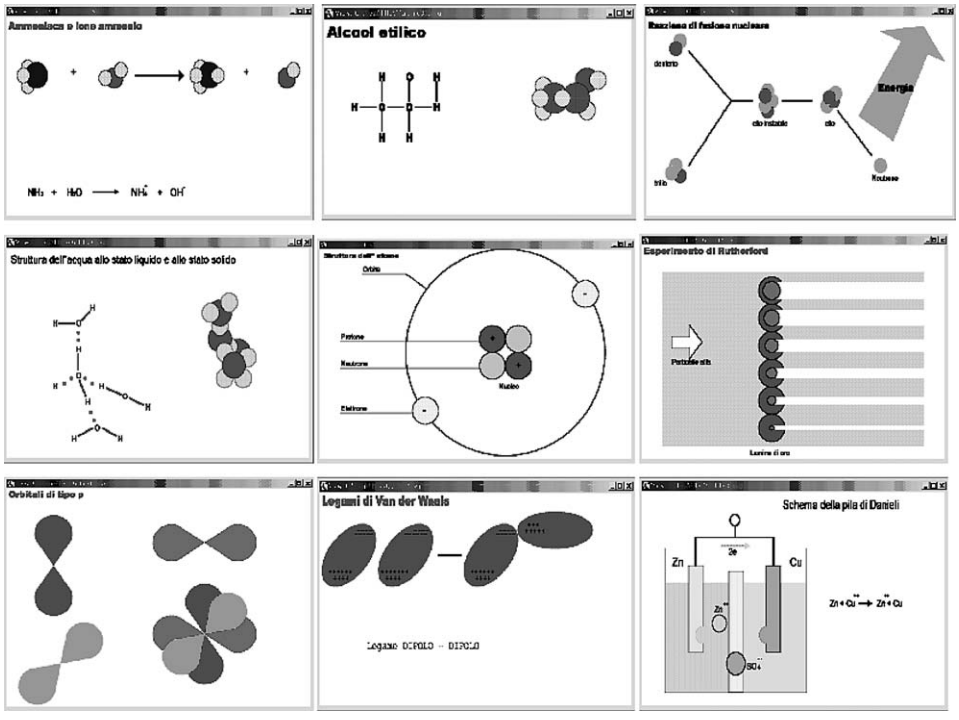

Figure 9. A sample of graphics used in the experiments picturing chemical items.

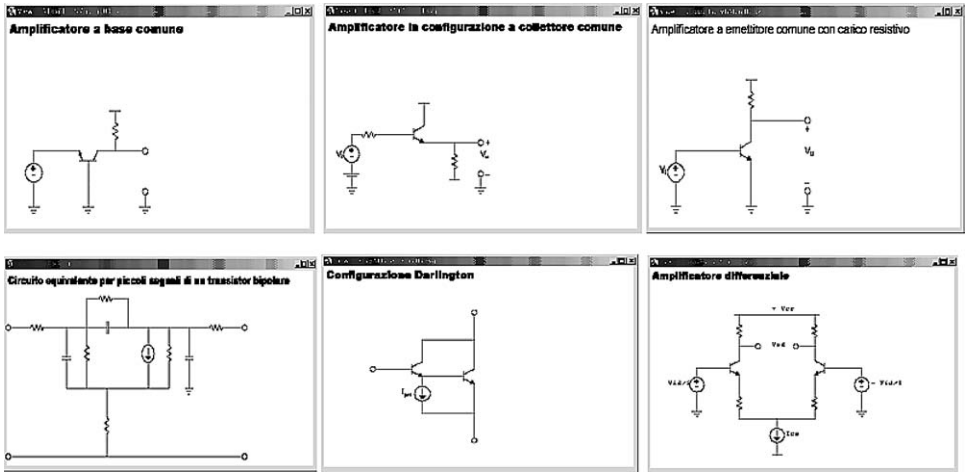

Figure 10. A sample of graphics used in the experiments picturing electronic circuits.

that rank orderings had to be based on the degree of conformance of the database documents with the queries. They were allowed to group documents when considered equivalent in content, and for each query, to discard documents that were judged wholly dissimilar from the query.

Having obtained two classifications, which were not univocal, we created the final ranking merging the previous similarity rankings according to a minimum ranking criterion. The final ranking of each document with respect to a query was determined as the minimum one among the two available.

Then we submitted the same set of 20 queries to the system, whose knowledge base was loaded only with the 48 documents of the test set. The resulting classification gave us what was called a system-provided ranking. Again following the framework in [23], we adopted the R_{norm} [6] as quality measure of the retrieval effectiveness.

R_{norm} values are in the range [0, 1]; a value of 1 corresponds to a system-provided ordering of the database images that is either identical to the one provided by the human experts or has a higher degree of resolution, lower values correspond to a proportional disagreement between the two.

Figure 12 shows results for queries shown in figure 11; the final average is $R_{\text{norm}} = 0.967$. R_{norm} results are nearly always equal to 1, this means that the system provided results are almost always similar to user-provided ones.

In order to provide also other typical information retrieval measures we report here results in terms of precision and recall, determined using the the same users' provided rankings. Average measures are the following ones: $Precision = 0.95$ $Recall = 0.87$. Details for single queries are shown in figure 13.

Figure 14 shows results for Query 3, which was the one with the worst Precision and Recall. As a matter of fact documents that would have drastically improved the ranking where just below the overall similarity threshold.

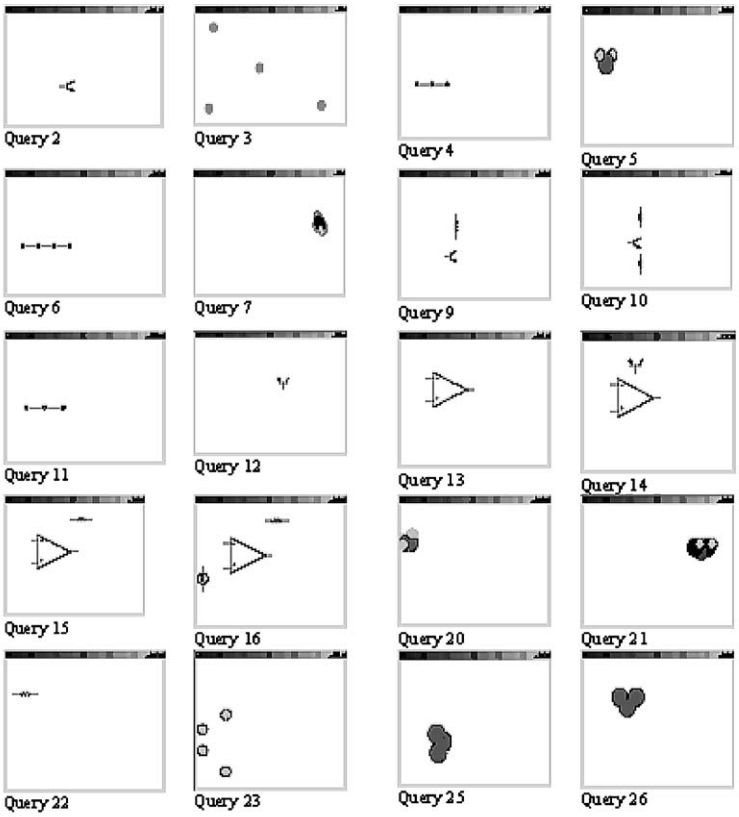

Figure 11. The set of tested queries.

It is noteworthy that we also carried out a simple test using a full text retrieval tool. Element types were included in the indexed terms, while various other tags were considered stop-words. We indexed the same *SVG* documents and carried out retrieval with the same queries previously used and a *cut-off* at 6 documents. Average precision and recall were the following ones: $Precision = 0.58$ $Recall = 0.54$.

6. Related work

As obvious this work is closely related to content based image retrieval. While we do not report here on several papers on low-level feature based approaches, see e.g., [2] for a recent survey, we briefly revise here some of the related work on retrieval by spatial similarity and logic based approaches.

Retrieval by spatial similarity dates back to the work by Chang et al. [11]. In that work the modeling of "symbolic images" or icons was presented, with icons identified by a

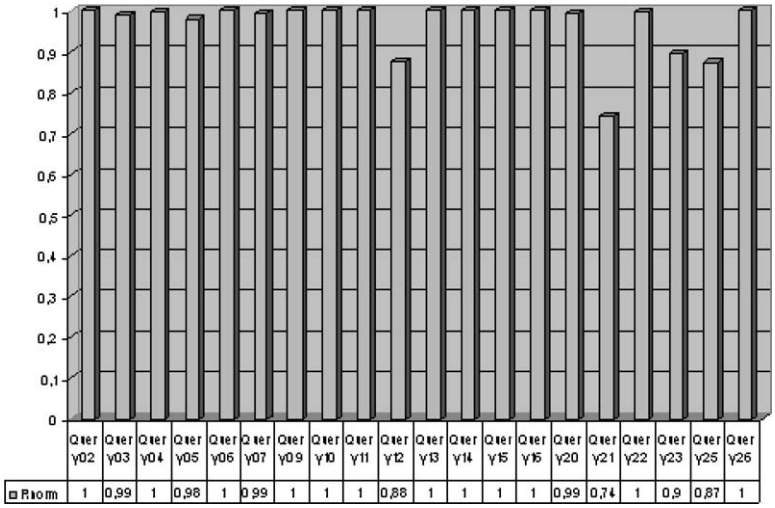

Figure 12. Detail of R_{norm} values.

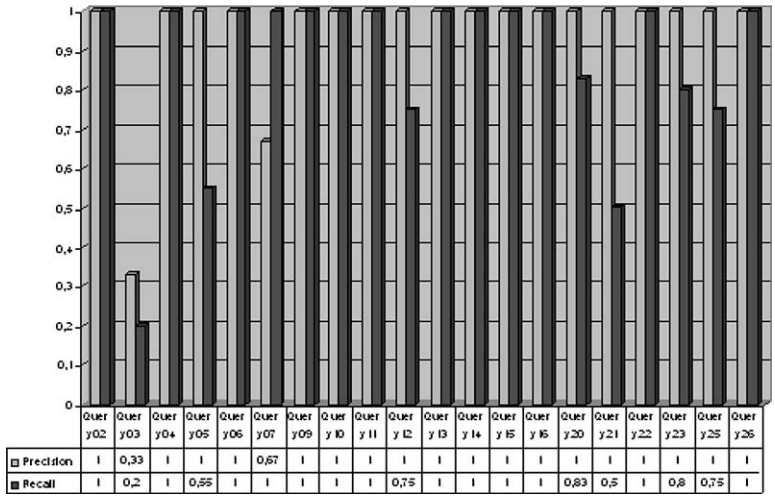

Figure 13. Precision and recall values.

single point in the 2D-space in terms of *2D-strings*, each of the strings accounting for the position of icons along one of the two planar dimensions. In this approach retrieval of images basically reverts to simpler string matching. Also Del Bimbo [14] proposed a simple spatial logic for symbolic descriptions. Gudivada and Raghavan [23] consider the objects in a symbolic image associated with vertexes in a weighted graph. Edges—i.e., lines connecting the centroids of a pair of objects—represent the spatial relationships among the objects and are associated with a weight depending on their slope. The symbolic image

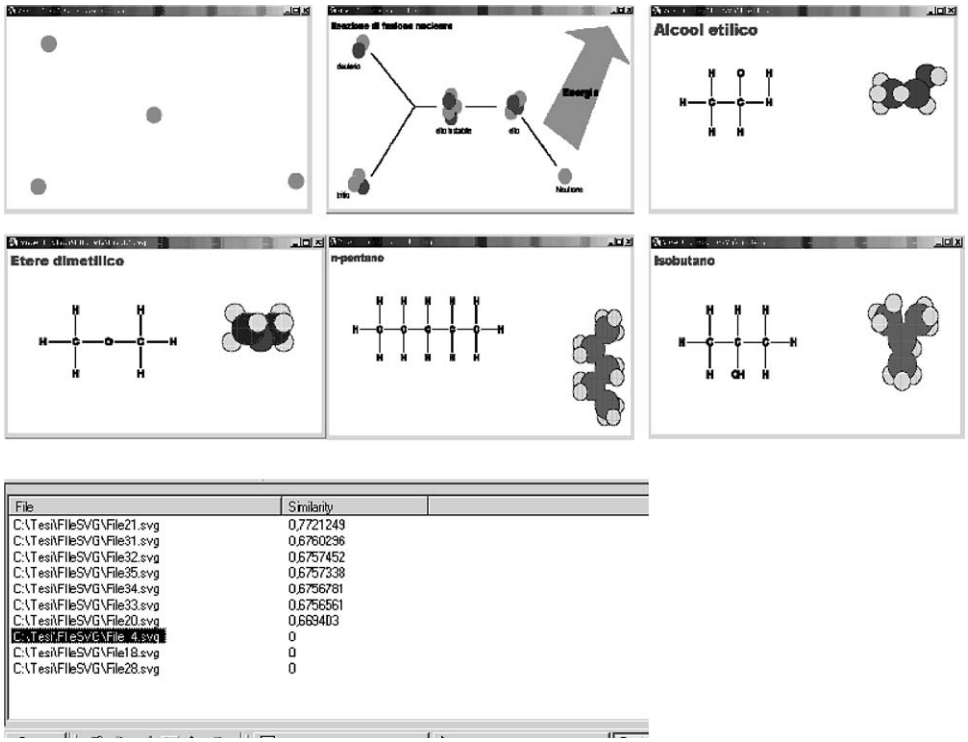

Figure 14. Results for query 3.

is represented as an edge list. Given the edge lists of a query and a database image, a similarity function computes the degree of closeness between the two lists as a measure of the matching between the two spatial-graphs. The similarity measure depends on the number of edges and on the comparison between the orientation and slope of edges in the two spatial-graphs. The algorithm is robust with respect to scale and translation variants in the sense that it assigns the highest similarity to an image that is a scale or translation variant of the query image. An extended algorithm includes also rotational variants of the original images.

More recent papers on the topic include those by Gudivada [22] and by El-Kwae and Kabuka [20], which basically propose extensions of the strings approach for efficient retrieval of subsets of icons. Gudivada [22] defines θR-strings, a logical representation of an image. Such representation also provides a geometry-based approach to iconic indexing based on spatial relationships between the iconic objects in an image individuated by their centroid coordinates. Translation, rotation and scale variant images and the variants generated by an arbitrary composition of these three geometric transformations are considered. The approach does not deal with object shapes, nor with other basic image features, and considers only the sequence of the names of the objects. The similarity between a database and a query image is obtained through a spatial similarity algorithm that measures the

degree of similarity between a query and a database image by comparing the similarity between their θR-strings. The algorithm recognizes rotation, scale and translation variants of the image and also subimages, as subsets of the domain objects. A constraint limiting the practical use of this approach is the assumption that an image can contain at most one instance of each icon or object. Di Sciascio et al. [15] extended this approach to include multiple instances.

El-Kwae and Kabuka [20] propose a further extension of the spatial-graph approach, which includes both the topological and directional constraints. The topological extension of the objects can be obviously useful in determining further differences between images that might be otherwise considered similar by a directional algorithm only computing the locations of objects in term of their centroids. The similarity algorithm proposed here extends the graph-matching one previously described by Gudivada and Raghavan [23]. The similarity between two images is based on three factors: the number of common objects, the directional and topological spatial constraint between the objects. The similarity measure includes the number of objects, the number of common objects and a function that determines the topological difference between corresponding objects pairs in the query and in the database image. The algorithm retains the properties of the original approach, including its invariance to scaling, rotation and translation and is also able to recognize multiple rotation variants. Cardoze and Schulman [9] solved the nearly-exact point pattern matching problem with efficient randomized methods, but without scaling. They also observed that best match is a more difficult problem than nearly-exact match. Also Chew et al. [12] proposed a method for best match of shapes, but they analyzed only rigid motions without scaling.

With reference to previous work on vision in artificial intelligence, the use of structural descriptions of objects for the recognition of their images can be dated back to Minsky's frames, and work carried out by Brooks [8]. The idea is to associate parts of an object (and generally of a scene) to the regions an image can be segmented into. The hierarchical organization of knowledge to be used in the recognition of an object was first proposed by Marr [27]. Reiter and Mackworth [31] proposed a formalism to reason about maps as sketched diagrams. In their approach, the possible relative positions of lines are fixed and highly qualitative (touching, intersecting).

Structured descriptions of three-dimensional images are already present in languages for virtual reality like VRML [26] or hierarchical object modeling. However, the semantics of such languages is operational, and no effort is made to automatically classify objects with respect to the structure of their appearance.

Meghini et al. [28] proposed a formalism integrating DLs and image and text retrieval, while Haarslev et al. [24] integrate DLs with spatial reasoning. Further extensions to the approach are described by Moeller et al. [29]. Both proposals build on the clean integration of DLs and *concrete domains* of Baader and Hanschke [4]. However, neither of the formalisms can be used to build complex shapes by nesting more simple shapes. Moreover, the proposal by Haarslev et al. [24] is based on the logic of spatial relations named RCC8, which is enough for specifying meaningful relations in a map, but it is too qualitative to specify the relative sizes and positions of regions in a complex shape.

Also for [25] DLs and concrete domains are at the basis of a logical framework for image databases aimed at reasoning on query containment. Unfortunately, the proposed

formalism cannot consider geometric transformations neither determine specific arrangements of shapes.

More similar to our approach is the one by Ardizzone et al. [3], where parts of a complex shape are described with a DL. However, the composition of shapes does not consider their positions, hence reasoning cannot take positions into account.

Relative position of parts of a complex shape are expressed in a constraint relational calculus in the work by Bertino and Catania [5]. However, reasoning about queries (containment and emptiness) is not considered in this approach. Aiello [1] proposes a multi-modal logic, which provides a formalism for expressing topological properties and for defining a distance measure among patterns.

Spatial relation between parts of medical tomographic images are considered by Tagare et al. [36]. There, medical images are formed by the intersection of the image plane and an object. As the image plane changes, different parts of the object are considered. Besides, a metric for arrangements is formulated by expressing arrangements in terms of the Voronoi diagram of the parts. The approach is limited to medical image databases and does not provide geometrical constraints.

Compositions of parts of an image are considered in the work by Sanfeliu and Fu [33] for character recognition. However, in recognizing characters, line compositions are "closed", in the sense that one looks for the specified lines, and no more. Instead in our framework, the shape "F" composed by three lines, is subsumed by the shape "Γ"—something unacceptable in recognizing characters. Apart from the different task, this approach does not make use of an extensional semantics for composite shapes, hence no reasoning is possible.

A logic-based multimedia retrieval system was proposed by Fuhr et al. [21]; the method, based on an object-oriented logic, supports aggregated objects but it is oriented towards a high-level semantic indexing, which neglects low-level features that characterize images and parts of them.

In the field of computation theories of recognition, we mention two approaches that have some resemblance to our own: Biederman's structural decomposition and geometric constraints proposed by Ullman, both described by Edelmann [19]. Unfortunately, neither of them appears suitable for realistic image retrieval: the structural decomposition approach does not consider geometric constraints between shapes, while the approach based on geometric constraints does not consider the possibility of defining structural decomposition of shapes, hence reasoning on them.

7. Conclusion and future work

We proposed a logical language for for describing *SVG* documents as composition of basic elements and transformations and gave an extensional semantics to queries, in terms of sets of retrieved documents. The composition of basic elements is made possible by the explicit use in our language of geometric transformations. The extensional semantics allows us to properly define subsumption (i.e., containment) between queries. The approximate reasoning allows to define ranking among documents in the retrieved set.

Borrowing from Description Logics, we stored descriptions in a subsumption hierarchy. The hierarchy provides a semantic index to the *SVG* documents in a database. The

logical semantics allowed us to define other reasoning services aside form the retrieval one, namely: the recognition of a shape arrangement in a *SVG* document or fragment, the classification with reference to a hierarchy of descriptions, and subsumption between descriptions.

A prototype system has been built implementing the approach and experiments on retrieval performances although not carried out on large databases, which by the way are currently unavailable, show the good correspondence of the system behavior with users judgement.

Acknowledgments

We wish to thank our former student D. Giordano for useful software implementations. This work was supported by MIUR FESR project *Sistemi Informativi di Immagini*, PON project *Tecnologie innovative per la valorizzazione e la fruizione dei Beni Culturali*, and CNR projects *LAICO* and *DEMAND*.

References

1. M. Aiello, "Computing spatial similarity by games," in AI*IA-01, F. Esposito (Ed.), No. 2175 in Lecture Notes in Artificial Intelligence, Springer-Verlag, 2001, pp. 99–110.
2. S. Antani, R. Kasturi, and R. Jain, "A survey on the use of pattern recognition methods for abstraction, indexing and retrieval of images and video," Pattern Recognition, Vol. 35, No. 4, pp. 945–965, 2002.
3. E. Ardizzone, A. Chella, and S. Gaglio,"Hybrid computation and reasoning for artificial vision," in Artificial Vision, Academic Press, V. Cantoni, S. Levialdi, and V. Roberto (Eds.), 1997, pp. 193–221.
4. F. Baader and P. Hanschke, "A schema for integrating concrete domains into concept languages," in Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI91), Sydney, 1991, pp. 452–457.
5. E. Bertino and B. Catania, "A constraint-based approach to shape m anagement in multimedia databases," MultiMedia Systems, Vol. 6, pp. 2–16, 1998.
6. P. Bollmann, F. Jochum, U. Reiner, V. Weissmann, and H. Zuse, "The LIVEproject-retrieval experiments based on evaluation viewpoints," in SIGIR-85, ACM New York, 1985, pp. 213–214.
7. A. Borgida, "Description logics in data management," IEEE Transactions on Knowledge and Data Engineering, Vol. 7, No. 5, pp. 671–682, 1995.
8. R. Brooks, "Symbolic reasoning among 3-D models and 2-D images," Artificial Intelligence, Vol. 17, pp. 285–348, 1981.
9. D. Cardoze and L. Schulman, "Pattern matching for spatial point sets," in Proceedings of the Thirtyninth Annual Symposium on the Foundations of Computer Science (FOCS98), Palo Alto, CA, 1998, pp. 156–165.
10. A. Celentano and E. Di Sciascio, "Features integration and relevance feedback analysis in image similarity evaluation," Journal of Electronic Imaging, Vol. 7, No. 2, pp. 308–317, 1998.
11. S. Chang, Q. Shi, and C. Yan, "Iconic indexing by 2D strings," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 9, No. 3, pp. 413–428, 1983.
12. L. Chew, M. Goodrich, D. Huttenlocher, K. Kedem, J. Kleinberg, and D. Kravets, "Geometric pattern matching under euclidean motion," Computational Geometry, Vol. 7, pp. 113–124, 1997.
13. I. Cox, M. Miller, T. Minka, and T. Papathomas, "The bayesian image retrieval system, picHunter," IEEE Transactions on Image Processing, Vol. 9, No. 1, pp. 20–37, 2000.
14. A. Del Bimbo, "Visual Information Retrieval, Morgan Kaufmann ed., 1999.
15. E. Di Sciascio, F. Donini, and M. Mongiello, "Spatial layout representation for query by sketch content based image retrieval," Pattern Recognition Letters, Vol. 23, No. 13, pp. 1599–1612, 2002a.

16. E. Di Sciascio, F. Donini, and M. Mongiello, "Structured knowledge representation for image retrieval," Journal of Artificial Intelligence Research, Vol. 16, pp. 209–257, 2002b.
17. E. Di Sciascio and M. Mongiello, "Query by sketch and relevance feedback for content-based image retrieval over the Web," Journal of Visual Languages and Computing, Vol. 10, No. 6, pp. 565–584, 1999.
18. F. Donini, M. Lenzerini, D. Nardi, and A. Schaerf, "Reasoning in description logics," in Foundations of Knowledge Representation, G. Brewka (Ed.) CSLI-Publications, 1996, pp. 191–236.
19. S. Edelmann, Representation and Recognition in Vision, The MIT Press, 1999.
20. E. El-Kwae and M. Kabuka, "A robust framework for content-based retrieval by spatial similarity in image databases," ACM Transactions on Information Systems, Vol. 17, No. 2, pp. 174–198, 1999.
21. N. Fuhr, N. Gövert, and T. Rölleke, "DOLORES: A system for logic-based retrieval of multimedia objects," in Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Developement in Information Retrieval (SIGIR 98), Melbourne, Australia, 1998, pp. 257–265.
22. V. Gudivada, "θR-String: A geometry-based representation for Efficient and effiective retrieval of images by spatial similarity," IEEE Transactions on Knowledge and Data Engineering, Vol. 10, No. 3, pp. 504–512, 1998.
23. V. Gudivada and J. Raghavan, "Design and eevaluation of algorithms for image retrieval by spatial similarity," ACM Transactions on Information Systems, Vol. 13, No. 2, pp. 115–144, 1995.
24. V. Haarslev, C. Lutz, and R. Möeller, "Foundations of spatioterminological reasoning with description logics," in Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98), 1998, pp. 112–123.
25. M.-S. Hacid and C. Rigotti, "Representing and reasoning on conceptual queries over image databases," in Proceedings of the Twelfth International Symposium on Methodologies for Intelligent Systems (ISMIS99), Springer-Verlag: Warsaw, Poland, 1999, pp. 340–348.
26. J. Hartman and J. Wernecke, The VRML 2.0 Handbook, Addison-Wesley, 1996.
27. D. Marr, Vision, W.H. Freeman and Co., Oxford, 1982.
28. C. Meghini, F. Sebastiani, and U. Straccia, "A model of multimedia information retrieval," Journal of the ACM, Vol. 48, No. 5, pp. 909–970, 2001.
29. R. Moeller, B. Neumann, and M. Wessel, "Towards computer vision with description logics: Some recent progress," in Proceedings of the IEEE Integration of Speech and Image Understanding, 1999, pp. 101–115.
30. B. Nebel, Reasoning and Revision in Hybrid Representation Systems, No. 422 in Lecture Notes in Artificial Intelligence, Springer-Verlag, 1990.
31. R. Reiter and A. Mackworth, "A logical framework for depiction and image interpretation," Artificial Intelligence, Vol. 41, No. 2, pp. 125–155, 1989.
32. Y. Rui, T. Huang, and S. Mehrotra, "Content-based image retrieval with relevance feedback in MARS," in Proceedings of the IEEE International Conference on Image Processing (ICIP'97), 1997, pp. 815–818.
33. A. Sanfeliu and K. Fu, "A distance measure between attributed relational graphs for pattern recognition," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 13, No. 3, pp. 353–362, 1983.
34. Scalable Vector Graphics Specification, http://www.w3.org/TR/SVG/. 2001.
35. U. Straccia, "Reasoning within fuzzy description logics," Journal of Artificial Intelligence Research, Vol. 14, pp. 137–166, 2001.
36. H. Tagare, F. Vos, C. Jaffe, and J. Duncan, "Arrangement: A spatial relation between parts for evaluating similarity of tomographic Section," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 17, No. 9, pp. 880–893, 1995.
37. J.D. Ullman, Principles of Database and Knowledge Base Systems, Vol. 1. Computer Science Press: Potomac, Maryland, 1988.
38. W.A. Woods and J.G. Schmolze, "The KL-ONE family," in Semantic Networks in Artificial Intelligence, F.W. Lehmann (Ed.), Pergamon Press, 1992, pp. 133–178. Published as a special issue of Computers & Mathematics with Applications, Vol. 23, No. 2–9.
39. J. Yen, "Generalizing term subsumption languages to fuzzy logic," in Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI'91), 1991, pp. 472–477.
40. L. Zadeh, "Fuzzy sets," Information and Control, Vol. 8, pp. 338–353, 1965.

Eugenio Di Sciascio received the laurea "cum laude" degree in Electronics Engineering from University of Bari in 1989 and the Ph.D. degree in 1994 from Technical University of Bari. In 1992 he joined the University of Lecce as an assistant professor. He is currently an associate professor in Information Technology Engineering at Technical University of Bari, where he is scientific responsible for several research projects. His research interests include, image and video processing, multimedia information retrieval, knowledge representation and knowledge based systems for E-commerce. On these topics he has authored or co-authored over eighty papers in international journals and international conferences.

Francesco M. Donini got the Master in Electronics Engineering from the University of Rome "La Sapienza" in 1988. He got the Ph.D. in Computer Science in the same university in 1992. From 1991 to 1998 he has been researcher/assistant professor at the Department of Computer and System Science of University of Rome "La Sapienza". Since 1998 he is an associate professor, first at Technical University of Bari, and currently at University of Tuscia in Viterbo. He is coauthor of many papers in international journals, as well as international conferences. The paper "Tractable concept languages", presented at the conference IJCAI-91 (1991), received the best paper award. He is responsible of local university research projects since 1996, and of CNR research projects. He is editor of the area "Concept-Based Knowledge Representation" for the journal ETAI—Electronic Transactions on Artificial Intelligence—published by the Royal Swedish Academy.

Marina Mongiello received the laurea "cum laude" degree in Computer Science from University of Bari in 1993 and the Ph.D. degree in Electronics Engineering from University of Catania in 2001. In 2002 she joined the Technical University of Bari as an assistant professor. Her research interests include knowledge based systems for E-commerce and multimedia information retrieval. On these topics she published many papers in international journals and conferences.