A Knowledge Based System for Content-based Retrieval of Scalable Vector Graphics Documents

Eugenio Di Sciascio
Politecnico di Bari
Via Re David, 200
I-70125, Bari, Italy
disciascio@poliba.it

Francesco M. Donini
Università della Tuscia
via San Carlo, 32
I-01100, Viterbo, Italy
donini@unitus.it

Marina Mongiello
Politecnico di Bari
Via Re David, 200
I-70125, Bari, Italy
mongiello@poliba.it

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

A Knowledge Based System for Content-based Retrieval of Scalable Vector Graphics Documents

ABSTRACT
Scalable Vector Graphics (SVG), the novel XML based language for describing two-dimensional graphics, is now a W3C standard and is likely to become popular on the Internet, due to its inherent advantages over raster image formats in several domains. We present a system for semantic based retrieval by content of SVG. The system is endowed of a web crawler for documents search and a graphical interface for query by sketch. The approach adopted in the system implements a simple description logic devised for the semantic indexing and retrieval of complex objects. Its syntax allows to describe basic shapes and complex objects as compositions of basic ones, and transformations. Its extensional semantics, which is compositional, allows to define retrieval, classification, and subsumption services. An experimental evaluation is also presented, which shows results obtained in terms of precision and recall, but also points out that there are still few SVG documents available on the Web.

Categories and Subject Descriptors
H.4.2 [Information Systems Applications]: Decision Support; I.2.4 [Knowledge Representation Formalisms and Methods]: Representation languages

1. INTRODUCTION
Scalable Vector Graphics (SVG) language [18] is now a W3C approved standard for describing two-dimensional graphics at the level of graphical objects rather than individual points, with descriptions based on XML. 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 Content-based image retrieval (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. SVG provides basic geometric shapes i.e., rectangle, circle, ellipse, line, polyline, polygon. 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. 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 g element contains the SVG fragment between a pair of syntactic parenthesis. 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. 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, named Current Transformation Matrix (CTM), which defines the mapping from the user coordinate system into the viewport coordinate system. Translation, Scaling, Rotation, Shearing can be applied to basic shapes or composition of shapes and multiple transformations can be applied, too, obtained as the product of the CTM matrices. The 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. To this aim we propose a knowledge representation approach that is based on a logical language. Obviously, we borrow from works on classical feature-level CBIR (see [2] for a recent survey), spatial similarity algorithms [5, 12, 9, 6], and logic based approaches [17, 15, 7, 14, 16, 1, 11, 8]. Our language, devised along the lines of Description Logics [10] (DL), 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. A Semantic Data Model ensues, in which the various characteristic features of each component are given an explicit notation through a geometric transformation. The semantics allows defining hierarchical descriptions, based on set containment between interpretation of descriptions. Retrieval corresponds in this setting to

2004 ACM Symposium on Applied Computing
interpretations satisfying a description.

To manage and query the SVG 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. 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 present our logical language in terms of syntax and semantics and available reasoning services. We then propose algorithms used to compute similarities in the indexing and retrieval phases. The prototype system implementing the logic is described in section 4, together with experiments carried out to validate the approach. Last section draws the conclusions.

2. PROPOSED LANGUAGE

To represent graphic objects in the SVG standard, basic features are included in the object definition as attributes. Nevertheless user-defined shapes can be, through transformations, made completely different from their prototypical basic shape, to visually resemble e.g., other basic shapes. We cope with this situation defining a proper semantics for SVG documents. Our language is based on a simple DL and is adapted from a formalism defined in [7]. Syntax. The syntax of our language is defined 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. Formally, basic shapes are denoted with the letter B, and have a contour e(B) characterizing them. We assume that e(B) is described in a space whose origin coincides with the centroid of B. Basic shapes, as defined in the SVG standard are circle, rectangle, ellipse, line, polyline, polygon. The possible transformations are those defined in SVG. We globally denote a transformation as τ. Transformations can be composed in sequences τ1 ⋯ τn, 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. The basic building block of our syntax is a basic shape component B[c, τ], which represents a basic shape B with color c and edge contour τ(e(B)). With τ(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

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

This syntax is just an internal representation of a composite shape for our system. 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. 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 D is a set of SVG fragments and the interpretation of an object composed by basic shapes is a subset of the domain D, in which the basic shapes have the proper features. 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 (I, D), where D is a set of SVG fragments, and I is a mapping from shapes and components to subsets of D. 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 ∈ D belongs to the interpretation of a basic shape component B[τ] if F contains a shape whose contour matches τ(e(B)). The definition can be extended to approximate recognition as follows. Notice 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 ∈ S, f_s(x) = 0 otherwise. We consider now the characteristic function of the set B[τ]. Let F be a SVG fragment; if F belongs to B[τ], then the characteristic function computed on F has value 1, otherwise it has value 0. To keep the number of symbols low, we use the expression B[τ] also to denote the characteristic function (with an argument F) to distinguish it from the set.

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

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

\[ B[\tau](F) = \max_{s \in F} \text{sim}(e(s), \tau(e(B))) \]

Note that \( \text{sim} \) 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 \( \tau \). 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

\[ B^*(\tau) = \max_{s \in F} \text{sim}(e(s), \tau(e(B))) \]

The maximization over all possible transformations maxc can be effectively computed by using a similarity measure \( \text{sim}_{\text{ax}} \), that is invariant with respect to transformations (see Section 3). In this way, a basic shape B can be used as a query to retrieve all fragments from D which are in B^*. 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 \( B_1[\tau_1] \cap \cdots \cap B_n[\tau_n] \). For approximate matching the interpretation of a composite shape is:

\[ C^*(F) = \max_{\tau} \{ \min_{i=1}^n \{ B_i[(\tau \circ \tau_i)^*(F)] \} \} \] (1)

3. SIMILARITY COMPUTATION

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 = \( B_1[c_1, \tau_1] \cap \cdots \cap B_n[c_n, \tau_n] \), and a SVG fragment F composed by shapes \( \{s_1, \ldots, s_n\} \). We assume \( n \leq m \), since all components of C must appear in F. Let \( \text{sim}(s, B[c, \tau]) \) be a similarity measure that considers a shape s with its color c(s) and a component B[c, τ] 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 that formula (1) — i.e., the measure of how
well $C$ is recognized in $F$ — can be computed also according to the following formula:

$$\max_{j: \{1, n\} \rightarrow \{1, m\}} \max_{\tau \in \Gamma} \left\{ \frac{1}{n} \min_{i=1}^{n} (\sin(s_{j(i)}, B_i, (\tau \circ \tau_i))) \right\}$$

(2)

where $j: \{1, n\} \rightarrow \{1, m\}$ 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 $B_i[\tau, \tau_i]$ is a component of $C$, then $j$ maps it into shape $s_{j(i)}$ of $F$. The maximization over $\tau$ 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 $\tau$ is still unfeasible [3, 4], we adopt some heuristics to evaluate the above formula. We decompose $\sin(s, B[\tau, \tau_i])$ as a sum of weighted contributions, with each contribution accounting for a feature. 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 summations are obtained. For similarity computation of each term we use the function $\Phi(x, f, x, y)$. 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 $f$, $y$, $y$, as follows.

$$\Phi(x, f, x, y) = \begin{cases} \frac{(1 - f) y - \cos\left(\frac{\pi}{y} f x\right)}{1 - f} & \text{if } 0 \leq x < f x, \\
\frac{1 - f y}{1 - f} & \text{if } x \geq f x. \end{cases}$$

where $f x > 0$ and $0 < y f < 1$.

**Shape similarity.** Similarity of the shapes (translation-rotation-scale invariant) is denoted by $\text{sim}_{\text{shapemap}}(e(s), e(B))$. 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 $\text{sim}_{\text{color}}(e(s), e(c))$ measures the similarity in terms of color appearance as the Euclidean distance between RGB components of $k$-th component of $D$ and the region $s_{j(i)}$. 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 $\tau$ — 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, $\text{sim}_{\text{transf}}$.** 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. **Spatial similarity.** The feature $\text{sim}_{\text{spatial}}(e(s), \tau(e(B)))$ 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; our algorithm extends previous ones [9, 12, 6, 13] 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 $\alpha_{i1}$ 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 $\beta_{j(1)j(i)}$ 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 $s_{j(1)}$ respectively. A pictorial representation of the angles is given in Figure 1. Then we let the difference $\Delta_{\text{spatial}}(e(s_{j(1)}), \tau(e(B_1)))$ be the maximal absolute difference between correspondent angles:

$$\Delta_{\text{spatial}}(e(s_{j(1)}), \tau(e(B_1))) = \max_{i, h=2, \ldots, n, i \neq h} \{|\alpha_{i1} - \alpha_{j(i)j(i)}|\}$$

(3)

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

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

If there exists a transformation bringing components into shapes exactly, every difference is 0, so Formula (3) yields 0, and so $\text{sim}_{\text{spatial}}$ raises to 1 for every component. The more an arrangement is scattered with reference to the other arrangement, the higher its maximal difference. Figure 1 pictures a SVG query and a database SVG fragment.

In the first step the algorithm considers the object $O_1^j$ in the query i.e., the resistor and the pivot vector with origin in the resistor and vertex in the next object, $O_2^j$, the voltage source; it computes the angles with $O_2^j$ obtained considering the pivot vector and the vector with origin in the next object $O_1^j$. In the same manner it computes the angle $\alpha_1$, considering the pivot vector and the vector with vertex in object $O_2^j$. In the next step, the algorithm considers a different pivot vector with origin in $O_2^j$ and vertex in the next object $O_3^j$ and extracts the remaining angle $\alpha_3$. The corresponding angles are extracted for the database fragment; in Figure 1 they are noted as $\beta_3$.

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

![SVG Query](image1.png)

**Figure 1: Representation of angles used for computing spatial similarity of component 1 and region $r_j(1)$.**
is obtained as a function of the maximum errors for all the groups of objects. We use the function \( \Phi(x, f_x, f_y) \).

More formally, the algorithm can be described as follows:

**Algorithm:** skim(g, \( E_u \))

- input: a set \( E_u \) of objects with \( u \) as the number of groups
- output: \( E_u \)

begin

for \( i \in \{1, \ldots, n\} \) do

\[ \Delta_y = \max_{a \in E_u} \Delta_y \]

end

return \( \Delta_y \)

end

**4. SYSTEM AND EXPERIMENTS**

Using our logical language as a formal specification, we implemented a prototype system. Figure 2 shows a snapshot of its user interface. The system includes a web crawler for indexing documents or fragments within a document. The user interface allows to create queries by sketch, by composition of basic shapes, which are automatically translated in the correspondent SVG description.

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 loaded into the knowledge while the crawler navigates sites. New graphics are created by composing basic shapes, described through attributes in SVG standard. 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. 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. Figure 3 shows the structure of the hierarchy with reference to an actual example. The insertion algorithm of the object \( O \) determines the set of parent nodes of the new node. 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. 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 \( N_i \) and the elements \( C_i \in G \) 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 \( H_0 \) 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:

$$X_{N_i} = \bigcup_{j=1}^{n_{D}} X_{D_j} \cup I_{N_i}$$

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 $G = \{N_0, \ldots, N_{n_{O} - 1}\}$ of parents of $O$, the set of documents to link to the new shape is:

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

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

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

is determined. The links to documents in the set $T_D \cap M_O$ are moved to the new node $N$ and links to documents in $[T_D - (T_D \cap M_O)]$ are copied in $N$ instead of being moved. 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 to be somewhere present. This tends to increase precision, also at the expenses of recall, thus we may accept false negatives, but prevent false positives. 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 dataset of SVG documents taken from available documents found on the web by the system. First it should be pointed out that currently there are still few SVG files on the web. A search initiated on may 6th, 2003 using as seeds the sites of companies that took part in the standard definition led to the discovery and indexing of 2347 SVG documents. Over 400 were duplicates and largest part of them are examples of the standard usage. We selected 50 documents and used them as queries. Results were encouraging and are summarized hereafter in terms of average precision, recall, and processing time.

**Precision**\_average = 0.784, **Recall**\_average = 0.868, **Processing-time**\_average = 1.002, **Precision**\_variance = 0.073, **Recall**\_variance = 0.023, **Processing-time**\_variance = 0.168.

Processing time refers to a PC with 2 GHz Pentium processor with 512 MB RAM. As a comparison we 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 stopwords. We indexed the same SVG documents and carried out retrieval with the same queries previously used and a cut-off at 10 documents. Average precision and recall were the following ones: **Precision** = 0.471, **Recall** = 0.644.

5. **CONCLUSION AND FUTURE WORK**

We have proposed a Knowledge based approach and a system for SVG documents retrieval. The language presents a syntax and a companion extensional semantics, which is fully compositional and enforces downward refinement. The approach enables semantic retrieval instead of retrieval based on the syntactic text. A prototype system has been built, implementing the approach. Obviously tests carried out so far are not enough to assess performances of the system and much is to be done, nevertheless they show encouraging results.

6. **REFERENCES**