SVG-to-RDF Image *Semantization*

Khouloud Salameh[1], Joe Tekli[2], Richard Chbeir[1]

[1] LIUPPA Laboratory, University of Pau and Adour Countries (UPPA)
64600 Anglet, France
{khouloud.salameh, richard.chbeir}@univ-pau.fr

[2] School of Engineering, Lebanese American University (LAU)
36 Byblos, Lebanon
joe.tekli@lau.edu.lb

Abstract. The goal of this work is to provide an original (semi-automatic) annotation framework titled *SVG-to-RDF* which converts a collection of raw Scalable vector graphic (SVG) images into a searchable semantic-based RDF graph structure that encodes relevant features and contents. Using a dedicated knowledge base, *SVG-to-RDF* offers the user possible semantic annotations for each geometric object in the image, based on a combination of shape, color, and position similarity measures. Our method presents several advantages, namely i) achieving complete *semantization* of image content, ii) allowing semantic-based data search and processing using standard RDF technologies, iii) while being compliant with Web standards (i.e., SVG and RDF) in displaying images and annotation results in any standard Web browser, as well as iv) coping with different application domains. Our solution is of linear complexity in the size of the image and knowledge base structures used. Using our prototype *SVG2RDF*, several experiments have been conducted on a set of panoramic dental x-ray images to underline our approach's effectiveness, and its applicability to different application domains.

Keywords: Vector images, SVG, RDF, semantic graph, semantic processing, image annotation and retrieval, visual features, image feature similarity.

1. Introduction

The need to index and retrieve multimedia data is becoming ever-more important, especially on the Web where image search and retrieval techniques do not seem to keep pace. Most existing Web image search engines (such as Google and AltaVista) and photo sharing sites (e.g., Flickr and Picasa) adopt the keyword (text-based) querying paradigm, usually returning a large quantity of search results, ranked by their relevance to a text-based query [27]. This can be extremely tedious and time consuming, since the returned results usually contain multiple topics mixed together, where the automated engines are guessing image visual contents using (in)direct textual clues [27]. An alternative approach is content-based image retrieval (CBIR), where images are indexed based on their visual content, using low-level color, texture, and shape descriptors, and are consequently processed via dedicated search engines (e.g., QBIC [5], Photobook [19], and Google *search-by-image*[1]). CBIR has been usually less successful than text-search engines since low-level features are usually unable to effectively capture the semantic meaning of the

[1] https://google.com/imghp

image [13]. This is known as the so-called *semantic gap* [14]: discrepancy between low-level image features and user semantics.

The main goal of our study is to convert, with as little human intervention as possible, a collection of raw images into a searchable semantic-based structure that encodes semantically relevant image content. We specifically target the semi-automatic annotation of vector images, mainly SVG (Scalable Vector Graphic) images [28]. In summary, SVG is an XML-based language for describing two-dimensional graphics and encoding three types of visual objects: vector graphic shapes, images and text. SVG images have interesting properties (resolution-independent and extremely small-size image coding) and are becoming increasingly popular in a wide range of applications covering: medical image annotation [8, 10], geographic map annotation [11, 18], manipulating graph charts as well as basic shape annotation to simplify data accessibility for the blind [1, 2].

Here, we introduce a framework titled *SVG-to-RDF* which allows to convert a collection of SVG images into an RDF (Resource Description Framework) [7] graph structure. The RDF data model is similar to classic conceptual modeling approaches such as entity–relationship or class diagrams, allowing to define statements about resources in the form of subject-predicate-object expressions. These expressions are known as triples in RDF terminology. The *subject* denotes the resource being described, the *predicate* denotes traits or aspects of the resource, expressing a relation between the subject and the object, and the *object* designates another resource or data values.

Our system automatically transforms an SVG image into an RDF graph describing the geometric objects in the image and their relations (in the form of RDF triples), and then offers the user possible semantic annotations for each geometric object encoded in the RDF graph, based on shape, color, and position similarity comparison with existing objects stored in a dedicated (RDF-based) knowledge base. The annotated RDF image graph could be in several cases integrated in the knowledge base helping extend its semantic expressiveness and hence provide more accurate annotation offers for future comparisons. Our original method presents several advantages over existing approaches, namely i) the complete *semantization* of image contents, ii) allowing sophisticated semantic-based data search and processing using standard RDF technologies (e.g., SPARQL [20]), iii) while being compliant with Web standards (i.e., SVG and RDF) in displaying images and annotation results in any standard Web browser, as well as iv) coping with different application domains by its generality and adaptability. To validate our approach, a prototype tool called *SVG2RDF* has been developed and tested on a collection of panoramic dental x-ray images. Experimental results were satisfactory and promising.

The rest of the paper is organized as follows. Section 2 presents an overview of our approach. Section 3 describes the components of our *SVG-to-RDF* image *semantization* framework. In Section 4, we present the experimental results obtained when evaluating our approach. Section 5 concludes the paper and discusses future directions.

2. *SVG-to-RDF* Image Semantization

An overview of our *SVG-to-RDF* annotation framework is shown in **Fig. 1.** It consists of five main components: i) *SVG feature extraction*, ii) *RDF graph representation*, iii) *Similarity computation*, iv) *User verification and feedback*, and v) *RDF knowledge base*. Our approach is general in that: 1) it can process both raster and vector images (since raster image contours can be automatically extracted and used to generate an SVG image), and also 2) it can be associated to different application domains.

Once an input SVG image is available, the first phase of the process consists in automatically extracting the visual features and semantic properties of the image. It is worthy to note that unlike traditional raster (visual) feature extraction methods that would require important processing time, feature extraction from SVG images (identifying geometric shapes, their colors, and related textual descriptions) can be undertaken very efficiently and quickly using XML-based parsing from the SVG source code.

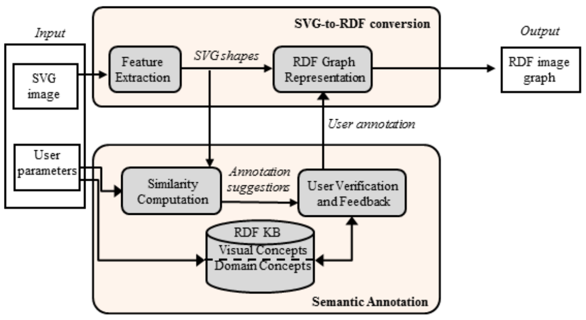


Fig. 1. *Simplified activity diagram describing our SVG-to-RDF framework.*

After their extraction, SVG features are represented in the form of *subject-predicate-object* triples into an RDF graph and consequently compared and mapped with those already stored in the dedicated RDF knowledge base, using corresponding comparisons. The RDF knowledge base presents domain-specific reference knowledge concerning the images being annotated (cf. Section 2.1). Finally, the generated annotations could be revised/modified by the user before validating the final image graph representation. When new application domain concepts and mappings are detected within the obtained graphs, they are injected into the RDF knowledge base to incrementally update it and increase continuously its semantic expressiveness.

In the following, we present in more details *SVG-to-RDF*'s main components.

2.1. RDF knowledge base

The RDF knowledge base provides domain experts, who are in charge of verifying/validating image annotations, with a set of predefined concepts and relations which are then extended by creating new instances of those concepts, based on the images being annotated. It can be generated *manually* by domain experts: including application domain concepts as well as their descriptions and mapping with the visual concepts (e.g., we adopt a reference dental knowledge base from [9] in our current study, cf. Section 3), or *automatically*[1] involving some machine learning techniques, using samples (a human expert manually annotates sample images with the intended semantic concepts, which are then provided as training data for a learning algorithm that induces rules to be used for assigning concepts to other images, thus incrementally building the knowledge base) [31].

Our RDF knowledge base is represented as an RDF graph (N, E) which **nodes** N are *subjects*, *objects*, or *subject/object properties* representing: i) SVG visual/geometric concepts (e.g., ellipse, circle, path), ii) application domain concepts (e.g., molar tooth, planet), and iii) corresponding property values (e.g., stroke, 50); and **edges** E are *Predicates* representing: i) relations between concepts (e.g., Circle *SubClassOf* Geometric Object, circle *IsA* planet, Teeth *HasInfluentialFacts* Symptom, etc.), and ii) property and value relations (e.g., ellipse *HasRadius* 50). **Fig. 2.** shows an extract of a sample knowledge base used in our study.

[1] This will be studied in a dedicated work.

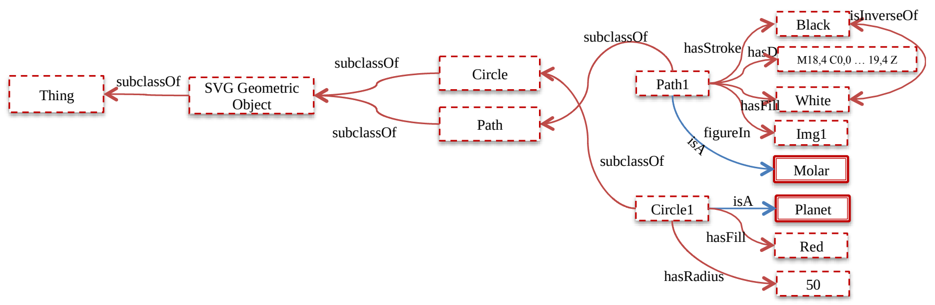


Fig. 2. *Extract of the RDF knowledge base used in our study*[1].

2.2. Feature Extraction

SVG allows encoding a variety of geometric objects using a set of predefined basic *shape* elements (*rectangle*, *circle*, *ellipse*, *line*, *polyline*, *polygon* and *path*), defining for each shape a set of descriptive attributes known as geometric object *properties*[2].

Since SVG is an XML-based coding, our SVG feature extraction component retrieve XML attributes using traditional XQuery and XPath expressions. This is a major advantage over traditional low-level feature extraction which may require extensive processing in comparison with fast XQuery/XPath processing (**Fig. 3.**).

2.3. RDF Graph Representation

Once the feature extraction is produced for a given SVG image, the image graph representation and annotation (using related components) are automatically executed. To do so, we adopt RDF (Resource Description Framework) [7] as a W3C data model designed to standardize the definition, the use and the reasoning of metadata, and we use it to describe and handle image representation. Initially, the graph representation contains only SVG visual elements of a given image mapped to RDF triples such that: RDF subjects represent SVG geometric elements (e.g., *circle*, *rectangle*, *path*, etc.), RDF predicates represent SVG element relations (e.g., *hasD*, *hasFill*, *hasStroke*, etc.), RDF objects represent properties values (e.g., *cx= "50"*, *stroke = "black"*, etc.).

Query: ***Select the visual attributes of path elements having a fill color = "white"***			
SVG Raw image	**SVG source code**	**XPath query**	**Result:** Extracted Visual Features

Img1.svg	<?xml version-"1,0"?>
	<svg width="20cm" height="20cm">
	<path d=" M18,4 C0,0 -26 -15 17,18
	C 9,33,3,18,8,31 S-1-25,7
	20.5 S6,19, 4,22 S15 - 31,15
	- 43 S26 - 4,19,4 Z"
	stroke="black" fill="white"/>
	</svg>
		//path[@fill="white"]/@*	d =" *M18,4 C0,0 -26 -15 17,18 C 9,33,3,18,8,31 S-1-25,7 20.5 S6,19, 4,22 S15 - 31,15 - 43 S26 - 4,19,4 Z"*
			stroke= "black"
			fill ="white"

Fig. 3. Example of SVG feature extraction using a typical XPath query statement.

[1] Subjects and Predicates URI, e.g., "http://svg2rdf.org#", are omitted here in order to simplify the graph.

[2] SVG includes a *text* element which we disregard in this paper: for clarity of presentation, and since text elements can be straightforwardly handled using traditional natural language processing techniques.

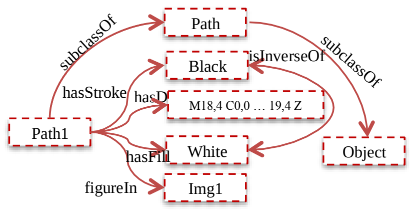
Fig. 4.a shows the extracted features of an SVG image, which is automatically transformed into an RDF graph as shown in **Fig. 4.b** Triple *Path1-figureIn-Img1* is added to indicate that the path is included in the image.

d =" *M18,4 C0,0 -26 -15 17,18 C 9,33,3,18,8,31 S-1-25,7 20.5 S6,19, 4,22 S15 - 31,15 - 43 S26 -4,19,4 Z"*

stroke= "black"

fill ="white"

a. Extracted Visual Features of *Img1.svg*.



b. Resulting RDF graph.

Fig. 4. Sample SVG feature extraction and resulting RDF triple representation (before annotation).

While the mapping of SVG tags to RDF triples is straightforward at this stage, yet it is the building block required to add semantic annotations: allowing additional triples to be added when applying the semi-automatic annotation component, in order to provide user-specific semantic meaning. For instance, the triple *Path1-IsA-Molar* can be added to provide a semantic meaning for the geometric elements in the image (e.g., *Path1* represents a *molar* tooth). The annotation process is described in the subsequent sections through the following two components: *similarity computation* and *user verification and feedback*.

2.4. Similarity Computation

Although SVG coding presents the syntactic/structural properties of vector images (in the form of basic geometric objects and properties), it does not provide any semantic meaning (e.g., the SVG coding in **Fig. 3.** does not reflect any semantic). Once the image graph is produced for a given SVG image, the similarity process compares each of the image graph's geometric objects with those stored in the RDF knowledge base using three main similarity criteria: i) shape similarity, ii) color similarity, and iii) position similarity[1]. Given two SVG geometric objects O_1 and O_2:

$$\mathrm{Sim}(O_1, O_2) = w_{Shape} \times \mathrm{Sim}_{Shape}(O_1, O_2) + w_{Color} \times \mathrm{Sim}_{Color}(O_1, O_2) + w_{Pos} \times \mathrm{Sim}_{Pos}(O_1, O_2) \quad \textbf{(1)}$$

where $w_{Shape} + w_{Color} + w_{Pos} = 1$ and $(w_{Shape}, w_{Color}, w_{Pos}) \geq 0$, such that $(\mathrm{Sim}_{Shape}, \mathrm{Sim}_{Color}, \mathrm{Sim}_{Pos}) \in [0, 1]$. We utilize the *weighted sum* function to combine the different similarities, allowing the user to fine-tune the weight of each criterion. Then, based on the aggregate similarity result greater than a given user or application-based predefined threshold $Thresh_{Sim}$, the system provides annotation offers for each geometric object in the image, corresponding to the most similar RDF node objects found in the knowledge base.
In the following, we briefly describe the similarity measures used by default in our system, note that the user can define his own similarity functions suitable to his domain application.

2.4.1. Shape Similarity

SVG shape similarity can be performed to compare geometric objects of the same type (comparing two *circles*, or two *rectangles*, etc.), or to compare objects of different shape

[1] A text similarity factor can be straightforwardly added when considering SVG *text* elements, using traditional text comparison techniques such as *string edit distance* and *N-gram* [6].

types (e.g., comparing a *circle* with a *rectangle*). For this purpose, Li et al. [11] introduce a set of mathematical formulas specially tailored for the task, which we adopt in our study.

On one hand, when comparing two objects of the same type, we start by identifying the invariants of the object type, i.e., points which remain invariant even if the geometric shape undergoes a transformation (e.g., translation, rotation, or both). This results in a general mathematical equation defined based on invariant points (as its coefficients), which can then be used to compare same-shape elements. For example, comparing two ellipses is accomplished using the quadratic conic curve similarity formula [11]:

$$\text{Dist}_{\text{Ellipse}}(O_1, O_2) = w_{\text{Major}}\,|a_1\text{-}a_2| + w_{\text{Minor}}\,|b_1\text{-}b_2| + w_{\text{Ecc}}\,|\varepsilon_1\text{-}\varepsilon_2| \qquad \textbf{(2)}$$

where (w_{Major}, w_{Minor}, w_{Ecc}) ≥ 0 and $w_{Major} + w_{Minor} + w_{Ecc} = 1$; a_1 and a_2 are the semi-major axis of O_1 and O_2 respectively; b_1 and b_2 are their semi-minor axis; and ε_1 and ε_2 are their eccentricities. Similar formulas are provided in [11] for comparing lines and rectangles.

On the other hand, when comparing two geometric objects having different types, such as comparing a *circle* with a *path*, the proximity of their contours is computed [11]. A contour is treated as a set of points, and hence contour proximity is measured in terms of the distances between the points: two contours are more similar, if the distance between their points is smaller. Hence, considering $A = \{p_1, p_2, \ldots, p_n, \ldots\}$ and $B = \{q_1, q_2, \ldots, q_m, \ldots\}$ the set of points describing the contours of objects O_1 and O_2 respectively, the distance between O_1 and O_2 can be evaluated as:

$$\text{Dist}_{\text{DiffShapes}}(O_1, O_2) = \mathit{max}(h(A, B), h(B, A)) \qquad \textbf{(3)}$$

$$\text{Where} \quad h(A,B) = \max_{p \in A} \min_{q \in B} |p - q| \quad \text{and} \quad h(B,A) = \max_{q \in B} \min_{p \in A} |q - p|$$

Note that, a particular case can be defined when comparing two objects of different types, while of them can be transformed into the other (comparing a *circle* with an *ellipse*). This can be done with a less expensive computation using the same quadratic conic curve similarity given in equation (2).

In our study, we adopt the formal definition of shape similarity as the inverse of a distance function[32], and thus deduce similarity scores from distances accordingly:

$$\text{Sim}_{\text{Shape}}(O_1, O_2) = \frac{1}{1 + \text{Dist}_{\text{Shape}}(O_1, O_2)} \in [0,1] \qquad \textbf{(4)}$$

2.4.2. Color Similarity

In addition to shape similarity, color is one of the most widely used features in image retrieval. On one hand, colors have been traditionally defined on a selected color space [15], such as RGB, LAB, HSV, etc., each one serving a different set of applications, where a given color is coded as a set of integer values. On the other hand, color ontologies have been recently introduced, e.g., [16, 24], in order to bridge the gap between low-level (numeric) color features and high-level (semantic) color descriptions, where colors are defined using *color names* (e.g., *red*, *blue*, *light blue*, etc.), and organized in an ontological graph structure based on their visual and semantic relatedness.

Since SVG allows coding colors in both: i) numerical format in the RGB feature space, and ii) using color names with 147 reference colors [28], we adopt both color representations in our approach to calculate the similarity between two colors by combining their semantic meaning and their visual properties as follows:

$$\text{Sim}_{\text{FillColor/StrokeColor}}(C_1, C_2) = w_{\text{HSV}} \times \text{Sim}_{\text{HSV}}(C_1, C_2) + w_{\text{Ont}} \times \text{Sim}_{\text{Ont}}(C_1, C_2)) \qquad \textbf{(5)}$$

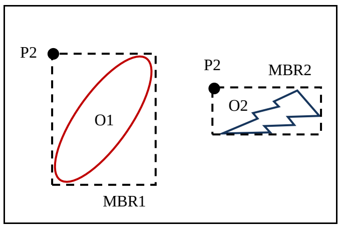
While SVG codes colors in numerical format in the RGB color space, yet we chose to convert RGB into the HSV color space, since HSV encoding is considered to be closer to human perception [30] and thus can be more semantically descriptive. Hence, to compare two colors (based on numerical format), we first convert their vectors from RGB to HSV using [30], and then calculate their scalar product. As for comparing color names, it can be achieved using any of several existing methods to determine the semantic similarity between concepts in a semantic network, e.g. [12, 21, 22]. These can be classified as i) edge-based: estimating similarity as the shortest path between the concepts being compared, and ii) node-based methods: estimating similarity as a function of the maximum the amount of information content concepts share in common [22]. In our approach, we combine (using weighed sum aggregation) two central edge and node-based approaches developed by *WuPalmer* [29] and *Lin* [12] (omitted here for lack of space).
Given two objects O_1 and O_2, we formally compute their color similarity as follows:

$$\text{Sim}_{\text{Color}}(\text{O}_1, \text{O}_2) = \begin{array}{l} w_{\text{FillColor}} \times \text{Sim}_{\text{FillColor}}(\text{FC}_1, \text{FC}_2) + \\ w_{\text{StrokeColor}} \times \text{Sim}_{\text{StrokeColor}}(\text{SC}_1, \text{SC}_2) \end{array} \quad \textbf{(6)}$$

where $(w_{\text{FillColor}}, w_{\text{StrokeColor}}) \geq 0$ and $w_{\text{FillColor}} + w_{\text{StrokeColor}} = 1$ such that $(\text{Sim}_{\text{FillColor}}, \text{Sim}_{\text{StrokeColor}}) \in [0, 1]$; FC_1 and FC_2 designate the fill colors of objects O_1 and O_2 respectively; and SC_1 and $SC2$ designate their stroke colors.

2.4.3. Position Similarity

In order to compare position similarity between two geometric objects O_1 and O_2, we generate their minimum bounding rectangles (MBR_1 and MBR_2) and then compute the *Euclidian* distance between the top-left vertices of their MBRs (P_1 and P_2), where the top-left vertex serves as a reference position point for SVG rectangle objects (cf. **Fig. 5.**), as indicated in Equation (7).



$$\text{Sim}_{\text{Pos}}(\text{O1}, \text{O2}) = \frac{1}{1 + \text{Dist}_{\text{Euclidian}}(\text{P}_1, \text{P}_2)} \in [0,1] \quad \textbf{(7)}$$

where $P_1(x_1, y_1)$ and $P_2(x_2, y_2)$ are the coordinates of the top-left MBR vertices.

Fig. 5. *Sample MBRs & reference points.*

2.5. User Verification and Feedback

The similarity-based annotation suggestions, which are automatically identified for each geometric object in the RDF image graph, are presented to the user according to a (user or application-based) predefined similarity threshold $Thresh_{Sim}$. Hence, RDF object nodes which similarities are lower than $Thresh_{Sim}$ are filtered out, retaining the most similar nodes which are then ranked and presented to the user according to their similarity scores w.r.t. the geometric object being annotated. The user can then verify and/or update the annotations according to the system annotation offers. Upon accepting the annotation offers, the latter are appended to the corresponding RDF image graph describing the image, thus producing a complete semantic representation of the image. Consequently, when identifying new application domain concepts and mappings, the RDF image graph is integrated in the RDF knowledge base, by appending the image graph nodes as instance nodes under their corresponding categories in the knowledge base (e.g., nodes representing circle objects are appended as instances under the category *geometric object*, nodes representing molar are appended under the category *tooth*, etc.).

3. Experimental Evaluation

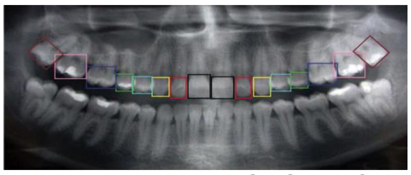
We have developed a prototype system[1], to test and evaluate our *SVG2RDF* image *semantization* framework, implemented using Java, and making use of the JENA API[2] in order to create, parse, and search RDF models (using SPARQL). While our approach is generic, yet we chose to test it in a real-world application scenario: clinical dental therapy. Our tests were designed to process a collection of dental panoramic x-ray images. After several meetings with multiple dentists specialized in dental surgery and orthodontia, we identified some of the critical information that is of interest to specialists when examining a dental panoramic image, namely: i) the shape of the tooth (e.g., the tooth looks poorly developed, decaying, etc.), ii) the tooth color (*white* for synthetic teeth, *dark gray* for decayed teeth, and *black* for lack of teeth), and iii) the position of the teeth (teeth are juxtaposed, evenly spaced, etc.). At this stage, the significance of similarity factors' weights is emphasized. Consequently, we considered that the three similarity criteria are at the same level of importance, hence we used equal weights (i.e., $w_{Shape} = w_{Color} = w_{Pos}$) However, in other applications, those criteria could have different impact in the process, so the user can change the values of the weights according to her preferences. To provide domain specific annotations, we adopted a reference dental knowledge base from [9], consisting of dental domain concepts (tooth, symptoms, etc.) and we extended it to include SVG geometric object constructs and properties (cf. **Fig. 4.**). Note that, initially our knowledge base does not contain any visual or semantic description of any SVG image; it only contains basic dental domain concepts and SVG basic geometric objects and properties. To simplify the process of creating SVG annotations on top of panoramic images, we used the minimum bounding rectangle (MBR) as a simple and suitable solution where MBRs designate annotated teeth (**Fig. 6.**).

3.1. Experimental Results

The main criteria used to evaluate the effectiveness of automatic annotation approaches is the amount of manual work required to perform the annotation task. This depends on: i) the quality of automatic annotations, and ii) the time to provide automatic annotations. It is worthy to mention that, to the best of our knowledge, there is no other existing approach that proposes to convert SVG to RDF and hence we don't provide here any comparative study.

3.1.1. Evaluating Annotation Quality

We conducted preliminary experiments on a collection of 10 panoramic dental images, each containing 16 teeth, i.e., 16 geometric shapes, consisting of the upper jaw teeth in panoramic dental X-ray images. Three dentists participated in the evaluation process.


a. Panoramic image annotated with SVG shapes.

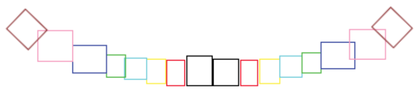
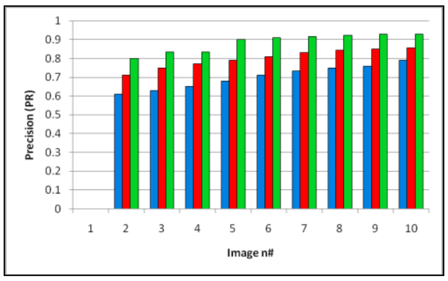

b. Resulting SVG image (annotations).

Fig. 6. Generating an RDF image graph annotating a raw dental panoramic image.

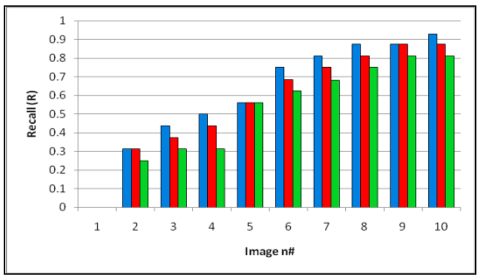
[1] Available online at: http://sigappfr.acm.org/Projects/SVG-To-RDF/
[2] https://jena.apache.org/

After producing the RDF image graph, the dentist (user) annotates the first image without any help, i.e., without any automatic annotation offers since the knowledge base only contains raw concepts in the beginning (general dental concepts from the reference dental knowledge base, and basic SVG constructs). Starting from the second image, automatic annotation offers can be produced, and presented to the dentist for evaluation. The system registers the dentist's answer as: *relevant* or *irrelevant* annotation, and computes *PR*, *R*, and *F-value* accordingly. The process was repeated for each dentist, using three different similarity thresholds Thresh_{Sim}: 0.5, 0.7 and 0.9 ($\in$ [0, 1]), hence resulting in a total of 10×16×3×3=1440 annotation tasks. Average results are presented in **Fig. 7.d** Note that in all our annotation tasks, all similarities factors were considered with equal weights (i.e., $w_{Shape} = w_{Color} = w_{Pos} = 0.3334$, $w_{Major} = w_{Minor} = w_{Ecc} = 0.3334$, $w_{Lengh} = W_{Slope} = 0.5$, etc.). In fact, in this study, we do not address the issue of assigning similarity weights, which we report to a dedicated subsequent study.

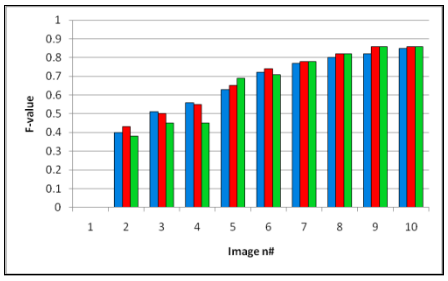
Graphs in **Fig. 7.a** and **Fig. 7.b** show that *precision* and *recall* levels can vary from 0 (minimum) to 1 (maximum) when the system returns 0 or 16 relevant annotation offers respectively (corresponding to each of the 16 geometric objects in the image). *Precision/recall* levels increase almost regularly as the number of images being annotated increases. This is because every time the user annotates an image, the new annotations are stored in the RDF knowledge base and thus become available as potential annotation offers. Then, when it comes to annotating the following image, the number of potentially accurate annotation offers increases, thus increasing *precision* accordingly. However, we notice in **Fig. 7.a** and **Fig. 7.b** that average *precision* levels tend to increase when the Thresh_{Sim} increases, whereas average *recall* levels tend to decrease. This is because increasing the similarity threshold: i) reduces the number of irrelevant annotation offers (reflected by increasing precision), yet also ii) filters out certain potentially relevant annotations which might be less similar to the object being annotated (decreasing *recall*).



a. Precision (PR) levels.



b. Recall (R) levels.



c. F-value levels

$\mathrm{Thresh}_{Sim} = 0.5$ 🟦 $\mathrm{Thresh}_{Sim} = 0.7$ 🟥 $\mathrm{Thresh}_{Sim} = 0.9$ 🟩

Thresh_{Sim}	**Precision**	**Recall**	**F-Value**
0.5	0.7013	0.6723	0.6733
0.7	0.8011	0.6312	0.6878
0.9	0.8856	0.5677	0.6667

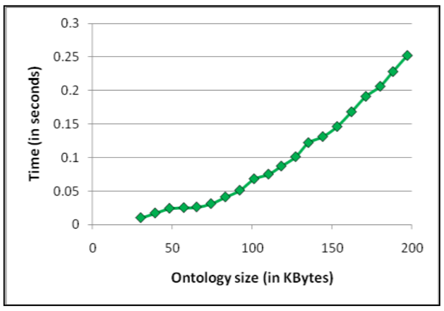
d. Average *PR*, *R*, and *F-value* results.

Fig. 7. Precision, recall, and f-value results.

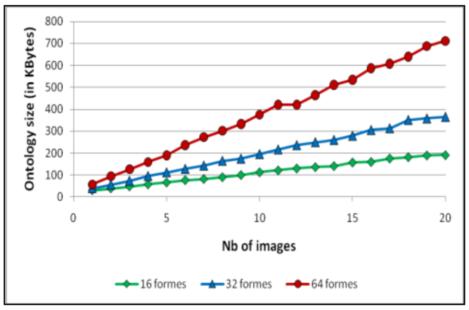
In cases where higher/lower *precision*/*recall* levels are obtained, the *f-value* measure (the harmonic mean of *precision* and *recall*) is central in evaluating the overall loss and gain in average *precision*/*recall*, in order to evaluate result quality. **Fig. 7.c** shows that *f-value* levels increase regularly w.r.t. the number of images annotated, varying from 0.4 (± ε) to 0.75 (± ε) with all three similarity thresholds. Also, results in **Fig. 7.c** show that *f-value* levels remain almost the same when varying the similarity threshold (e.g., *f-value* = 0.4 (± ε) for image n#1, and 0.75 (± ε) for image n#10, for all three $Thresh_{Sim}$ values). This illustrates the results mentioned above: when the threshold increases, *precision* increases yet *recall* decreases, hence inducing certain equilibrium with *f-value* levels. To sum up, *precision*, *recall*, and *F-value* results are clearly positive and promising, despite the similarity threshold's effect on *recall* (which tends to slowly decrease when increasing the similarity threshold), which we plan to investigate in an upcoming experimental study.

3.1.2. Evaluating Annotation Performance

In addition to testing the effectiveness of our approach in identifying meaningful annotations, we evaluate its time performance. The complexity of our method comes down to $O(|I|\times|KB|)$ where $|I|$ represents the size of the image being annotated (in number of geometric objects), and $|KB|$ the size of the reference knowledge base (in number of object nodes). Theoretical complexity analysis is omitted due to lack of space. We start by verifying our approach's linear time dependency on the size of the image and that of the RDF knowledge base. Timing experiments were carried out on a PC with an Intel Xeon 2.66 GHz processor with 2 GB RAM. **Fig. 8.a** shows that the time needed to produce automatic annotation offers for all geometric objects in an image grows in a linear fashion with the knowledge base size. In addition, **Fig. 8.b** also shows that the size of the knowledge base increases in an almost perfect linear fashion with the number of images being annotated and appended to the knowledge base. This explains the increasing slopes of the chart lines in **Fig. 8.b** since every geometric object is annotated and then integrated as a new geometric description in the knowledge base.



a. Images made of 16 geometric objects each.



b. knowledge base graph size variation w.r.t. the number of sequentially annotated images

Fig. 8. Time performance and knowledge base size variations w.r.t. image size (in number of geometric objects per image).

4. Related Work

Many general-purpose raster image retrieval and processing systems have been proposed in the literature, and can be roughly categorized as: text-based [4, 27] and content-based [13, 14]. More recent methods have investigated XML-based solutions [25, 26] organizing images into an XML (MPEG-7) document tree hierarchy, and then applying image search and retrieval operations on the obtained XML multimedia tree. Most of these approaches suffer from the *semantic gap* [14, 23], given that the meaning of an image is rarely evident using traditional keyword and/or content-based descriptions. The interested reader can refer to surveys in [4, 13].

Few approaches have specifically targeted SVG image processing, e.g., [3, 8, 10, 17, 18]. The work in [3] suggests the organization of features extracted from SVG images in the form of an aggregation tree, where each tree node represents an SVG geometric object or an aggregated set of objects and is described by an MBR (Minimum Bounding Rectangle) and a shape description, taking into consideration the topological relationships between the objects (e.g., *disjoint*, *meet*, *overlap*, etc.). The aggregation tree is constructed using object-aggregation rules defined based on topological relations, e.g., two *disjoint* objects *p* and *q* are grouped under a higher level object *n* consisting of a new MBR encompassing the ones of *p* and *q*. The study in [3] presents an on-going work, aiming to index SVG images toward easier information retrieval. Another approach in [10] introduces a hierarchical SVG image abstraction layer for medical imaging, organizing low level features and high level semantic information in an image abstraction layer where content pieces are represented in XML and SVG. The authors then describe a web-based tool that visualizes, manipulates, and searches the abstraction layer using XQuery. Similar works investigating the processing and retrieval of SVG images using XML data search and manipulation techniques have been proposed in [8, 18]. In [11], the authors introduce a library of shape similarity measures designed to compare SVG geometric objects. This approach has been adopted in our framework and some of the measures are presented in Section 3. An approach which is relatively comparable to ours in presented [17] introducing a tool allowing users to manually associate semantic annotations to a sketch based query specification. Here, images are drawn and transformed into SVG coding, whereas user annotations are transformed into an RDF fragment appended to the SVG image code. Nonetheless, this approach solely focuses on manual user annotation and does not address semi-automatic annotation. Also, the resulting RDF code is appended to the SVG image source code which limits RDF semantic processing capabilities. In addition, the authors process images separately, in contrast with our approach which introduces the concept of unified reference RDF ontology to gather the collective semantics of an image repository, allowing annotation suggestions and improving image semantic processing.

5. Conclusion

This paper introduces a framework: *SVG-to-RDF* for transforming a collection of SVG images into RDF graphs. The system automatically transforms each input SVG image into a basic RDF graph, and then offers the user semantic annotation offers for each geometric object in the image, based on shape, color, and position similarity comparisons with existing objects already stored in a reference RDF knowledge base. When new concepts and mappings are detected, the annotated RDF image graph is then integrated in the knowledge base extending its semantic expressiveness. Experiments show that our approach is of average linear complexity w.r.t. image and knowledge base size, and provides promising annotation results.

We are currently investigating the extension of our approach to allow whole image search, as opposed to searching and annotating individual geometric objects within an image. In this context, dedicated reference ontologies and user-defined semantics would have to be considered to better assess image and geometric object relatedness. We also plan to study the effect of different similarity criteria (shape, color, and position) on annotation quality, proposing (if possible) weighting schemes that could help the user tune her input parameters to obtain optimal results.

References

[1] Abu Doush I., et al., *Multimodal Presentation of Two-Dimensional Charts: An Investigation Using Open Office XML and Microsoft Excel.* ACM TACCESS, 2012. 3(2):1-50.

[2] Awada Y., et al., *Towards Digital Image Accessibility for Blind Users via Vibrating Touch Screen: A Feasibility Test Protocol.* Inter. Conf. on Signal Image Tech. & Internet Systems (SITIS), 2012. pp-547 - 554.

[3] Bai S., et al., *Revised Aggregation-tree Used in Metadata Extraction from SVG Images.* DMIN'06. pp 325-328

[4] Datta R.; Joshi D.; Li J. and Wang J.Z., *Image Retrieval: Ideas, Influences and Trends of the New Age.* ACM Computer Surveys, 2008. 40(2):1-60.

[5] Faloutsos C. *et al.*, *Efficient and Effective Querying by Image Content.* JIIS J., 1994. 3 (3:4):231–262.

[6] Hall P. and Dowling G., *Approximate String Matching.* Computing Survey, 1980. 12(4):381-402.

[7] Hayes P., *RDF Semantics.* W3C Recommendation, http://www.w3.org/TR/rdf-mt/. 2004. [cited 26 May 2014].

[8] Jiang K., et al., *Information Retrieval through SVG-based Vector Images Using an Original Method.* Proc. of IEEE Inter. Conference on e-Business Engineering (ICEBE'07) 2007. pp. 183–188.

[9] Kiani M., et al., *Ontology-Based Negotiation of Dental Therapy Options.* Advances in Semantic Computing (Eds. Joshi, Boley & Akerkar), 2010. Vol. 2, pp 52 – 78.

[10] Kim E., et al., *A Hierarchical SVG Image Abstraction Layer for Medical Imaging.* Society of Photo-Optical Instrumentation Engineers (SPIE) Conference, 2010. 7628, 7.

[11] Li D., et al., *Shape similarity computation for SVG.* Int. J. Comp. Science and Eng., 2011, 6:1/2.

[12] Lin D., *An Information-Theoretic Definition of Similarity.* Inter. ICML Conf., 1998. pp. 296-304.

[13] Liu Y.; Zhang D.; Lu G. and Ma W.-Y., *A Survey of Content-Based Image Retrieval with High-Level Semantics* Pattern Recognition, 2006. 40(1):262-282.

[14] Long F. *et al.*, *Fundamentals of Content-based Image Retrieval.* MM IR Management, 2003.

[15] Manjunath B.S., *Color and Texture Descriptors.* IEEE CSVT Trans., 2001. 6:703-715.

[16] Mezaris V. *et al.*, *An Ontology Approach to Object-based Image Retrieval.* Proceedings of the International Conference on Image Processing (ICIP). Vol. 2, pp. 511-514.

[17] Noah S. and Sabtu S., *Binding Semantic to a Sketch Based Query Specification Tool.* The International Arab Journal of Information Technology, 2009. Vol. 6, No. 2. pp. 116

[18] Peng Z.R. and Zhang C., *The roles of geography markup language (GML), scalable vector graphics (SVG), and Web feature service (WFS) specifications in the development of Internet geographic information systems (GIS).* Journal of Geographic Systems, 2004. pp 95-116.

[19] Pentland A.; Picard R.W. and Scaroff S., *Photobook: Content-based Manipulation for Image Databases.* International Journal of Computer Vision, 1996. 18 (3):233–254.

[20] Prudhommeaux E. and Seaborne A., *SPARQL Query Language for RDF.* W3C Recommendation, 2008. http://www.w3.org/TR/rdf-sparql-query/. [cited 26 May 2014].

[21] Resnik P., *Using Information Content to Evaluate Semantic Similarity in a Taxonomy.* Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 1995. Vol 1, pp. 448-453.

[22] Richardson R. and Smeaton A., *Using WordNet in a Knowledge-based approach to information retrieval.* Proceedings of the BCS-IRSG Colloquium on Information Retrieval, 1995. pp 1-16.

[23] Smeulders A. *et al.*, *Content-based Image Retrieval at the End of the Early Years.* IEEE Trans. of Pattern Analysis and Machine Intelligence, 2000. 22(12):1349–1380.

[24] Stanchev P.L.; Green D. Jr. and Dimitrov B., *High Level Color Similarity Retrieval.* Inter. Journal on Information Theory and Applications, 2003. 10(3):363-369.

[25] Torjmen M., et al., *XML Multimedia Retrieval: From Relevant Textual Information to Relevant Multimedia Fragments.* INEX: Initiative for the Evaluation of XML Retrieval, 2009.

[26] Tsikrika T., et al., *Structured Document Retrieval, Multimedia Retrieval, and Entity Ranking Using PF/Tijah.* INEX: Initiative for the Evaluation of XML Retrieval, 2008. pp. 273–286.

[27] Wang S. *et al.*, *IGroup: Presenting Web Image Search Results in Semantic Clusters.* CHI 2007.

[28] W3C, *Scalable Vector Graphics (SVG)*. http://www.w3.org/Graphics/SVG/. [cited 26 May 2014].

[29] Wu Z. and Palmer M., *Verb Semantics and Lexical Selection.* Proceedings of the 32nd Annual Meeting of the Associations of Computational Linguistics, 1994. pp. 133-138.

[30] Foley, van Dam, Fiener, Hughes, Computer Graphics: Principles and Practice, Addison Wesley, Reading, MA. 1990.

[31] Alpaydin, Introduction to Machine Learning. MIT Press, Cambridge, MA, 2004

[32] Ehrig M. and Sure Y., Ontology Mapping - an Integrated Approach. Proc. of the European Semantic Web Conference (ESWC), 2004, 76-91.