Unsupervised Knowledge Representation of Panoramic Dental X-ray Images using SVG Image-and-Object Clustering

Khouloud Salameh[1], Farah El Akoum[2], and Joe Tekli[2*]
[1] *Computer Science and Engineering Department*
American University of Ras Al Khaimah (AURAK), 10021, Ras Al Khaimah, UAE
[2] *Electrical and Computer Engineering Department*
Lebanese American University (LAU), 36 Byblos, Lebanon

Abstract. Given that the meaning of an image is rarely self-evident using traditional keyword and/or content-based descriptions, the general goal of this study is to convert, with minimal human intervention, a stream of web vector graphics into a searchable knowledge graph structure that encodes semantically relevant image contents. To do so, we introduce an original framework titled SSG which automatically converts a stream of SVG images and objects into a semantic graph. We introduce an incremental clustering approach to semantically annotate SVG images and their constituent objects in a fast and efficient manner, using an aggregation of shape, area, color, and location similarity measures. We then produce an RDF graph representation of the input image and integrate it in a reference knowledge graph, incrementally extending its semantic expressiveness to improve future annotation tasks. This achieves semantization of vector image contents with minimum human effort and training data, while complying with native Web standards (i.e., SVG and RDF) to preserve transparency in representing and searching images using Semantic Web stack technologies. Our solution is of linear complexity in the number of images and clusters used. We have conducted a large battery of experiments to test and evaluate our approach. We have created a labelled SVG dataset consisting of 22,553 objects from 750 images based on panoramic dental x-ray images. To our knowledge, it is the first significant dataset of labelled SVG objects and images, which we make available online as a benchmark for future research in this area. Results underline our approach's effectiveness, and its applicability in a practical application domain.

Keywords. Vector graphics, SVG, semantic graph, semantic processing, RDF, incremental clustering, image annotation, visual features, image feature similarity.

1. Introduction

Image datasets have become ever-more available, especially on the Web considered as the largest multimedia database to date [1]. However, the value of their content depends on how easy they are to search and manage [2]. Thus, the need to efficiently represent and manage images is becoming increasingly important. Existing Web image search engines and photo sharing sites (such as Google Images[1] and Flickr[2]) chiefly utilize the keyword (text-based) representation model, where image visual contents are described using indirect textual clues [3]. They typically return a large amount of search results ranked according to their relevance to the keyword query, which can be tiresome and time consuming for the user [3]. Another approach is content-based image retrieval (CBIR), where images are described based on their low-level visual content, e.g., color, texture, and shape descriptors (e.g., Google *search-by-image*, Picsearch[3]) [4, 5]. Yet low-level features are usually unable to effectively capture the high-level semantic meaning present in images [6], which is known as the *semantic gap* problem: the difference between the visual expressiveness of low-level image features and the meanings provided by user semantics [4, 6]. Few recent approaches proposed to augment the textual descriptions of Web images, using techniques like probabilistic image tagging (using the image tags from the logs of related users to infer new tags, e.g., [7, 8]), and semi-supervised image annotation based on visual and Web contents (training supervised learners to perform annotation based on existing images with predefined labels, e.g., [9, 10]). While promising, yet these solutions involve large amounts of training data and significant training time which are not always obtainable and require significant manual labor to prepare [3].

---

* Corresponding author. Tel.: +961-9-547-262; E-mail address: joe.tekli@lau.edu.lb. Joe Tekli is also an adjunct researcher with the SPIDER research team, LIUPPA Laboratory, University of Pay and Pays Adour (UPPA), 64600, Anglet, Aquitaine, France.

[1] https://www.images.google.com [2] https://www.Flickr.com [3] https://www.picsearch.com/

The goal of our study is to convert, with minimal human intervention and as little training data as possible, a stream of raw Web vector graphics into a searchable semantic-based structure that encodes semantically relevant image contents. We specifically target the annotation of SVG (Scalable Vector Graphics)[3] images due to their interesting properties, i.e., resolution-independence and extremely small-size image coding. Vector images are becoming popular in a range of practical applications covering medical image annotation [11, 12], geographic map annotation [13, 14], and data accessibility for visually impaired users [15, 16]. We introduce an original SVG Semantic Graph framework titled *SSG*, which takes as input a stream of SVG images made of geometric objects, and produces as output a knowledge graph (KG) structure made of RDF (Resource Description Framework)[4] triples. *SSG* consists of four main modules for: i) converting SVG images into KG representations, ii) computing SVG image similarity using an aggregation of vector object similarity measures (an early version of this module is described in [17]), iii) performing efficient SVG image and object clustering to allow unsupervised labelling, and then iv) integrating the produced labels as semantic annotations in the KG representation, extending its semantic expressiveness. As a cold start, *SSG* requires only a minimum of one single vector graphic representative for each cluster to jumpstart the labelling process. The representatives are incrementally updated as new images are annotated by the system. Our solution is of linear complexity in the number of images and clusters used. It is fully automated and does not require training data or human effort. At the same time, it is transparent and explainable to the user who can choose to be involved through the whole process: from fine-tuning the similarity measures, to choosing the cluster representatives, and validating the system generated annotations and KG representations. *SSG* also complies with native Web standards, namely SVG and RDF, to preserve transparency in displaying, representing, and searching images using Semantic Web stack technologies. Compared with the existing literature in vector image representation and processing (cf. Section 3), the main contributions of this study are summarized as follow: i) we introduce a semantically augmented RDF-based representation to describe vector images, compared with existing solutions which use plain SVG coding, ii) we introduce the concept of unified reference KG to gather the collective semantics of an SVG image repository (using our augmented RDF-based representation), allowing annotation suggestions and improving image semantic processing, compared with existing solutions which represent images as separate standalone entities, iii) we introduce incremental SVG image and object clustering to perform unsupervised labelling, in contrast with existing solutions which process images separately by performing pairwise image similarity computation.

We have conducted a large battery of experiments to test and evaluate our approach. We have created a labelled SVG dataset consisting of 22,553 objects from 750 images based on panoramic dental x-ray images. To our knowledge, it is the first significant dataset of labelled SVG objects and images, which we make available online as a benchmark. Results underline our approach's effectiveness, and its applicability in a practical application domain.

The rest of the paper is organized as follows. Section 2 describes the motivation of our work. Section 3 briefly reviews the state of the art in image feature representation, semantics and vector image processing. Section 4 develops our *SSG* image semantization framework. Section 5 provides the complexity analysis. Section 6 presents and discusses our experimental results, before concluding in Section 7 with future directions.

2. Motivation

2.1 Applications of Vector Images versus Raster Images

Images can be grouped in two main categories: i) raster images, consisting of a set of pixels; and ii) vector images made of geometric entities such as circles, rectangles, triangles, and polygons, etc. Most existing approaches in the

[3] https://www.w3.org/TR/SVG2/ [4] https://www.w3.org/RDF/

literature focus on the processing of raster images [2, 4], which are typically produced by digital photo-taking cameras, and are capable of presenting complex pictures having a variety of colors and shapes. However, vector images are becoming more and more popular in several application areas requiring the manipulation of small-size, resolution-independent, and simple images made of basic lines and shapes. These applications range over: medical image annotation (adding basic shapes on top of medical images to identify organ tissues and tumors) [11, 12], geographic map annotation (highlighting special places and destinations on a map) [13, 14], video film annotation [18] (using vector graphics to define both spatial and temporal regions in a video film, and label its color features and contents accordingly), as well as manipulating graph charts and simplifying accessibility to data and geometric shapes for blind users (producing simplified contour-based images to simplify data accessibility and navigation for the blind) [15, 16]. In this context, SVG (Scalable Vector Graphics) [19] was introduced as an open W3C standard language for describing vector images. Based on the XML syntax, SVG allows the encoding of vector graphic shapes, images, and text, which facilitate including high-level descriptions within SVG content. SVG images can be created with a text editor, or more efficiently using a drawing software (e.g., SVG-edit[4], JANVAS[5], etc.).

While the advantages and the practical applications of vector graphics highlight the importance of this category of images, e.g., [18, 20], yet, most existing image retrieval systems process vector images similarity to raster images [11], regardless of the properties offered by the former. This underlines two major limitations: i) undergoing expensive low-level feature selection and extraction, while disregarding the readily available geometric object features which can be extracted much more efficiently (cf. Section 4.1), and ii) handling low-level features which are size and resolution dependent and which can affect retrieval quality, in contrast with vector graphics' features which are both resolution and size independent.

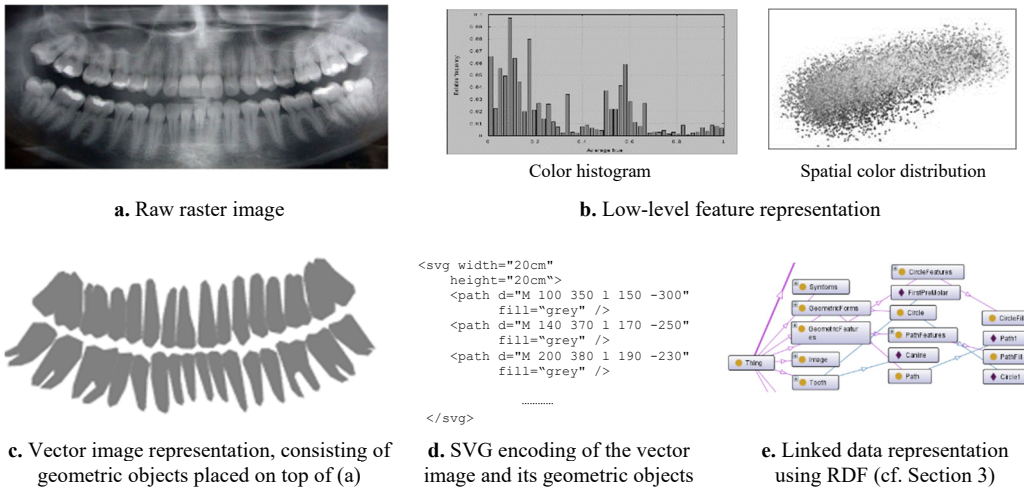
2.2 Case of Panoramic Dental X-ray Images for Clinical Dental Therapy

Consider the following example in Figure 1 that illustrates a clinical dental therapy scenario, describing the specific motivation behind this work. Dentists regularly process continuous streams of dental health records and dental images from their patients. More specifically, dentists specializing in surgery and orthodontia process dental panoramic x-ray images to identify various critical information including: i) the shape of the teeth (e.g., the tooth looks *poorly developed*, *decaying*, etc.), ii) the location of the teeth (teeth are *juxtaposed*, *evenly spaced*, etc.), and iii) the teeth's color (*ivory* for healthy teeth, *white* for synthetic teeth, *dark gray* for decayed teeth, or *black* for lack of teeth, etc.). This requires manual effort and highlights the following needs: i) identifying the different teeth in a panoramic x-ray image, ii) recognizing the type of each tooth (i.e., *incisor*, *canine*, *premolar*, and *molar*), iii) recognizing the presence of different types of critical information regarding teeth shape, location, and color, and iv) semantically linking the latter information with the patients' dental records.

Vector image annotation and processing can be utilized to answer the aforementioned needs. Considering the sample x-ray image in Figure 1.a, traditional image retrieval techniques would extract and process low-level image features (e.g., dominant color, color histogram, cf. Figure 1.b) which require considerable processing time and are usually unable to effectively describe the semantic meaning. However, the SVG image in Figure 1.c can be processed through its vector features (e.g., path stroke, ellipse fill color, cf. Figure 1.d) which are extracted much more efficiently and can be easily augmented with semantic information. Vector objects can be added on top of the panoramic x-ray images to identify different teeth and recognize their types and related information according to the dentists' needs (e.g., *object1 hasShape path1*, *object1 isA molar*, *object1 isLocatedIn upperJaw*). Also, being a W3C standard, we can augment SVG through the Semantic Web stack to benefit from its semantic processing capabilities and link it with open data repositories (using RDF triples, cf. Figure 1.e).

---

[4] Available at: https://code.google.com/p/svg-edit/
[5] Available at: http://www.janvas.com/site/home_en.php



a. Raw raster image

b. Low-level feature representation

c. Vector image representation, consisting of geometric objects placed on top of (a)

d. SVG encoding of the vector image and its geometric objects

e. Linked data representation using RDF (cf. Section 3)

Figure 1. Sample dental panoramic x-ray image (a) described using traditional low-level features (b), versus vector image and feature representation (c, d), and linked data representation (e, cf. Section 3)

In this context, we highlight the following challenges to achieve SVG semantic processing: *challenge 1* – how to represent SVG images in a meaningful way for seamless integration with Semantic Web technologies, *challenge 2* – how to process the vector images' shape, location, and color properties to perform object recognition and annotation, *challenge 3* – how to configure the annotation process in order to inject domain insights and user feedback (e.g., allowing dentists to configure, adjust, and modify the annotation process according to their needs) for a more adapted and accurate annotation task. Hence, we aim to provide a solution for knowledge-based vector image representation, allowing to address the challenges mentioned above. We aim to consider the intrinsic properties of SVG vector graphics, and build on the descriptive power of the Semantic Web stack, namely RDF, while relying on unsupervised data clustering to perform the semantic annotation task.

3. Related Works

Most existing Web image representation solutions focus on raster images made of a set of pixels described using low-level features. Recent approaches have addressed semantic augmentation using supervised learning techniques. Only a few approaches have addressed vector image representation and processing. Hence, we categorize existing solutions as: text-based, content-based, hybrid, and more recently XML-based, MPEG-7 based, and vector-based methods.

3.1 Text-based and Content-based Methods

In text-based systems, e.g., [21, 22], images are manually or automatically annotated by text descriptors, which are then exploited via classic database or text retrieval systems to perform image search [23]. On the one hand, the text-based paradigm has been adopted by most current Web search engines (e.g., Google and Bing) and photo sharing sites (e.g., Flickr and Imgur), due to its well proven scalability in handling the tremendous amounts of images published on the Web. However, text-based systems are usually characterized by poor result quality, since the automated engines are guessing image visual contents using indirect textual clues [22], and are usually unable to confirm whether the retrieved images actually contain the desired concepts expressed in the user queries [24]. In addition, text-based systems usually produce a large quantity of image search results ranked by their relevance to the text-based user query. This can be extremely tedious and time consuming since the returned results usually contain multiple topics mixed together, where users could entirely miss their search goal due to cognitive overload [3, 25]. In content-based systems,

e.g., [5, 26] images are indexed based on their visual content, e.g., color, texture, and shape descriptors, and are processed via search engines specially devised to handle, compute and compare low level image feature descriptors (e.g., dominant color, color layout, color and edge histograms, etc.) [27]. Yet, as these descriptors are low-level, they seem effective only in matching images almost identical in content [28]. In other words, they seem useful locally, i.e., when applied on a subset of similar images (retrieved a priori through some other means, e.g., using textual or user provided evidences), but fail when matching relatively disparate images [6]. In addition, low-level features are usually unable to effectively capture the high-level semantic meaning present in the image, which is known as the *semantic gap* problem [4, 6].

3.2 Hybrid Feature Representation Methods

Various hybrid methods have been developed, integrating both text-based and content-based image processing capabilities [23]. Most methods in this category target Web images, e.g., [29, 30] where both low-level and text-based image clues are available such as: the Web links of image files (e.g., URLs) which have a clear hierarchical structure including useful information such as the image Web category, and Web documents in which images are imbedded (e.g., HTML) which can encompass textual metadata, e.g., image title, Web page title, ALT-tag, etc. Yet, several studies have shown that such metadata can only annotate images to a certain extent (e.g., the image title is usually abbreviated and might be meaningless, the ALT-tag might be missing), and do not utterly solve the *semantic gap* problem [3].

On the other hand, moving on from traditional low-level image features (e.g., color, shape, and texture features), various studies have investigated *high-level semantics* [4, 31], i.e., deriving semantic descriptions generated automatically based on low-level features. This can be done using different techniques, namely: i) using dedicated ontologies to relate low-level features with high-level concepts (e.g., color ontologies where colors are defined using color names – *red*, *blue*, etc. – linked with numerical representations [32]), ii) using supervised learning models to associate low-level and high-level features using trained classifiers based on sample data provided by experts (e.g., texture categorization into pre-defined classes – *sea*, *clouds*, *forest*, etc. – based on training numerical spaces [33]), and iii) generating semantic templates to support high-level semantic image retrieval based on low-level features (e.g., retrieval of named events, of pictures with emotional significance such as "*find pictures of a joyful crowd"*, e.g., [34, 35]). The main premise with this family of hybrid techniques is to try to simulate the visual concept space in terms of lexical concepts as perceived by humans, which remains an inherently complicated task and an ongoing challenge in image retrieval [4]. Techniques such as probabilistic user-based image tagging (using the tagging logs from the histories of similar users to infer new tags, e.g., [7, 8]), and semi-supervised image annotation based on visual and Web contents (training different deep learning algorithms to annotate new images based on a training image set with predefined labels, e.g., [9, 10]) show promising results. Yet they require training data and training time, which are not always available.

3.3 XML-based Representation Methods

Some XML-based solutions have been introduced, e.g., [36-38], organizing images into an XML document tree hierarchy, and then applying image search and retrieval operations on the obtained XML multimedia tree. The general process consist of two main steps: placing images into a hierarchy, defining multiple evidence scores such as: image ascendants, brothers, and children, evaluated using an existing XML retrieval system, and then retrieving multimedia fragments from those relevant images [39, 40]. Authors in [41, 42] organize the textual content within and surrounding a multimedia object into disjoint and hierarchically organized entities called *Region Knowledge* (*RKs*), following the hierarchical structure of the document containing that multimedia object: the self-region, its sibling elements, its parent element, its ancestor elements, and so on; where the largest region is the document itself. Then, the authors use the vector space model to describe and evaluate each *region* using TF-IDF and hierarchical weighting schemes. Even though this method exploits the document structure, yet it solely focuses on textual descriptions (text-based), and does not actually consider the relations between images (content-based) and *regions* when the latter's relevance is

evaluated. Other XML-related approaches use a linear combination of evidences to merge the results of content-based and XML-based retrieval [37, 38].

While most XML-based methods capture a certain hierarchical organization inherent in multimedia information, nonetheless, most approaches show a high dependence on the underlying XML hierarchical structure and retrieval system used, rather than the actual visual and semantic properties of images.

3.4 MPEG-7 Feature Representation Methods

Some image retrieval approaches have extended XML-based solutions toward MPEG-7 retrieval [43-45]. MPEG-7 (Multimedia Content Description Interface) is a standardized description of various types of multimedia information including images and videos [46]. MPEG-7 provides a range of standardized *Descriptors* (*D*): representing low-level features (date of create, author, time, place, etc.) and high-level features (dominant color, scalable color, edge histogram, etc.), and *Description Schemes* (*DS*): defining the structure and semantics of the relationships between different descriptors and description schemes). A dedicated *Description Definition Language* (*DDL*) allows the creation of new descriptors and description schemes, adapting MPEG-7 to specific applications. As a result, several approaches have fine-tuned XML-based image retrieval to handle standardized MPEG-7 descriptors, e.g., [43-45]. Different methods have been developed to calculate similarity among images using low-level MPEG-7 descriptors, namely color (e.g., scalable color, dominant color) and texture (e.g., edge histogram) descriptors. The authors in [47] adopt Earth Mover's Distance as a measure between image signatures extracted as dominant nodes with corresponding weights from an image. In [48], the authors suggest organizing the descriptors within a hierarchical decision fusion framework using fuzzy logic, combining different types of image features into an integrated feature space. Another approach in [49] introduces a neural network-based system using a self-organizing mapping from the image descriptor space to a two-dimensional grid made of artificial neural units. The system uses the *query-by-image* paradigm, requiring the user to formulate pictorial cues and image queries, and allowing the latter to provide relevance feedback about the results, adjusting the neural network' weights accordingly.

While most MPEG-7 approaches target low-level multimedia descriptors and processing, yet, existing methods generally undermine image semantics. While it provides a unified way of describing many low-level image descriptors, yet, MPEG-7 can represent the same textual (semantic) metadata in multiple different ways which can complicate image processing. For example, using MPEG-7 to annotate an image depicting a *FirstPreMolar* juxtaposed to a *Canine* can be done in multiple ways using i) the *free text* tag, ii) the *keyword* tag, or iii) the *semantic* tag (cf. Figure 20.a, b, and c). This complicates data search and retrieval since a single query will not be enough to retrieve all similar metadata having similar characteristics and describing similar objects. Nonetheless, the same image can be annotated in a unique representation using an RDF-based representation such as ours (cf. Figure 2.d), which would allow the selection and retrieval of all the similar metadata using a single SPARQL query.

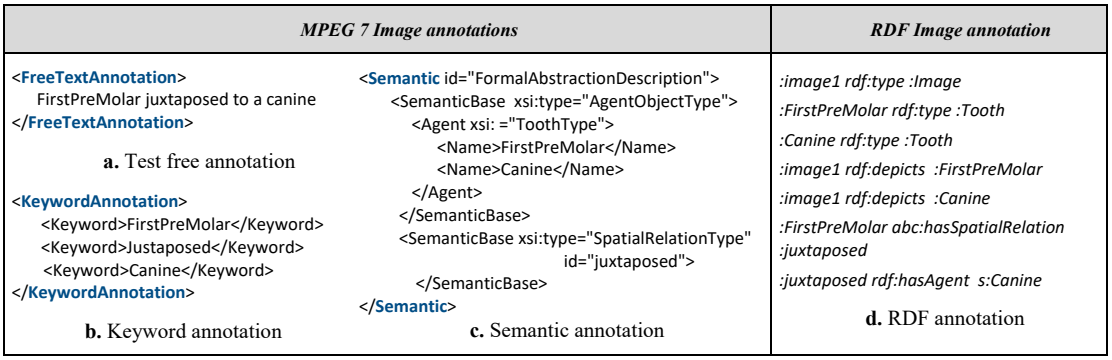


Figure 2. MPEG 7 image annotations versus RDF image annotation

Other limitations regarding the semantic capabilities of MPEG-7 include: i) the lack of a unified procedure to decompose a multimedia object (e.g., an image) and identify the relations between its constituents [50], and ii) the lack of capabilities allowing the deduction of facts (semantic descriptions) based on existing ones (i.e., identifying the semantic relations between multimedia objects, as well as between objects and query keywords), which is central to allow semantic-based search and processing (such as semantic clustering and classification) [51]. Some approaches have suggested creating MPEG-7 ontologies (based on RDF and OWL) to better handle MPEG-7 semantic descriptions [52]. In contrast, our approach allows a formalized decomposition of image content using SVG-based RDF descriptions, while providing an open framework for semantic annotation and fact deduction readily available for use with emerging RDF deductive systems, e.g., [53, 54]. Such capabilities underline the basic pillars necessary for the development and deployment of RDF and Semantic Web databases [55, 56]

3.5 Vector-based Representation and Similarity Methods

Few approaches have specifically targeted SVG image representation and processing [57, 58]. The work in [59] suggests organizing the 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 a 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.* Another approach in [12] 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 retrieval of SVG images using XML data search and manipulation techniques have been proposed in [11, 14]. These approaches exploit XML syntactic processing capabilities, yet do not address semantic annotation or semantic-based retrieval.

A few studies have introduced vector shape similarity measures designed to compare SVG geometric objects, e.g., [11, 13, 60, 61]. The authors in [13] introduce a number so similarity measures to compare same shape and different shape objects. The main motivation in [13] is to support geographical location based services: identifying similar geographic sites represented via similar SVG geometric objects, using approximate similarity-based SVG object search. For similar shapes, invariant points are identified and used to produce a general mathematical equation that can be used for comparison. For different shapes, the proximity of two object contours is computed as the distance between their corner points. The authors in [11] compute the similarity between two vector objects as the sum of shape, color, and position similarity measures. They compute an interior angle sequence defined as the angle formed by 2 adjacent sides of a polygon using Formula 2 (cf. Section 4.2.1). They also calculate a length sequence defined as the length of a line segment of path data, using Euclidean distance. Then they normalize by dividing each length value by the maximum length (L_i/L_{max}). A similar approach in [60] adopt edit distance to compute the interior angle sequence and length sequence between two polygon shapes. However, the authors do not mention how the edit-distance measure is computed. In [62], the authors extract a set of points designating the shape's perimeter using the current transformation matrix (CTM). They also introduced a smoothing function to transform a distance into a similarity measure. The authors in [63] introduce a tool allowing users to manually associate semantic annotations to a sketch based query specification. Images are drawn and transformed into SVG coding, whereas user annotations are transformed into an RDF fragment appended to the SVG image code. Yet this approach solely focuses on manual user annotation and does not address semi-automatic annotation. The authors in [61] extract shape contour and introduce a technique for centroid distance computation, as the distance from every contour point to the centroid of the object. They also consider the angles between the line from the centroid point to the i^{th} contour point and the tangent line of the object at the contour point, compared using cosine similarity. In our study, we build on the approaches mentioned above and use some of their similarity measures in designing our *similarity computation* component (cf. Section 4.2).

To sum up, i) most existing SVG solutions represent images using native SVG coding, compared with our semantically augmented RDF-based representation, ii) they represent images as separate standalone entities, compared with our solution which introduces the concept of unified reference KG to gather the collective semantics of an image repository, and improve image semantic processing, iii) they process images separately performing pairwise image similarity computations, in contrast with our approach which introduces SVG image and object clustering, comparing images with cluster representatives to improve annotation quality and performance.

3.6 Image Clustering Methods

Various clustering solutions have been proposed to organize raster image repositories and search results. They mainly fall under two main groups: i) partitional and ii) hierarchical. Partitional clustering algorithms attempt to divide data objects (e.g., images) into non-overlapping subsets, i.e., the clusters, by maximizing intra-cluster similarity and minimizing inter-cluster similarity. K-means [64] is one of the most popular algorithms in this category and attempts to recursively minimize the distance between objects in a cluster and a special object designated as the center of the cluster (computed as the average between all objects in the cluster). Similar algorithms such as k-medians, k-medoids, and BSAS (Basic Sequential Algorithmic Scheme) have also been suggested [65-67]. Partitional algorithms are usually intuitive, easy to implement, and relatively efficient (e.g., k-means is of average $O(n \times k \times i)$ time where n is the number of images, k is the number of clusters, and i is number of iterations). Nonetheless, they usually require the user to specify certain parameters like the number of clusters which is not always known in advance. A few algorithms such as x-means [68] and YAK [69] have attempted to counter the previous limitation, yet, they rely on empirical heuristics and their usage remains limited [3].

Hierarchical clustering generates a set of nested clusters organized in a hierarchy, called *dendrogram*, where the root node of the dendrogram represents the whole dataset and each leaf node represents an individual data object (e.g., image). The cluster hierarchy is produced based on the similarity between individual data objects and clusters. In [70], the authors introduce a three-step hierarchical clustering approach for Web images: i) the first step clusters the input images based on the concepts extracted from their textual metadata (URL of the containing webpage and image anchor text), ii) the second step accepts as input the clusters of the first step and merges them based the images' textual contexts (surrounding text in the containing webpage), iii) the third step accepts as input the clusters produced in the second step and expands the context of each cluster using Wikipedia semantics to merge the ones sharing the most similar contexts. A similar approach is described in [71] where the authors consider labels added using social tagging, photo taking metadata, and low-level image features. In [72], the authors introduce the folding tree hierarchical algorithm: it starts from the individual images which represent leaf nodes, and merges the most similar ones based on their visual features, to form the first level clusters represented as inner tree nodes. Tree traversal is then performed level-by-level from the leaves to the root. Hierarchical clustering algorithms usually produce better results compared with their partitional counterparts, yet they are more computationally complex (requiring at least $O(n^2 \times \log(n))$ time where n is number of images being clustered) [3].

Note that neither partitional nor hierarchical clustering solutions can be straightforwardly utilized in our study, since our SVG image repository is not initially populated for processing (except for a few seed images): it is populated incrementally as more and more images are added and requested for annotation. Hence, we design a new incremental clustering solution to process the stream of incoming SVG images, one-by-one, as they are added to the dataset.

4. SVG Semantic Graph (*SSG*) Framework

An overview of our *SSG* image semantization framework is depicted in Figure 3. It consists of four main modules: i) SVG-to-KG conversion, ii) SVG similarity computation, iii) SVG image and object clustering, and iv) SVG annotation and KG integration. An input SVG image is first processed for feature extraction by identifying its

constituent objects and their properties, and converting them into a KG representation in the form of RDF *subject-predicate-object* triples (cf. *Challenge 1*). Object features are processed for similarity computation using dedicated shape, area, location, and color similarity measures. The image is run through an incremental clustering process, grouping it with the most similar cluster representatives based on their aggregate object similarities. The image is then labeled according to its associated cluster, and is integrated in the reference KG accordingly (cf. *Challenge 2*). The reference KG's semantic expressiveness continuously increases as new images are labelled by the system. The user can choose to fine-tune each module, by choosing the most representative features, fine-tuning the similarity measures' weights, choosing the cluster seed representatives, and validating the system generated annotations before integration in the reference KG (cf. *Challenge 3*). We describe each module in the following sub-sections.

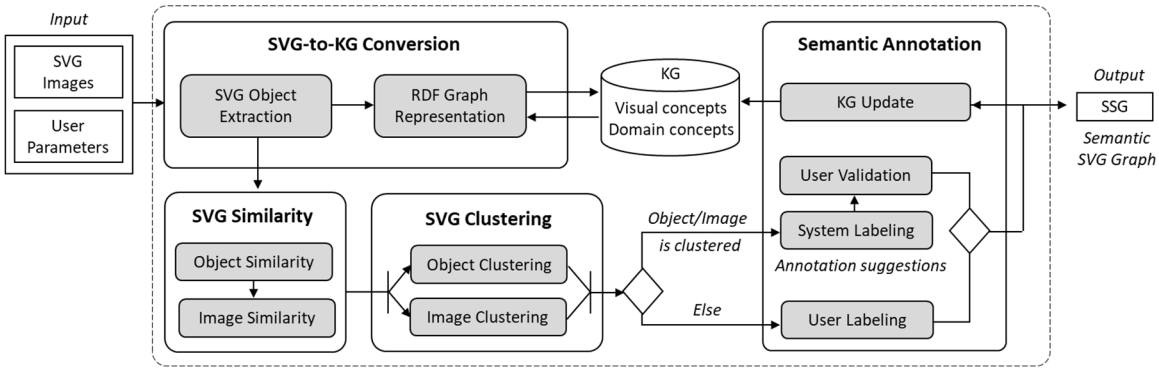


Figure 3. Simplified diagram describing our *SSG* image semantization framework

4.1 SVG-to-KG Conversion

The SVG-to-KG representation module allows converting an SVG image into a knowledge representation that can be integrated in a reference KG. We first define the SVG and KG data representations adopted in our study, and then describe our SVG-to-KG conversion algorithm.

Table 1. Basic SVG geometric objects and their properties

Objects (shapes)	**Attributes** (properties)	**Visual presentation**	**SVG source code** (tagging)	**Description**
Circle	*cx, cy, r, stroke, fill*		<circle cx="50" cy="50" r="40" fill="red" stroke="black" >	*cx* and *cy* define the coordinates of the circle's center, *r* its radius, *fill* and *stroke* its fill/contour colors
Ellipse	*cx, cy, rx, ry, stroke, fill*		<ellipse cx="200" cy="80" rx="100" ry="50" fill = "purple" />	*rx* and *ry* define the ellipse' radiuses along the x and y axes respectively
Rectangle	*x, y, width, height, stroke, fill*		<rect x="50" y="50" width="100" height="300" stroke = "red"/>	*x* and *y* define the coordinates of the top-left vertex of the rectangle
Line	*x1, y1, x2, y2, stroke*		<line x1="0" y1="200" x2="200" y2="0" stroke = "red" />	*x*1 and *y*1 define the line's start point coordinates, and *x*2 and *y*2 its end point coordinates
Polygon/ Polyline	*points, stroke, fill*		<polygon points="220,10 300,210 170,250 123,234" />	*points* defines the *x* and *y* coordinates for each corner of the polygon, e.g., (220, 10), (300, 210), (170, 250), and (123, 234) respectively.
Path	*d, stroke, fill*		<path d="M130,0 L50,10 L230,300 Z" />	*d* is the sequence of points in the path, e.g., starting at point (130, 0) with a line to (50, 10), then a line to (230, 300), and closing at (130, 0))

4.1.1 SVG Representation

SVG allows encoding a variety of geometric objects having different features. Formally:

Definition 1 – SVG Image Document: It is an XML-based document representing structured and self-contained information consisting of *elements* representing geometric object *shapes*, and *attributes* defining, for each shape, a set of descriptive features known as geometric object *properties* (cf. Table 1) ●

Since SVG is an XML-based coding, SVG feature extraction comes down to retrieving attributes using traditional XQuery and XPath queries. This is a major advantage over traditional low-level feature extraction from raster images which may require extensive processing (depending on the nature and dimensionality of the features being extracted [26, 73]), in comparison with fast XQuery/XPath processing (which is based on optimized database-style semi-structured data processing [74, 75], cf. Figure 4).

4.1.2 KG Representation

SVG coding provides the structural properties of vector images, yet it does not provide any semantic meaning (e.g., the SVG coding in Figure 4 consists of a path object shape, but does not reflect semantic meaning such as: *the path is a molar tooth*). To solve this issue, we propose to represent SVG images, objects, and their properties as RDF *subject-predicate-object* triples[6], producing a semantic image representation in the form of an RDF graph. The latter is referred to as our SVG Semantic Graph (*SSG*) representation. The *SSG* is then integrated with the reference KG representation to provide context and meaning to the SVG image. Formally:

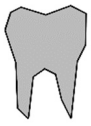
Query: *Select the visual attributes of path elements having a fill color = "dark gray"*			
SVG Raw image	**SVG source code**	**XPath query**	**Result:** Visual Features
 *Img1.svg*	<?xml version-"1,0"?> <svg width="20cm" height="20cm"> <path d=" M18,4 C0,0 -26 -15 17,18 C 9,33,3,18,8,31 S-1-25,7 20.5 S6,19, 4,22 S15 - 31,15 - 43 S26 - 4,19,4 Z" stroke="black" fill="dark gray"/> </svg>	//path[@fill="dark gray"]/@*	**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 =** "dark gray"

Figure 4. Example of SVG feature extraction using a typical XPath query statement

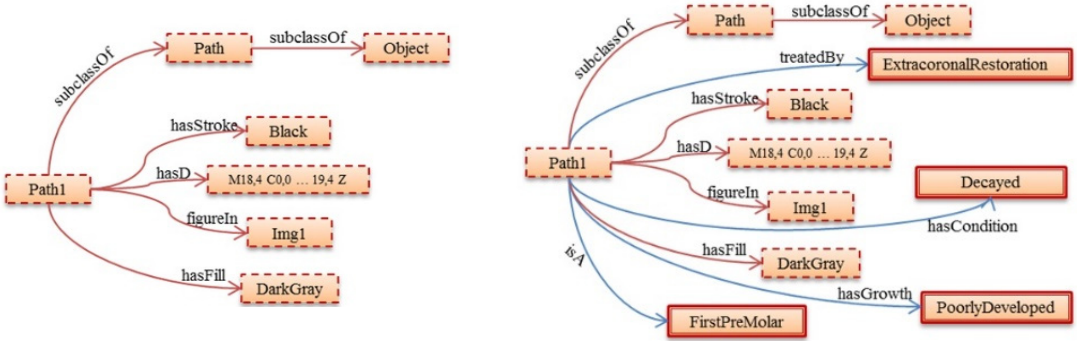
Definition 2 – SVG Semantic Graph (*SSG*): It is a graph (*N*, *E, L*) consisting of a collection of RDF *subject-predicate-object* triples describing SVG image content, where:

- Nodes $n_i \in N$ designate RDF *subjects* or *objects* representing:
  - Visual concepts describing geometric objects (e.g., ellipse, circle, path) extracted from the SVG image,
  - Domain concepts (e.g., molar tooth, canine tooth) added after annotation from the reference KG to provide semantic meaning,
  - and corresponding geometric object property values (e.g., 50 is the value of property *stroke* for an ellipse object, "red" is the value of its *fillColor*).
- Edges $e_i^j \in E$ connecting source/destination nodes $(n_i, n_j) \in N$ designate *predicates* representing:

[6] In an RDF *subject-predicate-object* expression, the subject denotes the *resource* being described, the predicate denotes a *trait* or *aspect* of the resource, expressing a relation between the subject and the object, where the object designates another resource or a data value [76].

- Relations between geometric objects (e.g., Path1-*subClassOf*-Path),
- Relations between domain concepts (e.g., Tooth-*hasInfluentialFacts*-Symptom),
- Relations in-between geometric objects and domain concepts (e.g., Path1-*isA*-Molar),
- Relations between geometric objects and their property values (e.g., ellipse1- *hasStroke*-50, ellipse1-*hasFillColor*-"red").

– Labels $\ell_i \in L$ designate node labels, and $\ell_i^j \in L$ designate edge labels, cf. Figure 5) ●



a. *SSG* resulting from extracted visual features of *Img1.svg* in Figure 4

b. *SSG* after annotation, including semantic triples (highlighted in bold)

Figure 5. Sample SVG feature extraction and the resulting KG representations

Figure 5.a shows an *SSG* graph which is automatically generated from the SVG image in Figure 4. Triple *Path1-figureIn-Img1* is added to indicate that the path is included in the image. Figure 6 shows an extract of the reference KG adopted in our study. It is based on a dental knowledge base from [77], which we manually extended to include SVG geometric object constructs and properties (cf. Figure 19). The reference KG is also represented following our *SSG* data model (cf. Definition 2), and provides domain experts with a set of predefined visual and semantic concepts and relations corresponding to the application domain at hand. The reference KG is dynamically extended by creating new concept and relation instances based on the images being annotated.

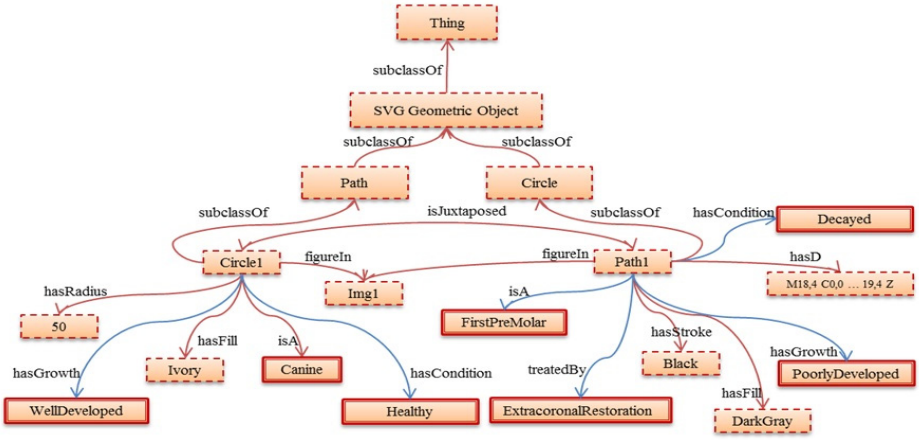


Figure 6. Extract of the reference KG used in our study

Algorithm *SVG-to-KG_Conversion*

Input: Img // SVG image document
KG // Knowledge graph represented using the SSG data model

Output: SSG // Semantic SVG image graph

```
Begin
   SSG = null                                          // Initializing SSG                          1
   List GO = Img.getGeometricObjects(Img)              // Retrieving geometric objects from Img     2
   For each(object oi in GO)                                                                        3
   {                                                                                                4
     SSG.addNode(oi)                                   // Object node insertion in the SSG          5
     For each(object oj ≠ oi in SSG)                                                                6
     {                                                                                              7
        ℓ^j_i = KG.getEdgeLabel(oi, oj)                // Acquiring edge labels from KG             8
        SSG.addEdge(oi, oj, ℓ^j_i)                                                                  9
     }                                                                                              10
     List ATT = oi.getAttributes()                     // List of attributes of object oi           11
     For each(attribute atk in ATT)                                                                 12
     {                                                                                              13
       SSG.addNode(atk)                                // Attribute node insertion in SSG           14
       ℓ^k_i = KG.getEdgeLabel(si, atj)                // Acquiring edge labels from KG             15
       SSG.addEdge(si, atj, ℓ^k_i)                                                                  16
     }                                                                                              17
   }                                                                                                18
   Return SSG                                                                                       19
End
```

Figure 7. Pseudo-code of *SVG-to-KG_Conversion* algorithm

4.1.3 SVG-to-KG Conversion

Algorithm *SVG-to-KG Conversion* for extracting SVG features and building the corresponding *SSG* graph is shown in Figure 7. It accepts as input an SVG image and a reference KG, and produces as output the corresponding *SSG* graph. The algorithm starts by extracting geometric objects from the SVG image, and inserting the latter as nodes in the *SSG* graph (lines 3-5). Geometric object nodes are connected with other nodes in the *SSG* graph by creating edges with their labels extracted from the KG (lines 6-10). Dedicated graph nodes are created for each of the geometric objects' attributes (lines 11-14), using dedicate edges with labels extracted from the KG (lines 15-16).

4.2 SVG Similarity Computation

Once the *SSG* graph is produced for a given SVG image, the annotation process is executed by comparing the image's geometric objects with those stored in the reference KG. The latter are run through an incremental clustering process (described in the following Section 3.4), grouping them with the most similar cluster representatives based on their aggregate object similarities. The similarity computation process is depicted in Figure 8. We consider four main similarity criteria: i) shape, i) area, iii) position, and iv) color. Given two SVG geometric objects o_1 and o_2:

$$\mathrm{Sim}(o_1, o_2) = \sum_{f \in F} w_f \times \mathrm{Sim}_f(o_1, o_2) \quad \in [0, 1] \tag{1}$$

where F={*Shape*, *Area*, *Position*, *Color*}, $\sum_{w_f} = 1$, $\forall_{w_f} \geq 0$, such that $Sim_f(o_1, o_2) \in [0, 1]$. While different aggregation functions can be used (e.g., *maximum*, *average*, etc.), we utilize the *weighted sum* function to combine the different similarities, allowing users to fine-tune the weight of each criterion according to their notion of similarity. Note that the fine-tuning of similarity weight parameters is an optimization problem such that parameters should be chosen to maximize annotation quality (through some cost function such as the *f-value* measure, cf. Section 6.3). This can be solved using a number of techniques that apply linear programming or machine learning in order to identify the best weights for a given problem class, e.g., [78-80]. Providing such a capability, in addition to manual tuning, would enable the user to start from a sensible choice of values (e.g., identical weight parameters to consider all similarity measures) and then optimize and adapt the process following the optimization function at hand. Users can also add or remove their own similarity functions suitable to their application domain. We do not further address the fine-tuning of parameters here since it is out of the scope of this paper, and will be addressed in a dedicated study.

In the following, we integrate existing measures from the literature and introduce new measures to handle each similarity criterion. Existing measures are explicitly highlighted and referenced when used.

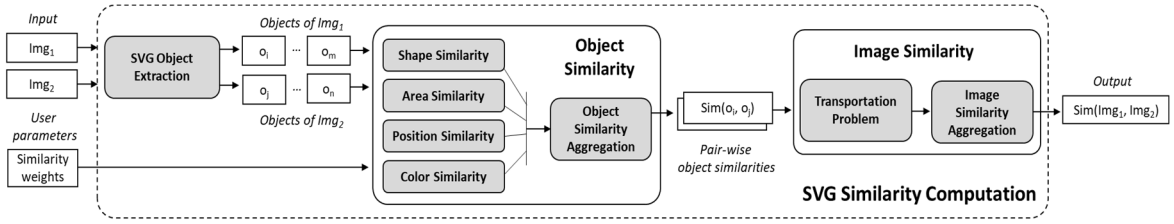


Figure 8. Simplified diagram describing our similarity computation process

4.2.1 Shape Similarity

SVG shape similarity can be performed by comparing geometric objects of the same type (e.g., comparing two *circles*, or two *rectangles*), or by comparing objects of different shape types (e.g., comparing a *circle* with a *rectangle*). On the one hand, comparing objects of the same type is achieved by comparing their mathematical properties using a set of well-defined mathematical formulas (e.g., comparing the radiuses of two circles, comparing the width and height values of two rectangles). The authors in [13] introduce a set of mathematical formulas specially tailored for the task of comparing same type SVG objects, which we adopt in our implementation. On the other hand, we compare geometric objects of different types by first i) transforming the objects into their most generic polygon representations, and then ii) comparing the resulting polygons. A polygon consists of a sequence of vector moments which makes it invariant to transformations. From each polygon, we extract: i) the interior angle sequence, where each angle is formed by two adjacent sides of the polygon (computed using Formula 2 from [11]) and ii) the side length sequence, consisting of the length values of the adjacent sides of the polygon (computed using Euclidian distance between the sides' edge points).

$$\theta_{a,b} = \cos^{-1}\left(\frac{\vec{a}.\vec{b}}{\|\vec{a}\| \times \|\vec{b}\|}\right) \quad \in [0, \pi] \qquad \textbf{(2)}$$

where a and b are two adjacent sides of a polygon, and $\theta_{a,b}$ is their internal angle (cf. Figure 9). We compute the similarities between pairs of angle/side sequences using cosine similarity, and then aggregate them to compute polygon shape similarity. More formally, given two polygon objects p_1 and p_2:

$$Sim_{Shape}(p_1, p_2) = w_{AS} \times Sim_{Cosine}(AS(p_1), AS(p_2)) + w_{LS} \times Sim_{Cosine}(LS(p_1), LS(p_2)) \quad \in [0, 1] \qquad \textbf{(3)}$$

where $AS(p_i)$ and $LS(p_i)$ represent the angle and side length sequences of polygon p_i respectively, w_{AS} and w_{LS} represent the angle and side sequences' similarity weights having $w_{AS} + w_{LS} = 1$ and $(w_{AS}, w_{LS}) \geq 0$. When comparing two sequences with different dimensions, we pad the sequence having the smaller dimensionality with zero weights in order to reach the same size of its bigger counterpart. This comes down to adding blank contour angles/sides to the smaller polygon, which is reasonable since polygons made of different numbers of angles/sides (i.e., of different resolutions) are indeed not the same. Their difference will be reflected through the cosine similarity computation (cf. Formula 3), where more/less zero weights would yield lower/higher similarity scores respectively. As for the ordering of the sequence dimensions, it depends on the ordering of the sequences themselves, which in turn depends on the starting point of each polygon object. In other words, similar polygons with different starting points can generate different angle/side sequences, resulting in low sequence similarities which is counter intuitive. To solve this problem, we adopt a solution from [61] by computing sequence similarity $2\times n$ times, where n is the length of the longest (angle/side) sequence among the polygons being compared. Each angle/side dimension is shifted left for the first iteration and then shifted right for the second iteration, allowing to consider all possible starting point configurations when computing similarity. Finally, the highest cosine similarity score is considered as the output sequence similarity.

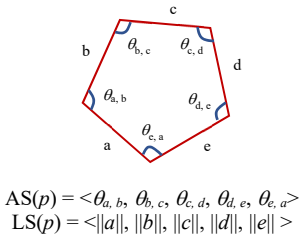


Figure 9. Sample polygon with its angle and size representations

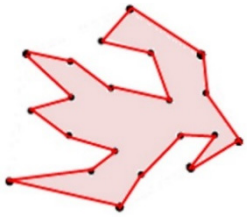


Figure 10. Area surface coverage using Gauss's area approach [81]

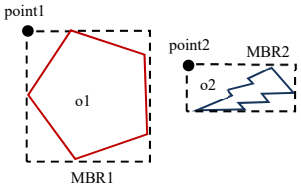


Figure 11. Sample minimum bounding rectangles and their reference points

4.2.2 Area Similarity

We identify the coverage area of a polygon based on the Cartesian coordinates of its contour points using Gauss's area formula [81] (cf. Figure 10). More formally, considering a polygon object p made of a sequence of i=1…n contour points having Cartesian coordinates (x_i, y_i):

$$\text{area}(p) = \frac{1}{2}\left|\sum_{i=1}^{n-1} x_i \times y_{i+1} + x_n \times y_1 - \sum_{i=1}^{n-1} x_{i+1} \times y_i - x_1 \times y_n\right| \quad \textbf{(4)}$$

Consequently, we compute the similarity between the coverage areas of two polygons p_1 and p_2 as the absolute normalized difference between their coverage areas:

$$Sim_{Area}(p_1, p_2) = \frac{|\text{area}(p_1) - \text{area}(p_2)|}{max(\text{area}(p_1), \text{area}(p_2))} \in [0,1] \quad \textbf{(5)}$$

4.2.3 Position Similarity

To compare the positions of two objects o_1 and o_2 in an SVG image, 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 ($point_1$ and $point_2$), where the top-left vertex serves as a reference location point for an SVG rectangle object (cf. Figure 11).

$$Sim_{Pos}(o_1, o_2) = \frac{1}{1 + Dist_{Euclidian}(point_1, point_2)} \in [0,1] \quad \textbf{(6)}$$

4.2.4 Color Similarity

Colors are traditionally defined on a selected color space, such as RGB or HSV [82], each serving a different set of applications, where each color is coded as a set of integers values. More recently, color ontologies have been introduced 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.) organized in an ontological graph structure [83, 84]. Since SVG allows coding colors using both: i) numerical format in the RGB feature space, and ii) color names with 147 reference colors [19], we adopt both color representations in our approach and calculate the similarity between two colors by combining their visual properties and their semantic meaning as follows:

$$\mathrm{Sim}_{\mathrm{FillColor/StrokeColor}}(c_1, c_2) = w_{\mathrm{HSV}} \times \mathrm{Sim}_{\mathrm{HSV}}(c_1, c_2) + w_{\mathrm{Ont}} \times \mathrm{Sim}_{\mathrm{Ont}}(c_1, c_2) \quad \textbf{(7)}$$

While SVG represents colors in numerical format following 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. 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 graph ontology, e.g. [85, 86]. In our implementation, we combine two semantic similarity approaches by *WuPalmer*[7] and *Lin*[8] due to their prolific usage in the literature. Consequently, given two objects o_1 and o_2, we compute their color similarity as the aggregation of their fill color and stroke color properties:

$$\mathrm{Sim}_{\mathrm{Color}}(o_1, o_2) = w_{\mathrm{FillColor}} \times \mathrm{Sim}_{\mathrm{FillColor}}(fc_1, fc_2) + w_{\mathrm{StrokeColor}} \times \mathrm{Sim}_{\mathrm{StrokeColor}}(sc_1, sc_2) \quad \textbf{(8)}$$

where $(w_{\mathrm{FillColor}}, w_{\mathrm{StrokeColor}}) \geq 0$ and $w_{\mathrm{FillColor}} + w_{\mathrm{StrokeColor}} = 1$ such that $(\mathrm{Sim}_{\mathrm{FillColor}}, \mathrm{Sim}_{\mathrm{StrokeColor}}) \in [0, 1]$, fc_1 and fc_2 designate the fill colors of objects o_1 and o_2 respectively, and sc_1 and sc_2 designate their stroke colors.

4.2.5 Image Similarity

We compute the similarity between two SVG images Img_1 and Img_2 based on the aggregated similarities of their constituent objects. We utilize the transportation optimization problem, e.g., [90, 91], to match the highest similarity objects from both images. The transportation problem seeks to associate a number of supply centers m (sources) with a number of demand centers n (destinations) to optimize supply delivery. In our case, we consider the objects of the first image $o_i \in Img_1$ to be the supply centers, and the objects of the second image $o_j \in Img_2$ to be the demand centers. Considering two images with $m = |Img_1|$ and $n = |Img_2|$ objects respectively, we construct an $m \times n$ matrix where the rows represent the objects of the first image and the columns represent the objects of the second image. Consequently, we match the nodes together using the transportation problem's *minimum* (*least*) *cost method* widely adopted in the literature, e.g., [90, 91]. We compute cost as the inverse of similarity, and hence we seek to minimize the cost (i.e., maximize the similarity) among the matching objects. We briefly describe the process as follows: (i) assign the supply center (object from the first image) with the demand center (object from the second image) having the highest pair-wise similarity, (ii) cross-out the row where the supply center is located, (iii) cross-out the column where the demand

---

[7] $\mathrm{Sim}_{\mathrm{Lin}}(c_1, c_2, CO) = \dfrac{2 \times \log p(c_0)}{\log p(c_1) + \log p(c_2)} \in [0,1]$

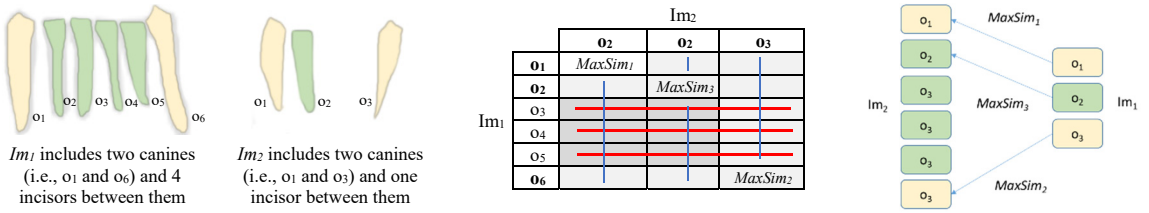
where CO designates a reference hierarchical color ontology, N_1 and N_2 are respectively the lengths of the paths separating colors c_1 and c_2 from their lowest common ancestor color c_0 in CO, and N_0 is the length of the path separating color c_0 from the root of CO [89].

[8] $\mathrm{Sim}_{\mathrm{WuPalmer}}(c_1, c_2, CO) = \dfrac{2 \times N_0}{N_1 + N_2 + 2 \times N_0} \in [0,1]$

where $p(C_i)$ denotes the occurrence probability of color c_i designating the frequency of occurrence of the name color c_i in a reference corpus [88], such as the Brown text corpus [87] adopted in our study.

center has been satisfied, (iv) repeat iteratively from (i) to assign the remaining objects until no row or no column is left. Figure 12 shows a visualuzation of transporation problem's minimum cost (maximum simialrity) method. Lastly, the similarity between two images Img_1 and Img_2 is computed as the sum of the similarities of their matchings objects, normalized by maximum image cardinality:

$$\mathrm{Sim}(\mathrm{Img}_1, \mathrm{Img}_2) = \frac{\sum_{o_i \in \mathrm{Img}_1,\, o_j \in \mathrm{Img}_1 \,/\, o_i \text{ matches } o_j} Sim(o_i, o_j)}{max(|\mathrm{Img}_1|, |\mathrm{Img}_2|)} \quad \in [0,1] \tag{9}$$



Im1 includes two canines (i.e., o1 and o6) and 4 incisors between them

Im2 includes two canines (i.e., o1 and o3) and one incisor between them

a. Sample images obtained from panoramic x-rays of human front lower teeth

b. Visualization of the transportation matrix (matching objects are shown in bold)

c. Visualization of the matching objects

Figure 12. Sample visualization of the transportation problem, used to identify matching SVG objects according to their maximum similarities

4.3 Image and Object Clustering

Our clustering approach is depicted in Figure 13. We adopt an incremental clustering process [92] which considers images and objects one by one in an incremental manner, and decides what to do with (where to put) them. Incremental clustering is especially useful when processing streams of data, which is the case in our scenario where streams of input SVG images and their constituent objects are incrementally processed and annotated by the system.

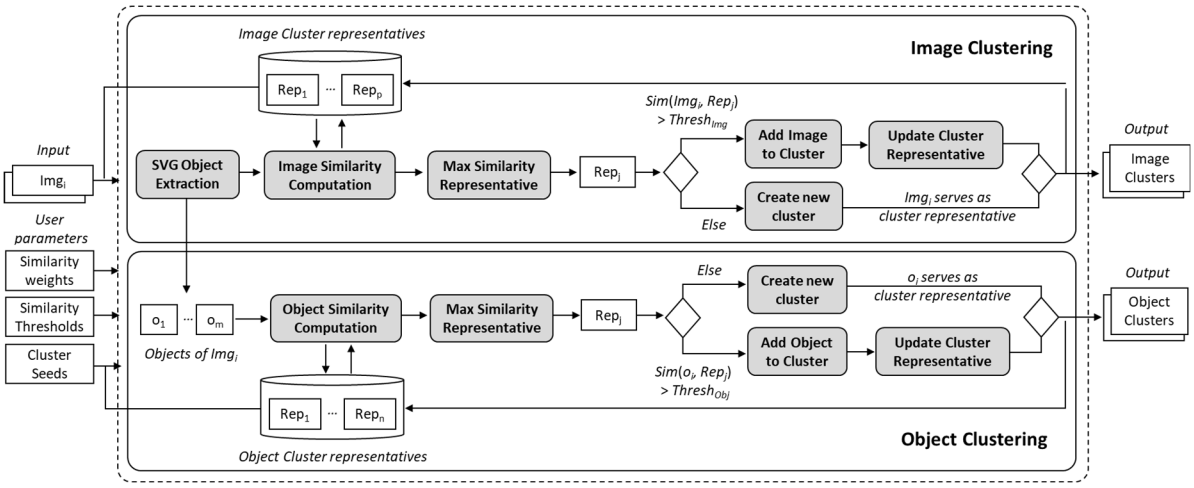


Figure 13. Simplified diagram describing our clustering process

4.3.1 Clustering Process

Our clustering process consists of two instances for i) clustering images and ii) clustering objects, and outputs two sets of clusters grouping similar images together and similar object together. The pseudocode of our *Image_Clustering* algorithm is shown in Figure 14. It takes as input the first image and compares it with each of the image cluster representatives (performing image similarity computation using the aggregated similarities of their constituent objects, cf. previous Section 4.2). The algorithm identifies the cluster representative having the highest similarity with the image (cf. Figure 14, lines 1-2), and then decides whether the image should be placed in the cluster (lines 10-12) – or whether a new cluster should be created around the image – based on a user chosen (or system-generated average) similarity threshold (lines 3-8). If the image is added to an existing cluster, the cluster representative is updated accordingly (lines 10-12, following a process described in Section 4.3.2) before processing the following image. Otherwise if a new cluster is created around the image, the image will serve as the cluster's initial representative. The algorithm continues in the same manner until all images have been clustered. Another instance of the same clustering process is applied on the image's individual objects, grouping them in object clusters. The output image and object clusters are used to perform image and object annotation, which we describe in Section 4.4.

```
Algorithm Image_Clustering

Input: S                                          // Stream of newly added images
       R                                          // Set of seed cluster representatives
       Thresh_Img                                 // Image similarity threshold
       Weights = {w_Shape, ..., w_StrokeColor)    // Set of similarity weights

Output: C                                         // set of image clusters

Begin
    For each Img ∈ S                                                              1
        Find max(Sim_Weights(Img, Rep_i)) where Rep_i ∈ R   // Find maximum similarity image   2
          If max(Sim_Weights(Img, Rep_i)) < Thresh_Img      // If similarity less than threshold,  3
          {                                                // then create a new cluster          4
             Create new cluster c_new                                                 5
             Add Img to c_new                                                         6
             Designate Img as Rep_New                                                 7
             Add c_new to C                                                           8
          }                                                                           9
          Else                                             // If similarity above threshold,     10
             Add Img to cluster c_i having Rep_i as representative   // then update existing cluster  11
             Update_Cluster_Representative(c_i)                                       12

    If user wishes to update threshold                                            13
       Set Thresh_Img as average similarity of all images in C                    14

    Return C                                                                      15
End
```

Figure 14. Pseudo-code of *Image_Clustering* algorithm

4.3.2 Cluster Representatives

The pseudocode of our *Cluster_Representatives_Update* algorithm is shown in Figure 15. It considers two approaches when defining the cluster representatives: i) maximum cumulative sum, and ii) higher membership heuristic. Following the first approach, the image that is most similar to all other images in the cluster is chosen as the cluster's representative (cf. Figure 15, lines 1-6). This is done by comparing each image with all others and computing the

cumulative sum of all pair-wise similarities. Consequently, the image with the maximum cumulative sum is chosen to be the representative. While effective, yet this approach is computationally expensive where *n* similarity computation operations are performed each time a new image is added, *n* being the number of images in the cluster. To solve this issue, we introduce the *higher membership heuristic* approach: i) for the first three images added to a cluster, we apply the *maximum cumulative sum* approach described previously (lines 9-11), ii) we compute the average cumulative sum score for the representative image by normalizing its cumulative sum with the number of similarity computation operations – we refer to this average as the representative's *membership* score with respect to the cluster (line 12), iii) starting from the forth image onward, we only compute the new image's similarity with the cluster representative (lines 13-15), iv) we consider a heuristic assumption: if the similarity score is greater than the representative's membership score, we consider that the new image represents the cluster better than the representative – and the new image becomes the new cluster representative, otherwise we keep the old representative unchanged (lines 14-15). This heuristic process is repeated for every other image, and requires constant unit time regardless of cluster size. Users can choose to apply the maximum cumulative approach or the higher membership heuristic approach according to their needs.

Algorithm *Cluster_Representatives_Update*

Input: c_i // Image cluster
MaxCumul // Boolean parameter
Weights = {w_{Shape}, …, $w_{StrokeColor}$) // Similarity weights

Ouput: Rep_i // Representative image of cluster c_i

```
Begin
   If MaxCumul is true                                                    // Maximum cumulative sum          1
   {                                                                                                      2
      For each Img_u ∈ c_i                                                                                3
         Compute sum(Sim_Weights(Img_u, Img_v)) where Img_v ∈ c_i                                          4
      Rep_i = Img_u having maximum sum(Sim_Weights(Img_u, Img_v))          // Representative with max similarity  5
   }                                                                                                      6
   Else                                                                   // Higher membership heuristic     7
   {                                                                                                      8
     For each Img_u ∈ c_i                                                                                 9
       If Img_i is among first three images in c_i                        // Computing maximum cumulative sum  10
         Compute sum(Sim_Weights(Img_i, Img_v)) where Img_v ∈ c_i          // for first three images, and normalizing  11
            Score_i = sum(Sim_Weights(Img_i, Img_v)) / |c_i|               // with size of cluster            12
       Else If Img_u is forth image or more in c_i                        // Starting from forth image onward,  13
         If Sim_Weights(Img_u, Rep_i) > Score_i                           // compute similarity with         14
            Designate Img_u as new Rep_i                                 // cluster representative          15
   }                                                                                                      16
   Return Rep_i                                                                                           17
End                                                                                                       18
```

Figure 15. *Update_Cluster_Representative* algorithm

4.3.3 Cold Start

As a cold start, the user can provide a minimum of one representative for each image cluster and object cluster in order to guide the clustering process. This allows users to define their clusters of interest and label them according to the reference KG. Consequently, objects and images added to the user clusters are assigned the corresponding cluster

labels, preparing for their integration in the reference KG (discussed in the following subsection). Failing to provide the cold start representatives means the system will decide about the initial clusters itself, and define its own behavior according to the data being clustered. The user can verify the produced clusters at a later stage, and label them according to the reference KG. Both scenarios, i.e., with and without cold start representatives, are useful in different applications. Yet we recommend the user provides cold start representatives sooner than later, in order to form the seed clusters from the beginning, allowing for an effective annotation of the images and objects according to the user's target labels. This would help reduce user effort in updating the clusters and their labels at a later stage.

```
Algorithm SVG_Annotation
Input: Img                                          // SVG Image
       Weights = {w_Shape, ..., w_StrokeColor}      // Set of similarity weights
       Thresh_Img, Thresh_Obj                       // Similarity thresholds
       KG                                           // Reference knowledge graph
Output: SSG                                         // Annotated SSG
Begin
  N = ϕ                                                                          1
  c_Img = Image_Clustering(Img, Weights, Thresh_Img)                             2
  ℓ_Img = c_Img.getLabel()                                                       3
  membership_Img = Sim(Img, Rep(c_Img))                                          4
  N = N ∪ createNode(ℓ_Img, addAttribute(membership_Img))                        5
  For each object o_i in Img                                                     6
  c_i = Object_Clustering(o_i, Weights, Thresh_Obj)                              7
  ℓ_i = c_i.getLabel()                                                           8
  membership_i = Sim(o_i, Rep(c_i))                                              9
  N = N ∪ createNode(ℓ_Img, addAttribute(membership_i))                          10
  SSG = SVG-to-KG_Conversion(Img, KG)                                            11
  For each (n_k in SSG)                                                          12
  {   N' = UserValidation(N)          // User validates annotations              13
      For each (n_i ∈ N')                                                        14
      {   SSG.addNode(n_i)            // Node insertion in the SSG               15
          For each(object n_j ≠ n_i in SSG)                                      16
          {   ℓ_i^j = KG.getEdgeLabel(n_i, n_j)   // Acquiring edge labels       17
              SSG.addEdge(n_i, n_j, ℓ_i^j)                                       18
          }                                                                      19
      }                                                                          20
  }                                                                              21
  Return SSG                                                                     22
End
```

Figure 16. Pseudo-code of *SVG_Annotation* algorithm

4.4 SVG Annotation and KG Integration

The annotation process is run for each clustered SVG image and its constituent objects, providing the user with annotation suggestions. The pseudo-code for algorithm *SVG_Annotation* is shown in Figure 16. It invokes the SVG image and object clustering algorithms described in the previous section (cf. lines 2, 7). An image/object added to a cluster inherits the cluster's label as its annotation, along with a membership degree representing the image/object's

similarity with respect to the cluster's representative (lines 3-4, 8-9). Considering for instance two clusters c_1 and c_2 labelled *canine* tooth and *incisor* tooth respectively (cf. Figure 12), a new image Img_i added to c_2, having $Sim(Img_1, Rep(c_2)) = 0.8$ will inherit label *incisor* with membership degree 80% (i.e., the system suggested annotation for Img_1 is 80% *incisor*). Users can choose to accept, update, or disregard the system annotation according to their perceptions and needs. When users accept or update a system generated annotation, its membership degree becomes 100%, highlighting the users' validation of the system suggestion (cf. Figure 16, line 13). The accepted annotations are then appended to the SSG representation of the image (lines 11-15) with the corresponding edge links acquired from the reference KG (lines 16-18).

Consequently, the *SSG* image graph is integrated in the reference KG, by appending the *SSG* graph nodes as instance nodes under their corresponding categories in the KG (e.g., nodes representing circle objects are appended as instances under the category *circle*, nodes representing molar teeth are appended under the category *molar tooth*, etc.). The pseudo-code for algorithm *KG_Update* is provided in Figure 17. It accepts as input an *SSG* image graph and the reference KG, and produces as output the updated KG. Nodes in the *SSG* graph which are not already present in the KG are appended to the latter (lines 2-6) with the corresponding edge links (lines 7-11). In addition, edges in the *SSG* graph which are not already present in the KG are appended to the latter (lines 13-19).

```
Algorithm KG_Update
Input:  SSG                                  // SVG Semantic Graph
        KG                                   // Reference knowledge graph
Ouput:  KG'                                  // Updated knowledge graph
Begin
   KG' = KG                                                                  1
   For each (node ni in SSG)                                                 2
   {                                                                         3
      If (ni ∉ KG')                                                          4
      {                                                                      5
        KG'.addNode(ni)                      // Node insertion in KG'        6
        For each(object nj ≠ ni in KB')                                      7
        {                                                                    8
          ℓ^j_i = KG.getEdgeLabel(ni, nj)    // Acquiring edge labels        9
          KG'.addEdge(ni, nj, ℓ^j_i)                                         10
        }                                                                    11
      }                                                                      12
      For each (edge e^k_i in SSG)                                           13
      {                                                                      14
        If ( e^k_i ∉ KG')                                                    15
            KG'.addEdge( e^k_i )             // Edge insertion in KG'        16
      }                                                                      17
   }                                                                         18
   Return KG'                                                                19
End
```

Figure 17. Pseudocode of *KG_Update* algorithm

5. Complexity Analysis

The time complexity of our solution simplifies to $O(N \times k)$, where N is the number of SVG images being processed, and k the number of clusters (i.e., the number of image and object categories). It comes down to the sum of the complexities of the modules:

- *SVG-to-KG conversion:* requires $O(N\times|Img|)$ to transform SVG images into *SSG* graph representations, where $|Img|$ is the number of objects in an SVG image.
- *SVG Similarity Computation:* requires $O(|Img| + |Img|^2)$ to compute object similarities and image similarities respectively, which simplifies to $O(|Img|^2)$.
- *SVG Image and object clustering*: requires $O(N\times k\times|Img|^2 + |Img|\times k)$ which comes down to the complexity of the incremental image and object clustering processes, where k is the number of clusters (i.e., the number of image and object categories).
- *SVG annotation and KG integration:* requires $O(|Img| + |KG|)$ where $|KG|$ represents the size of the reference knowledge graph.

Hence, overall similarity comes down to worst case $O(N\times|Img| + N\times k\times|Img|^2 + |Img|\times k + |Img|+|KG|)$. It comes down to $O(N\times k\times|Img|^2)$ as the largest factor, and simplifies to $O(N\times k)$ since usually $N >> |Img|$.

6. Experimental Evaluation

6.1 Prototype System

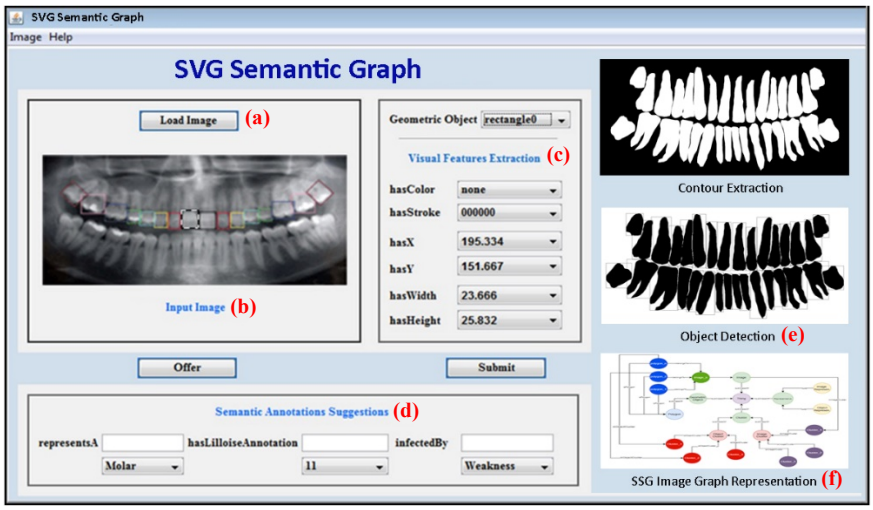


Figure 18. *SSG* prototype screen snapshot

We have developed a prototype system[9] to test and evaluate our *SSG* framework (cf. Figure 18). It is implemented using Java, making use of Neo4j to create, parse, and search our semantic graphs using the Cypher query language[10]. Users start by choosing the image they need to annotate (by clicking the *load image* button (Figure 18.a)). Users can also create on-the-fly SVG images on top of background raster images to annotate them (e.g., annotating a background panoramic dental x-ray image using SVG shapes (b)). Once the SVG image is loaded, the system automatically

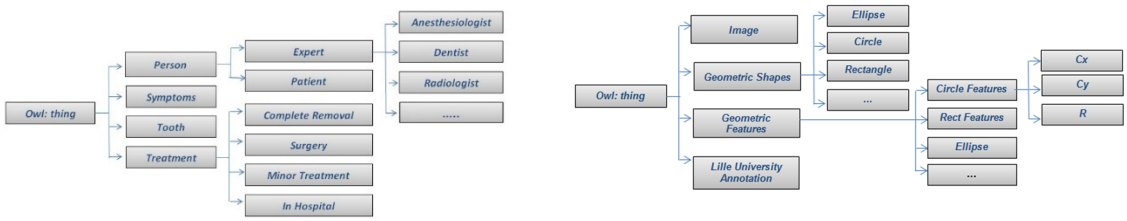
[9] Available online at: http://sigappfr.acm.org/Projects/SSG/

[10] We adopt Noe4j as a graph database to represent our semantic graphs, versus using a native RDF representation in Protégé [93] for instance, due to the latter's significant efficiency and processing speed compared with the latter. Other graph databases can be used as plug-and-play models according to the admin's preferences.

extracts SVG image features and produces the SSG graph representation, such that all the geometric object nodes in the SSG are filled in a combo-box (titled *GeometicObject*), and their visual properties are extracted and filled in other dedicated combo-boxes: *hasCx*, *hasCy*, *hasColor*, etc. (c). Users can then choose the geometric objects they wish to annotate, by clicking on the *offer* button to ask for annotation suggestions from the system (e). The *SSG* image graph is then displayed accordingly (f). Users can choose to validate/modify/disregard one/multiple system generated annotations, or add their own manual annotations (d). The annotation phase is concluded by clicking the *submit* button in order to save all annotations and append them to the reference KG.

6.2 Application Scenario and Data

While our framework is generic, yet we chose to test it in a real-world application scenario: clinical dental therapy (cf. Motivation in Section 2). Our tests are designed to process a collection of dental panoramic x-ray images in order to annotate their consistent teeth objects including: i) the type of each tooth (e.g., *incisor*, *premolar*), ii) the shape of the teeth (e.g., *poorly developed*, *decaying*, etc.), iii) the location of the teeth (*juxtaposed*, *evenly spaced*, etc.), and iii) the teeth color (*ivory* for healthy teeth, *dark gray* for decayed teeth, etc.). To provide domain specific annotations, we adopted a reference dental KG from [77], and we manually extended it to include SVG geometric object constructs and properties (cf. Figure 19).



a. Extract of original dental domain KG concepts [77][11]. **b.** Extract of SVG KB visual concepts developed in our study.

Figure 19. Reference dental KG[12].

6.3 Experimental Metrics

The main criteria used to evaluate the effectiveness of automatic annotation approaches are the amount of manual work and user effort required to perform the annotation task. This depends on: i) the quality of the produced annotations, as well as ii) the time needed to provide automatic annotations.

On one hand, to evaluate annotation quality, existing text/image annotation approaches in information retrieval (IR) propose to first manually produce annotations, and exploit the obtained results as a reference to evaluate the quality of the matches produced by the system [94, 95]. Thus, similarly to IR approaches, the *precision* and *recall* metrics can be utilized in comparing *"real"* and system generated annotations. *Precision* (*PR*) identifies the number of correctly generated annotations, w.r.t. the total number of annotations (correct and false) produced by the system. *Recall* (*R*) underlines the number of correctly identified annotations, w.r.t. the total number of correct annotations, including those not identified by the system. Having:

- *A* the number of correctly identified annotations (true positives),
- *B* the number of wrongly identified annotations (false positives),
- *C* the number of real annotations not identified by the system (false negatives).

---

[11] *Lille University annotation* is a tooth's annotation scheme, elaborated by the University of Lille in France, which can be integrated in any general purpose dental knowledge base [77].

[12] The complete knowledge graph is available online on the project prototype Web page.

Precision and *recall* are computed as follows:

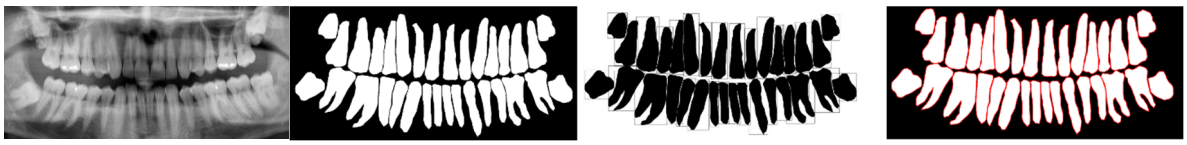
$$PR = \frac{A}{A+B} \in [0,1] \qquad R = \frac{A}{A+C} \in [0,1] \qquad F\text{-}value = \frac{2 \times PR \times R}{PR+R} \in [0,1] \tag{10}$$

High *precision* denotes that the annotation process achieved high accuracy in identifying correct annotations, whereas high *recall* means that very few correct annotations where missed by the system. In addition to evaluating *precision* and *recall* separately, it is a common practice to consider *F-value* as a combined measure, representing the harmonic mean of *precision* and *recall*. High *precision* and *recall*, and thus high *F-value* indicate in our case high annotation quality.

In addition, we evaluate time performance: i) measuring the time to automatically produce semantic annotations, ii) evaluating time w.r.t. the size of the reference ontology and how the latter evolves with the number of annotated images, and ii) comparing automatic (system) annotation time with manual (user) annotation time.

6.4 Experimental Data

We have created an SVG dataset consisting of 22,553 labelled objects from 750 images based on panoramic dental x-ray images. To our knowledge, it is the first significant dataset of labelled SVG objects and images, which we make available online as a benchmark for future research in this area. We first acquired dental x-rays from the UFBA_UESC dental images dataset [96], considering 750 images including structural variations regarding the number of teeth, restorations, implants, appliances, and the size of the mouth and jaws. We applied proper filters to clean the images, and performed edge detection using Python's OpenCV library. We generated contour points to approximate the shape of each tooth and created an SVG file for each object. Figure 20 depicts the process of cleaning, detecting, and generating the contours of each tooth in the image. Consequently, we organized the SVG images under 4 labelled clusters: *full teeth* (including exactly 32 teeth), images *missing a few teeth* (between 26 and 31), images *missing several teeth* (less than 26), and images of *supernumerary cases* containing extra teeth (more than 32). We also considered 10 object clusters: *incisor*, *canine*, *molar*, *premolar*, and *wisdom tooth* each being *upper jaw, lower jaw, healthy*, *synthetic*, and *decayed*. These seed images and objects form the first visual concepts in the reference KG, which will be populated gradually as new images and new objects are annotated by the system.



a. Input raster image **b.** Image filtering and cleaning **c.** Edge detection **d.** Contour generation

Figure 20. Sample images depicting our SVG image and object data preparation pipeline (the resulting SSG image graph following the annotation process is shown in Figure 21)

6.5 Evaluating Annotation Quality

6.5.1 Evaluating System Annotations

We conducted a battery of tests to evaluate the quality of our annotation process. The expert user first provides seed images and seed objects to form the representatives of the target clusters. In our experiments, we considered the seed image and object clusters produced in our experimental dataset (cf. Section 4.4). Consequently, the system starts offering annotation suggestions for each new image in the input stream, along with its constituent objects. The produced system annotations are compared against the user-labelled dataset (cf. Section 6.4), and are considered either

relevant (true positives) or irrelevant (false positives), allowing to compute PR, R, and F-value scores accordingly. Annotations are gradually added to the reference KG. The process is repeated using five different similarity thresholds $Thresh_{Sim}$: 0.5, 0.6, 0.7, and 0.8 ($\in$ [0, 1], cf. Figure 16), resulting in a total of 750×3 + 22,553×3 = 69,909 annotation tasks. Similarity weight parameters are considered with different values, after testing and choosing weight combinations that maximize result quality (i.e., w_{shape}=0.4, w_{area}=0.2, $w_{position}$=0.3, w_{color}= 0.1, and equal weights for the remaining parameters: $w_{major} = w_{minor} = w_{ecc} = 0.3334$, $w_{lengh} = w_{slope} = 0.5$, etc.). These weights can be tweaked by the user according to the application scenario. Note that fine-tuning the similarity weight parameters is an optimization problem that can be solved using a number of techniques that apply linear programming or machine learning to identify the best weights for a given problem class, e.g., [78-80]. Providing such a capability, in addition to manual tuning, would enable users to optimize the process according to their needs. We do not further address the fine-tuning of weights here since it is out of the scope of this paper, and will be addressed in a dedicated study. Average results are presented in Figure 22.

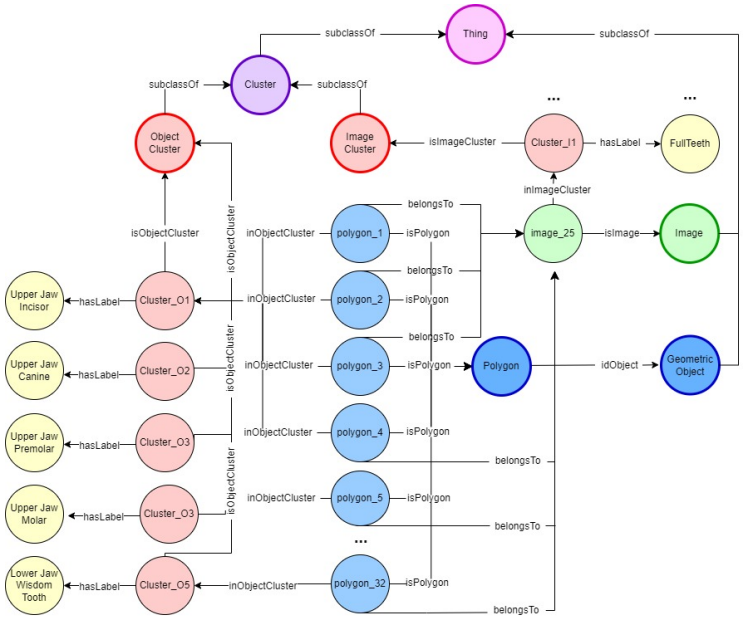
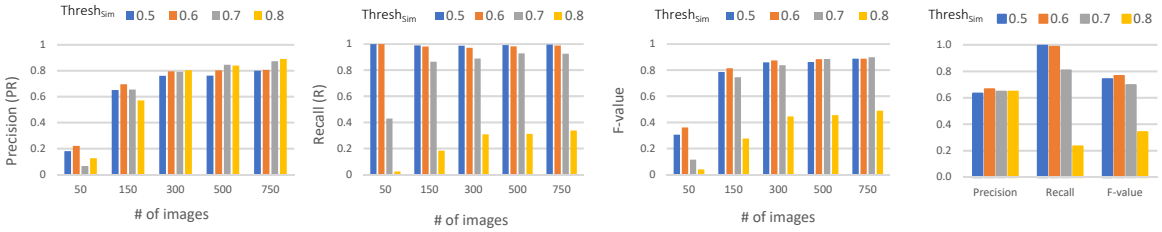


Figure 21. SSG image graph representation from the example in Figure 18.



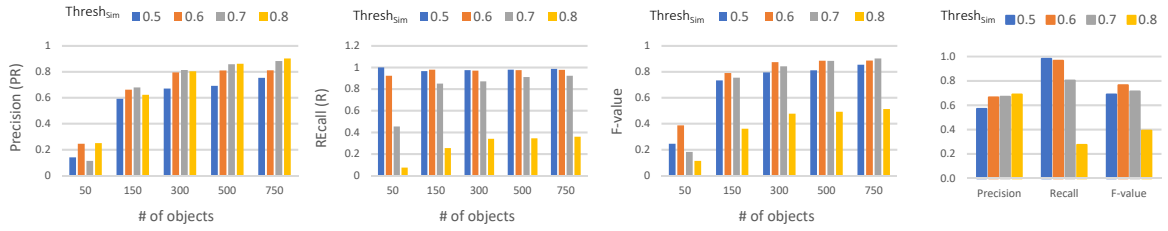
a. Precision **b.** Recall **c.** F-value **d.** Average results

Figure 22. Image annotation results

Precision *(PR)*: For each of the image annotation results, the worst (minimum) precision value (=0) is reached when the system returns zero relevant annotation offers to the user and the best (maximum) precision value (=1) is reached when 750 relevant annotation offers are returned to the user corresponding to 750 images. Results in Figure 22 show that annotation precision for the first 50 images is very low for all similarity thresholds (varying between PR=0.065-and-0.22 for $Thresh_{Sim}$=0.7 and 0.6 respectively). This is because the reference KG initially includes the seed images only, and is gradually populated with the incoming stream of 50 images, which might not be sufficient to fully form the target clusters. It increases throughout the experiment as the number of annotated images increases in the KG, reaching maximum levels when all 750 images have been annotated and added to the KG (varying between PR=0.8-and-0.89 for $Thresh_{Sim}$=0.5 and 0.8 respectively). As more images are fed into the system, it is learning and dynamically changing the representatives to choose the ones that resemble the clusters the most. The maximum precision value PR=0.89 is obtained for highest $Thresh_{Sim}$=0.8. This is because increasing the similarity threshold reduces the number of irrelevant annotation offers: a higher threshold means higher similarity between image and KG representatives, producing higher annotation accuracy).

Recall (*R*): Similarly, *recall* varies from 0 (minimum) to 1 (maximum) when the system returns 0 or 750 relevant annotation offers respectively (corresponding to each of the 750 images). Results show that as more images are annotated and fed into the reference KG, *recall* levels increase almost regularly. Moreover, the recall levels tend to decrease as the threshold increases. This is because increasing the similarity threshold reduces the number of annotation offers returned to the user: filtering-out certain potentially relevant annotations which might be less similar to the image being annotated. In other words, increasing the similarity threshold increases the risk of disregarding relevant results, reflected in our case by decreasing *recall* levels.

F-value: In cases where higher/lower *precision*/*recall* levels are obtained simultaneously, the *f-value* measure allows evaluating the overall loss and gain in average *precision*/*recall*, in order to evaluate result quality. Results in Figure 22.c show that average *f-value* levels increase from 0.74-to-0.764 with $Thresh_{Sim}$=0.5 and 0.6, and then decrease from 0.696-to-341 with $Thresh_{Sim}$=0.7 and 0.8. This reflects the increase and decrease in both *precision* and *recall* levels, and illustrates what we have seen before: when the threshold increases, *precision* increases yet *recall* decreases, hence inducing an increasing-and-decreasing slope with *f-value* levels.



a. Precision **b.** Recall **c.** F-value **d.** Average results

Figure 23. Object annotation results

Similar observations can be made for the object annotations results in Figure 23. We ran the tests on 5 random subsets from our experimental dataset, each consisting of 750 SVG objects. *Precision* increases with the increase in number of annotated objects and the increase in similarity threshold, reaching a maximum PR=0.902 with # of annotations = 750 and $Thresh_{Sim}$=0.8. Precision improves as more objects are annotated and integrated in the reference KG. Also, increasing the similarity threshold reduces the number of irrelevant annotation offers, thus increasing precision accordingly. Recall levels remain almost stable with the varying number of object annotations, yet decrease with the increase in similarity threshold, reaching a minimum R=0.113 with $Thresh_{Sim}$=0.8 and # of annotations = 50. This is because increasing the similarity threshold reduces the number of annotation suggestions: filtering-out certain potentially relevant annotations which might be less similar to the object being annotated. *F-value* levels increase and

decrease with the number of annotations and the similarity thresholds. This is expected and reflects the increase and decrease in both *precision* and *recall* levels mentioned above: improving as the KG appends more annotated objects and increases in expressiveness, while decreasing with the increase in threshold filtering-out certain potentially relevant annotations.

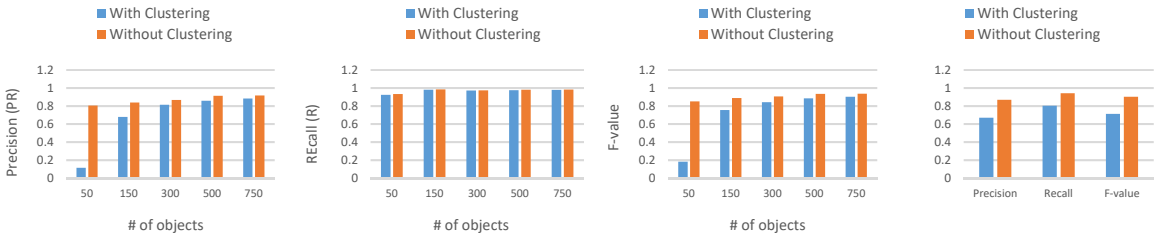
6.5.2 Ablation Study on the Clustering Process

We have conducted an ablation study to evaluate the impact of the clustering component on the SSG annotation pipeline. We evaluate the impact of clustering on both object annotation and image annotation levels. We ran the tests on our SVG image dataset, using the same 5 subsets of objects from our previous experiment each consisting of 750 SVG objects. Figures 24 and 25 report the average precision, recall, and f-value results obtained with similarity threshold $Thresh_{Sim} = 0.7$ (similar results were obtained with $Thresh_{Sim} = 0.5$, 0.6, and 0.8). Precision results show that annotation quality improves with the number of processed images and objects, where average precision levels are slightly higher without clustering compared with clustering. This is due to the fact that clustering is a data-centric process that requires a significant amount of data to execute effectively. In other words, the larger the amount of similar data grouped in every cluster (i.e., the larger the sizes of the image and object clusters), the higher the chances of having good representatives for the clusters taking into account cluster formation and diversity, and thus the better the quality of the upcoming clustering iterations allowing to incorporate new images and new objects more accurately in their respective clusters. Recall levels are consistent across both experiments, and produce almost equal results with and without clustering.

The impact of clustering is largely perceived when evaluating system performance, where it drastically reduces annotation time compared with the one-on-one – without clustering, similarity computation process (cf. Section 6.6.2).

a. Precision **b.** Recall **c.** F-value **d.** Average results

Figure 24. Image annotation results, obtained with and without clustering

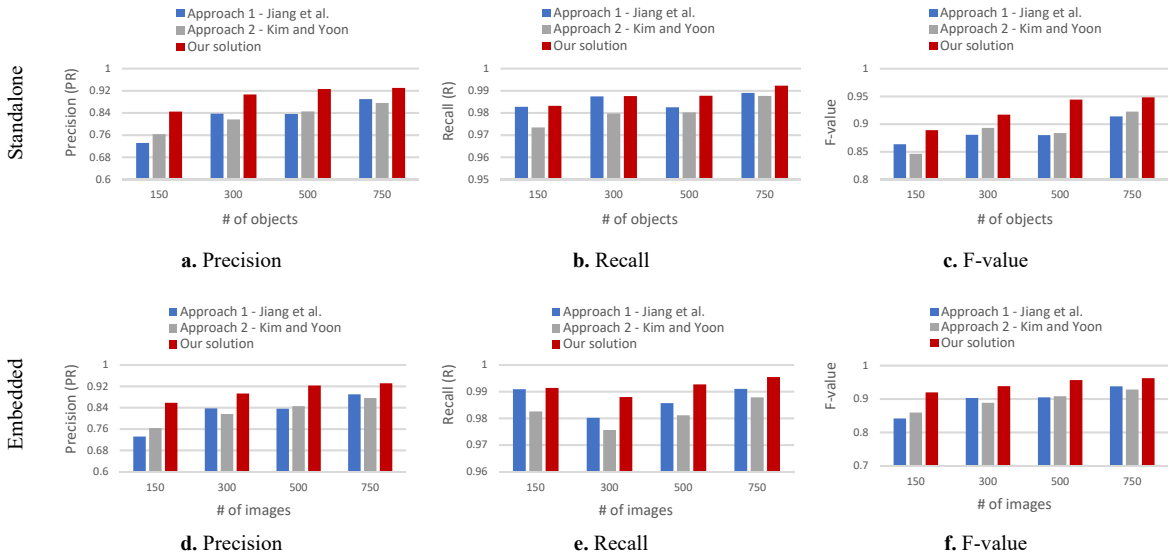


a. Precision **b.** Recall **c.** F-value **d.** Average results

Figure 25. Object annotation results, obtained with and without clustering

6.5.3 Comparative Quality Evaluation

We have also compared our solution with two existing SVG similarity evaluation measures. Since existing solutions only perform SVG object similarity computation and do not provide a pipeline for semantic image annotation, we perform two separate evaluations. First, we evaluate the similarity measures independently, as standalone solutions, where each input object is compared with the existing objects in the KG and is associated with the one having the highest similarity score, inheriting its label accordingly. Second, we evaluate the similarity measures as embedded solutions, after integrating them within our cluster-based annotation framework (i.e., we replace our similarity measure with each of the existing solutions and run the annotation pipeline accordingly). We ran the experiments on 5 random subsets from our experimental dataset, each consisting of 750 SVG objects. The same subsets were used for both standalone and embedded evaluations. Average precision, recall, and f-value results are shown in Figure 26. Results for both standalone and embedded experiments show that our integrated object similarity evaluation measure produces improved object annotation results compared with its predecessors. This is mostly due to the following:



a. Precision **b.** Recall **c.** F-value

d. Precision **e.** Recall **f.** F-value

Figure 26. Comparative quality results, considering standalone (a ,b, c) and embedded (d, e, f) evaluations

i) Existing solutions in [11, 60] combine shape, color, and position similarities and disregard area similarity which we consider in our integrated measure. In practice, SVG objects might have similar shapes (e.g., similar rectangles or similar hexagon shapes), while having different area coverages (e.g., pre-molar and molars might have similar shapes, yet they will be distinguished by their coverage areas), which highlights the need to consider a dedicated area coverage similarity measure.

ii) Existing solutions use distance measures to compare the angle and length sequences describing object shapes (i.e., [11] uses Euclidian distance and [60] uses edit distance), while we utilize the cosine similarity measure. Different from distance measures which are sensitive to the sequences' vector modules, cosine is a correlation measure which only considers vector angles and is completely insensitive to their modules. It compares vectors accordingly to their angle variations, making it more suitable with high-dimensional data where module variations are usually dismissed as noise. Variations in vector modules might distinguish between highly similar SVG objects (similarly to comparing high-dimensional text vectors in information retrieval, e.g., [94,

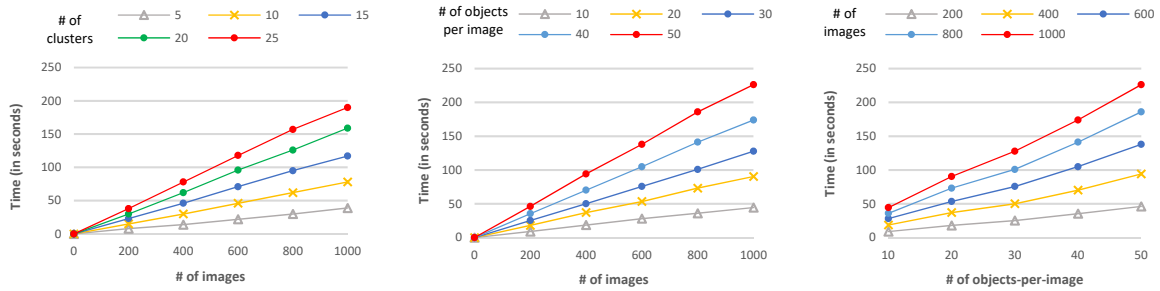
97])[13]. Other correlation measures like Pearson Correlation Coefficient can also be used in this context, yet we adopt cosine similarity due to its common usage in information retrieval literature, e.g., [94, 97].

We further highlight the performance impact of our embedded annotation pipeline, compared with standalone similarity-based annotation, in the following subsection.

6.6 Performance Evaluation

6.6.1 Evaluating System Performance

In addition to testing the annotation quality of our approach, we evaluate its time performance. The complexity of our method comes down to $O(N \times k \times |Img|^2)$ where N is the number of SVG images being processed, k the number of clusters (i.e., the number of image and object categories), and $|Img|$ the number of objects per image. It simplifies to to $O(N \times k)$ since usually $N >> |Img|$. Timing experiments were carried out on a PC with an Intel(R) Core(TM) i7-7500U 2.7 GHz processor with 16GB RAM. Figure 27 shows that the time needed to produce automatic annotations for geometric objects in an image grows in a linear fashion with the number of images (Figure 27.a), the number of clusters (Figure 27.b) used, and the number of objects per image (Figure 27.c). In addition, Figure 29 shows that the size of the KG increases in an almost perfect linear fashion with the number of images and objects being annotated and appended to the KG. The effect of increasing the number of objects per image (i.e., increasing image size) is also apparent in Figure 29, since every new object is annotated and then integrated as a visual concept instance in the reference KG, which increases the KG size accordingly.



a. Varying the # of images and clusters, while fixing the # of objects-per-image to 10

b. Varying the # of images and objects-per-image, while fixing the # of clusters to 5

c. Varying the # of clusters and images, while fixing the # of clusters to 5

Figure 27. Time performance results

6.6.2 Performance of Clustering Process

We have also evaluated the impact of the clustering component on the performance of the SSG annotation pipeline. Results in Figure 28 show the difference in time performance of SSG with clustering (i.e., our suggested process)

[13] The cosine measure only detects variations in vector angles, which highlight differences between the feature vectors' directions. This is commonly adopted in image annotation and retrieval where images of the same label might have significant feature vector module variations while sharing similar vector directions. This is especially useful with high-dimensional data where vector similarity is commonly evaluated as the similarity between the vectors' directions, versus vector module similarity which is considered to be too specific and oftentimes misleading especially with higher dimensionality, e.g. [94, 97].

versus SSG without clustering (i.e., replacing the clustering process with one-on-one image and object comparisons). Results clearly show the significant impact of clustering on reducing computation time by almost 25 fold. This highlights the reduction in processing time when images and objects are compared with the cluster representatives, versus comparing all images (and objects) against each other one-on-one to perform the annotation process. Hence, while SSG with clustering and SSG without clustering produced comparable quality results in Section 6.5.2, yet SSG with clustering allows to drastically reduce computation time as shown below, and is thus integrated in SSG's main process.

a. Varying the # of images while fixing the # of objects-per-image to 10, the number of images-per-cluster to 10, and the number of clusters to 5

b. Varying the # of images while fixing the # of objects-per-image to 10, the number of images-per-cluster to 10, and the # of clusters to 10

c. Varying the # of objects-per-image while fixing the # of images to 400, the number of images-per-cluster to 10, and the # of clusters to 10

Figure 28. Time performance results, evaluated with and without clustering.

6.6.3 Comparative Performance Evaluation

We also compare our solutions' performance with existing similarity-based solutions and with manual annotation time. Figure 30 compares average annotation time for a stream of random 200 images from our experimental dataset, each image containing 16 objects (consisting of the upper jaw teeth in panoramic dental X-ray images). Three graduate students participated in the manual annotation exercises, and were requested to manually annotate the teeth in every panoramic image. They were provided with "pause" and "resume" buttons in the annotation GUI[14] allowing them to pause and resume the annotation task as needed, according to their fatigue levels. Manual time was paused and recorded accordingly to account exactly for the amount of time spent on the annotation task. Average annotation times for each tooth are compiled and shown in Figure 30. Results underline the following observations:

- The impact of automatic annotation versus manual annotation is evident in the significant difference in time scale between both tasks: the average manual annotation time for an individual tooth (object) is 1.125 seconds, compared with 0.243, 0.236, and 0.008 seconds for automatic annotation following Approach 1, Approach 2, and our SSG solution respectively. The manual annotation task required around 1 hour to complete all 200 images, compared with 13, 12.6, and 0.45 seconds with automated Approach 1, Approach 2, and our solution respectively.
- Comparing the automated solutions together, results show that our solution does not seem to seriously reduce annotation time in the early stages of the experiment compared with existing methods, i.e., when annotating the first 50 images. That is because the reference KG only contains a reduced number of geometric object

[14] Graphical user interface

descriptions in the beginning, where the number of images is almost equivalent to the number of clusters. This defeats the purpose of performing clustering in the first place, since comparing with the cluster representatives comes down to comparing with almost every image in the KG. Nonetheless, the efficiency of performing clustering becomes apparent with the increase in the number of images, namely going beyond 50 images where clustering reduces annotation time by an increasing polynomial factor reaching almost 2,800% reduction compared with similarity-based annotation time (i.e., our solution required 27 seconds to annotate 200 images, versus 780 seconds with Approach 1 and 756 seconds with Approach 2).

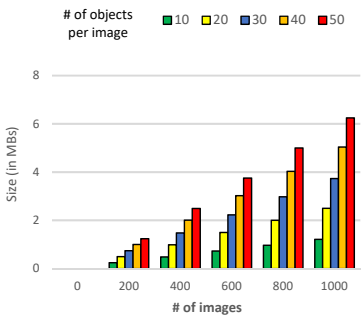


Figure 29. KG size variations w.r.t. the number of annotated images and image size (in # of objects)

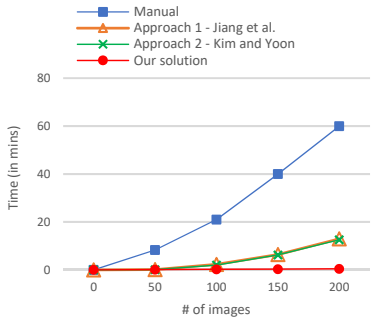


Figure 30. Comparing our solutions' performance with manual annotation and existing similarity-based approaches

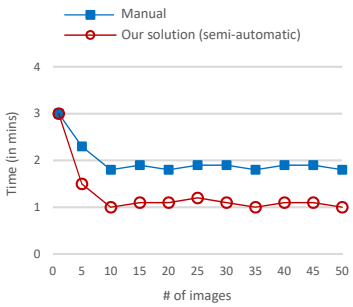


Figure 31. Comparing manual and semi-automated annotation time

In addition, we evaluate the efficiency of our solution in minimizing the amount of manual work needed to perform the annotation task. To do so, we compare our tool's semi-automatic processing with manual user annotation time. Semi-automatic time comprises of two subsequent intervals: i) the average time required by the system to automatically generate annotation offers, and ii) the average time required by the user to verify and (maybe) correct the resulting system annotations. Figure 31 compares average user time and semi-automatic annotation time for a stream of 50 random images from our dataset, each image containing 16 objects (consisting of the upper jaw teeth in panoramic dental X-ray images, totaling 50×16 = 800 manual annotation tasks). The same manual annotation process was adopted from the previous experiment. Average annotation times for each tooth are compiled and shown in Figure 31. Results show that the semi-automatic annotation process does not seem to seriously reduce time in the early stages of the experiment, i.e., when annotating the first couple of images. That is because the reference KG lacks geometric object descriptions in the beginning, and hence the system is not capable of providing useful automatic annotation offers at first. Nonetheless, as the KG becomes semantically richer by appending newly annotated images, results (starting from image #4 in Figure 31, i.e., starting from object #64) show that semi-automatic annotation reduces annotation time by a factor of 0.42 on average. In other words, the asymptotic values of the automatic and manual curves in Figure 31 seem to stabilize at around 1.1 (± ε) and 1.9 (± ε) respectively, highlighting an approximate 42% time reduction from manual to semi-automatic annotation.

7. Conclusion

In this paper, we introduce an original framework titled *SSG* which automatically converts a stream of SVG images and objects into a semantic graph representation. We introduce an incremental clustering approach to semantically annotate SVG images and their constituent objects in a fast and efficient manner, using an aggregation of shape, area, color, and location similarity measures. We then produce an RDF graph representation of the input image and integrate it in a reference knowledge graph, incrementally extending its semantic expressiveness to improve future

annotation tasks. Our solution achieves semantization of vector image contents with minimum human effort and training data, while complying with native Web standards (i.e., SVG and RDF) to preserve transparency in representing and searching images using Semantic Web stack technologies. Our solution is of linear complexity in the sizes of the image and knowledge graph used. We have created a large SVG dataset consisting of 22,553 labelled objects from 750 images based on panoramic dental x-ray images. To our knowledge, this is the first significant dataset of labelled SVG objects which we make available for future research in this area. Experimental results highlight our approach's effectiveness, and its applicability in a practical application domain.

Note that while designed for vector images, yet *SSG* can also be extended to raster images. This requires raster image contours to be extracted using traditional image segmentation and contour detection techniques [98], which are then used to generate vector graphics.

As continuing work, we are currently investigating the extension of our approach to integrate in our KG representation different kinds of objects including social media data [99] and video metadata [100]. This requires considering time and space dimensions [101] and extending our SSG representation model accordingly. We are also investigating auto-calibration and optimization techniques, e.g., [102, 103], to study the effect of different similarity measures (shape, area, location, and color) on annotation quality, aiming to suggest weighting schemes that could help users tune their input parameters to obtain optimal results. In the near future, we plan to investigate methods to mine the collective knowledge of an image collection compiled within its KG, using (semi-)automated RDF inference [54, 104] and fuzzy processing techniques [53, 105], which can help improve image accessibility, management, and exchange between automated Web agents and services.

Declarations

An early version of the SVG object similarity module and a preliminary annotation module is described in [17]. It consists of: i) an aggregate similarity measure for comparing SVG objects (the current paper redefines and extends the aggregate SVG object similarity measure to perform polygon similarity computation for non-identical object types, and adds a new SVG image similarity measure, cf. Section 4.2), ii) a basic similarity-based annotation process to recommend the most similar SVG object labels (the current paper introduces an unsupervised cluster-based framework, consisting of a two-layered clustering process to perform both SVG object and SVG image annotation, cf. Section 4.3), iii) a limited experimental evaluation described in one page (the present study performs an in-depth experimental evaluation, and introduces a new labelled SVG dataset consisting of 22,553 objects from 750 images – the first significant dataset of labelled SVG objects and images which we make available online, cf. Section 6). The present study also introduces a new knowledge graph representation model, allowing to convert SVG objects and images into semantic representations and provide annotation recommendations accordingly (cf. Section 4.1), as well as a dedicated motivation scenario (cf. Section 2) and an in-depth investigation of related solutions (cf. Section 3).

References

[1] Hong R., et al., *Multimedia encyclopedia construction by mining web knowledge.* Signal Processing 2013. 93(8):2361-2368.

[2] Wagenpfeil S., et al., *Fast and Effective Retrieval for Large Multimedia Collections.* Big Data and Cognitive Computing, 2021. 5(3): 33.

[3] Tekli J., *An Overview of Cluster-based Image Search Result Organization: Background, Techniques, and Ongoing Challenges.* Knowl. Inf. Syst., 2022. 64(3): 589-642.

[4] Jagtap J. and Bhosle N., *A Comprehensive Survey on the Reduction of the Semantic Gap in Content-based Image Retrieval.* International Journal of Applied Pattern Recognition, 2021. 6(3): 254-271.

[5] Dubey S., *A Decade Survey of Content based Image Retrieval Using Deep Learning.* IEEE Trans. Circuits Syst. Video Technol., 2022. 32(5): 2687-2704.

[6] Li X., et al., *Socializing the Semantic Gap: A Comparative Survey on Image Tag Assignment, Refinement, and Retrieval.* ACM Computing Surveys 2016. 49(1): 14:1-14:39.

[7] Papapanagiotou V., et al., *Improving Concept-Based Image Retrieval with Training Weights Computed from Tags.* ACM Transactions on Multimedia Computing, Communications, and Applications, 2016. 12(2): 32:1-32:22.

[8] Ruocco M. and Ramampiaro H., *Event-related Image Retrieval: Exploring Geographical and Temporal Distribution of User Tags* International Journal of Multimedia Information Retrieval, 2013. 2(4): 273-288.

[9] Ma L., et al., *Learning Efficient Binary Codes From High-Level Feature Representations for Multilabel Image Retrieval.* IEEE Transactions on Multimedia, 2017. 19(11): 2545-2560.

[10] Madduma B., R.S., *Image Retrieval based on High Level Concept Detection and Semantic Labelling* Intelligent Decision Technologies, 2012. 6(3): 187-196.

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

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

[13] Li D., et al., *Shape similarity computation for SVG.* Int. J. Computational Science and Engineering, 2011. Vol. 6, 1/2.

[14] 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. (6)95-116.

[15] Tekli J., et al., *Evaluating Touch-Screen Vibration Modality toward Simple Graphics Accessibility for Blind Users.* International Journal of Human Computer Studies (IJHCS), 2018. 110: 115-133.

[16] Engel C., et al., *SVGPlott: an Accessible Tool to Generate Highly Adaptable, Accessible Audio-Tactile Charts for and from Blind and Visually Impaired People.* PETRA 2019: , 2019. pp. 186-195.

[17] Salameh K., et al., *SVG-to-RDF Image Semantization.* 7th International SISAP Conference, 2014. pp. 214-228.

[18] Gaudenz H., et al., *VIAN: a Visual Annotation Tool For Film Analysis.* Computer Graphics Forum, 2019. 38(3): 119-129.

[19] World Wide Web Consortium. *Scalable Vector Graphics (SVG)*. http://www.w3.org/Graphics/SVG/ [5 Jan 2023].

[20] Spindler M., et al., *Translating Floor Plans into Directions.* Proceedings of the 13th international conference on Computers Helping People with Special Needs, 2012. Linz, Austria.

[21] Unar S., et al., *Detected Text-based Image Retrieval Approach for Textual Images.* IET Image Process, 2019. 13(3): 515-521.

[22] Parcalabescu L. and Frank A., *Exploring Phrase Grounding without Training: Contextualisation and Extension to Text-Based Image Retrieval.* CVPR Workshops, 2020. pp. 4137-4146.

[23] Cao M., et al., *Image-text Retrieval: A Survey on Recent Research and Development.* International Joint Conference on Artificial Intelligence (IJCAI), 2022. pp. 5410-5417.

[24] Khalid Y. and Noah S., *Semantic Text-based Image Retrieval with Multi-modality Ontology and DBpedia.* The Electronic Library, 2017. 35(6): 1191-1214.

[25] Moreno J., *Text-Based Ephemeral Clustering for Web Image Retrieval on Mobile Devices.* SIGIR Forum 2015. 49(1): 67.

[26] Ashraf R., et al., *MDCBIR-MF: Multimedia Data for Content-based Image Retrieval by using Multiple Features.* Multim. Tools Appl., 2020. 79(13-14): 8553-8579.

[27] Ahmad J., et al., *Describing Colors, Textures and Shapes for Content Based Image Retrieval - A Survey.* CoRR abs/1502.07041, 2015.

[28] Feng H. et al., *A Bootstrapping Framework for Annotating and Retrieving WWW Images.* Proceedings of the International ACM Multimedia Conference, 2004. pp. 960-967.

[29] Dong J., et al., *Cross-Media Similarity Evaluation for Web Image Retrieval in the Wild.* IEEE Transactions on Multimedia, 2018. 20(9): 2371-2384.

[30] Zhang B., et al., *An Automatic Image-Text Alignment Method for Large-scale Web Image Retrieval.* Multimedia Tools and Applications 2017. 76(20): 21401-21421 (2017).

[31] Ma K., et al., *Deep Blur Mapping: Exploiting High-Level Semantics by Deep Neural Networks.* IEEE Transactions on Image Processing, 2018. 27(10): 5155-5166.

[32] Kompatsiaris I., et al., *Multimedia Content Indexing and Retrieval using an Object Ontology.* Multimedia Content and the Semantic Web, 2005. pp. 339-371.

[33] Dhanalakshmi K. and Rajamani V., *An intelligent mining system for diagnosing medical images using combined texture-histogram features.* International Journal of Imaging Systems and Technology, 2013. 23(2):194–203.

[34] Chen H., et al., *The Classification and Retrieval of the Image Affective Semantics Based on Integration of Multi Features and SVM.* Journal of Information Hiding and Multimedia Signal Processing 2018. 9(4): 864-873.

[35] Pandey S., et al., *A Semantics and Image Retrieval System for Hierarchical Image Databases.* Information Processing and Management, 2016. 52(4): 571-591.

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

[37] Iskandar D., et al., *Social media retrieval using image features and structured text.* INEX: Initiative for the Evaluation of XML Retrieval, 2007. pp. 358–372.

[38] 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.

[39] Vanoirbeek C., et al., *A Lightweight Framework for Authoring XML Multimedia Content on the Web.* Multimedia Tools and Applications 2014. 70(2): 1229-1250.

[40] Pusnik M., et al., *XML Schema Quality Index in the Multimedia Content Publishing Domain.* Software Quality Analysis, Monitoring, Improvement, and Applications (SQAMIA'16) 2016. pp. 57-64.

[41] Kong Z. and Lalmas M., *XML Multimedia Retrieval.* In: Consens, M.P., Navarro, G.(eds.) SPIRE 2005. LNCS, vol. 3772, pp. 218–223. Springer, Heidelberg (2005), 2005.

[42] Kong Z. and Lalmas M., *Using XML logical structure to retrieve (Multimedia) objects.* In: Kov´acs, L., Fuhr, N., Meghini, C. (eds.) ECDL'07, LNCS, 2007. Vol. 4675, pp. 100–111. Springer, Heidelberg.

[43] Iakovidou C., et al., *Searching images with MPEG-7 (& MPEG-7-like) Powered Localized dEscriptors: The SIMPLE answer to effective Content Based Image Retrieval* 12th International Workshop on Content-Based Multimedia Indexing (CBMI) 2014. pp. 18-20.

[44] Molina R., et al., *Heterogeneous SoC-based Acceleration of MPEG-7 Compliance Image Retrieval Process.* Journal of Real-Time Image Processing, 2018. 15(1): 161-172 (2018).

[45] Phadikar B., et al., *Content-based Image Retrieval in DCT Compressed Domain with MPEG-7 Edge Descriptor and Genetic Algorithm.* Pattern Analysis and Applications, 2018. 21(2): 469-489.

[46] International Organization for Standardisation (ISO), *MPEG-7 Overview.* ISO/IEC JTC1/SC29/WG11, Coding for Moving Pictures and Audio, 2004. Martinez J.M., N6828.

[47] Wang S., et al., *Efficient image retrieval using MPEG-7 descriptors.* Proc. of the International Conference on Image Processing (ICIP) 2003, 509-512

[48] Kushki A. et al., *Retrieval of image from artistic repositories using a decision fusion framework.* IEEE Trans. Image Process., 2004. 13(3).277–289.

[49] Laaksonen J., et al., *PicSOM—Self-Organizing Image Retrieval With MPEG-7 Content Descriptors.* IEEE Trans. on Neural Networks, 2002. 13(4):841-853.

[50] Dasiopoulou S., et al., *Capturing MPEG-7 Semantics.* International Conference on Metadata and Semantics Research (MTSR'07), 2007. pp. 113-122.

[51] García R., et al., *Multimedia Content Description Using Semantic Web Languages.* Semantic Multimedia and Ontologies, 2008. pp. 17-54

[52] Doncel V., et al., *Overview of the MPEG-21 Media Contract Ontology.* Semantic Web 2016. 7(3): 311-332.

[53] Fan T., et al., *Storing and Querying Fuzzy RDF(S) in HBase Databases.* International Journal of Intelligent Systems, 2020. 35(4): 751-780 (2020).

[54] Straccia U. and Casini G., *A Minimal Deductive System for RDFS with Negative Statements.* International Conference on Principles of Knowledge Representation and Reasoning (KR'22), 2022. pp. 351–361.

[55] Schröder M., et al., *Bridging the Technology Gap between Industry and Semantic Web: Generating Databases and Server Code from RDF.* Inter. Conf. on Agents and Artificial Intelligence (ICAART'21), 2021. pp. 507-514.

[56] Angles R., et al., *Mapping RDF Databases to Property Graph Databases.* IEEE Access 2020. 8: 86091-86110.

[57] Schwab M., et al., *Scalable Scalable Vector Graphics: Automatic Translation of Interactive SVGs to a Multithread VDOM for Fast Rendering.* IEEE Transactions on Visualization and Computer Graphics, 2022. 28(9): 3219-3234

[58] Jiang X., et al., *Recognizing Vector Graphics without Rasterization.* Conference on Neural Information Processing Systems (NeurIPS'21), 2021. 24569-24580.

[59] Bai S., et al., *Revised Aggregation-tree Used in Metadata Extraction from SVG Images.* International Conference on Data Mining (DMIN'06), 2006. pp. 325-328.

[60] Kim B. and Yoon J., *Similarity Measurement for Aggregation of Spatial Objects.* ACM Symposium on Applied Computing (SAC'05), 2005. pp. 1213-1217.

[61] Abe k., et al., *Similarity Retrieval of Trademark Images by Vector Graphics Based on Shape Characteristics of Components.* International Conference on Computer and Automation Engineering (ICCAE'18), 2018. pp. 82-86.

[62] Di Sciascio E., et al., *A Logic for SVG Documents Query and Retrieval.* Multim. Tools Appl., 2004. 24(2): 125-153.

[63] 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.

[64] Lloyd S., *Least Squares quantization in PCM.* IEEE Transactions on Information Theory, 1982. 28(2):129-137.

[65] Villena-Román J., et al., *MIRACLE-GSI at ImageCLEFphoto 2009: Comparing Clustering vs. Classification for Result Reranking.* CLEF (Working Notes), 5 p., 2009.

[66] Park H, et al., *A K-means-like Algorithm for K-medoids Clustering and Its Performance.* Proceedings of the 36th CIE Conference on Computers & In-dustrial Engineering, 2006. pp.1222-1231.

[67] Bradley P., et al., *Clustering via Concave Minimization.* Advances in Neural Information Processing Systems, vol. 9, M. C. Mozer, M. I. Jordan, and T. Petsche, Eds. Cambridge, Massachusetts: MIT Press, 1997. pp. 368–374.

[68] Pelleg D. and Moore A., *X-means: Extending k-means with Effcient Estimation of the Number of Clusters.* In International Conference on Machine Learning (ICML), 2000. pp. 727-734.

[69] Wang H., et al., *Context-Based Clustering of Image Search Results.* Deutsche Jahrestagung für Künstliche Intelligenz (KI), 2009. pp. 153-160.

[70] Zhao K., et al., *Clustering Image Search Results by Entity Disambiguation.* European Conference on Machine Learning (ECML/14), 2014. (3): 369-384.

[71] Hirota M. *et al.*, *Constraint-based Clustering of Image Search Results using Photo Metadata and Low-level Image Features.* Proceedings of the 9th IEEE/ACIS International Conference on Computer and Information Science (ICIS'10), 2010, 165-178.

[72] Alamdar F. and Keyvanpour M., *Effective Browsing of Image Search Results via Diversified Visual Summarization by Clustering and Refining Clusters.* Signal, Image and Video Processing, 2014. 8(4): 699-721.

[73] Chen W., et al., *Feature Estimations Based Correlation Distillation for Incremental Image Retrieval.* IEEE Transactions on Multimedia, 2022. 24: 1844-1856.

[74] Rodríguez J., et al., *PAXQuery: Efficient Parallel Processing of Complex XQuery.* IEEE Transactions on Knowledge and Data Engineering 2015. 27(7): 1977-1991.

[75] Chen R., et al., *Parallel XPath Query based on Cost Optimization.* Journal of Supercomputing, 2022. 78(4): 5420-5449.

[76] Hayes P., *RDF Semantics.* W3C Recommendation, http://www.w3.org/TR/rdf-mt/. 2004.

[77] 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.

[78] Hopfield J. J., *The Effectiveness of Neural Computing.* IFIP World Computer Congress (WCC'89), 1989. 402-409.

[79] Azar D., et al., *A Combined Ant Colony Optimization and Simulated Annealing Algorithm to Assess Stability and Fault-Proneness of Classes Based on Internal Software Quality Attributes.* International Journal of Artificial Intelligence (ISSN 0974-0635), 2016. 14:2.

[80] Azar D. and Vybihal J., *An Ant Colony Optimization Algorithm to Improve Software Quality Predictive Models.* In Journal of Information and Software Technology, 2011. 53(4): 388-393.

[81] Braden B., *The Surveyor's Area Formula.* The College Mathematics Journal, 1986. 17(4):326–337.

[82] Manjunath B.S., *Color and Texture Descriptors.* IEEE Transactions on Circuits and Systems for Video Technology (CSVT), 2001. 6:703-715.

[83] Mezaris V.; Kompatsiaris I. and Strintzis M.G., *An Ontology Approach to Object-based Image Retrieval*. Proc. of the International Conference on Image Processing (ICIP). Vol. 2, pp. 511-514,

[84] Stanchev P. et al.., *High Level Color Similarity Retrieval.* International Journal on Information Theory and Applications, 2003. 10(3):363-369.

[85] Iranzo P. and Sáenz-Pérez F., *Implementing WordNet Measures of Lexical Semantic Similarity in a Fuzzy Logic Programming System.* Theory and Practice of Logic Programming, 2021. 21(2): 264-282.

[86] Li J., *Lightweight ontologies mapping and the Semantic Similarity based on WordNet.* Advances in Computer Science and its Applications, 2012. 1(2):111-117.

[87] Francis W. N. and Kucera H., *Frequency Analysis of English Usage.* Houghton Mifflin, Boston, 1982.

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

[89] Lin D., *An Information-Theoretic Definition of Similarity.* Proceedings of the International Conference on Machine Learning (ICML), 1998. pp. 296-304.

[90] Salazar R., *Operations Research with R - Transportation Problem.* Towards Data Science, 2019. https://towardsdatascience.com/operations-research-in-r-transportation-problem-1df59961b2ad.

[91] Salloum G. and Tekli T., *Automated and Personalized Meal Plan Generation and Relevance Scoring using a Multi-Factor Adaptation of the Transportation Problem.* Soft Computing, 26(5): 2561-2585 (2022)

[92] Ahmad A. and Khan S., *Survey of State-of-the-Art Mixed Data Clustering Algorithms.* IEEE Access 2019. 7: 31883-31902.

[93] Stanford Center for Biomedical Informatics Research. *Protégé Ontology Editor*. [5 Jan. 2023].

[94] Salton G. and Mcgill M.J., *Introduction to Modern Information Retrieval.* 1983. McGraw-Hill, Tokio.

[95] Zou F., et al., *Semi-supervised Cross-modal Learning for Cross Modal Retrieval and Image Annotation.* World Wide Web Journal, 2019. 22(2): 825-841.

[96] Silva G., et al., *Automatic Segmenting Teeth in X-ray Images: Trends, a Novel Dataset, Benchmarking and Future Perspectives* Expert Systems with Applications 2018. 107-15–31.

[97] Baeza-Yates R. and Ribeiro-Neto B., *Modern Information Retrieval: The Concepts and Technology behind Search.* ACM Press Books, Addison-Wesley Professional, 2nd Ed., 2011. p. 944.

[98] Tariq N., et al., *Quality Assessment Methods to Evaluate the Performance of Edge Detection Algorithms for Digital Image: A Systematic Literature Review.* IEEE Access, 2021. 9: 87763-87776.

[99] Abebe M., et al., *Generic Metadata Representation Framework for Social-based Event Detection, Description, and Linkage.* Knowledge Based Systems 2020. 188.

[100] Schiappa M. and Rawat Y., *SVGraph: Learning Semantic Graphs from Instructional Videos.* Computing Research Repository (CoRR), 2022. CoRR abs/2207.08001.

[101] Bai L., et al., *Querying Fuzzy Spatiotemporal RDF Data Using R2RML Mappings.* IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'20), 2020. pp. 1-8.

[102] Gal A., et al., *From Diversity-based Prediction to Better Ontology & Schema Matching.* Inter. WWW Conference, 2016. pp. 1145-1155.

[103] Ming M., et al., *A Harmony Based Adaptive Ontology Mapping Approach.* In Proceedings of the International Conference on Semantic Web and Web Services (SWWS'08), 2008. pp. 336-342.

[104] Lhez J., et al., *PatBinQL: a Compact, Inference-enabled Query Language for RDF Stream Processing.* IEEE BigData, 2018. pp. 4036-4044.

[105] Li G., et al., *Pattern Match Query over Fuzzy RDF Graph.* Knowledge Based Systems 2019. 165: 460-473.