Received 7 November 2022, accepted 5 December 2022, date of publication 15 December 2022, date of current version 22 December 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3229654

RESEARCH ARTICLE

Semantic Image Collection Summarization With Frequent Subgraph Mining

ANDREA PASINI, FLAVIO GIOBERGIA, (Graduate Student Member, IEEE), ELIANA PASTOR, AND ELENA BARALIS, (Member, IEEE)
Department of Control and Computer Engineering, Politecnico di Torino, 10129 Turin, Italy

Corresponding author: Flavio Giobergia (flavio.giobergia@polito.it)

This work was supported in part by the SmartData@PoliTo Center for Big Data and Machine Learning technologies.

ABSTRACT Applications such as providing a preview of personal albums (e.g., Google Photos) or suggesting thematic collections based on user interests (e.g., Pinterest) require a semantically-enriched image representation, which should be more informative with respect to simple low-level visual features and image tags. To this aim, we propose an image collection summarization technique based on frequent subgraph mining. We represent images with a novel type of scene graphs including fine-grained relationship types between objects. These scene graphs are automatically derived by our method. The resulting summary consists of a set of frequent subgraphs describing the underlying patterns of the image dataset. Our results are interpretable and provide more powerful semantic information with respect to previous techniques, in which the summary is a subset of the collection in terms of images or image patches. The experimental evaluation shows that the proposed technique yields non-redundant summaries, with a high diversity of the discovered patterns.

INDEX TERMS Frequent subgraph mining, image collection summarization, panoptic segmentation, scene graphs.

I. INTRODUCTION

Image collection summarization is fundamental to derive highlights from big image datasets without requiring manual effort. The obtained summaries can be used to provide previews of personal albums [1], [2] (e.g., Google Photos) or to identify and suggest thematic collections based on user interests (e.g., Pinterest, Flickr) [3]. Video-sharing platforms, such as YouTube, can benefit from these semantic digests to summarize the frames of a video and automatically derive tags and categories [4]. Additionally, in the deep learning field, summarization may help in the assessment of the coverage and diversity of the dataset used to train neural networks [5], [6].

Previous methods typically generate summaries based on low level visual features or image tags [7], [8]. The output summary is a subset of the collection, in terms of images or image patches. These approaches suffer from two different problems: (i) visual features and image tags do not effectively represent the semantic content, e.g., object classes and their relationships, and (ii) using a subset of the dataset as summary does not allow evaluating the semantic representativity of the result.

We propose *SImS* (*S*emantic *Im*age *S*ummarization), a method to extract abstract summarization patterns from large image collections. Differently from previous works, our technique describes the collection with a set of semantic patterns representing object classes and their relationships. More specifically, these output patterns are provided in the form of scene graphs, which report the object configurations found in the input data. The results of our method can be exploited to answer queries such as ‘‘Find the Pinterest boards that show a person on a bike’’ or, more complex, ‘‘two people on the same bike’’. Furthermore, the frequent scene graphs may be used to derive complex textual descriptions that also consider object relationships, such as ‘‘In the collection you can typically find scenes with a house surrounded by vegetation’’.

The associate editor coordinating the review of this manuscript and approving it for publication was Yiming Tang.

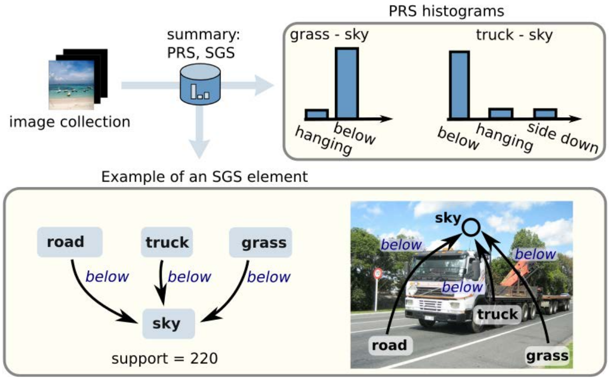


FIGURE 1. **Example of summary patterns extracted by SImS from the COCO dataset.**

SImS exploits frequent subgraph mining techniques to extract semantic summarization patterns from scene graphs. Scene graphs are computer vision models designed to represent the relationships among objects in the same image. We extend the semantic information of standard scene graphs by introducing fine-grained relationship types to describe the relative position between objects. Moreover, we design an algorithm to automatically derive the scene graphs from a dataset of images labeled with panoptic segmentation (i.e., the task of identifying the shape of the objects in an image and assigning a class label to each of them) [9]. Segmentation labels can be automatically obtained with state-of-the-art models [10], [11].

We define two types of patterns: i) *frequent pairwise object relationships* and ii) *frequent scene graphs*. Some examples of summary patterns extracted by our method are provided in Figure 1. Frequent pairwise object relationships describe common relative position configurations taken by pairs of objects belonging to specific classes. These patterns, stored in the form of histograms, represent the *Pairwise Relationship Summary* (PRS). Consider for example the two PRS histograms shown in Figure 1. They show that, in the considered image collection, the objects "grass" and "truck" are frequently found below "sky".

Frequent scene graphs characterize multiple-object configurations that describe the different scene types occurring in the input images. These patterns form the *Scene Graph Summary* (SGS). Differently from previous image summarization methods, the scene graphs together with their support value, indicating the frequency with which a pattern occurs in the collection, are an interpretable way of representing the summarized information. Figure 1 shows an example of SGS graph describing an outdoor road scene that occurs 220 times in the collection, in which "road", "truck", and "glass" are *below* "sky". Unfortunately, the extraction of the SGS by means of frequent subgraph mining algorithms has a very high computational complexity and generates redundant information that is undesired in a summary. By performing an effective preprocessing step, SImS overcomes both issues.

Patterns in the PRS and SGS provide a synthetic description of the visual content in the collection. They can be extracted from big real world image collections and exploited as a knowledge base that highlights common relationships between objects. Hence, they provide actionable knowledge for image understanding tasks [12], [13].

The contribution of our paper can be outlined as follows.

- *Semantics-aware summarization patterns.* We define and extract two types of semantics-aware summarization patterns (i.e., frequent pairwise object relationships and frequent scene graphs), which provide an explainable summary, highlighting the semantics of object relationships.
- *Novel scene graph relationships.* We define a novel set of fine-grained relationships correlating objects with their relative position in the image.
- *Automated scene graph construction.* We design an algorithm to automatically build scene graphs by analyzing images labeled with panoptic segmentation.
- *Effective and efficient scene graph preprocessing.* We design a scene-graph preprocessing technique to remove redundant information from the scene graph collection and significantly reduce the running time of the frequent graph mining process.

The paper is organized as follows. Section II presents the related works. Section III describes the main challenges of graph mining applied to our task, while Section IV presents the SImS summarization approach. Section V introduces our evaluation methodology and Section VI discusses the experimental results. Section VII draws conclusions and outlines future developments of our work.

II. RELATED WORK

This section firstly reviews literature works related to image collection summarization, highlighting differences with our method. Afterwards, it briefly describes common applications of scene graphs to computer vision and knowledge extraction techniques from image collections. Finally, we review state-of-the-art graph mining algorithms, which are used as building blocks by the proposed technique.

A. IMAGE COLLECTION SUMMARIZATION

The techniques to summarize image collections typically select a subset of images deemed to be the most significant samples [14], [15], [16]. These works describe images by means of raw level visual features, such as color histograms or Scale Invariant Feature Transform (SIFT). The main objective is to achieve a good summary *coverage*, which means all concepts in the collection should be represented, and *diversity*, which means that the result should not contain redundancies.

The most common methods [8], [17], [18] rely on the application of a clustering algorithm on visual features and the selection of the most relevant samples from each cluster. Specifically, in [17] the authors use Self Organizing

Maps (SOM) to cluster images, in [18] they use KMedoids, while [8] exploit Affinity Propagation clustering. A different approach, taken by [19], builds a dictionary with the minimum set of summary images that allow reconstructing the entire collection by means of a linear combination.

Summarization methods that only rely on visual features cannot fully capture the semantic meaning of the images. Reasoning on the high level concepts helps in the generation of more significant summaries. Reference [20] use LSA on image tags (retrieved from the Flickr dataset) to create groups of pictures. Each group is then summarized separately by clustering its members based on visual features. Alternatively, word tags can be used together with visual features in a multimodal analysis that builds a latent space whose elements reconstruct the entire dataset [7]. The authors in [21] assign class labels to each image with a generic classifier, then map these labels to domain concepts using the WordNet and DBpedia ontologies. The class vectors of each sample are exploited to build a similarity graph and the summary images are picked using the graph centrality metric.

All the previous summarization methods define summaries as a subset of the initial images. These methods are not able to explicitly provide the salient semantic characteristics of the proposed images. Differently from related works, our technique summarizes the high-level characteristics of the image collection in the form of frequent scene graphs. The resulting summary is both straightforwardly interpretable and suitable for automated processing, as it is encoded in a graph format.

B. SCENE GRAPHS

One of the main applications of scene graphs is image retrieval [22]. Reference [23] build scene graphs by inspecting object relationships in 3D meshes. Given a query image, graph kernels are exploited to retrieve similar pictures. Image retrieval was also performed by generating scene graphs from text queries [24], [25]. Despite their interesting applications, image retrieval tasks address a different problem. They cannot be directly used to derive summaries, but only to collect images based on a specific query. Similarly, image captioning based on scene graphs [26], [27] cannot be exploited to derive summaries, as these methods derive a caption for each single image.

Another frequent application of scene graphs is image generation. For example the authors in [28], [29], [30], and [31] use graph convolutional neural networks to generate realistic images from scene graphs. In these works scene graphs can be either manually generated (e.g., for the Visual Genome Dataset) [32] or inferred (e.g., for the COCO dataset) [9] by looking at the relative position (e.g., ''left of'', ''above'', ''inside'') between the object bounding boxes.

When generating scene graphs, in our work, we capture more complex position relationships (e.g., ''on'' vs ''above'') based on the segmentation masks of the objects, instead of bounding boxes or object centroids [29], [33]. The relative position between objects is computed by a classifier trained on a dataset that we labeled specifically for this task and is publicly available for reproducibility purposes.[1] Our approach also allows our model to be faster (during training and inference) and more interpretable with respect to other full-deep-learning techniques for generating scene graphs [34], [35].

C. KNOWLEDGE EXTRACTION

Sadeghi et al. [13] inspect object positions in image collections to extract visual knowledge and verify relation phrases. The idea of inspecting pairwise object relationships is conceptually similar to our PRS extraction phase, with the difference that the problem is modeled as the estimation of the most probable explanation (MPE) of the multinomial distribution that governs object relationships. Also in our previous work [36] we inspected pairwise relationships with the goal of detecting anomalies in semantic segmentation results. In this paper we generalize these conceptual representations, by moving from pairwise relationships to semantically richer scene graphs.

D. FREQUENT SUBGRAPH MINING

The aim of frequent subgraph mining (FSM) is the identification of all subgraphs, from a set of graphs or a single large graph, with frequency at least equal to a specified threshold denoted *minsup* [37]. The extraction of frequent patterns from a graph collection finds a variety of applications in chemical datasets [38], web map services [39], biological data [40], [41]. FSM algorithms may be classified by considering multiple properties as the input type (e.g., single/multiple graphs, directed/undirected, static/dynamic), the expected result (e.g., all frequent subgraphs or just a subset), and the algorithmic approach (e.g., Apriori or pattern growth-based) [37], [42], [43]. Considering the variety of applications, multiple algorithms have been proposed in the literature to identify frequent subgraphs [37]. Recently, given the increasing availability of data, several approaches have been proposed to handle big data [42]. These approaches are designed to deal with novel requirements such as the dynamicity and volume of data, by proposing dynamic, parallel, and distributed algorithms. Examples of approaches belonging to this category are the Ap-FSM [44] and MIRAGE algorithms [45], Spark-based approaches as [46] and [47], and gSpan-H [48], a parallel implementation of the gSpan algorithm.

In the proposed approach, we exploit FSM techniques to discover frequent graph patterns in the collection of scene graphs. As a result, any FSM technique which derives frequent subgraphs for a set of scene graphs could be accommodated and used. We exploit two well-known algorithms to mine frequent subgraphs in labeled graphs, gSpan [49] and SUBDUE [50]. gSpan [49] is an exact FSM algorithm based on a Depth First Search (DFS) strategy, which needs less memory and is thus suitable for large datasets.

[1] https://github.com/AndreaPasini/SImS

SUBDUE [50] is an inexact FSM technique based on graph compression. We considered these two algorithms because of their widespread adoption and their publicly available and maintained implementation. The experimental results, discussed in Section VI-D, show that the two approaches allow achieving good performance in a suitable time when coupled with our scene-graph preprocessing technique.

III. FREQUENT SCENE GRAPH MINING

The extraction of frequent scene graph patterns is a fundamental and challenging step in the generation of the Scene Graph Summary (SGS). Unfortunately, the direct application of FSM algorithms to mine patterns in scene graphs suffers from three different issues: (i) the *long running time*, (ii) the presence of *high-entropy relationships*, and (iii) the presence of *repeated items*.

A. RUNNING TIME

SUBDUE and gSpan may take many hours to produce a result (see Section VI-D), when applied directly to scene graphs. In the case of gSpan, the number of output frequent subgraphs is regulated by the *minsup* parameter, which specifies the minimum percentage of transactions that an output subgraph must belong to. Since gSpan is a DFS-based algorithm, with lower values of *minsup* the running time increases significantly. Setting higher values causes gSpan to discard low frequency patterns from the summary. Unfortunately, low frequency, but interesting, patterns typically occur in image datasets with high scene diversity, such as Microsoft COCO [9].

B. HIGH-ENTROPY RELATIONSHIPS

The direct application of FSM algorithms on scene graphs typically generates many redundant subgraphs, which do not provide useful information for the final summary.

This effect is due to the presence of high-entropy object relationships, that occur when two object classes do not have a predefined position (i.e., they appear with many different relative positions in different images). The absence of a predefined position between a pair of classes generates many similar frequent graphs, where the same objects are characterized by slightly different relationships. Consider for example the classes "handbag" and "towel". Figure 2 depicts a histogram with the frequencies of their position relationships occurring in the COCO dataset. The distribution of the relationship labels is almost flat and, since there is no preferred relative position, the generated frequent graphs include all these high entropy relationships. Indeed, Figure 3a and 3b show two examples of frequent graphs that only differ by the edge between "handbag" and "towel". These redundant frequent graphs do not provide useful information and generate an overcrowded SGS.

C. REPEATED ITEMS

Some particular object classes can be likely found multiple times inside an image. For example, consider the class "person" in crowded scenes or the class "car" in a road scene. Frequent subgraphs generated from these scenes typically contain the same element redundantly repeated multiple times. Furthermore, the resulting SGS may include frequent graphs in which the repeated item appears a different number of times. An example is represented in Figures 3c and 3d, in which "car" objects appear a different number of times in different frequent graphs.

TABLE 1. Relative position labels for a subject-reference pair.

Label	Description
above	s is above r without contact
below	s is below r without contact
on	s is on top of r with contact
hanging	s is below r with contact
side	s and r are not vertically aligned
side-up	s and r are not vertically aligned, s is in a higher position
side-down	s and r are not vertically aligned, s is in a lower position
inside	s pixels are inside r shape
around	s pixels are around r shape

IV. SEMANTIC IMAGE SUMMARIZATION

SImS exploits frequent subgraph mining techniques to extract semantic summarization patterns from a set of segmented images. The generated patterns include the Pairwise Relationship Summary (PRS) and the Scene Graph Summary (SGS). The general structure of the summary is covered in Section IV-A. Figure 4 depicts SImS architecture, whose building blocks are briefly outlined in the following.

Scene graph computation: This step transforms the images labeled with the panoptic segmentation format into scene graphs that describe the position relationships between objects (see Section IV-B).

Pairwise Relationship Summary generation: In this step, the input scene graphs are analyzed and common pairwise relationships between objects are extracted (see Section IV-C).

Scene graph preprocessing: This step prepares the graphs for the frequent subgraph mining process. It takes as input the PRS and the scene graphs (see Section IV-D).

Scene graph mining: An FSM algorithm extracts patterns from the preprocessed scene graphs, thus generating the scene graph patterns stored in the SGS (see Section IV-E).

A. SUMMARIZATION PATTERNS

SImS summaries are based on two types of patterns: (i) *pairwise object relationships* and (ii) *scene graphs*.

The scene graph data structure is used to describe both the input image collection and the results in the Scene Graph Summary. Some examples of scene graphs are provided in Figure 3.

A scene graph is defined as a directed graph $G = (V, E, l_v, l_e)$, where V is the set of vertices, E are the edges

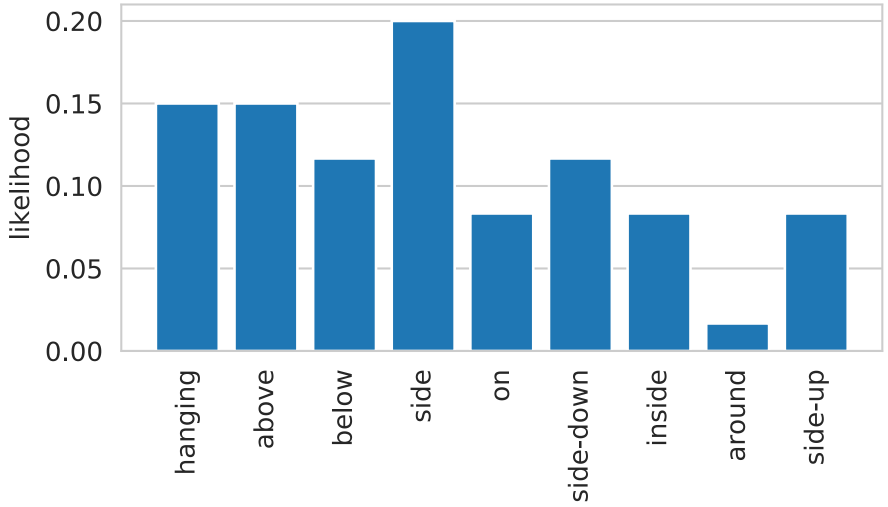


FIGURE 2. High-entropy relationships for the class pair handbag-towel.

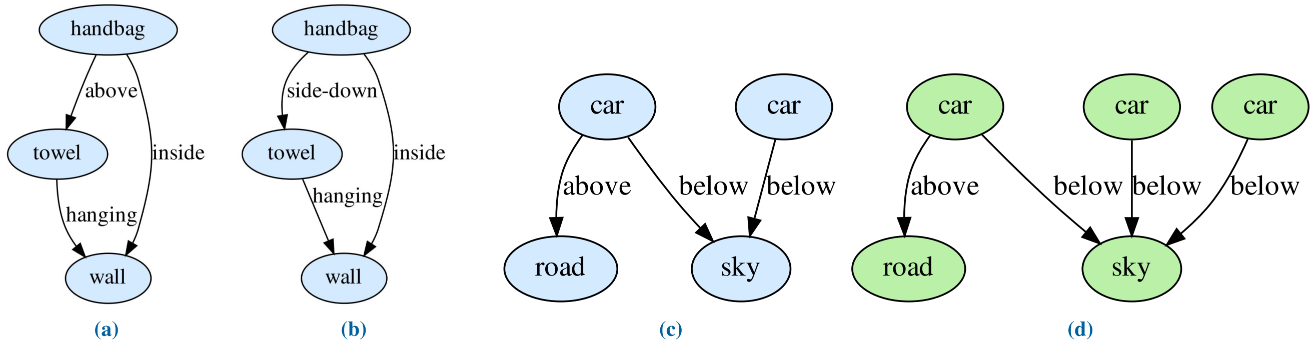


FIGURE 3. High-entropy relationships (a, b); repeated items (c, d).

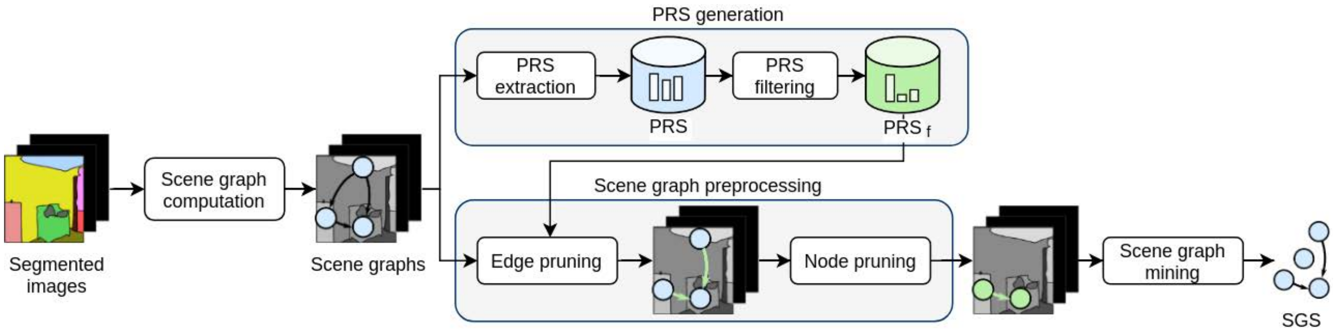


FIGURE 4. SImS architecture.

and l_v, l_e are functions assigning a label to each vertex and edge, respectively. Every vertex $v \in V$ represents an object in the image and $l_v(v)$ defines its object class (e.g., "car", "sky").

The edges in E take the form $e = (s, r) \in V \times V$ where s and r represent the *subject* and the *reference* of a relationship. The label $l_e(e)$ associated with an edge e corresponds to the relative position between the subject and the reference. We defined a set of 9 fine-grained position labels, thus extending the base ones [29], [33]. The position labels exploited by SImS are reported in Table 1. The relationships on/above (and below/hanging) are designed to distinguish the presence of a contact between the subject and the reference, enhancing in this way the semantic understanding of the visual scene. A further discussion on the considered relationships is provided in the following sections.

The Scene Graph Summary includes multi-object frequent patterns in the form of scene graphs. It is defined by the set:

$$SGS = \{G_1, \ldots, G_i, \ldots, G_{|SGS|}\}$$

where G_i is a scene graph extracted from the input collection with a Frequent Subgraph Mining algorithm and $|SGS|$ is the number of summary graphs.

The Pairwise Relationship Summary is composed of a set of units called *histograms*, which describe common pairwise object relationships that occur in the image collection.

Let $s_l \in L$ (subject label), $r_l \in L$ (reference label) be two object classes, where L is the set of all object class labels. The likelihoods of their different relative position labels are modeled as a discrete probability distribution. We describe the distribution with a *histogram*, defined as follows:

$$h(s_l, r_l) = \{P(p_1|s_l, r_l), ..., P(p_i|s_l, r_l), ..., P(p_n|s_l, r_l)\}$$

where every value $P(p_i|s_l, r_l)$ specifies the likelihood that a subject with class s_l and a reference with class r_l are related with a relative position p_i. For example, the histogram:

$$h(kite, sky) = \{P(below|kite, sky) = 0.12$$
$$P(inside|kite, sky) = 0.81, ...\}$$

specifies that a common pattern in the input dataset is represented by images where "kite" is *below* or *inside* "sky", with likelihoods 0.12 and 0.81 respectively.

We define the Pairwise Relationship Summary (PRS) as the set of histograms extracted from the image collection:

$$PRS = \{h(s_l, r_l) \mid (s_l, r_l) \in L \times L\}$$

where $L \times L$ is the set of all possible class pairs.

B. SCENE GRAPH COMPUTATION

A set of images labeled with the panoptic segmentation format is transformed into a collection of scene graphs that describe the position relationships between objects.

Panoptic segmentation locates and classifies every object instance in each image. More specifically, this labeling task assigns to each pixel of the analyzed image (i) a class label and (ii) an object identifier, merging together semantic segmentation and instance segmentation [9]. We denote as $(\mathcal{L}, \mathcal{Z})$ the result of panoptic segmentation for a given image, where $\mathcal{L}$ is the matrix that associates class labels to the pixels and $\mathcal{Z}$ is the matrix that specifies the object identifier for each pixel.

Our algorithm converts each segmentation result $(\mathcal{L}, \mathcal{Z})$ to a scene graph $G = (V, E, l_v, l_e)$. Every object identifier in $\mathcal{Z}$ is represented with a node $v \in V$ and its label $l_v(v)$ is extracted from the matrix $\mathcal{L}$.

Conversely, the object relationships cannot be directly retrieved from panoptic segmentation and are computed separately. Therefore, we infer the object relative positions based on a set of features we specifically designed for this task. This procedure is different from previous techniques that are only based on object bounding boxes or centroids [29], [33], which is limiting since these elements cannot describe the arrangement of the object shapes [36]. The obtained features are processed by a random forest classifier that predicts the relative position $l_e(e)$ for every subject-reference pair.

The classifier is trained on a dataset that we manually labeled due to the unavailability of existing online resources. Details on the dataset and the choice of the classifier are provided in Section VI-B. In the following paragraphs we describe the feature extraction step.

1) RELATIVE POSITION FEATURES

The feature extraction algorithm takes as input the matrix $\mathcal{Z}$ (containing object identifiers) and generates two categories of features: (i) *string-based* and (ii) *bounding-box-based*. *String-based* features ($f_1 - f_7$) analyze the matrix $\mathcal{Z}$ column by column to inspect how the subject s and the reference r are positioned along the vertical axis. This approach is inspired by [51] where the authors used the concept of *strings* for querying image databases. Instead, *bounding-box-based* features ($f_8 - f_{11}$) are designed to inspect the object positions when they are not vertically aligned.

String-based features (see Table 2) are computed as follows. For each horizontal position of the matrix $\mathcal{Z}$, the algorithm extracts the corresponding (vertical) column of pixels as a vector. To reduce the computational complexity of the following steps, consecutive pixels with the same object identifier are merged. We call *string* the resulting compressed column, denoted as:

$$\mathcal{Z}_x = \{z_1, \quad \cdots, \quad z_i, \quad \cdots, \quad z_n\}, x \in [1, width(\mathcal{Z})]$$

where x is the horizontal position of the column in $\mathcal{Z}$, z_i are the object identifiers in the column, and $width(\mathcal{Z})$ is the number of columns in the matrix $\mathcal{Z}$. The elements z_i in $\mathcal{Z}_x$ with lower i index are located near the top of the image, while those with higher index are located towards the bottom part.

To compute the features describing the position between the objects s and r, our algorithm considers all the strings $\mathcal{Z}_x$ in $\mathcal{Z}$ that contain both object identifiers. Table 2 describes a set of rules designed to detect the possible configurations of s and r in each $\mathcal{Z}_x$. For example, rule r_1 (s *on* r) applies when r is immediately after s in a string, which means that the subject is located in the upper part of the image with respect to the reference. Three examples of strings are shown in Figure 5, where a bird (subject) is compared with a house wall (reference). The picture exemplifies how an object pair may have strings that apply to different rules. Hence, a classifier is exploited to assign the final relative position label.

The features are computed by counting the number of strings that satisfy each rule. More formally:

$$f_i = \frac{count(r_i, \mathcal{Z}, s, r)}{min(w_s, w_r)}, \quad i \in [1, 7]$$

TABLE 2. Rules to extract relative position features from strings.

Feature	Rule	Label	Applies if	Example strings ($\mathcal{Z}_x$)
f_1	r_1	*s on r*	r is directly after s	$\{..., s, r, ...\}$
f_2	r_2	*s hanging r*	s is directly after r	$\{..., r, s, ...\}$
f_3	r_3	*s above r*	r is after s with interleaving region	$\{s, ..., r\}$
f_4	r_4	*s below r*	s is after r with interleaving region	$\{r, ..., s\}$
f_5	r_5	*s around r*	r is between s	$\{s, r, s\}$ or $\{s, ..., r, ..., s\}$
f_6	r_6	*s inside r*	s is between r	$\{r, s, r\}$ or $\{r, ..., s, ..., r\}$
f_7	r_7	*other*	none of the other statements applies	$\{r, s, r, s\}$

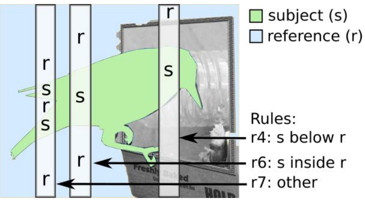


FIGURE 5. Example of string-based feature extraction.

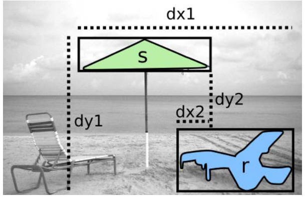


FIGURE 6. Example of bounding-box-based feature extraction.

where f_i is a feature, $count(r_i, \mathcal{Z}, s, r)$ is the number of strings in $\mathcal{Z}$ that contain s, r and satisfy rule r_i, w_s and w_r are the width of the subject and reference respectively. The width is computed as the horizontal pixel span that covers an object. The division by the minimum width normalizes the count relatively to the smallest object.

This normalization is inspired by human behavior when comparing objects. Specifically, it is more common for us to relate the position of the smallest object with respect to the bigger one. For example, we prefer "glass on table" rather than "table below glass". Hence, we intuitively design our features as the percentage of columns (i.e., strings) of the smallest object that satisfy a specific rule. Other normalization options are not suitable for our case. More specifically, dividing by the total number of strings that contain both s and r does not consider the strings with only either of the two objects, while dividing by $max(w_s, w_r)$ yields very low values for the features when one of the two objects is much larger than the other.

The previously defined features do not provide information when s and r are not vertically aligned (i.e., do not appear together in the same strings). Hence, they are not suitable for detecting the *side, side-up, side-down* labels. To address this issue, we defined four additional bounding-box-based features.

They are computed from the subject and reference bounding boxes, based on the following measures:

$$dx_1 = right(r) - left(s), \quad dx_2 = left(r) - right(s)$$
$$dy_1 = bottom(r) - top(s), \quad dy_2 = top(r) - bottom(s)$$

where the functions $top()$, $bottom()$ extract the vertical margins of the bounding boxes and $left()$, $right()$ extract the horizontal ones. Figure 6 depicts these values on an example image.

The final bounding-box-based features are generated by normalizing the previous quantities in the following way:

$$f_8 = dx_1/max(|dx_1|, |dx_2|), \quad f_9 = dx_2/max(|dx_1|, |dx_2|)$$
$$f_{10} = dy_1/max(|dy_1|, |dy_2|), \quad f_{11} = dy_2/max(|dy_1|, |dy_2|)$$

These normalized values can be interpreted as distances in percentage with respect to the horizontal span (i.e., $max(|dx_1|, |dx_2|)$ and vertical span (i.e., $max(|dy_1|, |dy_2|)$) covered by the two objects. In this way, the final features allow representing the distances between objects relatively to their size.

The set of features described in this section are finally fed to a random forest classifier (see Section VI-B for a discussion on the classifier selection) to assign the edge label $l_e(e)$ between each pair of objects (s, r) in the scene graphs. The complete set of edge label values is reported in Table 1.

C. PAIRWISE RELATIONSHIP SUMMARY GENERATION

Given a set of scene graphs, common pairwise relationships between object pairs with different classes are extracted to form the PRS. First, the PRS histograms defined in Section IV-A are generated from the input scene graphs (PRS extraction in Figure 4). Specifically, the histogram values are computed by counting the percentage of times that a subject with class s_l and a reference with class r_l satisfy each position relationship. The process has linear complexity with respect to the number of edges in the scene graphs being processed.

Next, the extracted PRS is filtered (PRS filtering in Figure 4) to keep only the most significant histograms and reduce the amount of information. To this aim, each histogram can be associated with two different measures to assess its importance in the PRS. The first is its *support*, which is the number of example pairs in the training graphs that are used to build the histogram. Higher support entails a more reliable histogram. The second measure is the *entropy* of the discrete probability distribution, which is lower when the distribution is unbalanced towards few values (i.e., positions in our case). Histograms with lower entropy are more significant because they identify class pairs that are characterized by very specific relative positions (e.g., "sky" is always above "river").

The output of the filtering step, PRS_f, is the set of histograms that satisfy constraints on a minimum support value ($minsup_h$) and a maximum entropy ($maxentr_h$). In Section VI-C we discuss the effect of varying the two thresholds on the resulting summary.

D. SCENE GRAPH PREPROCESSING

As discussed in Section III, avoiding high-entropy relationships and redundant node items is important to obtain a succinct and effective summarized representation of an image collection. To address these issues, we propose *edge pruning* and *node pruning* as preprocessing steps preceding the extraction of frequent scene graphs. Since the proposed techniques simplify the scene graphs in input to the mining process, preprocessing will also drastically reduce the running time, as we will see in Section VI-D.

1) EDGE PRUNING

To remove high-entropy relationships, we exploit the filtered pairwise knowledge base (PRS_f, see Section IV-C). Specifically, two scene graph objects with classes s_l, r_l are connected by a high-entropy relationship if $h(s_l, r_l) \notin PRS_f$. Every high-entropy relationship is removed from the input scene graphs. This pruning step generates scene graphs with a lower number of edges.

2) NODE PRUNING

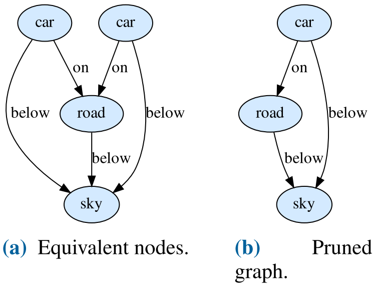
Repeated node items are detected by analyzing each single node inside a scene graph. This operation aims at detecting and pruning equivalent nodes that: i) have the same object class, and ii) are related to other objects in the same way. For example, if a scene contains many cars that are positioned in the same way with respect to "sky" and "road", then these objects could be merged into a single graph node, because they represent redundant information. We envision as future work the definition of aggregate nodes, representing groups of similar nodes. In the following we provide the definition of equivalent nodes.

We base node equivalence on two functions that describe the outbound and inbound edges of a specific vertex.

Definition 1 (Outbound Edge Description): Let $v \in V$ be a vertex in a scene graph $G = (V, E, l_v, l_e)$. Outbound edges are described with the following set:

$$out(v) = \{(l_e(e), l_v(v_k)) \mid e = (v, v_k) \in E\}$$

where e are the outbound edges of v with outbound nodes v_k.



(a) Equivalent nodes. (b) Pruned graph.

FIGURE 7. Node pruning.

The function $out(v)$ generates a set of tuples $(l_e(e), l_v(v_k))$ that specify the labels of the outbound edges and the outbound nodes.

Inbound edges can be described symmetrically with respect to outbound ones.

Definition 2 (Inbound Edge Description): Let $v \in V$ be a vertex in a scene graph $G = (V, E, l_v, l_e)$. Inbound edges are described with the following set:

$$in(v) = \{(l_e(e), l_v(v_k)) \mid e = (v_k, v) \in E\}$$

where e are the inbound edges of v with inbound nodes v_k.

Equivalent nodes share the same object class and the same labels for the inbound and outbound edges.

Definition 3 (Node Equivalence): Let $v_i, v_j \in V$ be two nodes. They are equivalent if:

$$l_v(v_i) = l_v(v_j) \wedge in(v_i) = in(v_j) \wedge out(v_i) = out(v_j)$$

Hence, this notion of equivalence allows identifying the nodes that carry equivalent semantic information inside the same scene graph.

Figure 7a depicts a scene graph where two equivalent nodes with the class "car" are related in the same way with the outbound nodes "road" and "sky". Figure 7b shows the resulting graph after the node pruning step, with only one of the two "car" nodes.

The previously defined node equivalence requires the two nodes to share exactly the same edge labels. This condition is hardly verified in very dense graphs with high entropy relationships. Hence, node pruning benefits from the application of edge pruning, that simplifies relationships and generates lower entropy graphs. For this reason, node pruning is applied after edge pruning during scene graph preprocessing.

E. SCENE GRAPH MINING

This step generates the final frequent graphs of the SGS. It entails the application of a frequent subgraph mining algorithm on a transactional dataset of labeled graphs. gSpan and SUBDUE are selected as the two most suitable state-of-the-art algorithms (see Section II). In Section VI-D we analyze the difference between the results obtained with the two graph mining algorithms. The output frequent graphs are then collected in the SGS.

To complete the information of the SGS from a visual point of view, we also provide an example image for each frequent subgraph. Specifically, we associate to each SGS graph G_i the image in the input collection whose scene graph is a super-graph of G_i and the number of nodes is the smallest (i.e., it only contains the necessary objects to represent the SGS graph).

Algorithm 1 SImS Algorithm

```
Require: imgs, minsup_h, maxent_h, minsup
Ensure: $PRS_f$, $SGS$
                                          ▷ Scene graph computation.
$S_{SG} = \{\}$
for $\mathcal{I} \in$ imgs do
    $\mathcal{L}, \mathcal{Z}$ = panoptic_segmentation($\mathcal{I}$)
    $f_1..f_7$ = get_string_features($\mathcal{Z}$)
    $f_9..f_{11}$ = get_box_features($\mathcal{Z}$)
    $V$ = objects($\mathcal{L}, \mathcal{Z}$)
    $E$ = random_forest.predict($V, f_1..f_{11}$)
    $S_{SG} = S_{SG} \cup G(V, E)$
end for
                          ▷ Pairwise Relation Summary (PRS) generation
$PRS$ = PRS_histograms($S_{SG}$)
$PRS_f = \{P | P \in PRS, support(P) \geq minsup_h, entropy(P) \leq maxent_h\}$
                                         ▷ Scene graph preprocessing
$S_{SG}$ = prune_edges($S_{SG}, PRS_f$)
$S_{SG}$ = prune_nodes($S_{SG}$)
                                               ▷ Scene graph mining
$SGS$ = subgraph_mining($S_{SG}$, minsup)
```

F. SImS COMPLEXITY

The main steps of the SImS algorithm are summarized in Algorithm 1.

By analyzing the steps of the SImS algorithm, we define its computational complexity as $O(kb^2hw + kb^3)$, where k is the number of images analyzed, all of which are assumed to have height h and width w. For each image, b objects are assumed to be found (i.e. b is the average number of objects found per image). The cubic dependency on b may appear troublesome, but it should be noted that, in a typical image, this value is in the order of ≈ 10 (in COCO for instance we extracted, on average, 11 objects for each image). Finally, we note that this complexity estimation does not include the panoptic segmentation and the subgraph mining steps, because the complexity of these steps depends on the selected approach and SImS does not rely on any specific approach to perform these steps.

V. EVALUATION METHODOLOGY

To assess SImS results, we rely on two frequently used evaluation metrics: *coverage* [21], [52] and *diversity* [7], [16]. Since ground truth summaries are not available, other evaluation metrics such as ROUGE [53], V-ROUGE [16] and VERT [54] are not applicable.

To analyze coverage, we exploit two coverage metrics: (i) *coverage*, and (ii) *coverage degree*. Coverage specifies the percentage of graphs in the input collection that are *represented* by graphs in the SGS. A graph is represented in the summary if one of its subgraphs is isomorphic to an SGS graph. Instead, the coverage degree describes the completeness of the summary in representing all the information available in the input scene graphs. More specifically, it measures the average representation degree of the input scene graphs. The representation degree of a scene graph is the percentage of its information that is included in the most representative SGS graph. Both coverage and coverage degree range in [0, 1].

Definition 4 (Represented Graph): Let G_i be an arbitrary scene graph in the input collection $\mathcal{G}$. G_i is represented in the *SGS* if the following function is true:

$$represented(G_i, SGS) = \begin{cases} 1, & if\ \exists\ G_j \subseteq G_i \mid G_j \in SGS \\ 0, & otherwise \end{cases} \quad (1)$$

where the operator $\subseteq$ indicates subgraph isomorphism.

Definition 5 (Coverage): Let *SGS* be a summary and $\mathcal{G}$ an image collection. We define *SGS* coverage with respect to $\mathcal{G}$ as:

$$coverage(SGS, \mathcal{G}) = \sum_{G_i \in \mathcal{G}} represented(G_i, SGS)/|\mathcal{G}| \quad (2)$$

where $|\mathcal{G}|$ is the number of images in $\mathcal{G}$.

Coverage ranges from $|SGS|/|\mathcal{G}|$ to 1. The minimum value is reached when each SGS graph represents only one image in the collection. A coverage of 1 (best quality) is reached instead when all the images are represented. This condition should be achieved with the lowest number of SGS graphs.

The representation degree measures the size of the maximal SGS graph representing an input scene graph G_i with respect to G_i itself.

Definition 6 (Representation Degree): Let G_i be an arbitrary scene graph in the input collection $\mathcal{G}$. The representation degree of G_i with respect to the summary *SGS* is:

$$\begin{aligned} \mathcal{R}_{degree}(G_i, SGS) &= \max_{G_j} |l_v(G_i) \cap l_v(G_j)|/|l_v(G_i)| \\ &\mid G_j \in SGS \wedge represented(G_i, \{G_j\}) \end{aligned} \quad (3)$$

where $l_v(G_i)$, $l_v(G_j)$ are the node labels of G_i and G_j respectively, and G_j are the summary graphs that represent G_i.

The fraction $|l_v(G_i) \cap l_v(G_j)|/|l_v(G_i)|$ computes the percentage of node labels in the scene graph G_i that are included in the SGS graph G_j. Hence, this value measures the amount of information that is represented by G_j. Among the summary graphs G_j, the metric considers the one with the maximum percentage of representative information.

Definition 7 (Coverage Degree): Let *SGS* be a summary and $\mathcal{G}$ an image collection. We define the *SGS* coverage degree with respect to $\mathcal{G}$ as:

$$coverage_{degree} = \sum_{G_i \in \mathcal{G}} \mathcal{R}_{degree}(G_i, SGS)/|\mathcal{G}| \quad (4)$$

Differently from coverage, this metric penalizes small SGS graphs (i.e., more general) that typically represent many input images. Indeed, this metric is equal to 1 only when all the nodes of the input scene graphs are represented by the SGS. This requirement is satisfied with a correct trade-off between the representativity and the detail level of summary graphs.

Diversity measures the average *dissimilarity* between summary items. A high-quality summary should be characterized by high diversity to avoid redundancy. In this paper we define two different metrics: (i) *node diversity*, and (ii) *edge diversity*.

Edge pruning (see Section IV-D) guarantees that node pairs with a specific class will be likely connected by edges with few predefined labels. Therefore, in the case of node diversity, graph dissimilarities can be computed by considering two graphs with the same node labels as semantically equivalent. Specifically, graph dissimilarity is defined as the complement of Intersection over Union (IoU) between their node labels.

Definition 8 (Node Dissimilarity): Let G_i, G_j be two graphs in the SGS. Their node dissimilarity nd is defined as:

$$nd(G_i, G_j) = 1 - \frac{|l_v(G_i) \cap l_v(G_j)|}{|l_v(G_i) \cup l_v(G_j)|} \tag{5}$$

where $l_v(G_i)$, $l_v(G_j)$ represent the node labels of G_i and G_j respectively.

Node diversity is defined as the average node dissimilarity of SGS graphs.

Definition 9 (Node Diversity): Let SGS be a scene graph summary. Its node diversity is defined as:

$$diversity_n(SGS) = \begin{cases} nd_{avg}(SGS) & \text{if } |SGS| > 1 \\ 1 & \text{otherwise} \end{cases} \tag{6}$$

where nd_{avg} is the average node dissimilarity across all pairs of summary graphs G_i and G_j,

$$nd_{avg}(SGS) = \frac{2 \cdot \sum_{G_i, G_j \in SGS, i<j} nd(G_i, G_j)}{|SGS| \cdot (|SGS| - 1)} \tag{7}$$

This measure takes values in the range [0, 1]. Higher values imply a better summary because the summary graphs contain non-redundant information.

Edge diversity also considers edge labels when comparing the graphs. To this aim, we define a function that describes the list of edges in a scene graph with a set of tuples.

Definition 10 (Edge Description): Let G_i be a scene graph in the SGS. Its edges are described with:

$$edges(G_i) = \{(l_v(v_j), l_e(e), l_v(v_k)) \mid e = (v_j, v_k) \in E\} \tag{8}$$

where e is an edge of G_i connecting the nodes v_j, v_k, while $l_v(v_j)$ and $l_v(v_k)$ are the node labels, and $l_e(e)$ is the edge label.

Similarly to node dissimilarity, we define the edge dissimilarity and edge diversity by exploiting edge descriptions.

Definition 11 (Edge Dissimilarity): Let G_i, G_j be two graphs in the SGS. Their edge dissimilarity ed is defined as:

$$ed(G_i, G_j) = 1 - \frac{|edges(G_i) \cap edges(G_j)|}{|edges(G_i) \cup edges(G_j)|} \tag{9}$$

Definition 12 (Edge Diversity): Let SGS be a scene graph summary. Its edge diversity is defined as:

$$diversity_e(SGS) = \begin{cases} ed_{avg}(SGS) & \text{if } |SGS| > 1 \\ 1 & \text{otherwise} \end{cases} \tag{10}$$

where ed_{avg} is the average edge diversity across all pairs of summary graphs G_i and G_j,

$$ed_{avg}(SGS) = \frac{2 \cdot \sum_{G_i, G_j \in SGS, i<j} ed(G_i, G_j)}{|SGS| \cdot (|SGS| - 1)} \tag{11}$$

where G_i and G_j are summary graphs.

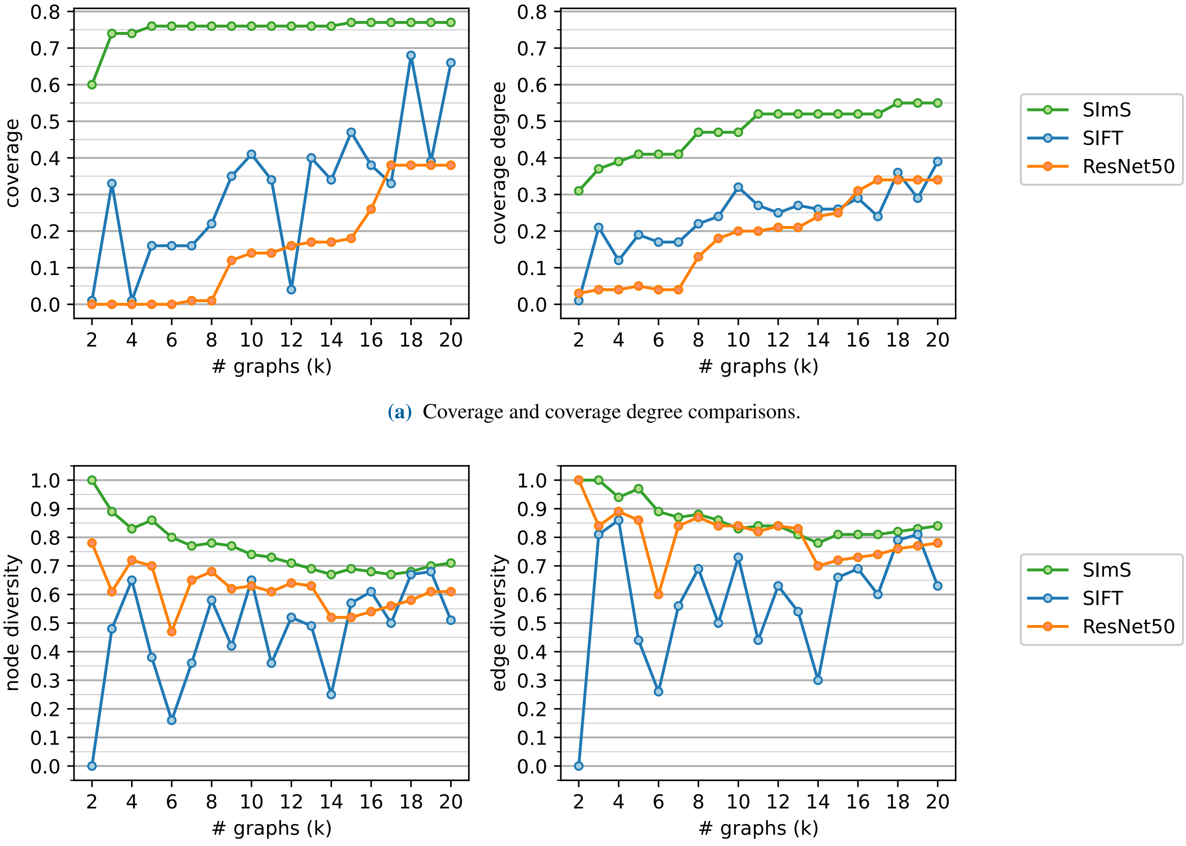
VI. EXPERIMENTAL EVALUATION

We evaluated SImS performance on the training set of the Microsoft COCO dataset [9]. This collection contains 118K manually segmented images in the panoptic segmentation format. Annotations include 53 stuff classes (uncountable objects, like "grass") and 80 object classes. We selected this dataset due to its wide variety of class labels. Other datasets including panoptic annotations, such as Cityscapes [55] and ADE20K [6], are less suitable due to the lower variety of represented scenes, but could be analyzed with appropriate changes on the input data preprocessing pipeline. The COCO dataset used for this study is available through the official website: `https://cocodataset.org`. We refer to the train split of the 2017 edition of the dataset. As a first step, we compare the performance of SImS with previous summarization methods (see Section VI-A). Next, we analyze SImS inner workings. We address (a) the evaluation of the relative position classifier used to build scene graphs (see Section VI-B), (b) the results of the PRS generation from the 118K images in COCO (see Section VI-C), (c) the effectiveness of the scene graph preprocessing steps, and (d) the effect of changing the frequent subgraph mining algorithm and its configuration (see Section VI-D for both).

When providing the running time, all the experiments refer to the following hardware configuration: Intel Xeon Gold 6140, CPU @ 2.30GHz, RAM 40 GB. SImS is implemented in Python 3, with the exception of the graph mining algorithms, for which we rely on the available online C implementations [49], [50]. Source code and the dataset for training the relative position classifiers are available in our repository: `https://github.com/AndreaPasini/SImS`

A. COMPARISON WITH OTHER SUMMARIZATION TECHNIQUES

SImS is compared with a common summarization baseline that relies on visual features and builds the summary with a subset of the image collection. Due to the unavailability of online implementations for other image collection summarization methods, we implemented [18], based on KMedoids clustering. This method is used as a baseline in most of the literature works cited in Section II. By following its standard implementation, we extract SIFT features from the input dataset, then represent each image with a Bag Of Words



(a) Coverage and coverage degree comparisons.

(b) Diversity comparisons.

FIGURE 8. Quantitative comparison between SImS and KMedoids on COCO Subset 1 (driving-skiing, 4865 images).

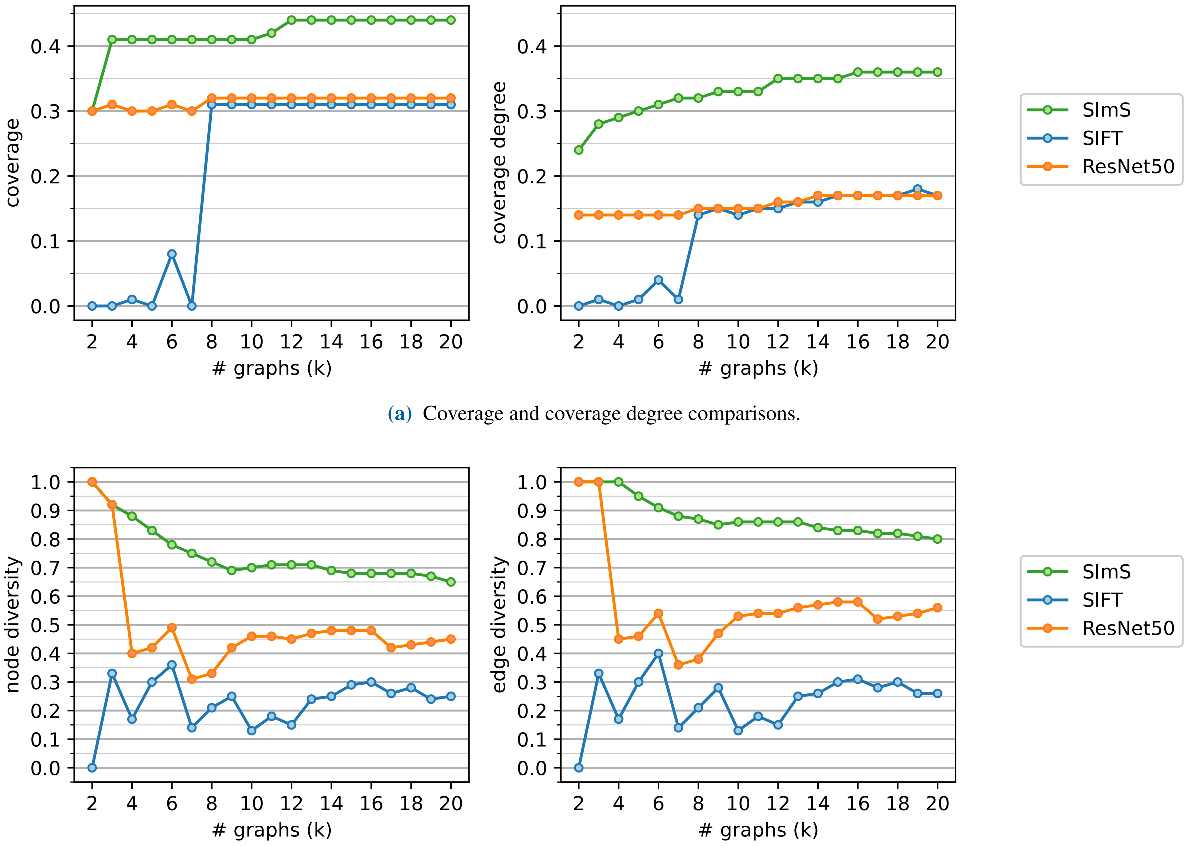
vector of length 1000. This version will be referred to as ''SIFT''. We also introduced an alternative feature extraction method. Using a version of ResNet50 without the fully-connected head, we extracted features in a 2048-dimensional space. Similarly to SIFT, we then applied KMedoids to this representation. We refer to this version as ''ResNet50''. For both methods, we apply KMedoids with different values of k to generate summaries with different sizes. Our implementations of [18] are available in our code repository.

Due to the scalability issues of KMedoids, we performed the experiments on two subsets of COCO. Specifically, we picked from COCO the images based on their captions. The first subset (Subset 1) includes the 4865 images containing the words ''skiing'' or ''driving'' in their caption, while the second subset (Subset 2), including 890 images, is selected with the words ''garden'' and ''church''. Other subsets of COCO could have been selected without significant changes in the analysis results. The choice of exploiting image captions to build the subset, instead of considering scene graph object classes, allows a fair comparison between the two summarization methods. In particular, it avoids bias caused by a selection based on the same information contained in the scene graphs used by SImS.

For Kmedoids the experiments are run with k in the range 2-20. In the case of SImS, we selected a reference hyperparameter configuration (config. 3 in Table 4, see Section VI-D) and picked the top-k frequent graphs in the SGS. For a fair evaluation, coverage and diversity are computed by extracting the scene graphs from the output images of both methods. To generate the scene graphs we use the procedure described in Section IV-B, while the output images in the case of SImS are derived from the SGS as explained in Section IV-E. Finally, the scene graphs obtained from KMedoids and SImS are preprocessed with edge and node pruning to extract the important information before computing the evaluation metrics.

Figure 8 and 9 provide coverage and diversity scores for the three methods on Subset 1 and Subset 2, respectively. Focusing on Subset 1, Figure 8a shows that SImS easily reaches a high coverage with few graphs ($k = 5$, coverage $= 0.76$), while both versions of KMedoids can reach at most coverage 0.68 with $k = 18$ (for SIFT). Similarly, the coverage degree is always higher for SImS, reaching value 0.55 at $k = 18$.

Figure 8b shows that node diversity has a descending trend for all curves. This should be expected since, intuitively, a lower number of selected graphs will result in those



(a) Coverage and coverage degree comparisons.

(b) Diversity comparisons.

FIGURE 9. Quantitative comparison between SImS and KMedoids on COCO Subset 2 (garden-church, 890 images).

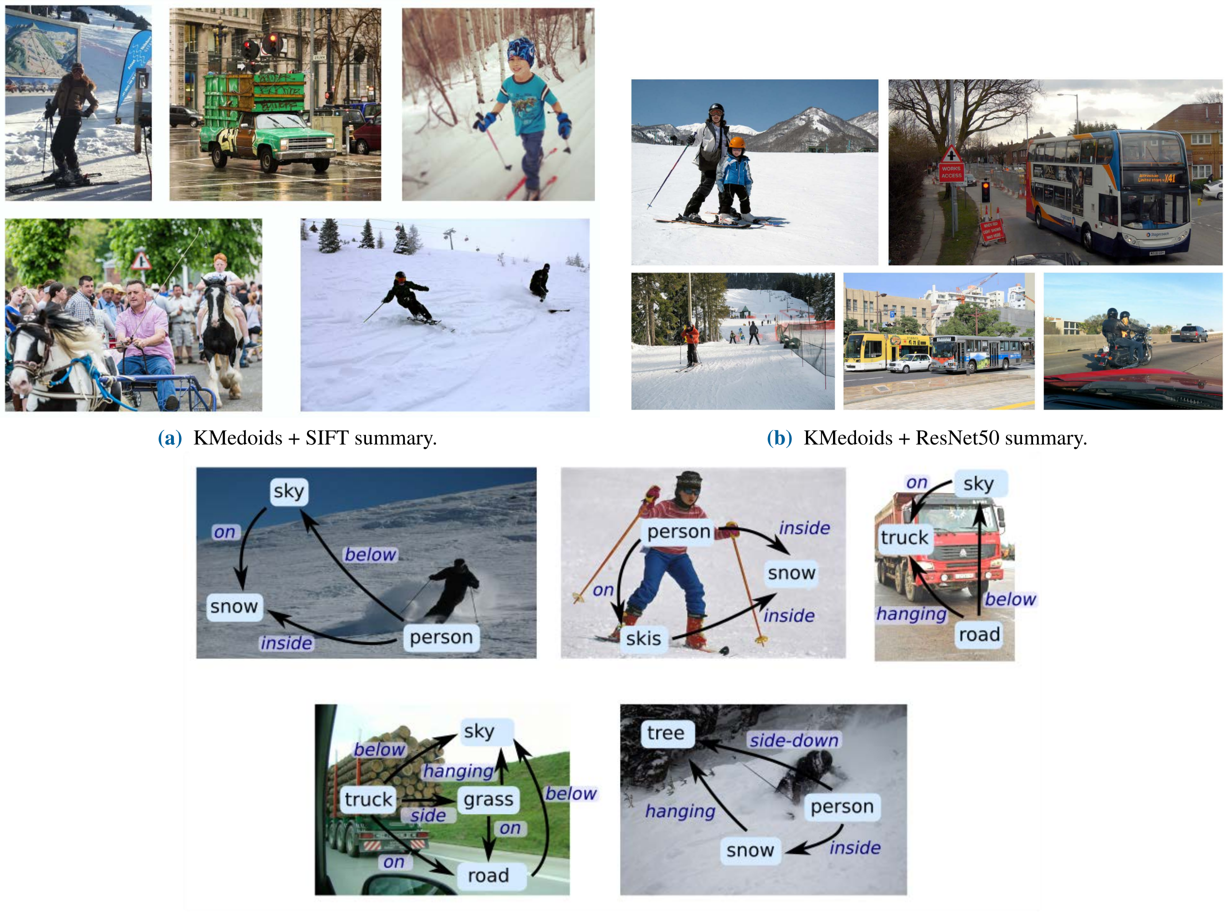
graphs being more dissimilar from one another. For SImS, the minimum value is 0.67 for $k = 14$. KMedoids node diversity is always lower than SImS for both versions. For a large k, it reaches its maximum value of 0.68 at $k = 19$. Higher values are reached by the ResNet50 version for small values of k. This, as already discussed, is to be expected given the nature of this metric. A similar trend is described by edge diversity, which inspects the information carried by edge labels. In this case, the ResNet50 representation reaches performance comparable to those of SImS in the range $k \in [7, 13]$. Finally, SImS shows better coverage and diversity results also in Subset 2 (Figures 9a and 9b). In general, SImS shows better performance in all aspects of the evaluation.

We also performed a qualitative comparison of the three methods. Figure 10 reports the results for $k = 5$ (Subset 1). Both methods show images depicting people skiing and cars/trucks. In Figure 10c, SImS graphs also highlight the important objects in these images. Coverage of both KMedoids approaches is typically lower since it includes concepts with a low frequency in the collection. For example, Figure 10a contains a picture that belongs to the "driving" topic and shows a man on a carriage. This image, albeit contributing to diversity, depicts a pattern that is much less frequent in the collection, because most images for this topic contain cars and trucks. Similarly, Figure 10b includes a picture of a motorcycle, another instance of an infrequent occurrence within the subset.

B. SCENE GRAPH COMPUTATION

The computation of scene graphs requires a classifier to label edges between each pair of objects in the graph (see Section IV-B). The dataset used for training the classifier has been generated by randomly selecting 700 images from the COCO training set. In each image we randomly selected a subject-reference pair and labeled it with one of the 9 relative position classes (e.g., "above", "below"). We finally balanced the dataset by means of a stratified sampling that yielded 540 images, including 60 samples for each class label. The final training set is openly available in our code repository.

To find the most suitable classifier, we considered the following set of classifiers available in the scikit-learn library [56]: decision tree, SVM, random forest, naive bayes, KNN. Each of them is evaluated in terms of F1 score with leave-one-out cross-validation and a grid search methodology.



(a) KMedoids + SIFT summary.

(b) KMedoids + ResNet50 summary.

(c) SImS summary.

FIGURE 10. Qualitative comparison on COCO subset ("skiing", "driving"), with 5 summary elements.

The obtained F1 scores, precisions and recalls are presented in Table 3. Random forest achieves the best macro-average F1 (0.928), with a grid search on maxdepth in the range 5-40 and a number of estimators in the range 10-200. The best configuration, according to macro-average F1 entails maxdepth = 20, number of estimators = 100.

The final generation of the complete set of scene graphs (118K graphs), including the feature extraction step and the classification step by means of the random forest classifier with its best configuration, took 4 hours on the training set of COCO.

C. PAIRWISE RELATIONSHIP SUMMARY GENERATION

The Pairwise Relationship Summary (PRS) represents in the form of histograms the distributions describing the position relationships between object pairs. Its generation entails two steps: (a) PRS extraction from the scene graphs set and (b) PRS filtering.

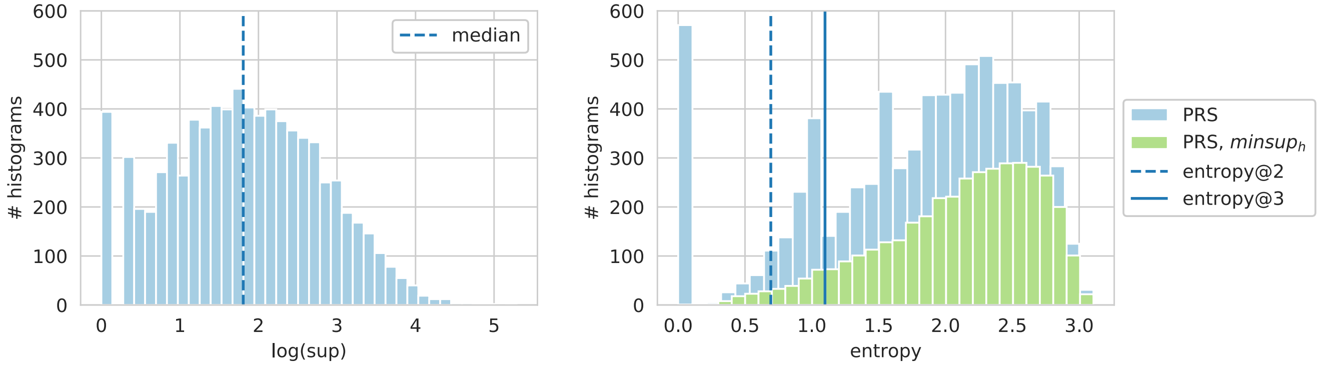
The extraction phase takes 20 seconds to analyze the 5M object pairs in the scene graphs set. The generated PRS includes 7867 histograms, whose support distribution is shown in Figure 11a. Since the support distribution is positively skewed, the horizontal axis is in log-scale, to obtain a gaussian-shaped distribution.

The PRS filtering phase selects the most significant histograms by enforcing the $minsup_h$ and $maxentr_h$ thresholds. The $minsup_h$ parameter is designed to increase the time performance of the FSM process. Higher values reduce FSM running time by removing outlier histograms that are supported by few training examples.

We analysed the sensitivity of the $minsup_h$ parameter by considering the effect of its variation in the value range [0%, 5%] (corresponding to [0, 10000] with absolute support). The experiments are performed on the whole COCO dataset, using the gSpan algorithm with $minsup = 0.01$. Figure 12 shows that when $minsup_h$ ranges in [0, 0.5%] ([0, 1000] with absolute support), coverage remains fixed to value 0.43, while diversity has slight fluctuations between 0.80 and 0.81. Additional increments of $minsup_h$ cause a slight decrease of both coverage and diversity. In all the other experiments we conventionally fixed $minsup_h = 64$, which corresponds to the median of the support distribution in log-scale (see Figure 11a).

TABLE 3. F1 score for the pairwise relative position computation.

		KNN	RBF-SVM	Decision tree	Random forest
above	F1 score	0.894	0.894	0.914	**0.949**
	precision	0.873	0.873	0.946	**0.966**
	recall	0.917	0.917	0.883	**0.933**
around	F1 score	0.931	0.939	0.950	**0.958**
	precision	0.964	**0.982**	0.950	0.966
	recall	0.900	0.900	**0.950**	**0.950**
below	F1 score	0.949	0.924	0.941	**0.967**
	precision	**0.966**	0.932	0.949	0.952
	recall	0.933	0.917	0.933	**0.983**
hanging	F1 score	0.900	0.845	0.902	**0.926**
	precision	0.900	0.875	0.887	**0.918**
	recall	0.900	0.817	0.917	**0.933**
inside	F1 score	0.922	0.938	0.924	**0.949**
	precision	0.964	**1.000**	0.932	0.966
	recall	0.883	0.883	0.917	**0.933**
on	F1 score	0.783	0.824	0.846	**0.934**
	precision	0.783	0.831	0.825	**0.919**
	recall	0.783	0.817	0.867	**0.950**
side	F1 score	0.816	0.750	0.793	**0.843**
	precision	0.785	0.808	0.787	**0.836**
	recall	**0.850**	0.700	0.800	**0.850**
side-down	F1 score	0.894	0.879	0.926	**0.933**
	precision	0.873	0.806	0.918	**0.933**
	recall	0.917	**0.967**	0.933	0.933
side-up	F1 score	**0.917**	0.855	0.824	0.891
	precision	**0.917**	0.789	0.831	0.898
	recall	0.917	**0.933**	0.817	0.883
macro	F1 score	0.890	0.872	0.891	**0.928**
	precision	0.892	0.877	0.892	**0.928**
	recall	0.889	0.872	0.891	**0.928**



(a) Support distribution.

(b) Entropy distribution.

FIGURE 11. PRS histogram statistics.

Figure 11b depicts the distribution of histogram entropy before and after the application of the $minsup_h$ filter. As expected, this filtering causes the removal of the bin with close-to-zero entropy histograms, which are typically supported by very few noisy samples.

The entropy threshold $maxentr_h$ allows the selection of histograms representing high quality distributions, that are concentrated on very few relationship types. To this aim, we consider two reference histograms with only 2 and 3 non-zero elements (among the 9 position relationships) and we compute their entropy values. We denote as *entropy*@2 and *entropy*@3 the entropy of these two reference histograms.

The number of histograms in the PRS_f filtered summary with $minsup_h = 64$, $maxentr_h = entropy@2$ is 77 (0.9% of the initial data). It becomes 277 (3%) when setting $maxentr_h$ to *entropy*@3. We selected *entropy*@3 as a better trade-off between the quality of the histograms and their number. Figure 13 shows various examples of histograms included in the PRS_f filtered with this setting. These examples formalize from commonly expected relationships. We find, for example, that "kite" is most likely to be in a relationship of type "inside" with "sky", whereas "sea" is either below or hanging from the sky. This highlights how different object interact differently with one another, as is expected. The provided examples are shown based on a subset of interactions that were expected to be found (i.e. a human-validated ground truth) and that have actually been identified. The distribution of relationship types matches the expectations that stem from common knowledge.

D. SCENE GRAPH SUMMARY GENERATION

The SGS generation phase is the core part of SImS. We ran different experiment configurations by (i) turning on/off the activation of the edge and node pruning steps, (ii) selecting either gSpan or SUBDUE as graph mining algorithm, and (iii) varying the value of *minsup* for gSpan. The experimental results, reporting the running time, the number of frequent graphs and their quality indices, are shown in Table 4.

Consider first the running time of the scene graph mining process. The first line of Table 4 shows that the direct application of gSpan, without edge and node pruning, takes roughly 16 hours to generate the frequent graphs with $minsup = 0.01$. When introducing edge pruning, it reduces to 4 hours and 30 minutes (config. 2), while, if the node pruning step is also activated, it drastically reduces to 3 seconds. When lowering *minsup* to 0.001, if the preprocessing steps are not enabled, the mining algorithm does not end within 2 days (config. 4). Conversely, with preprocessing enabled, the summary is generated in 7 seconds only (config. 5 in Table 4). Similarly, the experiments with SUBDUE (config. 6 and 7) show that the node pruning step allows reducing the running time from 12 hours to 17 minutes. When considering the running time, gSpan shows the best performance.

Consider now the effects of the preprocessing steps on the SGS content for config. 1-3. The application of edge pruning lowers the number of graphs from 6184 (config. 1) to 186 (config. 2). Conversely, node pruning slightly increases the number of frequent graphs (from 186 in config. 2 to 276 in config. 3) due to the simplification of the input collection.

The *Avg. N. nodes* and *Std. N. nodes* indicators in Table 4 provide the average and the standard deviation of the number of nodes in the frequent graphs. The average number of nodes tends to be lower after applying node pruning (from 5.29 in config. 1 to 3.11 in config. 3).

Coverage and diversity, described in Section V, assess the quality of the obtained summary. On the one hand, the designed preprocessing steps reduce the complexity of the input dataset and the summary size, while preserving the same coverage (0.43 for config. 1-3). On the other hand, edge and node pruning are fundamental to increase the node diversity of summary graphs (from 0.60 to 0.81).

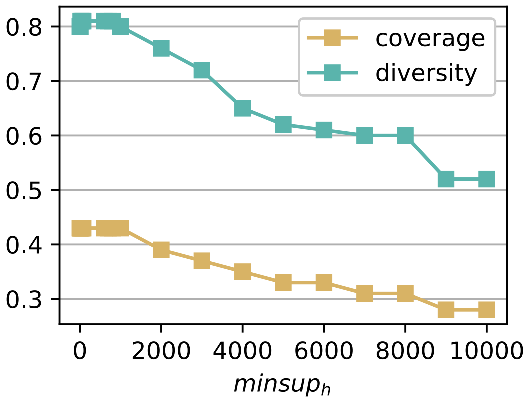


FIGURE 12. $minsup_h$ sensitivity.

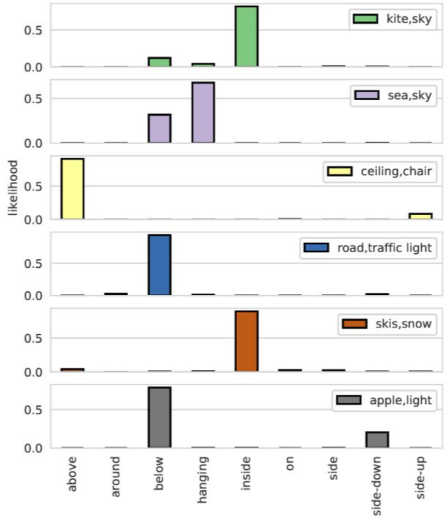


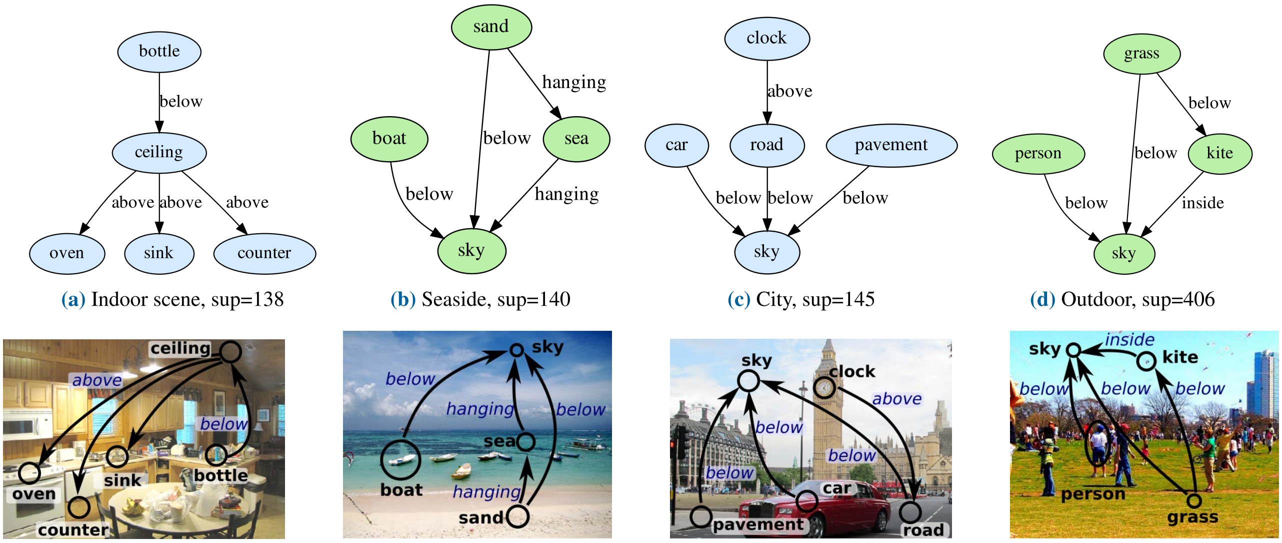
FIGURE 13. Example histograms in the PRS_f.

If we lower *minsup* to 0.001 (config. 5), we obtain more graphs (9865), with a higher number of nodes (5.24) that can be more interesting to characterize complex recurring scenes in the input collection. Coverage reaches 0.48, while node diversity is slightly lower (0.75) due to the high number of graphs.

SUBDUE (config. 7) performs worse with respect to gSpan also in terms of summary quality. It only generates 48 graphs with a low coverage and diversity. Interestingly, SUBDUE graphs tend to be bigger ($avg = 6.15$), because this algorithm aims at finding larger substructures that will provide a better compression of the graph collection.

TABLE 4. SGS generation results on whole COCO training set (118K images).

Configuration					*Statistics*					
Config.	Alg.	Minsup	Edge pruning	Node pruning	Scene graph mining time	N. graphs	Avg. N. nodes	Std. N. nodes	Coverage	Node diversity
1	gSpan	0.010	N	N	15h 55m	6184	5.29	2.05	0.43	0.60
2	gSpan	0.010	Y	N	4h 30m	186	4.20	2.22	0.43	0.69
3	gSpan	0.010	Y	Y	**3s**	276	3.11	0.97	0.43	**0.81**
4	gSpan	0.001	N	N	-					
5	gSpan	0.001	Y	Y	7s	9865	5.24	1.80	**0.48**	0.75
6	SUBDUE	-	Y	N	12h	184	19.89	8.94	0.24	0.35
7	SUBDUE	-	Y	Y	17m	48	6.15	2.35	0.24	0.38



(a) Indoor scene, sup=138 **(b)** Seaside, sup=140 **(c)** City, sup=145 **(d)** Outdoor, sup=406

(e) Indoor scene, example image **(f)** Seaside, example image **(g)** City, example image **(h)** Outdoor, example image

FIGURE 14. Examples of frequent graphs in the SGS, minsup = 0.001 (config. 5).

To draw conclusions, for the COCO dataset, the best configurations according to coverage, diversity and running time are config. 3 and config. 5, with the gSpan algorithm and activation of both scene graph preprocessing steps. Considering config. 5 ($minsup = 0.001$), which includes bigger and more interesting graphs, some examples of output scene graphs are shown in Figure 14 (a, b, c, d), while four example images including these graphs are depicted in Figure 14 (e, f, g, h). Typically, each graph represents a distinct scene type in the COCO images.

VII. CONCLUSION AND FUTURE WORK

In this paper we presented SImS, a semantic summarization technique for image collections. The proposed method can automatically derive a novel type of scene graphs from a set of segmented images and generate two types of summary patterns: (i) the PRS, representing pairwise object relationships, and (ii) the SGS, describing more complex object configurations. The experimental evaluation shows that, differently from previous techniques that are based on visual features or simple image tags and do not explicitly describe the relevant high-level concepts in the summary, our results are interpretable and rich of semantic information. Furthermore, our preprocessing method for scene graphs is effective in both reducing the amount of redundant information and significantly speeding up the graph mining process.

As future work we plan to extend the semantics of our scene graphs by including not only position relationships, but also actions (e.g., "person reads book") and object properties (e.g., "a red car"). Furthermore, we will incorporate in the node pruning step the definition of aggregate nodes (e.g., multiple "person" objects grouped into a single node with the "people" label). Finally, we will improve the summary characterization by introducing hierarchies to group SGS graphs based on their semantic content.

REFERENCES

[1] G. Kim, S. Moon, and L. Sigal, "Joint photo stream and blog post summarization and exploration," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3081–3089.

[2] I. Simon, N. Snavely, and S. M. Seitz, "Scene summarization for online image collections," in *Proc. IEEE 11th Int. Conf. Comput. Vis.*, Oct. 2007, pp. 1–8.

[3] Y. Li, T. Mei, Y. Cong, and J. Luo, "User-curated image collections: Modeling and recommendation," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Oct. 2015, pp. 591–600.

[4] P. Goyal, S. Sahu, S. Ghosh, and C. Lee, "Cross-modal learning for multi-modal video categorization," 2020, *arXiv:2003.03501*.

[5] H. Caesar, J. Uijlings, and V. Ferrari, "COCO-stuff: Thing and stuff classes in context," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1209–1218.

[6] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene parsing through ADE20K dataset," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 633–641.

[7] J. E. Camargo and F. A. González, "Multimodal latent topic analysis for image collection summarization," *Inf. Sci.*, vol. 328, pp. 270–287, Jan. 2016.

[8] Y. Zhao, R. Hong, and J. Jiang, "Visual summarization of image collections by fast RANSAC," *Neurocomputing*, vol. 172, pp. 48–52, Jan. 2016.

[9] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár, "Panoptic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 9404–9413.

[10] A. Kirillov, R. Girshick, K. He, and P. Dollár, "Panoptic feature pyramid networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 6399–6408.

[11] Y. Xiong, R. Liao, H. Zhao, R. Hu, M. Bai, E. Yumer, and R. Urtasun, "UPSNet: A unified panoptic segmentation network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 8818–8826.

[12] S. Aditya, Y. Yang, and C. Baral, "Integrating knowledge and reasoning in image understanding," 2019, *arXiv:1906.09954*.

[13] F. Sadeghi, S. K. Divvala, and A. Farhadi, "VisKE: Visual knowledge extraction and question answering by visual verification of relation phrases," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1456–1464.

[14] J. Camargo, F. González, and R. Torres, "Visualization, summarization and exploration of large collections of images: State of the art," in *Proc. Latin-Amer. Conf. Netw. Electron. Media (LACNEM)*, 2009.

[15] A. Singh and D. K. Sharma, "Image collection summarization: Past, present and future," in *Data Visualization and Knowledge Engineering*. Cham, Switzerland: Springer, 2020, pp. 49–78.

[16] S. Tschiatschek, R. K. Iyer, H. Wei, and J. A. Bilmes, "Learning mixtures of submodular functions for image collection summarization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 1413–1421.

[17] D. Deng, "Content-based image collection summarization and comparison using self-organizing maps," *Pattern Recognit.*, vol. 40, no. 2, pp. 718–727, Feb. 2007.

[18] Y. Hadi, F. Essannouni, and R. O. H. Thami, "Video summarization by *k*-medoid clustering," in *Proc. ACM Symp. Appl. Comput.*, 2006, pp. 1400–1401.

[19] C. Yang, J. Shen, J. Peng, and J. Fan, "Image collection summarization via dictionary learning for sparse representation," *Pattern Recognit.*, vol. 46, no. 2, pp. 948–961, 2013.

[20] J. Fan, Y. Gao, H. Luo, D. A. Keim, and Z. Li, "A novel approach to enable semantic and visual image summarization for exploratory image search," in *Proc. 1st ACM Int. Conf. Multimedia Inf. Retr.*, 2008, pp. 358–365.

[21] Z. R. Samani and M. E. Moghaddam, "A knowledge-based semantic approach for image collection summarization," *Multimedia Tools Appl.*, vol. 76, no. 9, pp. 11917–11939, May 2017.

[22] J. Johnson, R. Krishna, M. Stark, L.-J. Li, D. A. Shamma, M. S. Bernstein, and L. Fei-Fei, "Image retrieval using scene graphs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3668–3678.

[23] M. Fisher, M. Savva, and P. Hanrahan, "Characterizing structural relationships in scenes using graph kernels," in *Proc. ACM SIGGRAPH Papers*, 2011, pp. 1–12.

[24] M. de Boer, L. Daniele, P. Brandt, and M. Sappelli, "Applying semantic reasoning in image retrieval," in *Proc. 1st Int. Conf. Big Data, Small Data, Linked Data Open Data (ALLDATA)*, 2015, pp. 69–74.

[25] S. Schuster, R. Krishna, A. Chang, L. Fei-Fei, and C. D. Manning, "Generating semantically precise scene graphs from textual descriptions for improved image retrieval," in *Proc. 4th Workshop Vis. Lang.*, 2015, pp. 70–80.

[26] J. Gu, S. Joty, J. Cai, H. Zhao, X. Yang, and G. Wang, "Unpaired image captioning via scene graph alignments," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 10323–10332.

[27] X. Yang, K. Tang, H. Zhang, and J. Cai, "Auto-encoding scene graphs for image captioning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 10685–10694.

[28] O. Ashual and L. Wolf, "Specifying object attributes and relations in interactive scene generation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 4561–4569.

[29] J. Johnson, A. Gupta, and L. Fei-Fei, "Image generation from scene graphs," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1219–1228.

[30] B. Schroeder, S. Tripathi, and H. Tang, "Triplet-aware scene graph embeddings," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Oct. 2019, pp. 1–5.

[31] S. Tripathi, S. N. Sridhar, S. Sundaresan, and H. Tang, "Compact scene graphs for layout composition and patch retrieval," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2019, pp. 1–8.

[32] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, M. S. Bernstein, and L. Fei-Fei, "Visual genome: Connecting language and vision using crowdsourced dense image annotations," *Int. J. Comput. Vis.*, vol. 123, no. 1, pp. 32–73, 2017.

[33] C. Galleguillos, A. Rabinovich, and S. Belongie, "Object categorization using co-occurrence, location and appearance," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2008, pp. 1–8.

[34] J. Yang, J. Lu, S. Lee, D. Batra, and D. Parikh, "Graph R-CNN for scene graph generation," *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 670–685.

[35] R. Zellers, M. Yatskar, S. Thomson, and Y. Choi, "Neural motifs: Scene graph parsing with global context," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5831–5840.

[36] A. Pasini and E. Baralis, "Detecting anomalies in image classification by means of semantic relationships," in *Proc. IEEE 2nd Int. Conf. Artif. Intell. Knowl. Eng. (AIKE)*, Jun. 2019, pp. 231–238.

[37] C. Jiang, F. Coenen, and M. Zito, "A survey of frequent subgraph mining algorithms," *Knowl. Eng. Rev.*, vol. 28, no. 1, pp. 75–105, Mar. 2013.

[38] M. Kuramochi and G. Karypis, "Frequent subgraph discovery," in *Proc. IEEE Int. Conf. Data Mining*, Nov./Dec. 2001, pp. 313–320.

[39] W. Liu, L. Zhu, L. Chu, and H. Ma, "A common subgraph correspondence mining framework for map search services," *Multimedia Tools Appl.*, vol. 78, no. 1, pp. 747–766, Jan. 2019.

[40] S. Hill, B. Srichandan, and R. Sunderraman, "An iterative MapReduce approach to frequent subgraph mining in biological datasets," in *Proc. ACM Conf. Bioinf., Comput. Biol. Biomed.*, 2012, pp. 661–666.

[41] A. Mrzic, P. Meysman, W. Bittremieux, P. Moris, B. Cule, B. Goethals, and K. Laukens, "Grasping frequent subgraph mining for bioinformatics applications," *BioData Mining*, vol. 11, no. 1, pp. 1–24, Dec. 2018.

[42] B. Güvenoglu and B. E. Bostanoglu, "A qualitative survey on frequent subgraph mining," *Open Comput. Sci.*, vol. 8, no. 1, pp. 194–209, Dec. 2018.

[43] T. Ramraj and R. Prabhakar, "Frequent subgraph mining algorithms—A survey," *Proc. Comput. Sci.*, vol. 47, pp. 197–204, Jan. 2015.

[44] V. Bhatia and R. Rani, "Ap-FSM: A parallel algorithm for approximate frequent subgraph mining using pregel," *Expert Syst. Appl.*, vol. 106, pp. 217–232, Sep. 2018.

[45] M. A. Bhuiyan and M. Al Hasan, "An iterative MapReduce based frequent subgraph mining algorithm," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 3, pp. 608–620, Mar. 2015.

[46] F. Qiao, X. Zhang, P. Li, Z. Ding, S. Jia, and H. Wang, "A parallel approach for frequent subgraph mining in a single large graph using spark," *Appl. Sci.*, vol. 8, no. 2, p. 230, Feb. 2018.

[47] B. Jena, C. Khan, and R. Sunderraman, "SparkFSM: A highly scalable frequent subgraph mining approach using apache spark," in *Proc. IEEE Int. Conf. Data Mining Workshops (ICDMW)*, Nov. 2018, pp. 990–997.

[48] M. M. Sangle and P. Bhavsar, "gSpan-H: An iterative mapreduce based frequent subgraph mining algorithm," *Int. J. Adv. Res. Innov. Ideas Educ.*, vol. 2, no. 5, pp. 169–177, 2016.

[49] X. Yan and J. Han, "gSpan: Graph-based substructure pattern mining," in *Proc. IEEE Int. Conf. Data Mining*, Dec. 2002, pp. 721–724.

[50] D. J. Cook and L. B. Holder, "Substructure discovery using minimum description length and background knowledge," *J. Artif. Intell. Res.*, vol. 1, pp. 231–255, Feb. 1994.

[51] J. R. Smith and C.-S. Li, "Decoding image semantics using composite region templates," in *Proc. IEEE Workshop Content-Based Access Image Video Libraries*, Jun. 1998, pp. 9–13.

[52] P. Sinha, "Summarization of archived and shared personal photo collections," in *Proc. 20th Int. Conf. Companion World Wide Web*, 2011, pp. 421–426.

[53] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Proc. Workshop ACL Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 74–81.

[54] Y. Li and B. Merialdo, "VERT: Automatic evaluation of video summaries," in *Proc. 18th Int. Conf. Multimedia*, 2010, pp. 851–854.

[55] M. Cordts, M. Omran, S. Ramos, T. Scharwächter, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset," in *Proc. CVPR Workshop Future Datasets Vis.*, 2015.

[56] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: Experiences from the scikit-learn project," in *Proc. ECML PKDD Workshop, Lang. Data Mining Mach. Learn.*, 2013, pp. 108–122.

ANDREA PASINI received the master's and Ph.D. degrees in computer engineering from the Politecnico di Torino. His main research interests include machine learning, deep learning, big data, and data mining. His research also focuses on the integration of machine learning models with semantic knowledge extracted from data. He was a recipient of the Best Student Paper Award from IEEE AIKE 2019 with the article "Detecting Anomalies in Image Classification by Means of Semantic Relationships."

FLAVIO GIOBERGIA (Graduate Student Member, IEEE) received the dual master's degrees in computer engineering from the Politecnico di Torino and the Politecnico di Milano. He is currently pursuing the Ph.D. degree with the Database and Data Mining Group, Department of Control and Computer Engineering, Politecnico di Torino. His current research interest includes machine learning applications with limited labeled data availability, focusing on semi-supervised and transfer learning.

ELIANA PASTOR received the master's and Ph.D. degrees in computer engineering from the Politecnico di Torino. She is currently an Assistant Professor with the Department of Control and Computer Engineering, Politecnico di Torino. Her current research interests include trustworthy AI, explainable AI, and algorithms for big data.

ELENA BARALIS (Member, IEEE) received the master's degree in electrical engineering and the Ph.D. degree in computer engineering from the Politecnico di Torino. She has been a Full Professor with the Department of Control and Computer Engineering, Politecnico di Torino, since January 2005. She has published more than 200 papers in international journals and conference proceedings. Her current research interests include database systems and data mining, more specifically on mining algorithms for very large databases and sensor/stream data analysis. She has served on the Program Committees or as the Area Chair for several international conferences and workshops, among which VLDB, IEEE ICDM, SIGMOD, ACM SAC, DaWak, ACM CIKM, and PKDD.

Open Access funding provided by 'Politecnico di Torino' within the CRUI CARE Agreement