Laboratoire d'Analyses et Modélisation de Systèmes pour l'Aide à la Décision
UMR 7243

CAHIER DU LAMSADE

320

Mai 2012

A representation of contextual relationships knowledge in images

N. V. Hoang, V.Gouet-Brunet, M.Rukoz

LAMSADE Research Report

A representation of contextual relationships knowledge in images

Nguyen Vu Hoang[1,3], Valérie Gouet-Brunet[2], Marta Rukoz[1,3]

1 : LAMSADE - Université Paris-Dauphine - Place de Lattre de Tassigny - F75775 Paris Cedex 16
2 : CEDRIC/CNAM - 292, rue Saint-Martin - F75141 Paris Cedex 03
3 : Université Paris Ouest Nanterre La Défense - 200, avenue de la République - F92001 Nanterre Cedex
nguyenvu.hoang@dauphine.fr , valerie.gouet@cnam.fr , marta.rukoz@dauphine.fr

23 mars 2012

Abstract

This report is focused on the study of methods for image retrieval in collection of heterogeneous contents. The spatial relationships between entities in an image allow to create the global description of the image that we call the image context. Taking into account the contextual spatial relationships in the similarity search of images can allow improving the retrieval quality by limiting false alarms. We defined the context of image as the presence of entity categories and their spatial relationships in the image.

By studying statistically the relationships between different entity categories on LabelMe, a symbolic images databases of heterogeneous content, we create a cartography of their spatial relationships that can be integrated in a graph-based model of the contextual relationships, the principal contribution of this report. This graph describes the general knowledge of every entity categories. Spatial reasoning on this knowledge graph can help improving tasks of image processing such as detection and localization of an entity category by using the presence of another one. Further, this model can be applied to represent the context of an image. The similarity search based on context can be achieved by comparing the graphs, then, contextual similarity between two images is evaluated by the similarity between their graphs. This work was evaluated on the symbolic image database of LabelMe. The experiments showed its relevance for image retrieval by context.

Keywords : Image, similarity search, spatial relationships, image context.

1 Introduction

The interpretation of images by a machine requires to have a representation of images preprocessed manually or automatically. This representation can be built from visual features (such as color, shape of elements in images) or higher level information (such as spatial relationships between elements or models of these elements). To date, it is still difficult to build a robust model for automatic image

interpretation. On the contrary, humans prove their effectiveness in image processing tasks. We can say that what makes such humans' possible aptitudes is their ability to interpret visual features of images by using prior knowledge. Prior knowledge is very often related to the presence of multiple entities in an image and to the spatial information linking them, that can be called the context of the image. Humans can incorporate this knowledge to analyse the image and to create personal semantic concepts. According to [8], image interpretation can be classified into three levels of complexity :

- Level 1 : interpretation is based mainly on primary features such as color, texture, shape, segmented regions, interest points and/or the spatial location of image elements. These features are rather objective and their estimation is performed directly, it does not require any knowledge base. Many approaches of image retrieval or of image categorization can be classified into this level (e.g. Bag of Features and all its derived approaches).

- Level 2 : interpretation involves some degree of logical inference concerning the image content. At this level, queries are performed in order to retrieve entities of a given type or to retrieve a specific entity from another. The need of a processed knowledge base is obvious. This knowledge base can contain low-level information of entity categories (e.g color, shape), relationships between categories (e.g. correlations, conditional probabilities, spatial relationships), and more.

- Level 3 : interpretation is based on symbolic features. At this level, a significant amount of high-level reasoning about the meaning and purpose on the entities or on background of images can be involved. The result of this processing is that an image can be linked to a concept by a subjective judgement. To date, humans are only ones who can propose an effective interpretation of an image at this level.

Most of the approaches proposed lie between levels 1 and 2. It is still difficult to lie between levels 2 and 3 that refer to high-level semantic image retrieval [12]. The main effort is to connect low-level features to high-level semantics of images. The effective approaches to date are :

1. using machine learning methods in order to associate high-level concepts to low-level features,
2. taking into account user feedback in order to improve subjective concepts,
3. inferring visual content based on textual information extracted from image context,
4. using entity ontology in order to define high-level concepts.

Most content based image retrieval systems exploit a combination of two or more of these methods in order to perform high-level semantic image retrieval (see [24, 8, 18, 27, 6]). Although the results obtained are promising, designing systems that really understand image content at semantic level is still a open problem.

In this paper, we propose a model to represent a general knowledge about relationships such as spatial ones between entity categories existing in an image database. Furthermore, this model can be used to describe the context of a given image. The definition of image context is discussed in the section 2.2. Finally, we observe that an image may be linked to multiple subjective interactions, then, we hope to attribute a semantic meaning to each context image that can facilitate image retrieval or recognition tasks. Our work falls in the third and fourth categories of approaches listed above. In the

limit of our framework, we do not investigate the learning of concepts using machine learning methods and consider that these concepts are known.

In the section 2, we present the definition of image context and several spatial relationships between categories. The section 3 presents the concepts and definitions of our graph model. In the next, we discuss the evolution capacity of the graph to the new knowledge and spatial reasoning in the section 4 and 5. Finally, in the section 6, we present several experiments to evaluate our graph model.

2 Initial definitions

In this section, we present several definitions like spatial relationships and image context before presenting our principal work.

2.1 Spatial relationship

In our framework, we are firstly interested in the representation of spatial relationships between symbolic objects in images, called entities. In CBIR, embedding such information into image content description provides a better representation of the content as well as new scenarios of interrogation. The spatial relationships can be the unary, binary, and ternary relationships.

We call unary relationship, the relationship between an entity and its localization in an image, where localization is defined as a region or an area of the image. Areas of an image can be represented in different ways like quad-tree or quin-tree, see for example [20, 28]. Since we do not have any knowledge a priori of the location of the categories in the images, we propose to split images in a fixed number of regular areas (i.e. equal size areas). First, we divide each image in a fixed sized grid. Each cell of this grid, called atomic area, is represented by a code. Fig.1 and 2 depict a splitting in 9 or in 16 different basic areas and theirs codes, respectively. We then combine these codes to present more complex areas, by example for 9-area splitting, code 009 represents area () grouping together areas 001() and 008().

001	008	064
002	016	128
004	032	256

FIGURE 1 – Codes in unary relationship by splitting an image in nine areas.

00001	00016	00256	04096
00002	00032	00512	08192
00004	00064	01024	16384
00008	00128	02048	32768

FIGURE 2 – Codes in unary relationship by splitting an image in 16 areas.

A binary relationship links two entities of distinct categories together in an image. In last years,

there have been many approaches proposed for representing binary spatial relationships. They can be classified as topological, directional or distance-based approaches (see [11] for more details), and can be applied on symbolic objects or low level features. Here, we have focussed on relationships between the entities of the database described in terms of directional relationships with approach 9DSpa [17], of topological relationships [9, 10] and of a combination of them with 2D projections [19]. We do not use orthogonal [3] and 9DLT relationship [2] because of its inconveniences mentioned in [17].

A ternary relationship describes a relationship of a triplet of categories. To our knowledge, a few approaches were proposed to describe crisp triangular relationships of three symbolic entities. We can mention TSR approach [13] and our approach Δ-TSR (see [16]). By applying to a set of heterogeneous symbolic entities that do not have fixed shape and size, these approaches cannot described fully triangular spatial relationships between symbolic entities since they take into account only the center of each entity as representation of it.

2.2 Image context

In an image, the recognition or detection of entity category requires different information from the raw image data. According to [26], in the real world, there exists a strong relationship between the environments and entities found within it or between the entities. Entities are never in isolation. They can tend to co-vary with others entities and particular environments for providing a rich collection of contextual associations. The recognition or detection will be accurate and quick if entities usually appear in a familiar background. Then, initially, we can define that the context of an image describes all possible types of relationship between the entities in this image, or between the entities and background of this image. The use of image context can bring a strong interest not only for recognizing or detecting the entity category but also for image retrieval. For the recognition or detection of an entity category, it is evident to examine the general context of image if the local features are insufficient (e.g. entity is small, or appears partially). For image retrieval, the comparison of image contexts can help to filter out the false alarms before enter in the step of comparison of visual contents of images.

By using the visual features in image, the context can be described by relationships between local informations and global information of the image. This context definition can drive to a hard work of image processing. Another natural way of representing the context of an image is using the co-occurrence relationships of its entities. In the real world, the co-occurrence might happen at a global level, for example a bed room will predict a bed, or at a local level, for example a table will predict the presence of a chair. A probabilistic problem can be also associated in this case. More complex, the spatial relationships between entity categories in images can be taken into account. In general, that is difficult to have an exact definition of context ; each case of use can depend on a particular context definition.

Here, we try to study different relationship that could be present in images. According to [14], entities in an image can be THINGS (e.g. car, people) or STUFF (e.g. road, buildings, more precise that are the regions in images). In general, we can have five types of relationship :

- Thing-Thing : co-concurrent relationships, spatial relationships, etc.
- Stuff-Thing : texture regions that allows to predict the present of an entity category.
- Stuff-Stuff : relationships between regions of images.
- Scene-Thing : scene information such as scale, global direction that allows to determine the location of an entity category.
- Scene-Stuff : scene information such as scale, global direction that allows to determine the location of a region.

In our framework, we do not differentiate the entities present in image as "Thing" or as "Stuff" because we are interested in symbolic objects that are represented by polygons. These entities are classified simply by category. We define the image context by the presence of entity categories in image and by the spatial relationships between these entity categories. The presence of at least an instance of an entity category will confirm the presence of this one. The spatial relationships between categories in image will be represented in a general way (*e.g.* probabilities). There are two principal ways of using the context in a vision system :

- A priori : in this way, the context serves to locate the entities, to limit the searching region, and to decrease the retrieval time (for example the approaches proposed by [26, 14, 25]).
- A posteriori : the context serves to object recognition if the local information is not sufficient, it can help to reduce the ambiguities of the presents of objects in the same scene (for example the approaches proposed by [23, 22, 5]).

There has been a growing interest in exploiting contextual information for image retrieval, classification or object detection, recognition. Different techniques have been exploited to describe the context of image for this purpose. Intuitively, the spatial locations of objects and background scene from global view can be used as inside-image context. Further, the combination of object detection and classification tasks together can provide natural comprehensive context for each other without any external assistance. Moreover, a knowledge database, considered as a external element, based on machine learning SVM or probabilistic technique, allows to enhance other tasks as localization, detection, etc. Our approach proposed in the next sections is a priori one.

3 A Graph-based Knowledge Representation

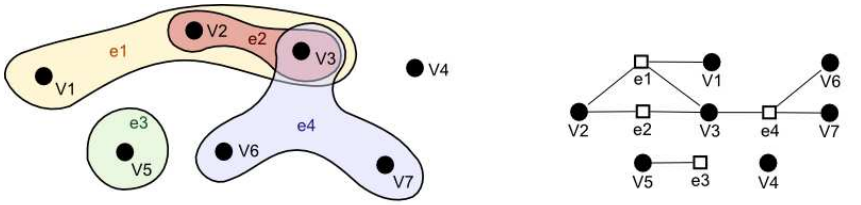
In this section, we present how to represent a knowledge between entity categories in images by using a graph. The concepts and definitions of this graph are presented in section 3.1. To avoid building an unreadable graph, the attributes of node and the graph constraints are discussed in section 3.2 and 3.3 respectively. Finally, section 3.4 presents a brief example of the use of the graph.

3.1 Concept and definitions

In reality, events can be expressed by two notions : entity and relationship. For example, if we have an event "our laboratory invited a professor last month", then this event can be represented by two entities : "our laboratory" and "a professor" that have a relationship "invite" of attribute "last month". We know that a general knowledge presents a general event based on concrete events which happened. For example, based on the previous event, a general event could be formed : "Laboratories

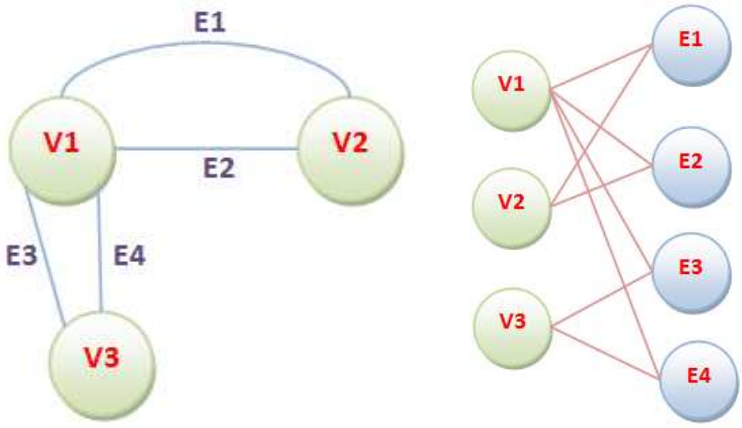
invite professors sometimes". "Laboratory" and "Professor" may be considered as entity categories that may be linked by a relationship of type "invite". Based on this argument, we would like to represent the learnt knowledge by using a graph concept developed only on two notions of "category" and of "relationship". In this graph, the instance of category (an entity) is not meaningful to guarantee a general representation of a knowledge. However, entities are always examined before building a graph. The reason was explained previously : a general event is based on particulars events.

Normally, in a classic graph, a vertex (or a node) represents a category and an edge represents a relationship. We observed that a "relationship" may be unary, binary, ternary or n-nary relationship, then a "relationship" can concern one or many categories. A classic graph can represent only the binary relationships between two vertices. By using a hypergraph, a generalization of a graph [4], an edge can connect any number of vertices (see Fig.3(a)). However, this connection of a set of vertices is represented by only an edge. We know that, between two or more "categories", there are many "relationships". It means that different edges can have the same end nodes, then a multigraph [1] can be another alternative graph. However, a multigraph allows to describe multiple relations between two and only two vertices (see Fig.4(a)). For our concept, we would like to use the advantages of these two graph models, and we know that a bipartite graph [29] can model the more general multigraph and hypergraph (see examples of representation of hypergraph in Fig.3(b) and of multigraph in Fig.4(b)).



(a) Hypergraph (b) Bipartite graph

FIGURE 3 – An example of hypergraph and its bipartite representation.



(a) Multigraph (b) Bipartite graph

FIGURE 4 – An example of multigraph and its bipartite representation.

We would like to present multiple relations between multiple vertices, that is why we decided to represent a relationship by a node. Our graph, denoted G, is a bipartite graph and contains two types

of nodes : a category node, denoted C and a relationship node, denoted R. We give a definition to each type of node in our graph G :

- A category node C represents the existence of a set of categories in a same environment, *i.e.* in the same database. For a set of categories $K = \{cat_i\}$, its representation node is $C_{\{cat_i\}}$ or C_K. C_K can own different attributes describing some information of K such as visual features or a dedicated object detection algorithm.

- Similarly, a relationship node R represents a *type* of relationship between categories in a set $J = \{cat_j\}$, we denote this node R_J^{type}. From a R_J^{type}, we can learn all possible configurations of relationship *type* involved from J. In our framework, we are especially interested in spatial relationships, for example, relationship *type* can be the topological spatial relationship [10] that describes different configurations : disjoint, joint, overlaps, insides , etc.

Now, graph G is defined as :

$$G = (V, E) \tag{1}$$

Knowing that V is the set of nodes in the graph :

$$V = \{C_K\} \cup \{R_J\} \tag{2}$$

This graph is an undirected graph. Then, E is the set of edges :

$$\begin{aligned} E = \quad & \{e_{(C_K,R_J)} | \forall C_K, R_J \in V \wedge K \subseteq J\} \\ & \forall C_K, R_J \in V; e_{(C_K,R_J)} \Rightarrow e_{(R_J,C_K)} \\ & \forall C_K, C_J \in V; \nexists e_{(C_K,C_J)} \\ & \forall R_K, R_J \in V; \nexists e_{(R_K,R_J)} \end{aligned} \tag{3}$$

Figure 5 gives an example of graph (at 3 levels).

3.2 Attributes of a node

On the other hand, our requirement is that graph-based representation must be simple to compute, clear to understand, and extendible to represent a new complex general knowledge. From this question, we define specific attributes associated to each node, category or relationship, in sections 3.2.1 and 3.2.2.

3.2.1 Level attribute

To avoid building an unreadable graph, based on the idea that say that a complex knowledge is developed from a set of basic ones, we split our graph into many different levels. A graph level indicates the number of categories concerned :$|\{cat_i\}|$ meaning cardinality of set $\{cat_i\}$. Thus, each level of graph

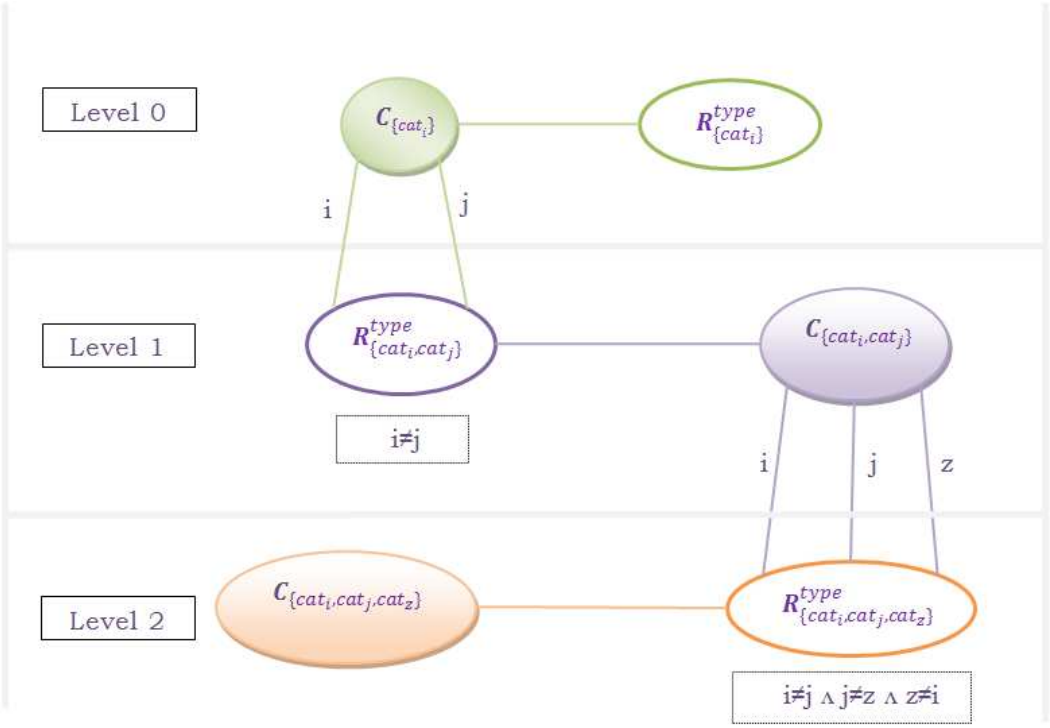


FIGURE 5 – An example of a 3-level graph. The green, violet, and orange colors illustrate nodes at level 0, 1, and 2 respectively.

is composed of a set of C and R nodes that have the same $|cat_i|$. Thus, we can consider the level as an attribute of the node, denoted lev, that is defined as : $lev = |cat_i| - 1$. Our idea is that a higher level node can be built from lower level nodes. A low level node can connect only to one higher level by edges between C nodes of lower level and R nodes of higher level.

In consequence, we redefine set E :

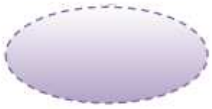
$$E = \{e_{(C_K,R_J)} | \forall C_K, R_J \in V \wedge K \subseteq J \wedge (C.lev = R.lev \vee C.lev + 1 = R.lev)\} \quad (4)$$

Note that we can expand the number of levels in the graph as we need. If we study N categories, then at the level l, we can have in maximum $\frac{N!}{(N-l)!}$ nodes C. A high number of levels can increase considerably the number of nodes in the graph, however, in reality, we can find many categories that never occur together. For example, in [15], by studying 86 different entity categories in a dataset of heterogeneous content, we found 879 couples of categories that never occur together among a set 7310 of possible couple and only 38031 present triplets in total among 102340 possible triplets.

In Fig.5, we show the concept for a 3-level graph. Concretely, we can model unary, binary, and ternary relationships with this graph.

3.2.2 Status attribute

We know that a knowledge can be either true, or false, or uncertain. In fact, for example, we are not certain to confirm the existence of aliens because of the limit of our scientific knowledge. "The earth is a square" is a false definition. In addition, a true confirmation could become a false one ; for example, nowadays the confirmation "the earth is a center of universe" is not true. To complete the graph conception, we propose to add an attribute on such status for each node of graph, denoted *stat*. A node in graph can be either TRUE, or FALSE, or UNCERTAIN. These statuses can be modified to deal with a new situation if necessary. Figs. 5, 8, and 9 show examples of TRUE nodes. To represent a UNCERTAIN node, we use a discontinuous line node as shown in Fig.6(a). Finally a FALSE node is represented by a strike-through node as shown in Fig.6(b). Note that FALSE and TRUE statuses are considered as confirmations, though, UNCERTAIN status will be used as induction that is not verified yet.



(a) Uncertain node representation

(b) False node representation

FIGURE 6 – Two additional statuses of a node.

A C node must own only one status at one time in the graph. In fact, for example, it is impossible that two confirmations : TRUE and FALSE about one category are present in parallel in the same graph. However, we know that a C node can be connected to many R nodes. A R is assigned to a set of categories J and has *type*, each R node determines a set of configurations Φ that are learnt from J. When adding *stat* attribute to R node, one remark is that three R nodes can own three different statuses but the same J and *type* can present in parallel to complete a knowledge of a *type* relationship studied from J but for different configurations. We denote the set of configurations in R_J^{type} : $R_J^{type}.\Phi$. Let $type.\Phi$ be the set of all possible configurations of the relationship *type*. In consequence :

$$R^{type}_{J|stat=true}.\Phi \cup R^{type}_{J|stat=uncertain}.\Phi \cup R^{type}_{J|stat=false}.\Phi \subseteq type.\Phi \tag{5}$$

And :

$$R^{type}_{J|stat=true}.\Phi \cap R^{type}_{J|stat=uncertain}.\Phi \cap R^{type}_{J|stat=false}.\Phi = \varnothing \tag{6}$$

Note that a configuration of relationship *type* belongs to only one R node among three different status R nodes of J and *type* (see Fig.7 for an example).

By adding a status to a node, there are two possible scenarios to represent a knowledge base :

– We begin our knowledge representation with a graph containing all possible combinations on categories and on relationships of universe (i.e. there is no limitation on the number of entity categories and of types of relationship). It is evident that these nodes are uncertain initially. Initially, the graph is very huge. The studying of several content databases will confirm or deny

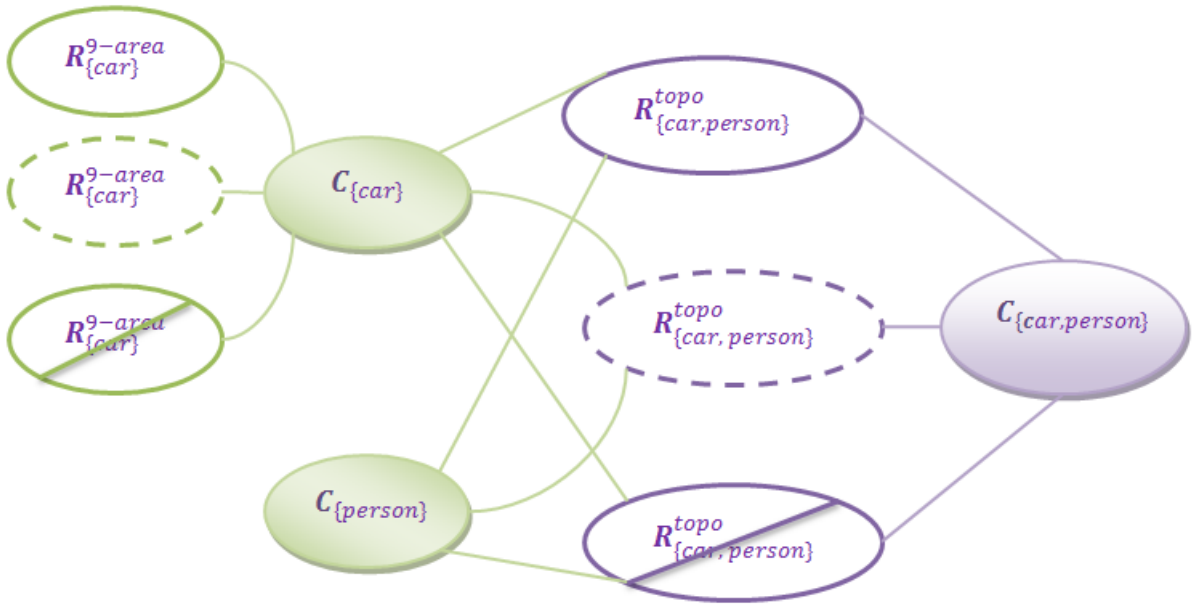


FIGURE 7 – Three different statuses of R nodes owning the same *type* and set J.

these nodes. But this scenario is complex and will not be taken into account in our framework.

– A knowledge graph is built directly from a given database. An initial graph contains only the true or false nodes. The number of categories and the types of relationship are limited. This graph evolute by adding new knowledge. Our work in the following will respect this scenario.

3.3 Graph constraints

Here another challenge is that the comparison of two graphs may not involve a costly computation. To avoid to complicate the graph that can impact on graph computation such as comparison, some constraints must be associated to it, on nodes (section 3.3.1), on statuses (section 3.3.2) and on edges (section 3.3.3).

3.3.1 Node constraints

The presence of a set of entity categories is represented by one C node only. Thus, a C node must be unique. It can be identified with a unique identification, denoted id and computed by function of equation 7 : $C_K.id = F_C^{ID}(K)$. Note that a cat_i can be identified by a unique id (that is an integer) and that N^{Categ} is the total number of categories examined in system. The id attribute of the C node is an integer computed from the id of $cat_i | cat_i \in K$. It will allow to locate quickly a C node in the graph by using a hash function.

$$F_C^{ID}(K) = \begin{array}{l} \sum_{i=1}^{|K|} cat_i.id * (N^{Categ})^{(i-1)} \\ \forall cat_i, cat_j \in K; i < j \Rightarrow cat_i.id < cat_j.id \end{array} \quad (7)$$

In consequence, we obtain : $\forall C_K \in V : \nexists C_J \in V | C_K.id = C_J.id$.

Similarly, a R_J node must be unique also. Its uniqueness is represented by a unique *id*. Differently to the *id* of the C node, this *id* is defined by three elements : set J of categories, *type* of relationship, and status *stat* of a node. It is computed by the function in equation 8 :

$$F_R^{ID}(K) = \quad type + toString(stat) + toString(F_C^{ID}(K)) \quad (8)$$

where $toString(s)$ allows to transform a data type (boolean, number, etc.) to a string. Then the *id* of a R node is a string that is unique. Thus, we obtain :

$$\begin{array}{c} \forall R_J, R_K \in V : (J \neq K) \vee (J = K \wedge R_J.type \neq R_K.type) \\ \vee (J = K \wedge R_J.type = R_K.type \wedge R_K.stat \neq R_J.stat) \end{array} \quad (9)$$

It is evident that, for a type of relationship, there may be many different R nodes. It means there are many instances of R^{type} for a given *type*. Nowadays, a computer can have a large memory, storage optimization is not the main subject. It is not necessary to look for a way to associate many C nodes to a R of given *type* efficiently. We address the computation optimization problem in the graph (e.g. comparison, node matching, etc.). In a large graph, it is always easy and quick to find a node by its *id*.

In our graph, one lower level is connected to only one next higher level. There is another important constraint on the link between R nodes and C nodes. Given a R node at level l, R is always connected to $l+1$ nodes C that are at level $l-1$, and is linked to only one C node at level l (see Fig.5).

3.3.2 Status constraints

To be able to compare the three statuses, we impose that TRUE>UNCERTAIN>FALSE. Finally, to avoid absurd representations in the graph, we propose three constraints concerning *stat* definition :

- The first constraint concerns the C and R nodes that are connected and at a same level :

$$\forall C_K, R_J \in V : (I = J \Rightarrow C_K.stat >= R_J.stat) \quad (10)$$

  Note that : $I = J$ can be replaced by $(\exists e_{(C_K,R_J)} \in E \wedge C_K.lev = R_J.lev)$.
- The second constraint concerns the lower level C node and the higher level R node that are connected :

$$\forall C_K, R_J \in V : I \subset J \Rightarrow C_K.stat >= R_J.stat \quad (11)$$

Here note that : $I \subset J$ can be replaced by $(\exists e_{(C_K,R_J)} \in E \wedge C_K.lev = R_j.lev - 1)$.

– The third constraint concerns two C nodes of two different levels :

$$\forall C_K, C_J \in V : I \subset J \Rightarrow C_K.stat >= C_J.stat \tag{12}$$

We can observe that the status of a category node plays an important role. It is certain that the status of a R node depends on the status of a C node. We cannot deny that the non-existence of a C node will reject all R nodes linked with it. For example, we can say that there are many researches concerning relationship between ALIEN category and EARTH category ; and one day, if we confirm that this ALIEN category do not exist (this category has a FALSE status), then all associated relationships become false. Furthermore, it is also evident that the status of a higher level node depends on the status of a lower level node. For example, if we confirm that in an environment, there is no A category ($C_{\{A\}}.stat$ is FALSE), then it is impossible for a couple of categories (A, B) to be present in this environment (thus, $C_{\{A,B\}}.stat$ becomes FALSE too).

3.3.3 Edge constraints

We impose two constraints for the edges between C nodes and R nodes :

– The first constraint is for two nodes of same level. Node C_K can be linked to node R_J at the same level if and only if they are assigned to the same set of categories, it means too $K = J$:

$$\forall R_J \in V : \exists C_K \in V | C_K.lev = R_J.lev \wedge K = J \wedge \exists e_{(C_K,R_J)} \in E \tag{13}$$

– The second constraint is for two nodes of different levels. A node C_K of a lower level can be linked to a node R_J of a next higher level if and only if $K \subset J$:

$$\forall R_J, C_K \in V | (R_J.lev = C_K.lev + 1) \wedge (K \subset J) : \exists e_{(C_K,R_J)} \in E \tag{14}$$

This last constraint confirms the fact that one lower level is connected to only one next higher level.

3.4 Examples

To make clear the model of our graph, we propose two examples in Fig. 8 and 9. In the first example, we describe spatial binary relationship, represented with the topological approach [9, 10] (see section 2.1), between two different categories : CAR and PERSON. We show the possibility of association of many R nodes to one C node. In the second example, the presence of triplet (CAR,PERSON,BUILDING) is confirmed by a presence of three couples of categories in a lower level that have a spatial ternary relationship, represented with the approach Δ-TSR [16](see section 2.1), between them. In general, if the number of categories is small, the graph is simple. However, the graph can evolve dynamically, supplement nodes can be added any time when a new knowledge is learnt.

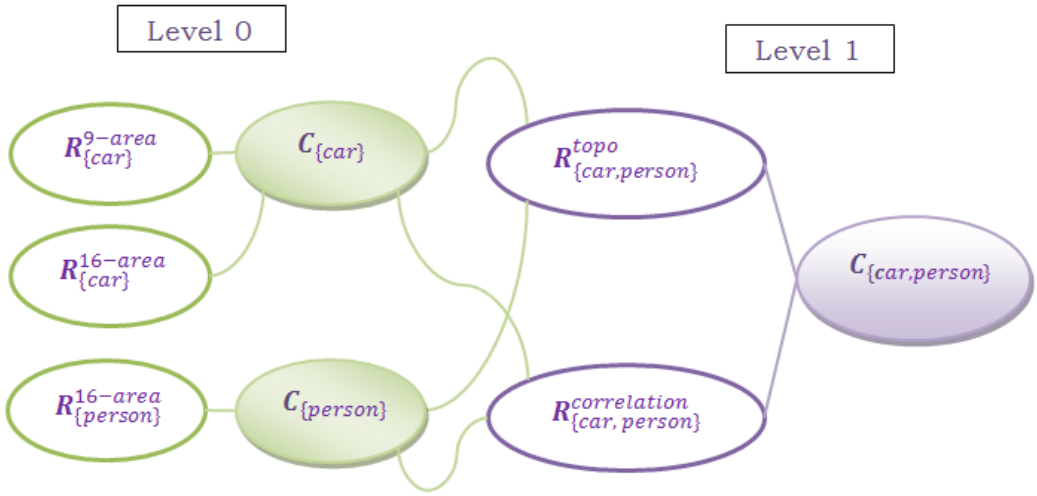


FIGURE 8 – Example of knowledge representation of relationships of two categories in two first levels by using different relationships : 9-area or 16-area relationships for unary relationships on location in the image, topological relationships or correlation for binary relationships.

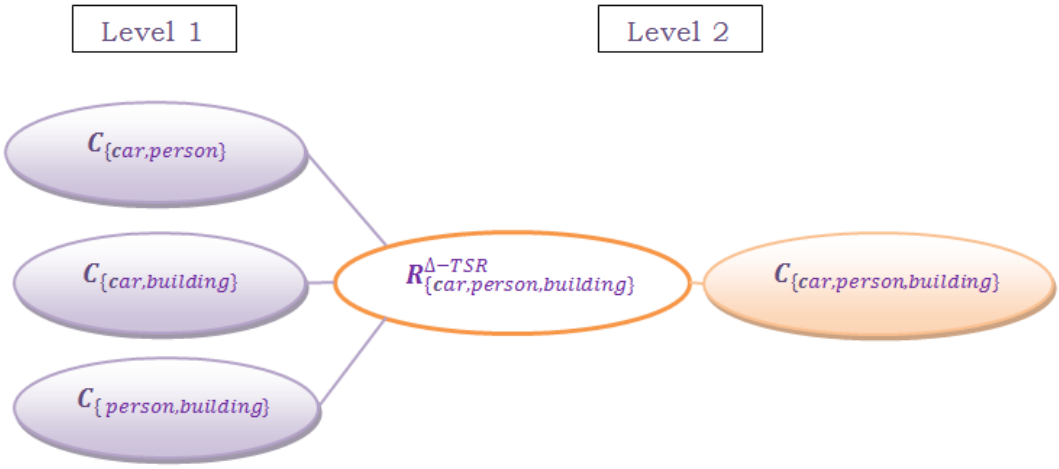


FIGURE 9 – Example of knowledge representation of three categories at level 2 by using Δ-TSR relationships that represent ternary relationships between entities.

4 Management of the graph

In this section, we present several possible manipulations on our graph-based model. The evolution capacity of the graph to the new knowledge is represented in section 4.1. Our graph-based model can be used also to describe the context of an image. This application will be presented in section 4.2. The advantage of this representation is to allow to accelerate the similarity computation of two images by comparing their graphs.

4.1 Evolution of the graph

Section 4.1.1 discuss the operation to update the node status. Section 4.1.2 present how to infer the new knowledge from an existing one.

4.1.1 Update of node status

It is evident that a knew knowledge may change the status of a node. This modification can have some consequences. Here, we say that a node degrades its status while the value of its status is decreased ; on the contrary, a node upgrades its status. For example, a C_K node degrades its status from TRUE downto FALSE and then upgrades its status FALSE upto UNCERTAIN. Here, we denote DG and UG the operations of status degradation/upgradation of a node respectively. There are two scenarios of status modification, one for C nodes and one for R nodes :

- While a C node changes its status, other nodes may change their statuses too in order to respect the three constraints of equations 10, 11, 12 p.11, 11, 12. These nodes can be R or C nodes. There are two cases :

  1. The first case is when a node C_K realises a DG. Then, every R node connected with it and having a status higher than the new status of C_K must undergo a DG. Every C node at the next level must undergo a DG too. Algorithm DG is defined as in Algo.1 p.15. Note that while a R node degrades/upgrades its status, it is probable to have two R nodes owning the same id, then it is necessary to merge these two R nodes into one.

  2. The second case is when a node C_K realises a UG. Here, we can have a paradox while a higher level C node upgrades its status to a status higher than lower level C node's one. To respect the imposed constraints, should the lower level C nodes upgrade their statuses too ? We examine an example below. Let suppose that in an given environment, an object detector cannot identify the existence of category A but confirms that B exists. Then, we obtain FALSE nodes A and (A, B) in the graph. Then, we see with our own eyes in this environment the existence of couple (A, B) together. Consequently, A must exists in examined environment. Thus, a new confirmation can reject an old confirmation. However, if we say (A, B) maybe exist together, that is not a confirmation, then this uncertain opinion cannot change the non-existence of A. From this example, we impose that a C node can upgrade its status only to TRUE status or to the lowest status of lower level C nodes concerning it. This UG operation is defined in Algo.2 p.15.

- On the contrary, we think that status modification of R node does not change the presence of C node. Therefore, a R node can undergo a UG/DG of its status such as its new status respects the three constraints of section 3.2.2. It means a R node cannot upgrade its status to a new status that is higher than one of any C node connected with it. Furthermore, these operations will not impact the status of any C nodes in the graph.

In summary, only the status modification of a C node can impact the status of every R or C nodes of lower/higher levels having a link with it. We can develop a recursive status modification of some

Data: C_K, *newStatus*
Result: downgrade $C_K.stat$ downto *newStatus*
$C_K.stat \longleftarrow newStatus$
for $(R_J/(R_J.lev = C_K.lev \vee R_J.lev = C_K.lev + 1) \wedge \exists e(C_K, R_J))$ **do**
  if $(R_J.stat > C_K.stat)$ **then**
    $R_J.stat \longleftarrow newStatus$
    if $(\exists R_K | K = J \wedge R_K.type = R_J.type \wedge R_K.stat = R_J.stat)$ **then**
      $R_J \longleftarrow merge(R_J, R_K)$
    end
  end
end
for $(C_Z/(Z \supset I)$ **do**
  if $(C_Z.stat > C_K.stat)$ **then**
    $DG(C_Z, newStatus)$
  end
end

Algorithm 1: *DG* degradation algorithm of C_K node.

Data: C_K, *newStatus*
Result: upgrade $C_K.stat$ upto *newStatus*
$lowestStatus \longleftarrow$FALSE.*val*
for $(C_J/(J \subset I)$ **do**
  if $(C_J.stat < lowestStatus)$ **then**
    $lowestStatus \longleftarrow C_J.stat$
  end
end
if $(newStatus \neq$TRUE.*val* $\wedge\ newStatus > lowestStatus)$ **then**
  return ;
end
$C_K.stat \longleftarrow newStatus$
if $(newStatus > lowestStatus)$ **then** // *newStatus* =TRUE.*val*
  for $(C_Z/(Z \subset I) \wedge (C_Z.stat \neq newStatus))$ **do**
    $UG(C_Z, newStatus)$
  end
end

Algorithm 2: *UG* upgradation algorithm of C_K node.

nodes from a C node status change.

Let us study the complexity of each operation. Let N^R be the average number of R nodes connected with one C node. Let N^C be the average number of C node connected with one C node by a R node. Let N^L be the total number of level in the graph. An upgradation operation of a C node at level l impacts only the C nodes of lower levels, thus, the complexity of upgradation is $O((N^C)^{(l-1)})$. On the contrary, an degradation operation of a C node at a level l impacts the C and R nodes of higher levels. Its complexity can reach $O((N^R)^{(N^L-l)} + (N^C)^{(N^L-l-1)})$.

Let us examine the example of Figure 7 p.10. Suppose that the C_{car} node degrades its status (see Fig.10 p.16) :

1. C_{car} node degrades its status to UNCERTAIN.
2. Consequently, respecting constraints on the status conducts that every TRUE R/C nodes concerned by CAR must change their statuses to UNCERTAIN.
3. A TRUE R node downgrades its status, it will merge to existing UNCERTAIN ones.

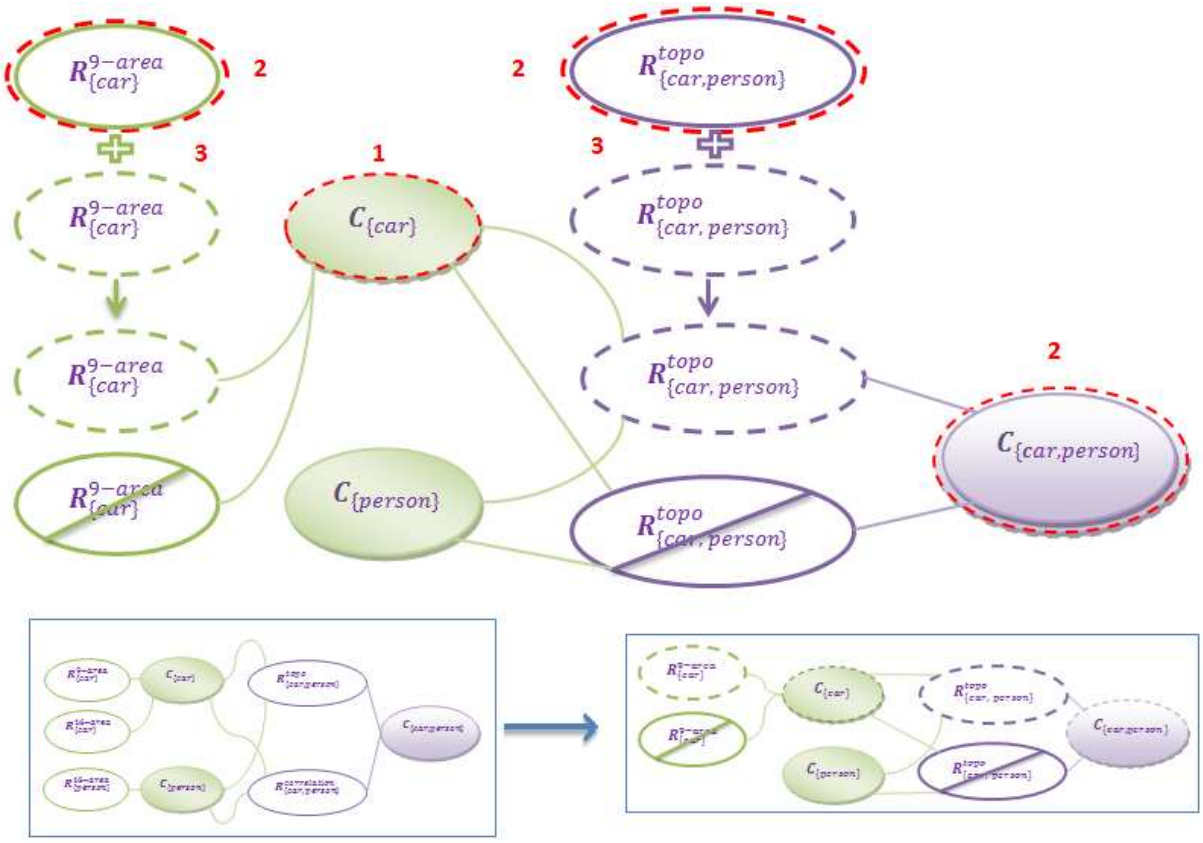


FIGURE 10 – The initial graph is presented in Fig.7. C node assigned to CAR category degrades its status downto UNCERTAIN.

In fact, if we are uncertain about the presence of CAR, we cannot be sure about the presence of couple CAR-PERSON. Further, every relationships concerning CAR are not certain too or do not exist. On the contrary, a higher level C node cannot impact the lower level C node when it degrades its status as we can see with the example of Fig.11 p.17.

We present another example on impact of UG operation of a C node. From the initial situation in Fig.12(a), a status upgradation of higher level C node can modify the status of lower level C node as shown in Fig.12(b). Note that, according to the above constraints, a higher level C cannot upgrade to UNCERTAIN if it exists at least one FALSE lower level C node concerned to this higher one. The UG operation of C node will not impact R nodes and higher level C nodes.

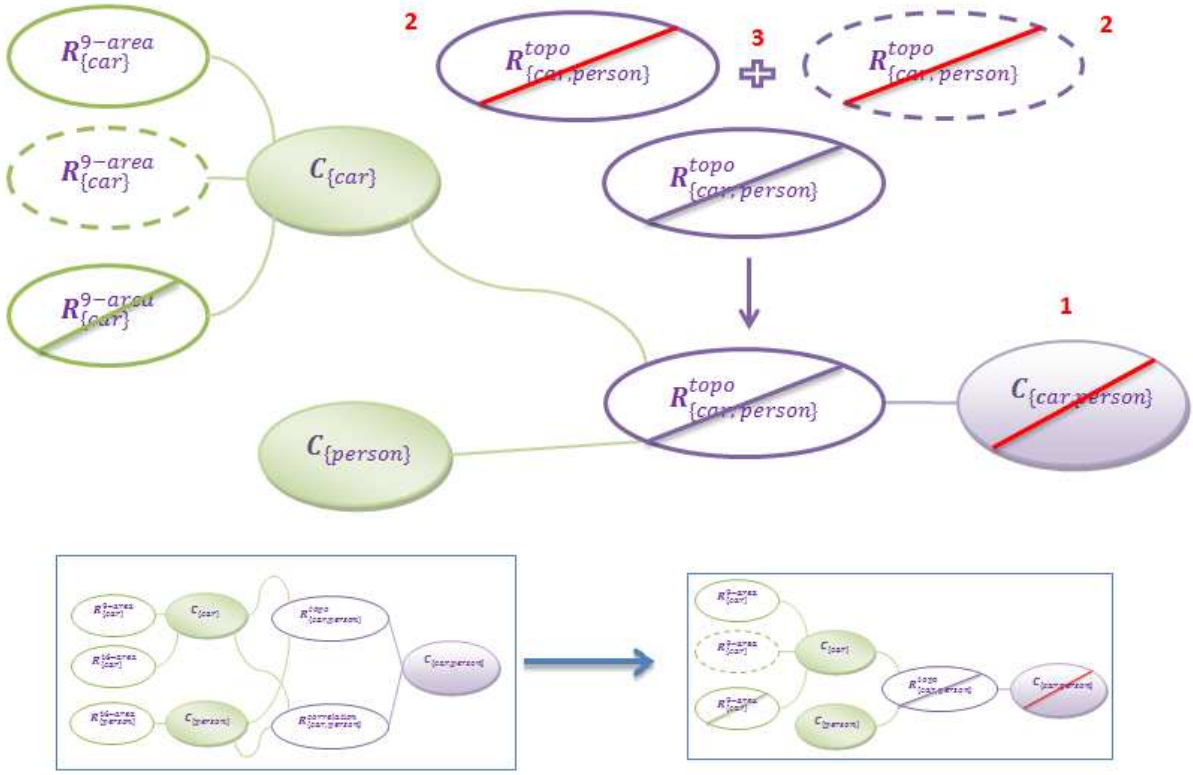


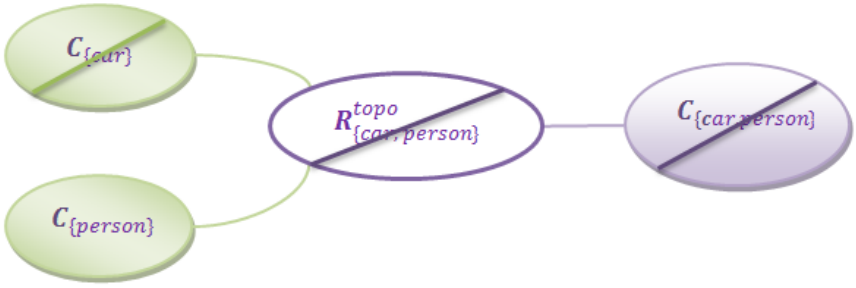
FIGURE 11 – The initial graph is presented in Fig.7. C node assigned to (CAR-PERSON) degrades its status downto FALSE.

4.1.2 Inference knowledge

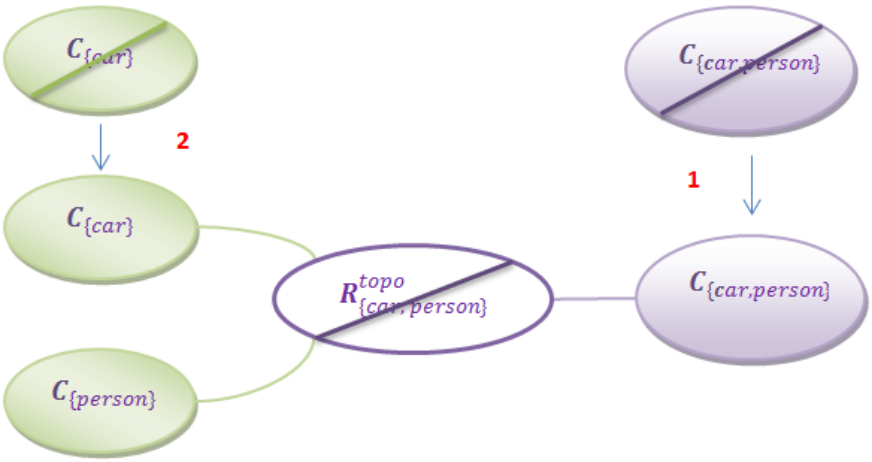
The first utility of our graph is to represent a set of knowledges on entity categories and on their relationships. From an image database, after the analysis of its visual content, we can construct such knowledge graph. This graph can help us to save, to organize, and to infer all statistical informations on the trends of spatial relationships involving entity categories effectively encountered in the database, with the aim of exploiting them in future CBIR applications, for improving tasks such as object recognition or retrieval.

To infer new knowledge from an existing one, several rules can be defined :

- A TRUE higher level C node can implicate TRUE lower level C nodes assigned to a subset of its set of categories. For relationship *type* between these C nodes, we can add an UNCERTAIN R node containing all configurations of *type* that are not yet represented. Fig.13 illustrates this rule. Let K be the set of categories represented by C. Let l the level of C. The complexity of this inference is $\prod_{i=0}^{l} \frac{(|K|)!}{(|K|-i)!}$.
- We can infer the relationship between two TRUE C nodes based on confirmed relationships between these two C nodes and the third intermediate TRUE C node. Note that for UNCERTAIN R node, we can associate each configuration to some probabilities or to a expected values. Fig.14 illustrates this rule. Here, to avoid complicate structure graph and useless knowledge, we would not like to introduce an implication based on existing UNCERTAIN nodes. It means that we cannot



(a) Initial graph



(b) Impact of status modification of a higher level C node on lower level C node.

FIGURE 12 – An example on relationships between two categories CAR-PERSON, and the impact of a status modification. Relationships are spatial binary relationships described with a topological model.

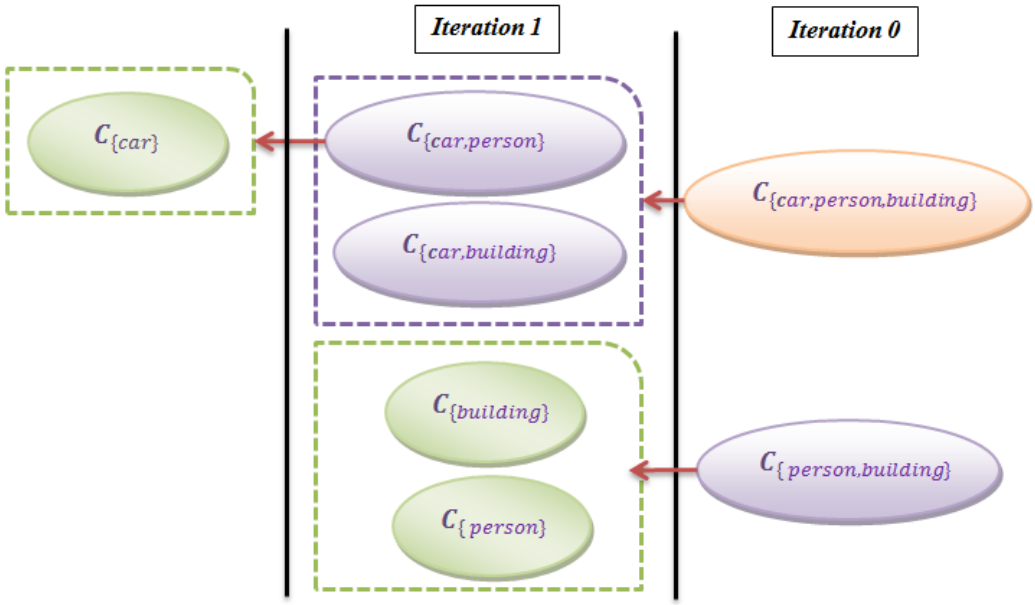
do a recursive implication, thus, the complexity of this inference is $O(1)$.

4.2 Graph-based representation of image context

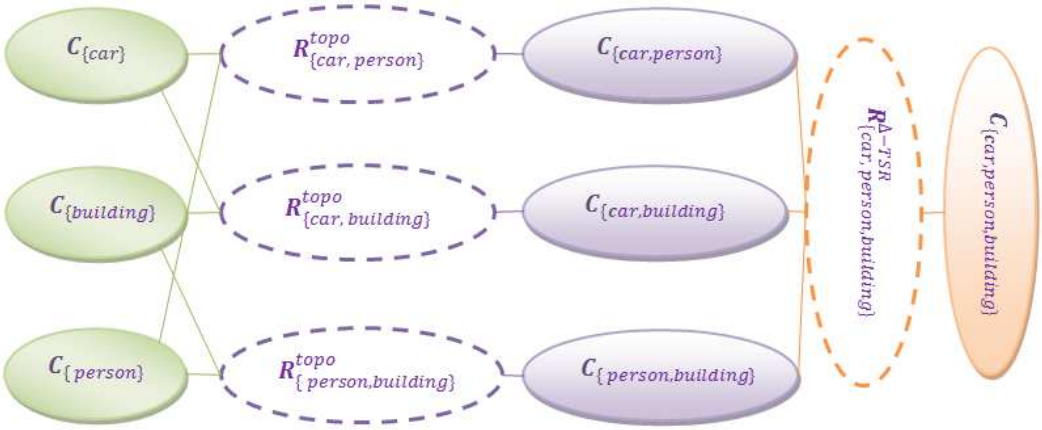
In section 2.1, we have defined the context of an image as the representation of every categories in this image and of their spatial relationships. Thus, the graph concept defined in previous sections can be applied to describe the context of an image. For example, the context of the image in Fig.15(a) can be represented by the graph shown in Fig.16. In this image, four entity categories : SKY, SUN, SEA, MOUNTAIN are detected. Suppose that we study only topological relationships. Note that the existence of the category nodes in level 1 allows to extend this graph to a higher level by studying ternary relationships. In this section, we explain how to compare the context of two images based on graph representation.

Section 4.2.1 defines the similarity between 2 category nodes, section 4.2.2 defines the one between 2 relationship nodes while section 4.2.3 defines the one between 2 graphs.

(a) Initial graph



(b) Inferences at step 1 : inferences of the C nodes

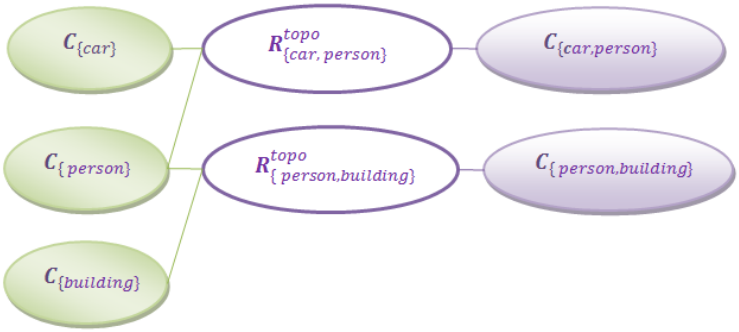


(c) Inferences at step 2 : inferences of the R nodes

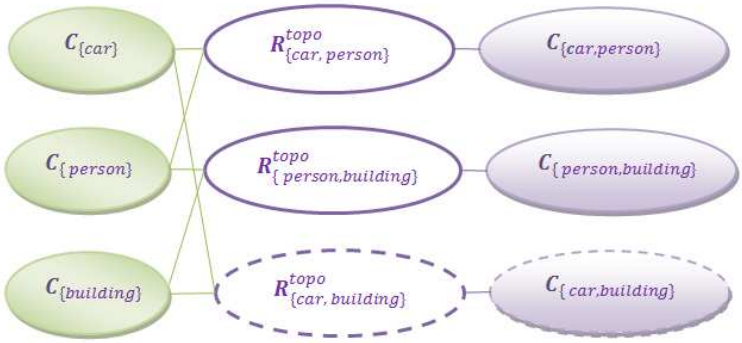
FIGURE 13 – Example of automatic inferences in the graph. The discontinuous line rectangles represent the sets of new nodes inferred from existing nodes.

4.2.1 Similarity between two category nodes

We can say that two similar contexts necessarily contain common entity categories. Therefore, firstly, to evaluate the similarity of context of images I_i and I_j, we evaluate the similarity of their entity categories.

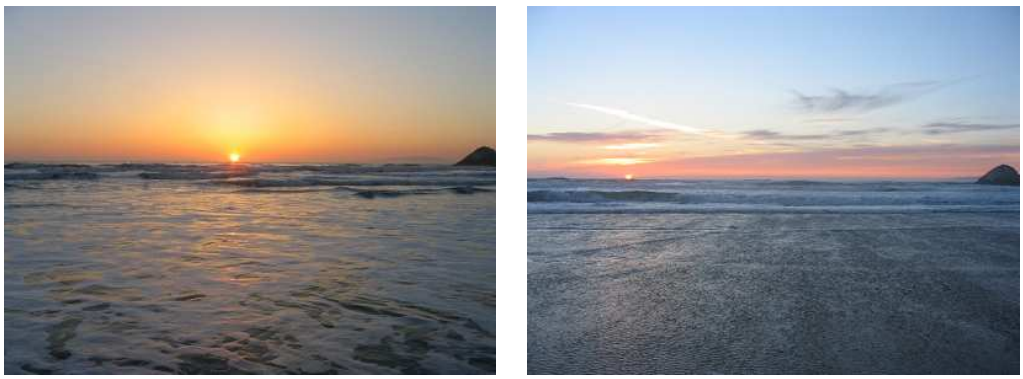


(a) Initial graph



(b) Inference of new uncertain nodes.

FIGURE 14 – Example of automatic inferences in the graph.



(a) (b)

FIGURE 15 – Example of two images that may have similar contexts in terms of entity categories and of topological relationships.

G_i, G_j are the graphs of I_i, I_j respectively. We denote $SC(G_i, G_j)$ the set of matched category couples that are in common between I_i and I_j : $SC(G_i, G_j) = \{(C_K^{G_i}, C_K^{G_j})\}$. We split this set in two subset : a set of (TRUE, FALSE) confirmed couples (SC^{TF})and a set of uncertain couples (SC^U). Our idea is that the computation of the similarity between two graphs is based on the weighted evaluation of these two sets. Let :

$$SC^U(G_i, G_j) = \{(C_K^{G_i}, C_K^{G_j}) | C_K^{G_i}.stat = \text{UNCERTAIN} \vee C_K^{G_j}.stat = \text{UNCERTAIN}\}$$
$$SC^{TF}(G_i, G_j) = \{(C_K^{G_i}, C_K^{G_j}) | C_K^{G_i}.stat \neq \text{UNCERTAIN} \wedge C_K^{G_j}.stat \neq \text{UNCERTAIN}\}$$

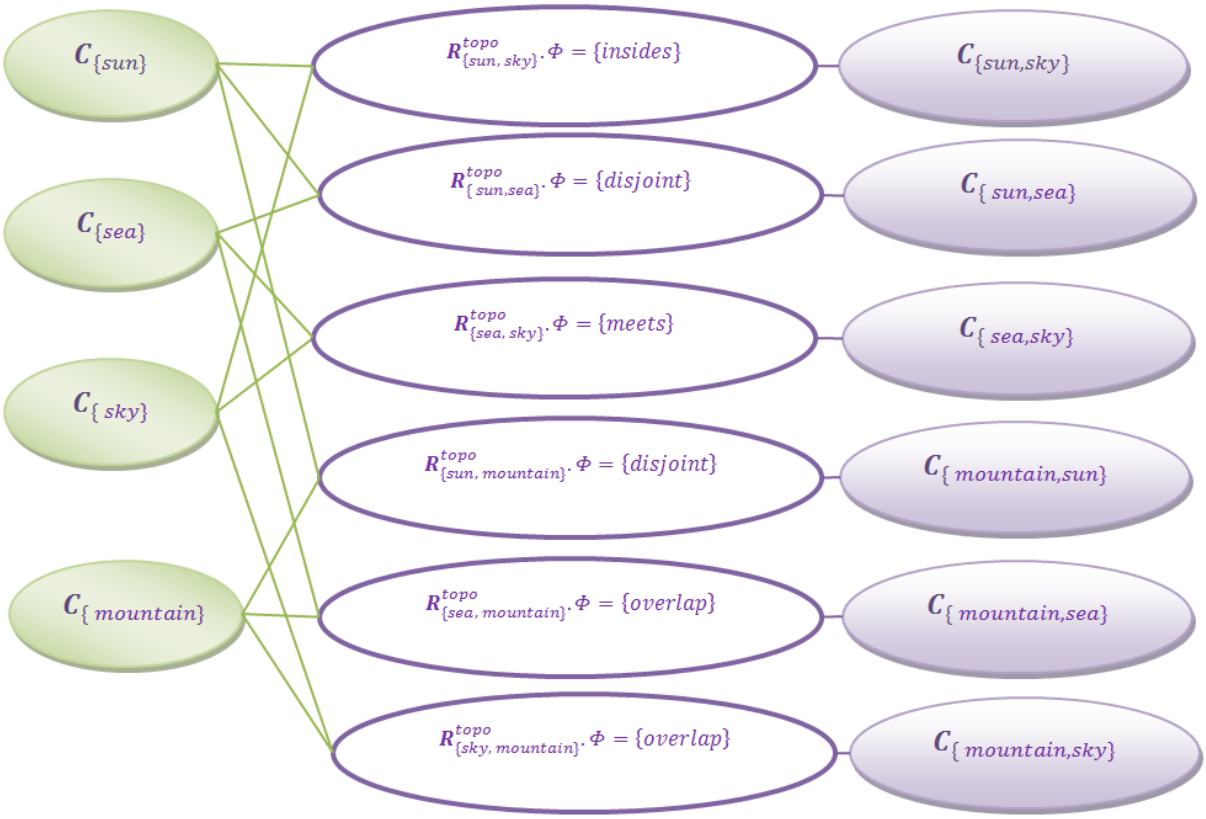


FIGURE 16 – Image context representation by a graph for images of Fig.15.

Thus, $SC(G_i, G_j) = SC^U(G_i, G_j) \cup SC^{TF}(G_i, G_j)$. Let $u = |SC^U(G_i, G_j)|$, $tf = |SC^{TF}(G_i, G_j)$. Then, $u + tf = |SC(G_i, G_j)|$, it is also the number of nodes C in each graph that participate to the matching. We call $sim^{SC}(G_i, G_j)$ the category similarity function of two graphs, that can be defined as :

$$sim^{SC}(G_i, G_j) = \frac{2 \times (u + tf)}{(|G_i.\{C_K\}| + |G_j.\{C_J\}|)} \times e^{CN} \tag{15}$$

Knowing that : $e^{CN} = p^U \times \frac{\sum_{k=1}^{u} sim^{CN}(SC_k^U(G_i,G_j))}{u} + (1 - p^U) \times \frac{\sum_{k=1}^{tf} sim^{CN}(SC_k^{TF}(G_i,G_j))}{tf}$

where :

- $SC_k^X(G_i, G_j)|X \in \{U, TF\}$ is the k^{th} couple of entity categories of $SC^X(G_i, G_j)$,
- $G_i.\{C_K\}$ the set of the C nodes in G_i,
- p^U is weight parameter on evaluation of UNCERTAIN part.

$sim^{CN}(SC_k^X(G_i, G_j)$ varies in [0..1], it evaluates the similarity of couples SC_k, then $sim^{SC}(G_i, G_j)$ also varies in [0..1]. $sim^{SC}(G_i, G_j) = 1$ when two images contains the same set of entity categories. If $\sum_{k=1}^{|SC(G_i,G_j)|} sim^{CN}(SC_k(G_i, G_j)) = |G_j.\{C_J\}|$, then I_i contains all of categories of I_j. Here, we can define the similarity of two C nodes in two graph as :

$$sim^{CN}(C_K^{G_i}, C_J^{G_j}) = \begin{cases} 0 \text{ if } K \neq J, \text{ otherwise} \\ sim(C_K.stat, C_J.stat) \end{cases} \tag{16}$$

$sim(C_K.stat, C_J.stat)$ evaluates the similarity of status of C_K and C_J.

For example, $sim(C_K.stat, C_J.stat) = 1$ when $C_K.stat = C_J.stat$ or $sim(C_K.stat, C_J.stat) = 0$ when $C_K.stat = true \wedge C_J.stat = false$.

However, if we take into account other attributes for category node in each graph, in this case, we have to consider the comparison of image visual contents also. We can redefine the sim^{CN} as :

$$sim^{CN}(C_K^{G_i}, C_J^{G_j}) = \begin{cases} 0 \text{ if } K \neq J, \text{ otherwise} \\ \frac{\sum_k^{|SetOfAttributes|} sim(C_K.attribute_k, C_J.attribute_k)}{|SetOfAttributes|} \end{cases} \tag{17}$$

where $C_K.attribute_k$ is the k^{th} attribute of C_K and $sim(C_K.attribute_k, C_J.attribute_k)$ the similarity of the k^{th} attribute between C_K and C_J. These attributes are status, furthermore, they can be the size, shape, or color histogram of I (in this case, the similarity is computed from the intersection of two histograms of two nodes or any usual description of graph).

The matching of category nodes in two graphs is not complex, it can be realized based on their *id* and by using a hash table. This operation has a complexity of $O(N)$ with $N = min(|G_i.\{C_K\}|, |G_j.\{C_J\}|)$.

4.2.2 Similarity between two relationship nodes

The comparison of context of two images can be studied deeper by taking into account R nodes in the two graphs of these images. Then, similarly to the comparison of C nodes in the two graphs, we define the similarity function to compare R nodes. We denote $SR(G_i, G_j)$ the set of matched relationship node couples from I_i and I_j : $SR(G_i, G_j) = \{(R_K^{G_i}, R_K^{G_j}) | R_K.type = R_K.type\}$. Here, because of the existence of different statuses of the R node, we impose a constraint of matching :

- a TRUE R_K^{type} node in G_i can be matched with a TRUE R_K^{type} node in G_j,
- a FALSE R_K^{type} node in G_i can be matched with a FALSE R_K^{type} node in G_j,
- a UNCERTAIN R_K^{type} node in G_i can be matched with any R_K^{type} node in G_j.

In the same way, we can split $SR(G_i, G_j)$ in two subsets $SR^U(G_i, G_j)$ and $SR^{TF}(G_i, G_j)$. Let NR^{G_i} the number of R nodes in G_i and NR^{G_j} the number of R nodes in G_j that participate to the matching. Consequently, we call $sim^{SR}(G_i, G_j)$ the relationship similarity function of two graphs that can be defined as :

$$sim^{SR}(G_i, G_j) = \frac{NR^{G_i} + NR^{G_j}}{|G_i.\{R_K\}| + |G_j.\{R_J\}|} \times e^{RN} \tag{18}$$

Knowing that : $e^{RN} = p^U \times \frac{\sum_{k=1}^{u} sim^{RN}(SR_k^U(G_i,G_j))}{u} + (1-p^U) \times \frac{\sum_{k=1}^{tf} sim^{RN}(SR_k^{TF}(G_i,G_j))}{tf}$

$sim^{RN}(SR_k(G_i, G_j)$ is the function that allows to evaluate the similarity of two R nodes. Then, we can take into account any information from these nodes such as frequency of each $conf \in \Phi$.

4.2.3 Similarity between two image graphs

Finally, we can evaluate the similarity of two image graph based on the similarity of category nodes and relationship nodes. We call $SIM(G_i, G_j)$ the function evaluating similarity between two graphs. Then :

$$SIM(G_i, G_j) = p \times sim^{SC}(G_i, G_j) + (1-p) \times sim^{SR}(G_i, G_j) \tag{19}$$

Knowing that p is a weight parameter and SIM varies in [0..1].

For example, we compare the contexts of the two images from Fig.15. Suppose that we study only the topological spatial relationships and choose $p = 0.5$ (equation 19) to give equal importance to categories and relationships. The context representation of these two images can be seen in Fig.16. We obtain $simSC(G_i, G_j) = 1$ and $simSR(G_i, G_j) = 1$. Finally, $SIM(G_i, G_j) = 1$, we can say these two images have the same context.

We can add other attributes for category nodes in each graph, for example information on size to each category node (SKY, SUN, SEA, MOUNTAIN). In the first image, SKY and SEA occupy around 39.2% and 58.3% of image respectively. In the second image, they occupy around 45.6% and 54.2% respectively. Suppose that we define $sim^A(C_I.size, C_J.size)$ as :

$$sim(C_I.size, C_J.size) = 1 - \frac{2 \times |C_I.size - C_J.size|}{C_I.size + C_J.size} \tag{20}$$

Symbol	Meaning
C	category node
R	relationship node
lev	level of node
$stat$	status of node
$type$	type of relationship
Φ	set of configurations
$F_C^{ID}(K)$	function computing the id of the C node from a set of entity categories
$F_R^{ID}(K)$	function computing the id of the R node from a set of entity categories
UG/DG	upgrade/downgrade the status of a node
$SC(G_i, G_j)$	set of matched category couples from graphs G_i and G_j
$SC(G_i, G_j)$	set of matched relationship couples from graphs G_i and G_j
$sim^{CN}(G_i, G_j)$	similarity between two C nodes from graphs G_i and G_j (section 4.2.1)
$sim^{SR}(G_i, G_j)$	similarity between two R nodes from graphs G_i and G_j (section 4.2.2)
$SIM(G_i, G_j)$	similarity between two image graphs (section 4.2.3)
G_K	knowledge graph

TABLE 1 – Meaning of symbols used.

Consequently, we obtain $sim^{SC}(G_i, G_j) \simeq 0.93$. We can find too a different score if we add other spatial relationships like $9DSpa$ which describe directional relationships [17].

5 Spatial reasoning

In this section, we present two ways for automatically building an image context graph by using the knowledge graph : graph building based only on the knowledge graph in section 5.1 and graph building based on the knowledge graph and the annotation of image in section 5.2. An overview of these main algorithms proposed is presented in section 5.3. From this strategy, several scenarios of query based on the image context graph can be applied, we present them in section 5.4.

5.1 Image context graph building based only on knowledge graph

We present how to build the context graph of an image I, denoted G_I, based only on visual content of I and on a knowledge graph G_K. We suppose that we have built G_K from a training image database and that G_K contains several categories of entities and useful knowledge on unary, binary relationships on them as well as on their visual features (*e.g.* color histograms, interest points, etc.) or a dedicated algorithm of category detection (the visual attributes of a category in G_K can be for example learnt

from the set of visual attributes of N instances of this one). Note that, every nodes in G_K have a TRUE or FALSE status because we suppose that the training database G_K represents the universe and that all knowledge collected from this database is verified. The presence of an entity category, a couple or a triplet of categories together at least in an image can be represented by a TRUE C node. The relationships between them can be represented by a TRUE R node. However, for multiple categories and a given configuration of a relationship type, if we cannot find at least one example in the training database, this configuration is presented in a FALSE R node.

The building of G_I is based on the knowledge of G_K by exploiting the visual features of I, for the confirmation of the existence of an entity category in I. Differently to G_K, a final G_I will contain only TRUE or UNCERTAIN nodes. Certainly, if the presence of an entity category in I is not confirmed with a high probability by a detection tool for example, we represent this one by a UNCERTAIN C node. On the other hand, every categories from the training database that are not present in I can be represented by FALSE C nodes in G_I. However, this last representation is useless and complicate the representation of G_I because of a great number of absent categories and then we do not put such nodes in G_I. Consequently, the absence of C node in G_I expresses the absence of entity category associated with it in I.

Algo.3 describes the main steps of this construction. Before starting Algo.3, we have to define the five following parameters :

1. Type of unary relationship $type^u$ to be studied in I, for example, $type^u \in$ {9-area, 16-area} (see section 2.1) ;
2. Type of binary relationship $type^b$ to be studied in I, for example, $type^b \in$ {9Dspa, topological} (see section 2.1) ;
3. Probability threshold $\varphi_p / 0 < \varphi_p \leq 1$: this parameter allows to filter all configurations of a given type of relationship with a low frequency or a low presence probability in G_K. For example, by using unary relationship, in I, we can begin by verifying the presence of a category on areas where its presence probability learnt from G_K is higher than φ_p ;
4. Probability threshold $\varphi_t / 0 < \varphi_t \leq 1$ for deciding of the TRUE status of a category or a relationship node in G_I : when the detection probability of an entity of given category cat_i reaches φ_t by using the visual features of an area in I in comparison with the visual features associated with category node in G_K, a TRUE category node assigned to cat_i can be created in G_I. It means that category cat_i is present in I ;

5. Probability threshold $\varphi_u / 0 < \varphi_u < \varphi_t$ for deciding of the UNCERTAIN status of a category or a relationship node in G_I : similarly to φ_t, when the detection probability of an entity of cat_i varies in $[\varphi_u..\varphi_t]$, a UNCERTAIN category node assigned to cat_i can be created in G_I. It means that category cat_i may be present in I.

These parameters will be used in the following algorithms. The main idea of Algo.3 is that, in G_K, each TRUE category node at level 0, that is assigned to an entity category, will be examined if it can

be present in G_I. A supplementary function, named "checkCN", used by this algorithm is presented in Algo.4. This function allows to verify the presence in G_I of a given category node of G_K by comparing the visual features of I to the visual features contained in this category node. Algo.4 can be used as the algorithm to detect the presence of a given entity category in image I. Since its presence is confirmed with a probability, a TRUE or UNCERTAIN category node will be added into G_I. To reduce the complexity of Algo.3, while a TRUE C node is added to G_I, all other C nodes that cannot occur with this node will be deleted from the tail of nodes to be checked. When the level-0 of G_I is completely studied by using G_K, the next level of G_I can be studied, i.e. the binary, ternary relationships in I. To illustrate the concept, we only study here binary relationships of $type^b$, which are studied in function "buildBinaryRelation" of Algo.3 (and fully described in Algo.5). Note that, to be able to study the binary relationships between two categories, during the building of level 0 in G_I, all possible locations detected for these categories can be stocked as meta-data of G_I.

Complexity of Algo.3, Algo.4, Algo.5 : The three algorithms are presented with the paradigm of Object-oriented programming (OOP), they are entirely implementable and applicable. Let study the complexities of these algorithms :

- N^{Cat} denotes the number of entity categories presented in G_K,
- N^{Cat}_{Avg} the average number of categories present in an image,
- N^{Conf}_{Avg} the average number of configurations assigned to a R node,
- N^{Entity}_{Avg} the average number of instances of a category in an image,
- $O(D)$ the complexity of the detection processing for a category.

In general, the complexity of Algo.4 is $O_2 = O(N^{Conf}_{Avg} \times O(D))$, the one of Algo.5 is $O_3 = O(\frac{(N^{Cat}_{Avg})^2}{2} \times (N^{Entity}_{Avg})^2)$. Then, Algo.3 has a maximum complexity of $O_1 = O(N^{Cat} \times O_2 + O_3)$. For example, from the statical studies on the symbolic database studied in [15], we have : $N^{Cat} = 86$, $N^{Cat}_{Avg} = 5$ $N^{Entity}_{Avg} = 33$, $N^{Conf}_{Avg} < 138$ with 9-area splitting.

Function : ***buildGraph_01***
Input: Image I, knowledge graph G_K, type of unary relationship $type^u$, type of binary relationship $type^b$, threshold parameters : φ_p, φ_t, φ_u
Output: Graph G_I representing the context of I

$G_I \longleftarrow \varnothing$; # Initialize the context graph of I
$L_{CN} \longleftarrow G_K.get\_ListOf\_TrueCNode\_AtLevel(0)$;
Get the set of TRUE category nodes at level 0 from G_K

while $L_{CN} \neq \varnothing$ **do**
  $cn_{check} \longleftarrow getRandom\_CNode\_From(L_{CN})$;
  # cn_{check} is a C node of G_k to be located in I

  $rn^{type^u}_{check} \longleftarrow cn_{check}.getRNode\_ByTypeAndStatus(type^u, \text{TRUE})$;
  # Get a TRUE R node of type $type^u$ that is connected to cn_{check}

  if $\exists rn^{type^u}_{check}$ **then**
    $listConf \longleftarrow rn^{type^u}.\phi$; # Get the list of configurations associated with rn^{type^u}
    $foundTrueCN \longleftarrow checkCN(I, G_I, G_K, cn_{check}, type^u, listConf, \varphi_p, \varphi_t, \varphi_u)$;
    # Verify if cn_{check} can be present in G_I. The status of $G_I.cn_{check}$ depends on φ_t, φ_u. *checkCNode* is defined in Algo.4

    if $foundTrueCN = true$ **then**
      Remove from L_{CN} other C nodes that cannot occur with cn_{check} ;
      # If cn_{check} is present as a TRUE node in G_I, we remove all C nodes in G_K that are connected to cn_{check} by only FALSE R nodes. [1]
    end
  end

  $L_{CN}.remove(cn_{check})$; # Since cn_{check} is checked, we can delete it from L_{CN}
end

$buildBinaryRelation(G_I, type^b)$;
Study other spatial relationships from G_I such as binary relationships, see this function in Algo.5
Return G_I ;

Algorithm 3: Algorithm of context graph G_I building for an image I from a knowledge graph G_K. See Tab.2 on page 39 for the definition of the main used variables/methods.

1. We study the non-occurrence in the same image of two level-0 category nodes in G_K. The determination of this situation is quickly done by examining the status of the C node at level 1 that is connected to these level-0 ones. If the status of this node is FALSE, this pair of nodes cannot occur together. It means that every binary relationship nodes between these two level-0 C nodes are FALSE.

Function : ***checkCN***

Input: Image I, context graph G_I, knowledge graph G_K, category node cn of G_K to be located in I, type of unary relationship $type^u$, list of configurations $listConf$ to be verified on cn, threshold parameters : φ_p, φ_t,φ_u

Output: $foundTrueCN$ indicates if we find a TRUE C node

$foundTrueCN \longleftarrow false$;

\# In unary relationship, $conf$ indicates the area and the presence probability at this area of entity category represented by cn

for $(conf \in listConf \wedge conf.probab >= \varphi_p)$ **do**

  $res \longleftarrow I.detectObject(conf, cn)$; \# This function returns detection probability $res.probab$ and possible location of entity $res.location$ in $conf$. Note that, in the worst case, the detection tool does not provide the location of the detected object, and then $res.location$ is $conf$

  $nodeStatus \longleftarrow$ TRUE ;

  if $(res.probab > \varphi_t)$ **then**

    $foundTrueCN \longleftarrow true$; \# If it exists at least one entity of category presented by cn in I, the status of the C node assigned to this category in G_I must be TRUE

  end

  else if $(res.probab > \varphi_u)$ **then**

    $nodeStatus \longleftarrow$ UNCERTAIN ;

  end

  $cn_{G_I} \longleftarrow G_I.getCNode\_ById(cn.id)$;

  \# cn_{G_I} can be *null* or a C node having the UNCERTAIN or TRUE status

  if $(\nexists cn_{G_I})$ **then**

    $cn_{G_I} \longleftarrow G_I.newCNode\_AssignedTo(cn.getCategory(0), nodeStatus)$;

    \# cn_{G_I} is assigned to the same category as cn and has status value of $nodeStatus$

  end

  if $(foundTrueCN = true)$ **then**

    $cn_{G_I}.setStatus(\text{TRUE})$;

    \# In all cases, cn_{G_I} must change his status to the TRUE status

  end

  $rn_{G_I} \longleftarrow cn_{G_I}.getRNode\_ByTypeAndStatus(type^u, nodeStatus)$;

  \# rn_{G_I} is a R node connected to cn_{G_I}

  if $(\nexists rn_{G_I})$ **then**

    $rn_{G_I} \longleftarrow G_I.newRNode\_AssignedTo(cn_{G_I}.getCategory(0), type^u, nodeStatus)$;

    \# rn_{G_I} is assigned to the same category of cn_{G_I} and has type of unary relationship $type^u$ and status value of $nodeStatus$

    $G_I.setLink(cn_{G_I}, rn_{G_I})$;

  end

  $rn_{G_I}.addConf(conf)$;

  $G_I.addMetaData(cn_{G_I}, res.location, nodeStatus)$;

  \# We add the meta data concerning location and status of cn_{G_I} to G_I, this information can be used to study binary or ternary relationships between categories in I

end

Return $foundTrueCN$;

Algorithm 4: Algorithm of the function **checkCN()**. See Tab.2 on page 39 for the definition of the main used variables/methods.

Function : ***buildBinaryRelation***
Input: Context graph G_I, type of binary relationship $type^b$
Output: Context graph G_I that is completed with binary relationships

$L_{CN} \longleftarrow G_I.get\_ListOf\_CNode\_AtLevel(0)$;
# Get the set of category nodes at level 0 from G_I
for *(*$i \in [0..size(L_{CN}) - 2]$*)* **do**
  $cn_i \longleftarrow L_{CN}.get(i)$;
  for *(*$j \in (i..size(L_{CN}) - 1]$*)* **do**
    $cn_j \longleftarrow L_{CN}.get(j)$;

    $setCat \longleftarrow \{cn_i.getCategory(0), cn_j.getCategory(0)\}$;
    # cn_i and cn_j are at level 0 then they contain only one entity category.
    $cn_{ij} \longleftarrow G_I.getCNode\_BySetCat(setCat)$;
    # cn_{ij} is a category node at level 1 in G_I and represents the presence of *setCat* in I
    if *(*$\nexists cn_{ij}$*)* **then**
      $cn_{ij} \longleftarrow G_I.newCNode\_AssignedTo(setCat, min(cn_i.stat, cn_j.stat))$;
      # According to status constraints, a higher level C node cannot have a status superior to the status of lower ones
    end

    $listMetaData_i \longleftarrow G_I.getMetaData(cn_i)$;
    $listMetaData_j \longleftarrow G_I.getMetaData(cn_j)$;
    # Get all possible meta-data of cn_i and cn_j from G_I. This meta data is created in Algo.4
    for *(*$data_i \in listMetaData_i$*)* **do**
      for *(*$data_j \in listMetaData_j$*)* **do**
        $minStat \longleftarrow min(data_i.getCNode().stat, data_j.getCNode().stat)$;
        $rn_{ij} \longleftarrow cn_{ij}.getRNode\_ByTypeAndStatus(type^b, minStat)$;
        if *(*$\nexists rn_{ij}$*)* **then**
          $rn_{ij} \longleftarrow G_I.newRNode\_AssignedTo(setCat, type^b, minStat)$;
          # rn_{ij} is a R node at level 1 and is assigned to the same category of cn_{ij}
          $G_I.setLink(cn_{ij}, rn_{ij})$;
          $G_I.setLink(cn_i, rn_{ij})$;
          $G_I.setLink(cn_j, rn_{ij})$;
        end
        $conf \longleftarrow getBinaryConf(type^b, loc_i, loc_j)$;
        $rn_{ij}.addConf(conf)$;
      end
    end
  end
end

Algorithm 5: Algorithm of the function **buildBinaryRelation()**. See Tab.2 on page 39 for the definition of the main used variables/methods.

5.2 Image context graph building based on incomplete image annotation and knowledge graph

In this section, we present how to build an image context graph from its annotations and the knowledge graph G_K. The annotations are the location of some given categories, they can be manual annotations or the result of an object detection task. The annotations are supposed incomplete, several categories of G_K which can be present in I are not annotated. These annotations will not be put back into question even if they do not coincide with G_K. From these annotations, G_I can be initialized : initially G_I contains only TRUE nodes that represent the presence of some entity categories and some unary and binary spatial relationships deduced from the location of the entities annotated in I.

We can easily adapt Algo.3 to these new initial conditions, see this adaptation in Algo.6. One main difference between these two algorithms is the input variable G_I : in Algo.6, G_I is non empty. Because G_I contains initially some category nodes, we can check only the category nodes in G_K that can occur with these ones. Consequently, the complexity of Algo.6 is reduced considerably compared to Algo.3.

Complexity of Algo.6 : Let study the complexity of this algorithm. N^{Cat} denotes the number of entity categories presented in G_K, N^{Cat}_{Filter} the number of categories that are consistent with categories present in initial graph G_I. Then the complexity of Algo.6 is $O(N^{Cat}_{Filter} \times O_2 + O_3)$ (see the definitions of O_2 and O_3 in section 5.1) in knowing that $N^{Cat}_{Filter} << N^{Cat}$.

Function : ***buildGraph_02***
Input: Image I, initial context graph G_I, knowledge graph G_K, type of unary relationship $type^u$, type of binary relationship $type^b$, threshold parameters : φ_p, φ_t, φ_u
Output: Graph G_I representing the context of I

$L_{CN} \longleftarrow G_K.get\_ListOf\_TrueCNode\_AtLevel(0)$;
# Get the set of TRUE category nodes at level 0 from G_K
for $cn \in G_I.get\_ListOf\_TrueCNode\_AtLevel(0)$ **do**
    Remove from L_{CN} other C nodes that cannot occur with cn ;
    # If cn is present as a TRUE node in G_I, we remove all C nodes in G_K that are connected to cn by only FALSE R nodes
end

# The following is the same as in Algo.3
while $L_{CN} \neq \varnothing$ **do**
    $cn_{check} \longleftarrow getRandom\_CNode\_From(L_{CN})$;
    # cn_{check} is a C node of G_k to be located in I

    $rn_{check}^{type^u} \longleftarrow cn_{check}.getRNode\_ByTypeAndStatus(type^u, \text{TRUE})$;
    # Get a TRUE R node of type $type^u$ that is connected to cn_{check}

    if $\exists rn_{check}^{type^u}$ **then**
        $listConf \longleftarrow rn^{type^u}.\phi$; # Get the list of configurations associated with rn^{type^u}
        $foundTrueCN \longleftarrow checkCN(I, G_I, G_K, cn_{check}, type^u, listConf, \varphi_p, \varphi_t, \varphi_u)$;
        # Verify if cn_{check} can be present in G_I. The status of $G_I.cn_{check}$ depends on φ_t, φ_u. *checkCNode* is defined in Algo.4

        if $foundTrueCN = true$ **then**
            Remove from L_{CN} other C nodes that cannot occur with cn_{check} ;
            # If cn_{check} is present as a TRUE node in G_I, we remove all C nodes in G_K that are connected to cn_{check} by only FALSE R nodes.
        end
    end

    $L_{CN}.remove(cn_{check})$; # Since cn_{check} is checked, we can delete it from L_{CN}
end

$buildBinaryRelation(G_I, type^b)$;
Study other spatial relationships from G_I such as binary relationships, see this function in Algo.5

Return G_I ;

Algorithm 6: Algorithm of context graph G_I building from initial incomplete context graph and a knowledge graph G_K. See Tab.2 on page 39 for the definition of the main used variables/methods.

Now let us consider the case of binary relationships. We can improve Algo. 6 by using the binary relationships from knowledge graph G_K for better filtering the possible locations of category in check. We define the two following functions ; the output of the first function is used as one of the input of the second one :

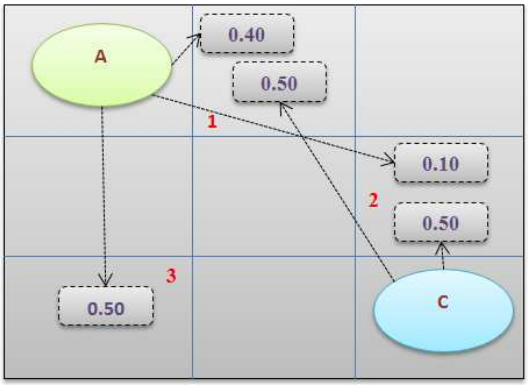
1. $findLocFromBR()$: defined in Algo.7, it allows to determinate the set of possible locations in I of a category (presented by C node cn_{check}) from G_K by using the binary relationships learnt from G_K and the existing C nodes in G_I. The presence of this category is confirmed by a TRUE or UNCERTAIN C node added in G_I. The output of this function is the list of possible locations and probabilities of cn_{check}.
2. $filterLocWithUR()$: allows to filter all locations predicted in the previous function that do not agree to the unary relationships of G_K. This function is presented in Algo.8. In this algorithm, we use a probability fusion method to compute the final probability for each possible location. We will detail this point in the following.

Algo.9 combines these two functions to improve Algo.6. Filtering on the entity location, by using binary relationships, can considerably reduce the number of final possible locations. Consequently, N_{Avg}^{Conf} is notably reduced, then also $O_2 = O(N_{Avg}^{Conf} \times O(D))$ (see section 5.1).

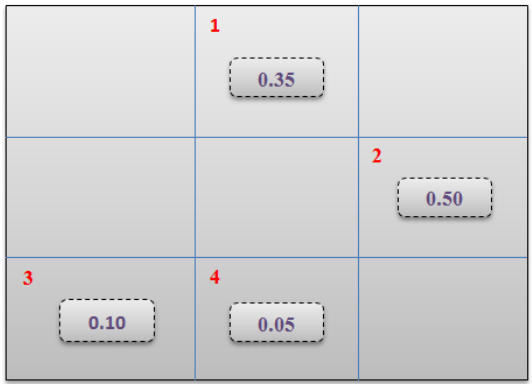
In function $filterLocWithUR()$ (Algo.8), we can encounter the following problem : how to combine the different knowledge to determinate the confidence of the results ? Fig.17 illustrates this problem. I contains entities of category A and C, we would like to verify if entities of category B could be in I, knowing that A, B, C and binary spatial relationships between A and B, B and C are known in G_K. From G_K, based on A and binary relationships between A and B, we can deduce three areas 1, 2 and 3 where B may be present with probabilities 0.40, 0.10, and 0.50 respectively. Based on C and binary relationships between C and B, there are two areas 1 and 2 with probabilities 0.50 and 0.50 respectively. Furthermore, from G_K with the unary relationship of B, we know that B can be present in four areas, as in Fig.17(b). We denote $P(cat_B|setCat, A_i)$ the presence probability of category B at area A_i in I when set of categories $setCat$ is present. According to literature on data fusion [7, 30] there are several strategies to merge these informations and to conclude about B :

1. Conjunctive strategy : for each area in I, keep the lowest probabilities and recalculate them by normalizing with the sum of the kept probabilities. For example, Fig.18(a) presents the results of this strategy by applying it on the example of Fig.17. When the probability is absent, we consider the value 0.
2. Disjunctive strategy : for each area in I, keep the highest probability and recalculate them by normalizing with the sum of kept probabilities. For example, Fig.18(b) presents the results of this strategy by applying it on the example of Fig.17.
3. Compromise strategy : It is based on the idea of the correlated probabilities fusion model of [21] :

$$f(g, h, c) = \frac{g \times h \times e^{m \times c}}{g \times h \times e^{m \times c} + (1 - g) \times (1 - h) \times e^{-m \times c}} \tag{21}$$



(a) Using binary relationship knowledge to predict the location of a given entity category B in I from existing entity categories A and C.



(b) Confidence on possible locations (unary relationships) of B learnt from G_K.

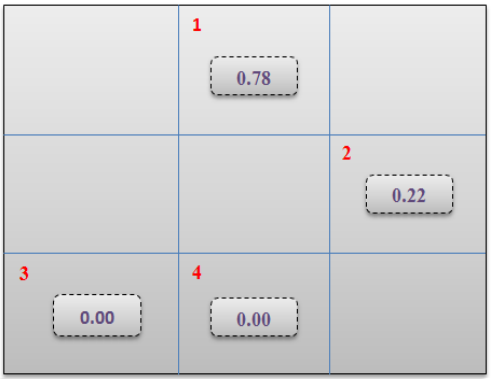
FIGURE 17 – An example of prediction of category B location in I by using learnt spatial knowledge.

where g and h are the probabilities considered, c the correlation, e is the exponential function, and $m \in [0,1]$ is the weight applied on c.
We extend this definition to a set of probabilities and define $P(cat_B|setCat, A_i)$ as a fusion function :

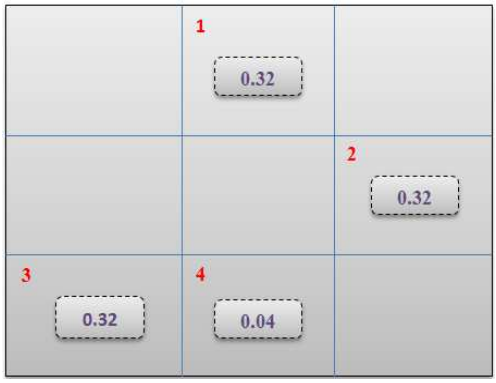
$$FFusion(\{p_i\}, c) = \frac{\prod_{j=1}^{|\{p_i\}|} p_j \times e^{m \times c}}{\prod_{j=1}^{|\{p_i\}|} p_j \times e^{m \times c} + \prod_{j=1}^{|\{p_i\}|} (1 - p_j) \times e^{-m \times c}} \tag{22}$$

where $setCat$ is the set of categories present in image, $p_j = P(cat_B|cat_j, A_i)$ the presence probability of cat_B at area A_i when cat_j is present, $c = max(correlation(cat_i,cat_j))/cat_i, cat_j \in setCat$. This strategy contains more information in comparison to the two previous strategies. The use of the correlation between a couple of existing categories (for example, the correlation obtained in [15]) can reinforce or reduce the prediction confidence about the presence of a category considered based on this couple. For example, if couple (A, C) has a high correlation, it means that the confidence of the prediction of B based on A and based on C is reinforced ; on the contrary, a low correlation of (A, C) can reduce this confidence. We have the global form of $FFusion$ is $\frac{a}{a+b}$ and the more correlation c increases, the more a increases and b decreases, consequently, the more $FFusion$ increases.

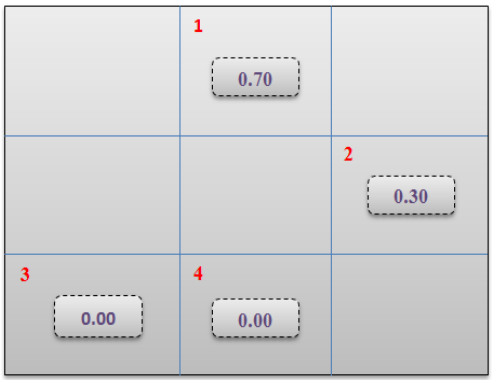
Because we think that the correlation between categories is a relevant information to integrate in the analysis, we choose the compromise strategy to compute the presence probability of one category based on the existing ones (see Algo.8).



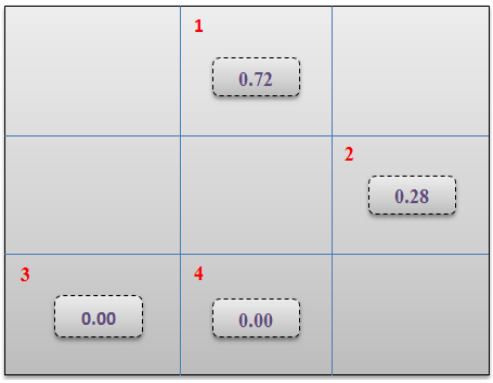
(a) By using the conjunctive strategy, probabilities 0.35, 0.10, 0.0, 0.0 are kept for areas 1, 2, 3, 4 respectively. By normalizing with their sum 0.45, we obtain the results above.



(b) By using the disjunctive strategy, probabilities 0.50, 0.50, 0.50, 0.05 are kept for areas 1, 2, 3, 4 respectively. By normalizing with their sum 1.55, we obtain the results above.



(c) By using the compromise strategy, with $m = 0.5$ and $c = 0.7$, merged probabilities for areas 1, 2, 3, 4 are 0.419, 0.182, 0.0, 0.0 respectively. By normalizing with their sum 0.601, we obtain the results above.



(d) By using the compromise strategy, with $m = 0.5$ and $c = 0.1$, merged probabilities for areas 1, 2, 3, 4 are 0.28, 0.11, 0.0, 0.0 respectively. By normalizing with their sum 0.39, we obtain the results above.

FIGURE 18 – Results of the application of three data fusion strategies to the example of Fig.17.

Function : ***findLocFromBR***

Input: Context graph G_I, knowledge graph G_K, category node cn_{check} of G_K to locate in I, type of unary relationship $type^u$, type of binary relationship $type^b$, threshold parameter φ_p

Output: List of possible unary configurations $LConf$

$LConf \longleftarrow \varnothing$;

for $(cn_{G_I} \in G_I.get\_ListOf\_TrueCNode\_AtLevel(0))$ **do**

  $setCat \longleftarrow \{cn_{G_I}.getCategory(0), cn_{check}.getCategory(0)\}$;

  # cn_{G_K} and cn_{check} are at level 0 then they contain only one entity category. In consequence, *setCat* has 2 items.

  $cn_{G_K} \longleftarrow G_K.getCNode\_BySetCat(setCat)$;

  # cn_{G_K} is a category node at level 1 in G_K

  $rn_{G_K} \longleftarrow cn_{G_K}.getRNode\_ByTypeAndStatus(type^b, \text{TRUE})$;

  for $(conf^{type^b} \in rn_{G_K}.\phi)$ **do**

    if $(conf^{type^b}.probab > \varphi_p)$ **then**

      $conf^{type^u} \longleftarrow getUnaryConfFrom(conf^{type^b}, cn_{G_I})$;

      # Determinates $conf^{type^u}$ the area in I of cn_{check} from $conf^{binary}$ and cn_{G_I}

      $conf^{type^u}_{other} \longleftarrow getByConf(LConf, conf^{type^u})$;

      # In set $LConf$, looking for a $conf^{type^u}_{other}$ having the same configuration as $conf^{type^u}$

      if $(\exists conf^{type^u}_{other})$ **then**

        # Store the detection probability and the category that allows to deduce this one before computing the fusion of probabilities in Algo.8

        $conf^{type^u}_{other}.addCategory(cn_{G_K}.getCategory(0))$;

        $conf^{type^u}_{other}.addToListOfProbab(conf^{type^u}.probab)$;

      end

      else

        $LConf.add(conf^{type^u})$;

      end

    end

  end

end

Algorithm 7: Algorithm of function **findLocFromBR()**. See Tab.2 on page 39 for the definition of the main used variables/methods.

Function : ***filterLocWithUR***
Input: Relationship node rn^{type^u} taken from G_K, list of possible unary relationship configurations $LConf$, probability threshold φ_p
Output: List of possible unary configurations $LConf$ after filtering

for *(*$conf \in LConf$*)* **do**
  $conf^{rn} \longleftarrow getByConf(rn^{type^u}.\phi, conf)$;
  # In set $rn^{type^u}.\phi$, looking for the $conf^{rn}$ having the same configuration as $conf$
  if *(*$\nexists conf^{rn} \vee conf^{rn}.probab < \varphi_p$*)* **then**
    $LConf.remove(conf)$;
  end
  else
    $conf.addToListOfProbab(conf^{rn}.probab)$;
  end
end

# Compute final detection probability for each $conf \in LConf$ according to strategy of fusion 3 and equation 22
for *(*$conf \in LConf$*)* **do**
  $maxCorr \longleftarrow computeMaxCorrFrom(conf.getCategories())$;
  # Get the highest correlation of a couple of categories among the list of categories stored for $conf$ in Algo.7
  $p \longleftarrow FFusion(conf.listOfProbab, maxCorr)$;
  # Compute the fusion probability in according to equation 22 $conf.setFinalProbab(p)$;
end

Sort $LConf$ by probability ;

Algorithm 8: Algorithm of function **filterLocWithUR()**. See Tab.2 on page 39 for the definition of the main used variables/methods.

Function : ***buildGraph_03***
Input: Image I, initial context graph G_I, knowledge graph G_K, type of unary relationship $type^u$, type of binary relationship $type^b$, threshold parameters : φ_f, φ_t, φ_u
Output: Graph G_I representing the context of I

$L_{CN} \longleftarrow G_K.get\_ListOf\_TrueCNode\_AtLevel(0)$;
# Get the set of TRUE category nodes at level 0 from G_K
for $cn \in G_I.get\_ListOf\_TrueCNode\_AtLevel(0)$ **do**
    Remove from L_{CN} other C nodes that cannot occur with cn ;
    # If cn_{check} is present as a TRUE node in G_I, we filter all C nodes in G_K that are connected to cn_{check} by only FALSE R nodes $L_{CN}.remove(cn)$;
end

while $L_{CN} \neq \varnothing$ **do**
    $cn_{check} \longleftarrow getRandom\_CNode\_From(L_{CN})$;
    $LConf \longleftarrow findLocFromBR(G_I, G_K, cn_{check}, type^u, type^b, \varphi_p)$;

    $rn_{check}^{type^u} \longleftarrow cn_{check}.getRNode\_ByTypeAndStatus(type^u, \text{TRUE})$;
    $filterLocWithUR(LConf, rn_{check}^{type^u}, \varphi_p)$;

    $foundTrueCN \longleftarrow checkCN(I, G_I, G_K, cn_{check}, type^u, LConf, \varphi_t, \varphi_u, \varphi_p)$;
    # Verify if cn_{check} can be present in G_I. The status of $G_I.cn_{check}$ depends on φ_t, φ_u. *checkCNode*() is defined in Algo.4

    if $foundTrueCN = true$ **then**
        Remove from L_{CN} other C nodes that cannot occur with cn_{check} ;
        # If cn_{check} is present as a TRUE node in G_I, we filter all C nodes in G_K that are connected to cn_{check} by only FALSE R nodes.
    end

    $L_{CN}.remove(cn_{check})$;
end

$buildBinaryRelation(G_I, type^b)$;
Study other spatial relationships from G_I such as binary relationships, see this function in Algo.5

Return G_I ;

Algorithm 9: Improved algorithm of context graph G_I building from initial incomplete context graph and a knowledge graph G_K by using binary relationship knowledge. See Tab.2 on page 39 for the definition of the main used variables/methods.

5.3 Overview of graph building

In Fig.19, we present an overview of the three main algorithms (Algo.3, Algo.6, and Algo.9). Algo.6 improves Algo.3 by filtering the list of categories considered before verifying their presence in I. Algo.9 improves Algo.6 by using the binary relationships learnt from G_K to filter possible locations of a category considered. We report the main object variables and functions used in these three algorithms in Tab.2.

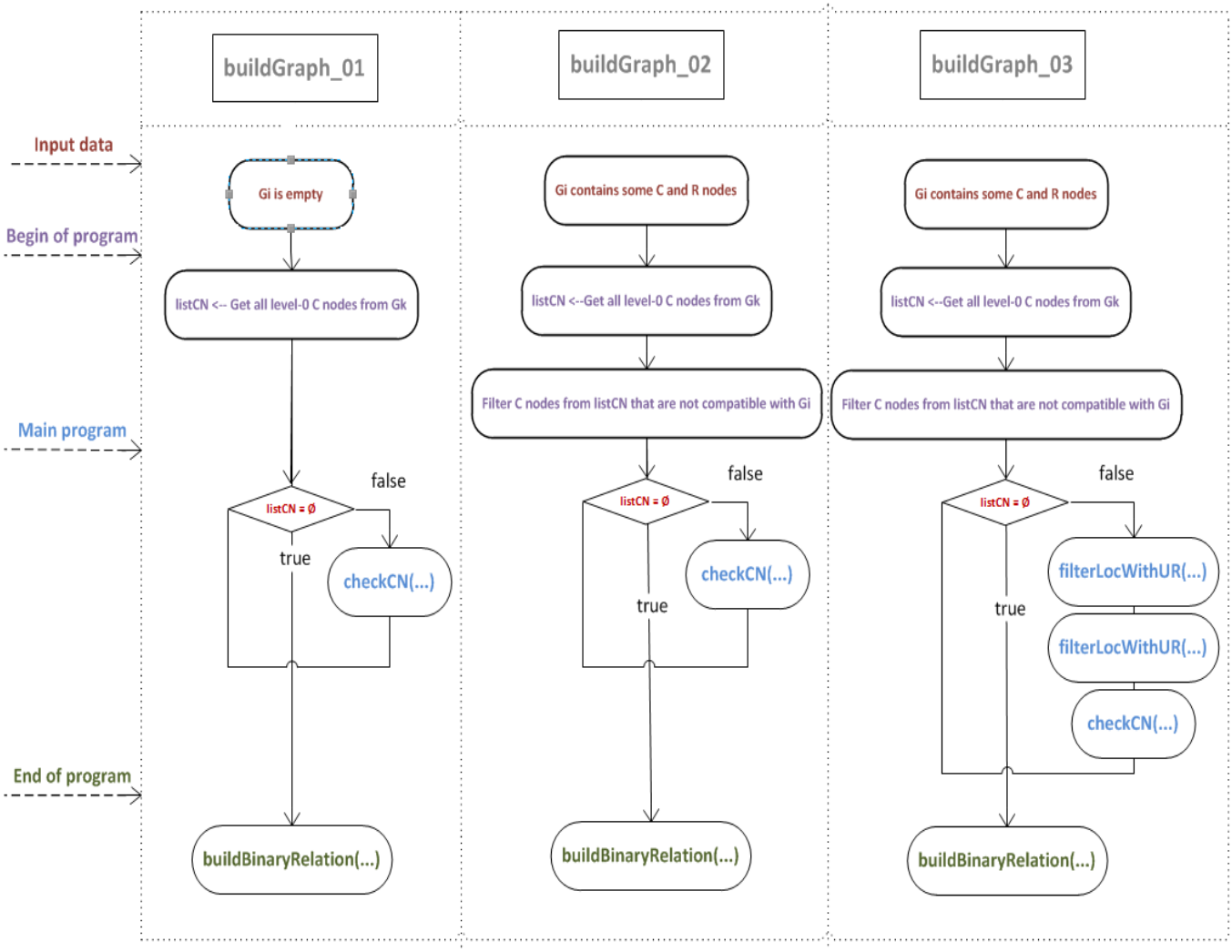


FIGURE 19 – An overview of the three graph building algorithms for a given image.

Name of variable/method	Meaning
G	Knowledge/image context graph
cn	Object variable of type C node
rn	Object variable of type R node
$conf$	Object variable of type configuration
L	List of object variables, for example L_{CN} is a list of C nodes
$L.remove(o)$	Remove object variable o from L
$L.add(o)$	Add object variable o to L
$getRandom\_CNode\_From(L_{CN})$	Get a C node from list L_{CN} randomly
$cn.getRNode\_ByTypeAndStatus(type, stat)$	Get a R node that has $type$ and $stat$ as attributes and is connected to cn
$G.getCNode\_ById(id)$	Get a C node from G in function of a given identificator id
$G.getCNode\_BySetCat(setCat)$	Get a C node from G in function of a given set of categories $setCat$
$G.newCNode\_AssignedTo(setCat, stat)$	Create a new C node in G that is assigned to a set of categories $setCat$ and that has a status $stat$
$G.newRNode\_AssignedTo(setCat, stat)$	Create a new R node in G that is assigned to a set of categories $setCat$ and that has a status $stat$
$G.setLink(cn, rn)$	In G, add an arc between cn and rn

TABLE 2 – Table of some used variables/functions and their meaning in Algo.3, 4, 5, 6, 7, 8, 9.

From these strategies, several scenarios of query based on the image context graph can be applied, we present them in the next section 5.4.

5.4 Scenarios of interrogation

Suppose that we have a knowledge graph G_K from a training image database where each image is completely annotated, and that we have a test database DB_{test} where each image is not completely annotated. By building the initial graph for each image in DB_{test}, by referencing the knowledge on G_K, from the previous algorithms, we can imagine forming some scenarios of query as following :

1. Is category A present in the given image I, knowing that A is present in G_K ?
2. What are the images in DB_{test} containing category A ? Where is the searching area of A in these images ? Where is exactly A, knowing that A is present in G_K.
3. Given two categories A and B, what are the configurations $conf$ of the relationship $type$ between them in DB_{test} ? Which configuration is the most frequent ?
4. What are the images where A and B have a $conf$ of relationship $type$?
5. Given category A, and configuration $conf$ of relationship $type$ (e.g. $type$ could be 9DSpa, topological), which category B have relation $conf$ with A ?
6. In image I where A is present, is B present ?
7. Given two categories A, B, and location L_A of A in an image, which configuration $conf$ of relationship $type$ can we found between A and B ? Where is the searching area for B ? Where is B ?
8. Given a query image, what are the images in DB_{test} that are similar to this query one ?
9. Given a prototype context (i.e a context description by a graph), what are the images from DB_{test} that have the similar context ?

In the next section, we integrate some of these scenarios into our experiments. The experiments in section 6.1 correspond to queries 1,2 while the ones in section 6.2 correspond to queries 5,6,7. The other queries are studied in section 6.3.

6 Experiments

This section is dedicated to the evaluation of our context graph-based model for object detection and image retrieval by example in a collection of images.

In the following experiments, the knowledge graph, G_K, is built from statistical results obtained in [15] on LabelMe [2] with annotated image database DB. This graph presents a general knowledge on 86 different entity categories, on their frequent locations in images (with 9-area/16-area splitting), and their binary/ternary relationships (spatial relationships, co-occurrence relationships).

To describe visual content of each entity category, we use the Δ-TSR description that allows to describe the triangular relationships between interest points (see section 2.1 or [16]). They are stored as description attributes of category nodes in G_K. In fact, for each entity present in images of DB , we extract all interest points within its boundary (annotated by a polygon). Points are extracted and characterized with SIFT. The size of the visual vocabulary is $N_L = 8000$. In consequence, each entity category can own many Δ-TSR descriptions. The detection of an entity in the query image is done by comparing its Δ-TSR description to each Δ-TSR one of the entity category (this step is encapulated in function $I.detectObject(conf, cn)$ of Algo.4 p.28). The comparison can be stopped immediately when the similarity measure reaches a given threshold.

The test database DB_{test} contains a set of 10000 images randomly chosen from LabelMe in daily contents. The content of these images is very heterogeneous and already annotated. In order to guarantee the quality of the database we verified carefully each annotated image for consistency :

- Firstly, we manually consolidated synonymous labels by correcting orthographic mistakes and merging labels having the same meaning.
- Secondly, we added missing annotations to entities of the considered categories, except for too small size entities or entities belonging to a category having a high frequency of already annotated entities in the image, such as "leaf", "window", "flower", etc. In this way, the statistical results should not be biased by these missing annotations.

In all experiments concerning algorithms presented in section 5 p.24, we choose threshold $\varphi_p = 0.1$ (the frequency of location), $\varphi_t = 0.3$ (the threshold of true positive), $\varphi_u = 0.15$ (the threshold of uncertainty). These thresholds were determined from different experiments : we vary thresholds φ_p, φ_u and φ_t in the interval [0.05,0.2], [0.1,0.3] and [0.2,0.5] respectively, and then we choose the values of thresholds that give the best results in general.

In sections 6.1 and 6.2, we evaluate the ability of describing the image context of the proposed algorithms of section 5. Then, in section 6.3, we evaluate the approach for the image retrieval based on context in using our context graph-base model. This evaluation shows the interest of using this graph-based model in image retrieval in comparison to Δ-TSR approach only.

2. LabelMe: http://labelme.csail.mit.edu.

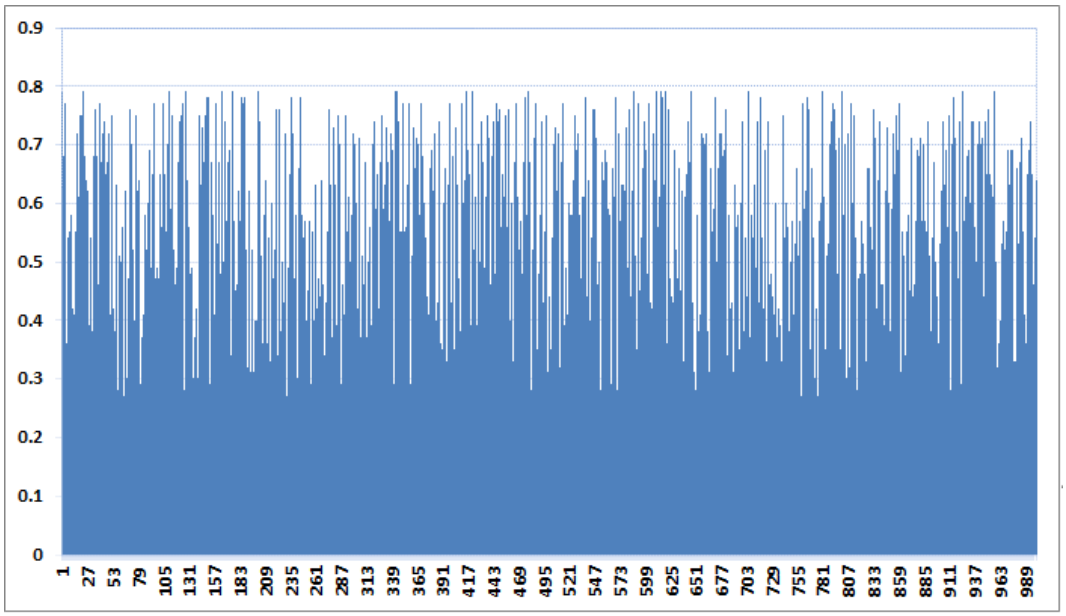


FIGURE 20 – Distribution of similarity measures between automatically built graphs and ground-truth graphs.

6.1 Evaluation of image context graph building by using unary relationships

In this section, we evaluate the ability of automatic image context graph building with algorithm *buildGraph_*01 (Algo.3 p.27) by only using the unary relationships from G_K. In section 6.1.1, we present the comparison between the graphs built by this algorithm and ground-truth graphs. Then in section 6.1.2, we evaluate the ability for object dectection and localization of *buildGraph_*01. Finally, in section 6.1.3, we present its ability of object detection compared to Δ-TSR.

6.1.1 Global comparison with ground-truth graphs

In this section, we evaluate the similarity between context graphs built with algorithm *buildGraph_*01 and context graphs built from image annotations (what we call a ground-truth graph). The similarity measure is computed based on the function of equation 19 p.23. Fig.20 displays the distribution of the similarity measures of 10000 couples of graphs.

The average of similarity measure is 0.531. The highest and lowest measures are 0.79 and 0.27 respectively. The low score is caused by false positives (entity is not present but is detected). Clearly, the absence of an entity category leads to the absence of its relationships that penalizes more the similarity between graphs. However, in average, *buildGraph_*01 gives an initial acceptable result.

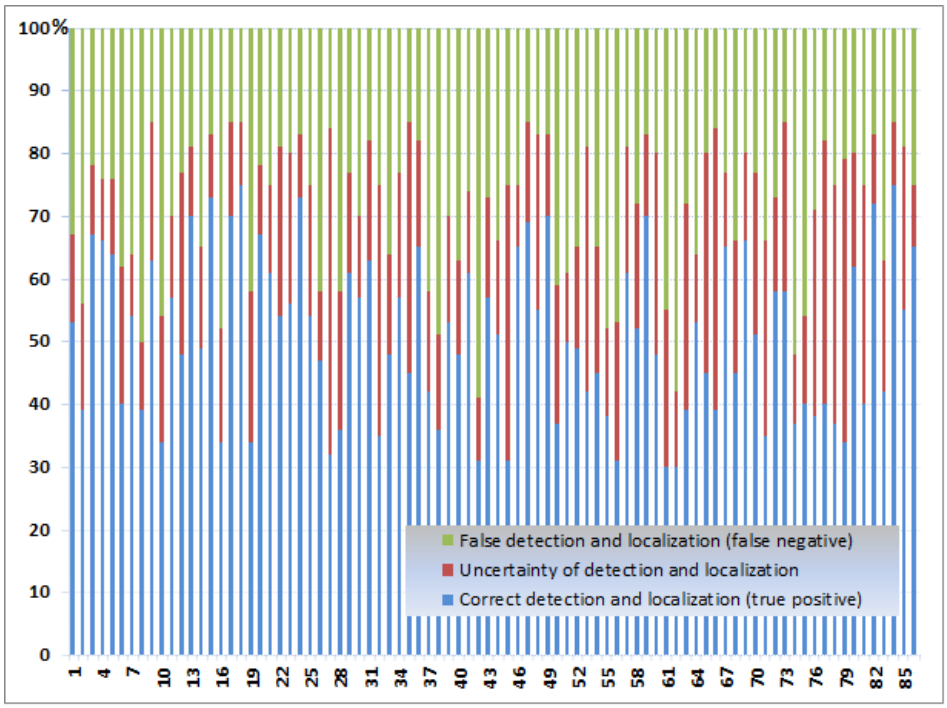


FIGURE 21 – Average precision of detection and localization step with *buildGraph*_01 (Algo.3) for each entity category when this last one is present.

6.1.2 Ability of object detection and localization vs. ground-truth

By using *buildGraph*_01, firstly, we observed that in average for each image, around 42 categories are checked for around 5 different categories detected. Secondly, there are about 14 and 21 location configurations among 138 and 649 different configurations to verify for each category with 9-area and 16-area splitting models respectively. To go deeper, we compare the information contained in the context graph for each image to the one of the annotation. We evaluate the average of precision for each entity category in two cases :

- the category is present in a given location in image,
- the category is absent in a given location in image.

Fig.21 shows the distribution of the mean precision (by number of instances of category) obtained with *buildGraph*_01 (Algo.3) for each entity category from G_K. Fig.21(-) indicates the percentage of true positives (according to threshold φ_t) while Fig.21(-) indicates the percentage of false negatives (i.e. entity is present in a given location but it is not detected in this location). The uncertainty of the detection is presented in Fig.21(-). In general, the mean precision of *buildGraph*_01 reaches 0.526 for true positives, 0.163 for uncertainty and 0.311 for false negatives.

Similarly, we evaluate the precision of *buildGraph*_01 on the absence of each category. The mean score of false positive is 0.134 (see the distribution in Fig.22), while the mean one of uncertainty is 0.153. Nevertheless, the mean score more than 0.71 of true negatives can confirm a robustness for object detection and localization of our model.

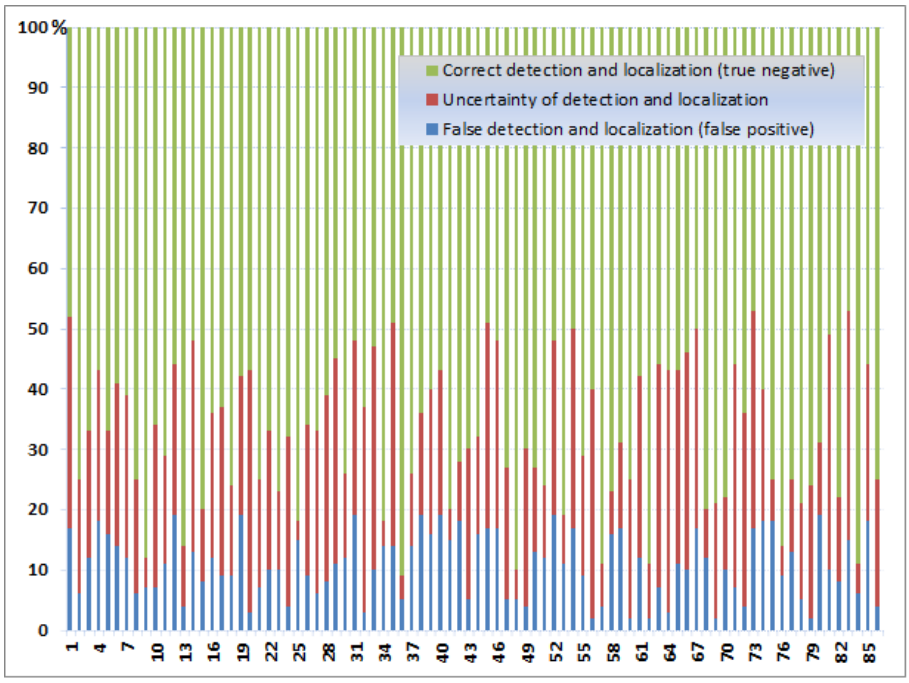


FIGURE 22 – Average precision of detection step with *buildGraph_01* (Algo.3) for each entity category when it is absent.

We illustrate category detection with our approach in Fig.26 of appendix. In this query image, the algorithm detected "car", "tree", "fence" and other categories. The yellow rectangles indicate a confirmation of the detection while the blue ones show an uncertainty.

6.1.3 Ability of object detection and localization vs. Δ-TSR

In this section, we evaluate the impact of using context knowledge for object detection and localization. Firstly, we compare the ability of detection with *buildGraph_01* with the one of Δ-TSR. Note that Δ-TSR cannot allow to localize the entity, this method detect only the presence/absence of a category in an image by giving a similarity score. Then, to compare to Δ-TSR, we evaluate *buildGraph_01* also on the criterion of presence/absence of a category in a whole image not taking into account its location. The comparison is shown in Tab.3 for Δ-TSR and strategy *buildGraph_01* (with and without considering location to be comparable with Δ-TSR).

We observe that detection with Δ-TSR is slightly improved of 1%. This improvement results from the matching of category descriptions on the whole image descriptions. Certainly, *buildGraph_01* limits the number of localization to verify, then it collects fewer descriptions from images to compare with category descriptions. Similarly, uncertainty with Δ-TSR are slightly improved of 0.8%, in consequence, the false positive are decreased of 2%. However, this advantage can become an important inconvenient when category C_i is absent. True negative of Δ-TSR decrease of 5%, that means the score of false negative and uncertainty are worse. It results from a false matching between descriptions

Category is		*builGraph*_01 take into account location	*builGraph*_01 do not take into account location	Δ-TSR
present	True positive	0.526	0.534	**0.543**
	Uncertainty	0.163	**0.158**	0.166
	False positive	0.311	0.308	**0.291**
absent	False negative	0.134	**0.127**	0.167
	Uncertainty	**0.153**	0.159	0.169
	True negative	0.713	**0.714**	0.664

TABLE 3 – Comparison of average precisions with the two algorithms.

of C_i with other descriptions in the image.

On the other hand, we show that the use of context knowledge can reduce the computation costs notably. We compare the average cost of detection for each category in two models. Note that no indexing structure is used for the two models. The comparison is presented in Tab.4. Logically, by exploiting context knowledge, the object detection of *buildGraph*_01 reduces considerably the number of comparison while remaining a competitive quality.

	*builGraph*_01	Δ-TSR
Average number of categories to check (averaged on the set of images)	42	86
Average number of Δ-TSR signatures used per image (averaged on the set of images)	1571	4315
Average number of Δ-TSR signatures of each category in G_K	442	442
Average number of signatures of each Δ-TSR signatures in G_K	73	73

TABLE 4 – Computation cost comparison between *builGraph*_01 and Δ-TSR approaches, in terms of volume of Δ-TSR descriptors and of volumes of categories to manipulate.

6.2 Evaluation of image context graph building adding binary relationships

In this section, we study the improvement brought by strategies *buildGraph*_02 (Algo.6 p.31), *buildGraph*_03 (Algo.9 p.37) in comparison to *building*_01 (Algo.3 p.27). Because *buildGraph*_02 and *buildGraph*_03 need an initial image context graph, for each image, we choose randomly an annotated entity category from image annotation as the initial information to build its initial context graph. The comparison of these three algorithms is reported in Tab.5.

We observed that :

	builGraph_01	*builGraph_02*	*builGraph_03*
Average number of categories to check (averaged on the set of images)	42	21	21
Average number of location configurations with 9-area and 16-area splittings	14/21	14/21	7/12
True positive	0.526	0.526	0.532
False positive	0.311	0.311	0.257
False negative	0.134	0.134	0.116
True negative	0.713	0.713	0.741

TABLE 5 – Comparison between *builGraph_01*, *builGraph_02*, and *builGraph_03*.

- *buildGraph_02* and *buildGraph_03* reduce considerably the number of category to verify. In average, there are only 21 categories to check for each image.

- *buildGraph_01* and *buildGraph_02* approximately inspects, in average, 14 and 21 different location configurations with 9-area and 16-area splitting respectively for each entity category. Meanwhile, in using 9DSpa relationships as binary relationship, *buildGraph_03* inspects only 7 and 12 different location configurations for each type of unary relationship.

- *buildGraph_02* does not improve the score of *buildGraph_01* in terms of detection precision because *buildGraph_02* only uses co-occurrence relationships to reduce the number of categories to check.

- *buildGraph_03* improves slightly the score of true positive when a category is present, reduces the one of false positive (the average score is 0.116). When a category is absent, the score of true negative is increased about 3%. The ones of false positive and uncertainty are lowered 2% (see the score of *buildGraph_01* in Tab.3).

We illustrate category detection with *buildGraph_03* in Fig.27 in appendix. By giving the information of "building" category, the algorithm shows different locations detected for other entity categories than can have a relationship with "building". We illustrate the algorithm with "car", "fence", "tree". By using binary relationship, *buildGraph_03* can be used to respond to queries 5, 6 or 7 presented in section 5.4. In comparison with *buildGraph_01*, *buildGraph_03* reduced the uncertainty in detection for remaining true positive.

6.3 Evaluation for image retrieval

In this section, we evaluate our context graph-based model for the scenario of image retrieval. All the techniques are evaluated under the paradigm of image retrieval by example, in terms of quality of the responses by computing Precision and Recall (P/R) curves (or at least mAP, i.e. mean Average Precision, for secondary results). These measures are averaged over all the images of the tested datasets taken as queries. We propose the evaluation with a database built by giving priority to the annotations.

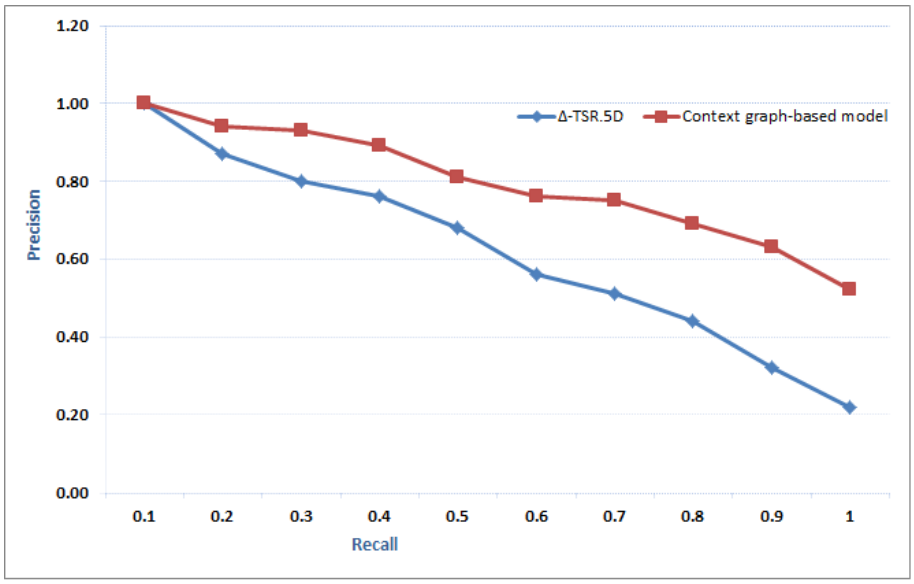


FIGURE 23 – Precision/Recall curves of Δ-TSR and context graph-based model for image retrieval.

In this experiment, we classify DB_{test} in different classes defined by the set of annotated categories present in each image (see example of Table 7 in appendix). For example, every images containing "car", "person", "building", "road" belong to the same class. In our framework, we defined randomly many classes by group of categories and chose 20 ones that contain the most images (see Tab.6 in appendix). The number of categories in each class varies between 1 and 5. The comparison of the P/R curves for the two approaches (Δ-TSR and the context graph-based model) gives the Fig.23.

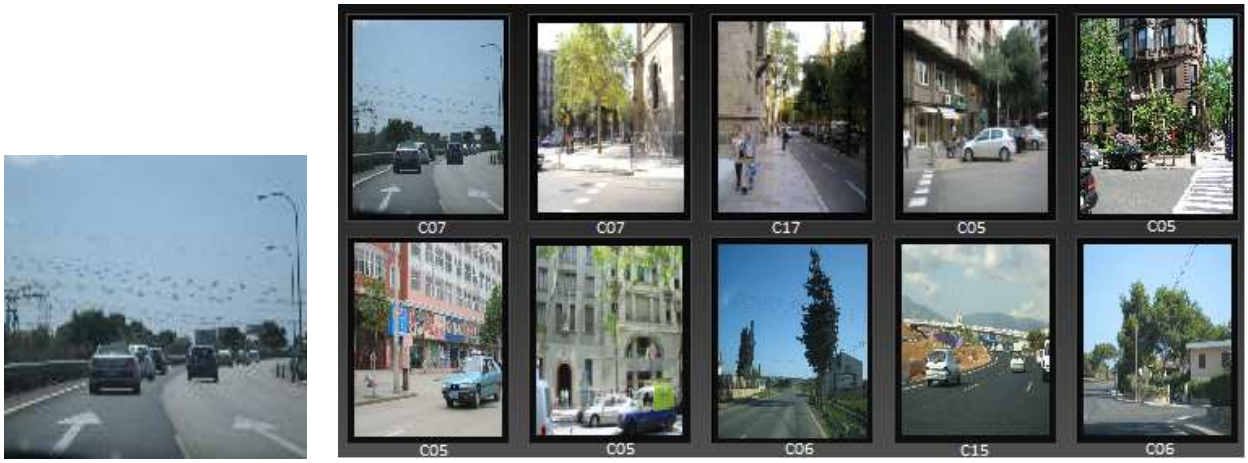
We observe that the graph-based model is superior. Its mAP is 0.79 while mAP of $\Delta\text{-TSR}_{5D}$ is 0.61. The result of $\Delta\text{-TSR}_{5D}$ is worse because of the important variability of the visual content in each class of this test database. The similarity measure in the graph-based model is always based on the presence of entity categories and spatial relationships between them, then the better precision is logical.

6.3.1 Example of retrieval

To conclude these experiments, we give an example of image retrieval by example with Δ-TSR in Fig.24 and with the context graph-based model in Fig.25.

7 Conclusion

In this paper, we have presented and evaluated a graph-based model to represent the spatial knowledge existing between entity categories in an image. This graph is totally extendible for adding a new



(a) Query image (b) Results with Δ-TSR$_{5D}$

FIGURE 24 – The ten first images similar to the query image in terms of visual content (Δ-TSR.

(a) Query image (b) Results with context graph-based model

FIGURE 25 – The ten first images similar to query image in terms of context.

knowledge. The unification of knowledge is guaranteed robustly by using three notions of node situation : TRUE, UNCERTAIN, FALSE. Moreover, this knowledge graph helps to localize and detect objects in an image if we can collect some initial information from other categories. And more generally, in the other cases, the location knowledge of an entity category allows to restrict the number of locations for its detection if we do not have any a priori information in the image.

We apply this graph model to represent the spatial context of an image. This representation allows a quick comparison between images based on the context similarity. Each node in the graph has an unique identification, then the matching between two nodes of two graphs can be done quickly with a hashing structure. The context similarity of two images is evaluated on the presence of categories and the similarity of their spatial relationships. So, our graph-based model can be classified with those using the image context a priori for image processing. In that way, for image retrieval for example, the model can strengthen the obtained images filtering , then, the execution time. We have also proposed three algorithms to represent automatically the context of image by using a knowledge graph as external assistance for object detection and localization. We demonstrated also the robustness of these three algorithms for the building of context of image. We have also experimented its relevance for image retrieval by context by comparing our graph-based model to Δ-TSR that allows a similarity search based on visual content.

References

[1] V. K. Balakrishnan. *Graph Theory*. McGraw-Hill, 1997. 6

[2] C. Chang. Spatial match retrieval of symbolic pictures. *Journal of Information Science and Engineering*, 7(3) :405–422, 1991. 4

[3] S. Chang and E. Jungert. A spatial knowledge structure for image information systems using symbolic projections. *International Journal of Computational Cognition*, pages 79–86, 1986. 4

[4] G. Chartrand. Introductory graph theory. *New York : Dover*, page 116, 1985. 6

[5] M. Choi, J. Lim, A. Torralba, and A. Willsky. Exploiting hierarchical context on a large database of object categories. *Conference on Computer Vision and Pattern Recognition*, 2010. 5

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

[7] D. Dubois and H. Prade. A review of fuzzy set aggregation connectives. *Information Sciences*, 36 :85–121, 1985. 32

[8] J. Eakins. Towards intelligent image retrieval. *Pattern Recognition*, 35 :3–14, 2002. 2

[9] M. Egenhofer and K. Al-Taha. Reasoning about gradual changes of topological relationships. In *Proceeding of the International Conference GIS - From Space to Territory : Theories and Methods of Spatio-Temporal Reasoning on Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, pages 196–219, London, UK, 1992. Springer-Verlag. 4, 12

[10] M. J. Egenhofer and R. D. Franzosa. Point set topological relations. *International Journal of GIS*, pages 161–174, 1991. 4, 7, 12

[11] V. Gouet-Brunet, M. Manouvrier, and M. Rukoz. Synthèse sur les modèles de représentation des relations spatiales dans les images symboliques. *Revue des Nouvelles Technologies de l'Information*, (RNTI-E-14) :19–54, 2008. 4

[12] V. Gudivada and V. Raghavan. Content-based image retrieval systems. *Computer*, 28(9) :18–22, 1995. 2

[13] D. Guru, P. Punitha, and P. Nagabhushan. Archival and retrieval of symbolic images : An invariant scheme based on triangular spatial relationship. *Pattern Recognition Letters*, 24(14) :2397–2408, 2003. 4

[14] G. Heitz and D. Koller. Learning spatial context : Using stuff to find things. *European Conference on Computer Vision (ECCV)*, 2008. 4, 5

[15] N. Hoàng, V. Gouet-Brunet, and M. Rukoz. A cartography of spatial relationships in a symbolic image database. *Computer Analysis of Images and Patterns*, 6854 (1) :377–385, 2011. 8, 26, 33, 41

[16] N. Hoàng, V. Gouet-Brunet, M. Rukoz, and M. Manouvrier. Embedding spatial information into image content description for scene retrieval. *Pattern Recognition*, 43(9) :3013–3024, 2010. 4, 12, 41

[17] P. Huang and C. Lee. Image Database Design Based on 9D-SPA Representation for Spatial Relations. *TKDE*, 16(12) :1486–1496, 2004. 4, 24

[18] Y. Liu, D. Zhang, G. Lu, and W. Ma. A survey of content-based image retrieval with high-level semantics. *Pattern Recognition*, 40(1) :262–282, 2007. 2

[19] M. Nabil, J. Shepherd, and H. Anne. 2D projection interval relationships : A symbolic representation of spatial relationships. In *Symposium on Large Spatial Databases*, pages 292–309, 1995. 4

[20] J. Park, V. Govindaraju, and S. N. Srihari. Genetic engineering of hierarchical fuzzy regional representations for handwritten character recognition. *International Journal on Document analysis and recognition*, 2000. 3

[21] K. Pradeep and S. Mohan. Probability fusion for correlated multimedia streams. *ACM Multimedia*, 2004. 32

[22] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie. Object in context. *IEEE International Conference on Computer Vision*, 2007. 5

[23] A. Singhal, J. Luo, and W. Zhu. Probabilistic spatial context models for scene content understanding. *Computer Vision and Pattern Recognition*, 2003. 5

[24] A. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *Pattern analysis and machine intelligence*, 22 :1349–1380, 2000. 2

[25] Z. Song, Q. Chen, Z. Huang, Y. Hua, and S. Yan. Contextualizing object detection and classification. *Conference on Computer Vision and Pattern Recognition*, 2011. 5

[26] A. Torralba, A. Oliva, M. Castelhano, and J. Henderson. Contextual guidance of eye movements and attention in real-world scenes : The role of global features on object search. *Psychological Review*, 113(4) :766–786, 10.2006. 4, 5

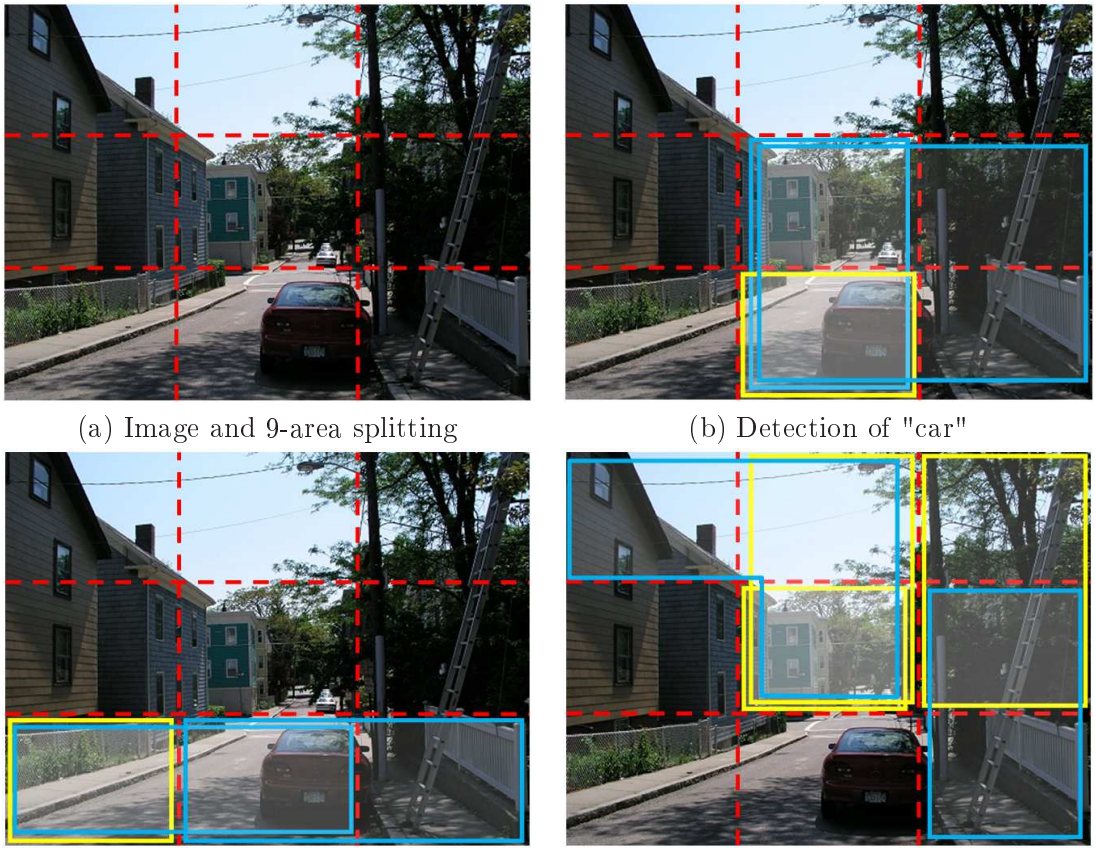
[27] N. Vasconcelos. From pixels to semantic spaces : Advances in content-based image retrieval. *Computer*, 40(7) :20–26, 2008. 2

[28] W. Wang, A. Zhang, and Y. Song. Identification of objects from image regions. *International Conference on Multimedia and Expo*, 2003. 3

[29] E. Weisstein. *Bipartite Graph*. MathWorld, 1999. 6

[30] R. Yager. Connectives and quantifiers in fuzzy sets. *Fuzzy Sets and Systems*, 40 :39–75, 1991. 32

APPENDIX



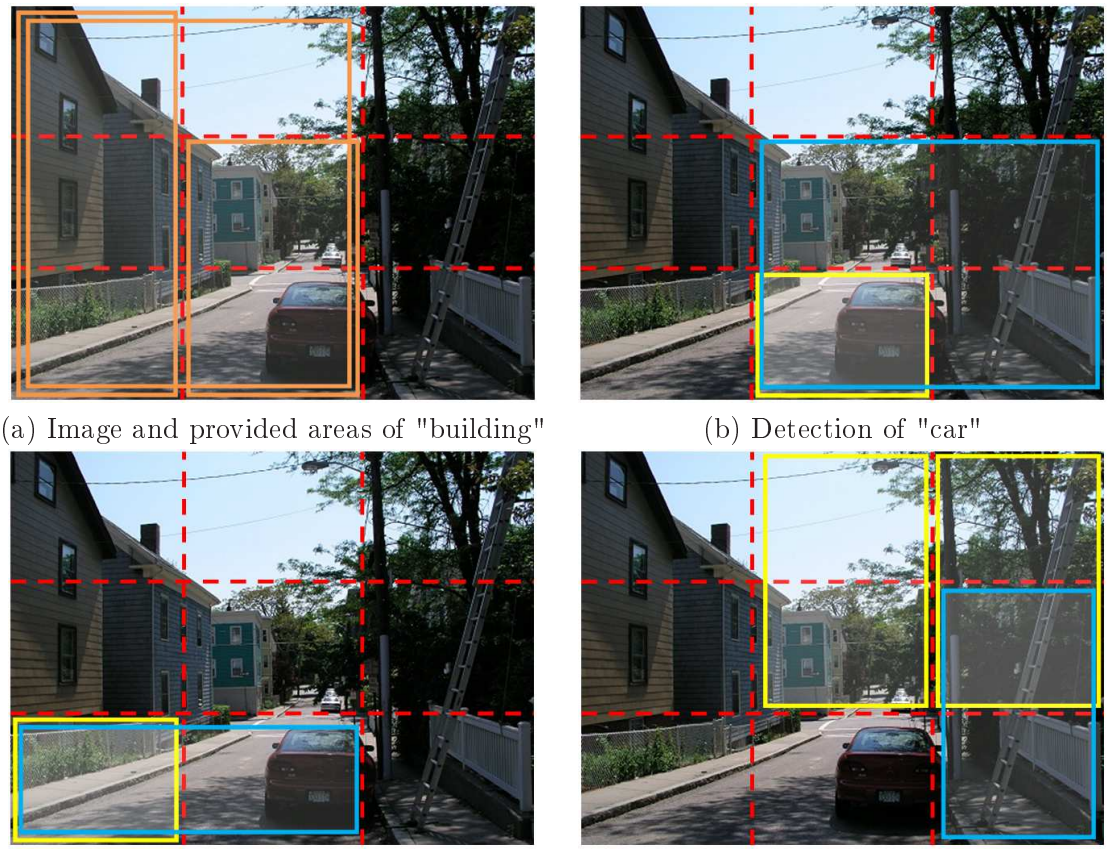
(a) Image and 9-area splitting

(b) Detection of "car"

(c) Detection of "fence"

(d) Detection of "tree"

FIGURE 26 – Entity categories detected with *buildGraph_*01 and their locations with 9-area splitting : blue rectangle for uncertain location and yellow rectangle for a sure one.



(a) Image and provided areas of "building"

(b) Detection of "car"

(c) Detection of "fence"

(d) Detection of "tree"

FIGURE 27 – Entity categories detected with *buildGraph_03* from the given category "building" and their locations : blue rectangle for uncertain location and yellow rectangle for sure one.

ID Class	Categories
C01	bicycle, car, road
C02	bird, grass, lake, tree, path
C03	bird, plant
C04	boat, building, lake, sky
C05	building, car
C06	building, fence
C07	car, road, tree
C08	chair, window
C09	chimney, roof, building, tree
C10	door, fence, wall
C11	flag, road
C12	grass, flower
C13	lake, tree, sky
C14	lamp, tree
C15	moutain, road
C16	motobike, table, traffic-light
C17	person
C18	sea, sky, mountain
C19	side-walk, tree
C20	table, umbrella

TABLE 6 – 20 image classes in experiment of section 6.3.

person	building, car	boat, lake, sky, building
		
		
		
		
		
lake, tree, sky	bycicle, car, road	side-walk, tree
		
		
		
		
		

TABLE 7 – Different classes of test database for evaluation in section 6.3.