Combining semantic technologies with a content-based image retrieval system – Preliminary considerations

P. Chmiel, M. Ganzha, T. Jaworska, and M. Paprzycki

Citation: AIP Conference Proceedings **1895**, 100001 (2017);

View online: https://doi.org/10.1063/1.5007405

View Table of Contents: http://aip.scitation.org/toc/apc/1895/1

Published by the American Institute of Physics

Combining Semantic Technologies with a Content-based Image Retrieval System – Preliminary Considerations

P. Chmiel[1,a)], M. Ganzha[1,2,b)], T. Jaworska[2,c)] and M. Paprzycki[2,3,d)]

[1]*Warsaw University of Technology, Koszykowa 75, 00-662 Warsaw, Poland*
[2]*Systems Research Institute, Polish Academy of Sciences, Newelska 6, 01-447 Warsaw, Poland*
[3]*Warsaw Management Academy, Kaweczynska 36, 03-772 Warsaw, Poland*

[a)]piotrchmiel90@gmail.com
[b)]Maria.Ganzha@ibspan.waw.pl
[c)]Tatiana.Jaworska@ibspan.waw.pl
[d)]Marcin.Paprzycki@ibspan.waw.pl

Abstract. Nowadays, as a part of systematic growth of volume, and variety, of information that can be found on the Internet, we observe also dramatic increase in sizes of available image collections. There are many ways to help users browsing / selecting images of interest. One of popular approaches are Content-Based Image Retrieval (CBIR) systems, which allow users to search for images that match their interests, expressed in the form of images (query by example). However, we believe that image search and retrieval could take advantage of semantic technologies. We have decided to test this hypothesis. Specifically, on the basis of knowledge captured in the CBIR, we have developed a domain ontology of residential real estate (detached houses, in particular). This allows us to semantically represent each image (and its constitutive architectural elements) represented within the CBIR. The proposed ontology was extended to capture not only the elements resulting from image segmentation, but also "spatial relations" between them. As a result, a new approach to querying the image database (semantic querying) has materialized, thus extending capabilities of the developed system.

1 INTRODUCTION

Historically speaking, at first, images were collected and browsed ([1]), next they were processed in different ways and, at present, they are being retrieved according to user preferences. While image processing is developing, *e.g.*, in the form of ever-improving, and more complex, methods for image segmentation, recognition, matching and retrieval (see, for instance, [2, 3, 4, 5, 6] and references collected there), one of interesting questions remains, how to use, as efficiently as possible, the existing collections of images. To address this problem, among others, multiple Content-Based Image Retrieval (CBIR) systems have been developed. They provide GUI's that allow users to search for images that match their interests, expressed in the form of images, *i.e.*, query by example. In other words, the main idea of a CBIR is to answer a query formulated as: which images in the collection are closest to the "selected / input image(s)". On the one hand, this allows to "move away from linguistic description of images", which seem to mimic the way we (humans) deal with images ([7, 8]). On the other, this prevents one from utilizing existing "captured knowledge" about the world (*e.g.*, information that, in most cases, roofs are on top of the building). This being the case, we have decided to extend an existing CBIR system by infusing it with techniques originating from, broadly understood, semantic technologies.

The CBIR system that we have access to (and are in the process of extending), is being developed at the Systems Research Institute, Polish Academy of Sciences (SRI PAS)(for complete description of all aspects of already developed system, see [9, 10, 11, 12]). At the time of writing of this contribution, the system uses four original modules, implemented in Matlab, for image segmentation and classification. These modules have been applied to images of detached houses. Note that number of available modules (and their properties) evolves, as they are a part of an ongoing research project. As a result of image processing, images (and their classified segments) are stored in a relational

Application of Mathematics in Technical and Natural Sciences
AIP Conf. Proc. 1895, 100001-1–100001-12; https://doi.org/10.1063/1.5007405
Published by AIP Publishing. 978-0-7354-1579-9/$30.00

(Oracle) database. Represented classes of elements (segments) have been stipulated according to the (external) architectural elements of building that the image processing techniques are capable of capturing. They include, among others: door, window, roof, stairs, *etc.*

The existing GUI, allows users to design a query, expressed in a special graphical form, which is transformed into a mathematical form, and sent to the internal search engine. The search engine selects, from the database, images that are the most similar to the user request. Specifically, the matching process consists of three steps: (1) comparison of image signature, describing the number of graphical segments of each class in the whole image, (2) comparison of a spatial segment layout, and (3) comparison of particular segments, in pairs, between the query and images from the database.

While the existing CBIR is quite robust, and systematically improved, its shortcoming is similar to the majority of CBIR's. It represents only knowledge found "within images" while ignoring ("contextual") information about the world that subjects of these images are part of. To address this shortcoming, we have decided to take the existing CBIR system as a starting point, and extend it by adding capabilities originating from semantic technologies. As illustrated in Figure 1, our current contribution (discussed in this paper) is augmentation of the CBIR system by adding a module (Ontobrain), which connects semantic mechanisms with the database of preclassified image segments. In other words, we have started process of development of a hybrid "Semantic CBIR" (SCBIR).

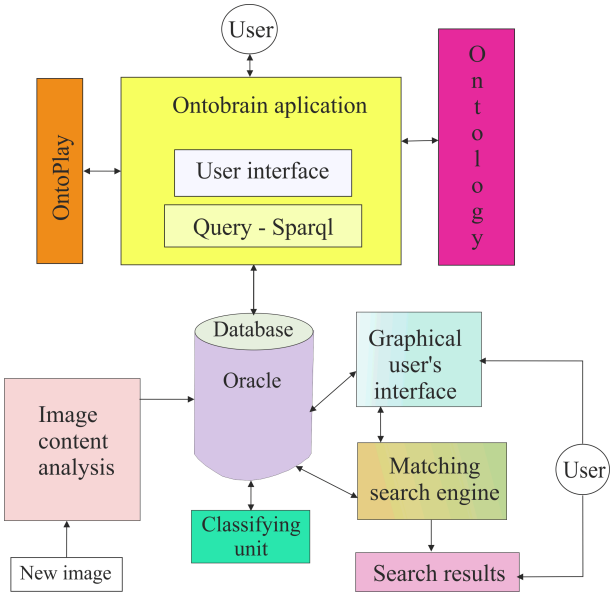

FIGURE 1. Connection between the existing CBIR system with semantic module for a common image database

Here, it should be observed that one of issues that reduce uptake of semantic technologies is lack of knowledge how to develop, implement and use them. This being the case, for the SCBIR to be accepted and tried outside of the area of (discussed here) images of family homes, it has to allow users, unfamiliar with semantic technologies, to formulate queries in a way that will hide from them the fact that semantic mechanisms are being used. Furthermore, interested users should be able to view the captured domain knowledge (in the form of an ontology) in a user-friendly graphical interface. Finally, user should be capable of making simple modifications to the ontology, based on their expertise and reactions to the displayed images (and their fragments). However, the latter aspects of the, already implemented, Ontobrain module, are outside of focus of this contribution, and thus are only briefly mentioned in Sections 4 and 5.

In this context, in what follows, in Section 2 we discuss how we have developed the ontology of detached homes and their surroundings. Next, in Section 3, we describe the key technical aspects of the Ontobrain (semantic) module. We follow, in Sections 4 and 5 with a brief description of features of the system, prepared for users who would like to go beyond simple image "browsing". Finally, Section 6 contains an example of the work of the SCBIR system, used for image retrieval.

2 DEVELOPING ONTOLOGY OF A BUILDING

In order to be able to apply semantic technologies, the first step is to develop a formal representation of the domain of interest (*i.e.*, an ontology [13]). In our case, the domain of interest can be identified as *real estate* or, more precisely, *detached houses and their surroundings*. Following good practices of ontology development (see, for instance, [14]), our initial idea was to reuse an existing building ontology; possibly by modifying it somewhat, in order to match the information available in the database (collected within the CBIR). Here, it is important to stress that we have decided / had to develop an ontology to work within the scope of an *already existing system*. Specifically, as stated above, there exists a database of segmented and classified images. Therefore, the domain ontology, should include already existing classes and match identifiable relationships between them. With this in mind, we have surveyed available sources, and found the following ontologies of buildings (related to buildings):

1. **dbpedia.org**. Representation of building domain, found in DBpedia [15], is focused on possible types of buildings, such as: hospitals, hotels, casinos, libraries or museums, which is not very useful for our purposes. Hence, it is "incomplete", from our point of view, and provides only limited information about the building. Specifically, it shows types of the building and synonyms of word *building*, while it does not provide any architectural elements of the building, and does not capture relations between them.
2. **gaz.owl** is an ontology of an "environment" [16]. It contains over 17000 classes and instances of environments. It contains a class Building Parts with subclasses. Besides a few classes of architectural elements of the building such as: *roof* or *window*, conceptualized on the same level of the hierarchy in the ontology, it contains classes such as: *room* or *kitchen*. Those classes are not architectural elements of the building, conceptualized from the perspective of images stored in the existing CBIR system (they are inside the building, while the CBIR captures *outside elements* only). Because of existence of only few classes related to (external) building parts, and too high level of generality of represented concepts, we could not use it in our project.
3. **eTRIMS – E-Training for Interpreting Images of Man-Made Scenes**. This project concentrates on structural learning, where relations between components and compositional hierarchies played central role in object categorization [17, 18]. As one of its results, an ontology of architectural elements was proposed. However, it comprises only a small number of classes. For instance, outdoor accessories, such as *pillars*, *chimneys* or *railings* are missing. This is a problem, as these elements are present in the existing CBIR database. Therefore, while this ontology could be used as a starting point for the Ontobrain, we have decided that it would require too many modifications to be useful.
4. **Ontology-Based Classification of Building Types Detected from Airborne Laser Scanning Data**. In this case, a scientific paper provides some information about an ontology containing hierarchy of buildings [19]. However, it also does not meet our needs. The publication focuses on the ontologies and semantic technologies that are used in GIS and in Remote Sensing. Furthermore, the constructed system is, probably, commercially used because the developed ontology is not publicly available.
5. **Ontology Across Building, Emergency, and Energy Standards**. This article presents a general building ontology [20]. It contains some connections between properties and within the ontology items. The constructed system has been made for services partners and government or public safety. That is why this system is mainly focused on the inside of the building, but our database contains images of the outside of the building. Hence, again, this ontology was not useful in the context of our work.
6. **Swoogle**. Swoogle is an ontology search engine [21]. It offers ontological matching the keywords. As a result of multiple searches, we managed to find, for instance, some examples of roof ontologies, but not of the whole building, with its constitutive parts (which might be interesting in our future work). Thus, Swoogle also did not provide us with useful information.

As a result of failure of our search for a ontology that would be usable in context of our work, we have been forced to construct our own residential real estate (detached houses) ontology. However, it should be stressed that, if a (more) suitable ontology already exists, but we have missed it, it can be used to replace the one that we have developed. Furthermore, such a replacement will be easy to achieve, as it will not require major changes in the SCBIR.

2.1 Ontology Construction

In order to construct the required ontology, at first, we have analysed the existing CBIR database, to establish, which architectural elements have been recognized in the system [11]. Here, let us note that, in general, this step may not be as

obvious as it seems. Generally, when someone is confronted with a database only (with no in-depth documentation), it may be rather difficult to extract semantic information from it. Specifically, (relational) databases are created to provide efficient access to information, while not being designed to utilize semantics of the data. This being the case, the semantics of data woven in tables and keys can be observed, primarily, thorough queries and responses (for more information, see [22]). However, in the case of the CBIR used in our work, we had direct access to the needed information concerning all aspects of the data that has been stored, which made the development of an appropriate ontology easier.

In the CBIR database, images (objects) are natural candidates for individuals, while classes of segments are equivalents of ontological classes. The CBIR database contains 42 classes, primarily specifying elements recognizable within the building façade, and outdoor accessories, which were located in the processed images (*e.g.*, swimming pools, fences, or plants). Therefore, naturally, the developed ontology has 42 core classes. In this way, we have assured that each element that the image processing techniques are capable of recognizing, has been captured in our formal representation of the domain.

Separately, additional classes have been added, on the basis of academics publications ([23, 24]), devoted to architecture and civil engineering studies. As a result, we have constructed ontology, which is divided into three main parts: *Building Parts* marked in Figure 2 with 1, *Things Around The Building* marked with 2, and *Technical* marked with 3. This ontology (all of its parts) consists of 151 classes, 86 properties and more than 3000 individuals.

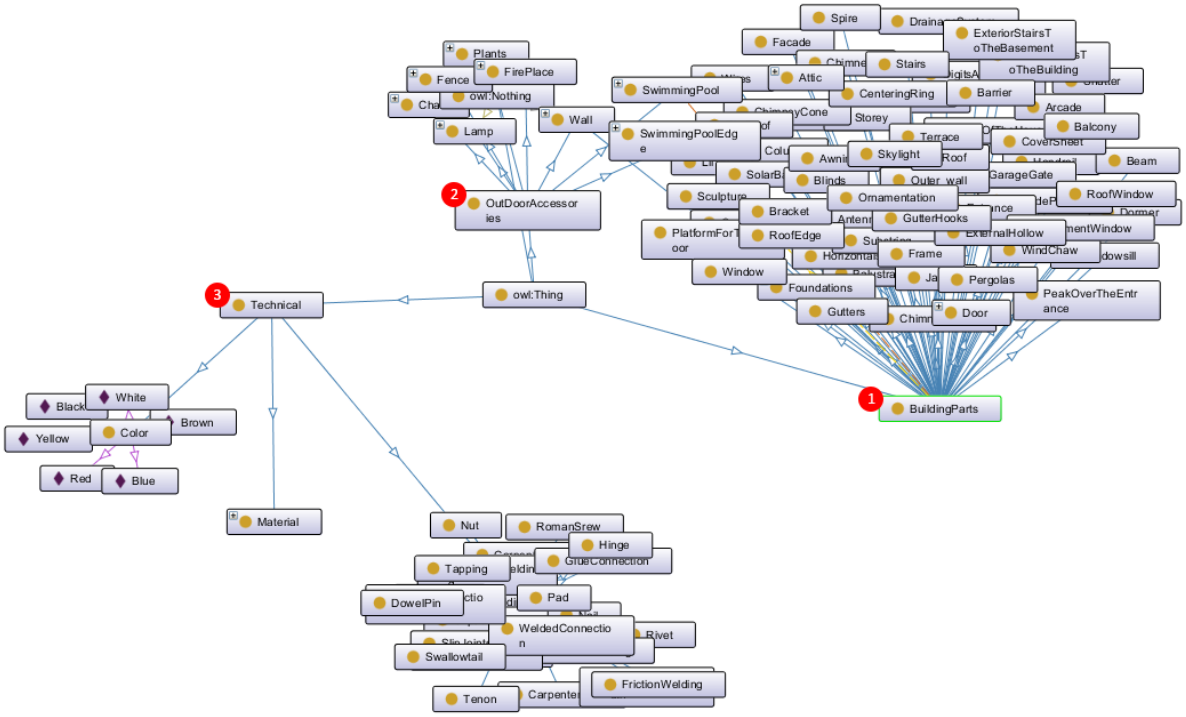

FIGURE 2. Building ontology structure

The first part is composed of 112 subclasses that represent "external parts of the building", such as: *roof, window, wall, door*, *etc.* The second part contains 9 classes that ascribe/attribute elements surrounding the building, such as: *swimming pool, plant, fence*, *etc.* The third part represents aspects of the image that (currently) do not exist in the CBIR database, but complemented it. The *Technical* class has 30 subclasses such as: *Color*, *Materials*, or *Building Joints*, which have been added on the basis of [24, 25].

One of important concerns was to decide how many levels the ontological hierarchy should be divided into. For example, the general question was: if a given individual should be assigned to a class or a new subclass should be created for it. For instance, when adding the class *Color* we had to decide if a subclass *Brown* should be added, or

if *Brown* should be placed as a representative of class *Color*. We decided that the level of details should stop at the class *Color*. It would be hard to justify use of class *Brown*, without describing it without other attributes. Moreover, the application we have been developing has no need to use deeper hierarchy for this class.

Note that new classes have been added to enhance the ontology, in order to obtain more complete description of objects (buildings). Here, the following reflection can be made. In the process of developing the ontology of detached houses and their surroundings, we have addressed additional aspects of the building and the area that it is placed in. These aspects have already been added to the CBIR database (though, neither they have representative images, nor processing software to add them, at least for the time being). This, in turn, may lead to an upgrade of existing image recognition modules (or adding new ones) that will be able to recognize these (new) elements. In this way, the very fact that the domain ontology has been developed, may further advance the developed system, by pointing to aspects of the image that are worthy capturing / recognizing.

2.2 Properties in The Ontology and Spatial Relationships of Architectural Elements

One more interesting issue that had to be resolved, was: how to represent spatial relationships among architectural elements. For instance, in a building, a roof is above a wall. However, it is impossible to deduce that fact solely from the class hierarchy in the ontology. In our system, classes such as *Wall*, *Roof* and *Window* are situated "on the same level" in the class hierarchy.

Our initial idea, to recognize the fact that roofs are over of the wall, whereas windows are located somewhere "on / within the wall", was to add two general properties *isAbove* and *isOn*, and a set of Domains and Ranges of these properties, to the list of the classes in the constructed ontology. However, this could not solve our problem, and would not provide any new information about the spatial relationships. For instance, let's assume that we have only one property *isAbove*. Using this property, we would like to express that a *Roof* is above a *Wall*, while a *Wall* is above a *Foundation*, and an *Antenna* is above a *Roof*. Let's assume that the class from the Range is *above* the class from the Domain of the property (it could also be the other way around – but we omit the details). In this case, in order to express that:

1. a *Wall* is above a *Foundation*, we add class *Foundation* to the Domain and class *Wall* to the Range;
2. a *Roof* is above a *Wall*, we add class *Wall* to the Domain and class *Roof* to the Range;
3. an *Antenna* is above a *Roof*, we add class *Roof* to the Domain and *Antenna* to the Range.

As a result, the Domain of the property *isAbove* is a set of three classes *Foundation*, *Wall* and *Roof*. The Range of the property *isAbove* is a set of three classes *Wall*, *Roof* and *Antenna*. Let us now assume that we would like to reason within this ontology, and check if a *Roof* is above a *Wall* (or if this it the other way around). According to our assumption, class from the Range is above of the class from the Domain. In our Range we have *Wall*, *Roof* and *Antenna*. Let us check each class: an *Antenna* is above a *Roof*, a *Roof* can be ignored (a *Roof* is not above of a *Roof*). Henceforth, we deduce that a *Wall* is above a *Roof*, which is wrong. The same reasoning can be made for the class *Wall* and it will lead us to the conclusion that a *Roof* is above of a *Wall*. Hence, this solution does not work.

To avoid this, we have decided to use a different approach; to include additional properties, which *unambiguously* specify, which elements of the building are above of, or within / on another element. These, added, properties have names with prefixes that start with *isAboveOf* or *isOn*, while the suffix is the name of the class. For instance, in the developed ontology, we have defined properties such as: *isAboveOfRoof*, *isAboveOfWall*, *isOnWall*, *etc.* All these properties have Range set to one class – the one from the suffix. The Domain of the property is set to the list of classes, which are allowed to be "within the element" or "on top of it". In the case of the class *Wall* these could be, for instance: *Roof*, *Chimney*, *Skylight*, *etc.*

3 DEVELOPING THE SEMANTIC MODULE – PRACTICAL CONSIDERATIONS

As indicated, in Section 1, our main goal (during this stage of the project) was to extend capabilities of an existing CBIR system, by adding to it a module that will allow users to utilize semantic technologies for image retrieval. To achieve this goal, we have built a WEB application, consisting of four abstract layers, represented in Figure 3. Let us describe each of these layers in some detail.

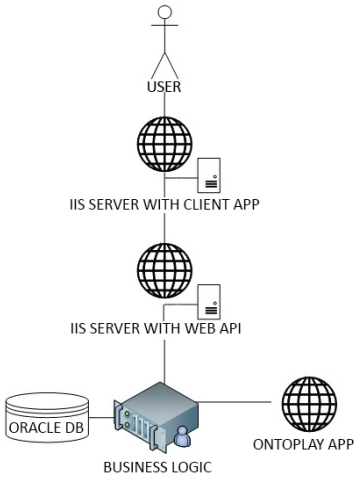

FIGURE 3. Architecture of semantic module

3.1 Client Layer

The Client Layer is responsible for displaying the graphical user interface in a WEB browser. It has been developed using the Angular Framework [26], HyperText Markup Language (HTML) and Cascading Style Sheets (CSS). The Client Layer has been divided into modules, which mirror tools described below (see, Section 4). These modules do not depend on each other. This approach allows easy development of the application, achieved by adding new modules without any side-effects on the other parts of the application. Defining each tool as separate module has one more advantage: each module can define its own template, style(s) and routing, making such tools more universal.

3.2 API Layer

As a WEB application, the Client Layer receives data from the server. The WEB API is a communication interface that returns data, in a JSON format (a browser-friendly data format). JSON-encapsulated data can be easily read by the browsers and maintained by Angular. To improve readability, for programmers, the WEB API has been arranged into Controllers (equivalent of modules), similarly to modules in the Client Layer. This way, one Controller is responsible for providing data for a single module. Controllers do not have and business logic inside them. To obtain the data they call the Data Layer.

3.3 Data Layer

Following good programming practices, we added layer that maintains our database, reads and updates it, communicates with the OntoPlay application (see, Section 3.5.2), and provides data from these two sources (if necessary). It also merges the data originating from them (for example, when getting an image of the object – an Individual from the Ontology, and an image from the database are needed). Only this layer has direct access to the ontology and to the database. It is divided into logical business parts: (i) management of ontology and individuals, and (ii) running, so called, "long term tasks" (see, Section 5.1).

3.4 Database

The structure of the original database (see, [11]) has not been changed. We have only extended it by adding new tables or, at least, columns to the already existing tables. The relational database is connected with a .NET Application, by the Oracle Data Provider for .NET.

3.5 External tools and libraries

In order to create the ontology and to "connect" it with the WEB application we have used a number of tools and libraries. Let us briefly describe them.

3.5.1 Protege

Protege was created at the Stanford University as a stand alone desktop application, which was used to construct the building ontology. It provides the graphical interface that allows one to create an ontology from the scratch, in order to generate hierarchy of the classes, define relationships between them, and manage instances or properties. External plugins also allow to validate the developed ontology [27].

3.5.2 OntoPlay

One of the frameworks that allows us to combine ontology, WEB applications and a database is OntoPlay [28, 29]. It provides a dynamic ontology-driven interface, which is ontology-agnostic and allows us to develop of web applications in an easy way. OntoPlay has a REST API, which was used in the Data Layer. It also has a WEB interface that can be used without third party applications. It allows the user to upload her/his own ontology and manage it. OntoPlay is built with Appache JENA, and includes Ontology Manager, which allows the user to view classes, instances of the classes, or properties in the ontology.

3.5.3 dotNetRDF

The dotNetRDF is a framework, developed for .NET applications. It offers an interface to manage RDF, SPARQL and ontologies, which is very useful when need arises to load ontology into an application, and fast view its necessary aspects, without calling external applications [30].

4 APPLICATION TOOLS

The developed semantic module has been implemented as a WEB application, which can be used by anyone with an Internet access. Within this application / module we have created six tools that allow using the CBIR database and the ontology together (making up the SCBIR). Let us describe each one of them in some details.

4.1 Ontology Viewer

The Ontology Viewer presents ontology in a user friendly way. It should be stressed that, thank to use of the OntoPlay, this viewer would work not only with the current (detached houses) ontology, but with any ontology that would be connected to the SCBIR. For users, who are not familiar with semantic technologies, it provides four screens:

1. Ontology general information and statistics;
2. Class Viewer – to show what classes are in the ontology and how they are hierarchically structured;
3. Properties Viewer – allows viewing properties of the ontology and their domains and ranges;
4. Instances Viewer – is screen, which lists all instances, occurring in the ontology, which have been grouped according to the class that they represent.

Obviously, this tool has been developed to be used by specialists / interested users. For a casual user, there is no need to delve into the actual structure of the ontology.

4.2 Objects from Database

The second tool presents all objects from the database. Having prepared ontology, in the developed application, one can try to manually assign a class from the ontology to a specific object. In order to maintain the main assumption of the Semantic Internet – the uniqueness of the URI – each object, added to the ontology, is represented by URL of the ontology, plus its database ID. Window of this tool, allows user to filter objects by their IDs, database class, ontology class, or file name. It also presents a thumbnail of the object. Details of a given object are presented in a special (separate) window, including a bigger image of the object (so it is easier to identify what it is), as well as all properties resulting from image processing, imported from the database object table. The most important issue is the

identifications history, and an interface that allows one to identify the object. Each user can suggest change of the object class. The Interface of this tool allows the user to fully describe object, according to the ontology restrictions (functionality, again, provided by the OntoPlay). Obviously, the change is not immediately applied and saved to the ontology. It has to be verified by an expert (*e.g.*, the owner of the image database). All proposed changes are serialized and temporarily stored in the database, until accepted or rejected by the authorized person.

5 ADMINISTRATION OF USER CHANGES

A module for the administrator of the SCBIR, allows one to view the changes that users have suggested for ontology and accept, or reject, them. If a rejection is made, the serialized object is removed from the database and there are no changes in the ontology. In case of accepting the suggestion, the serialized object is send to the OntoPlay, which is being used to manage changes in the ontology. At first, the module checks the existence of the individual labeled by its ID. If such object exists it is removed from the ontology. In its place, object send to the OntoPlay is added to the ontology

5.1 Long-term Tasks

In the CBIR database, there are more than 9000 segmented objects, in which $\sim$ 5800 segments have been classified. In the database, an intermediate table, called Identifications, with classified segments, exists. After using the four classifiers, available in the existing CBIR system, the $\sim$ 5800 classified segments were verified and are quite credible. Let us note that, in order to offer more options to the user, sometimes, two classes were assigned to an element. For instance, a gutter can be a vertical or a horizontal element. Additionally, there were segments difficult to recognize using either of the classifiers, which were labeled manually.

It would be time-consuming to explore all of these objects. That is why a "long-term task" was created, which works as "fire and forget" task. It takes all objects from the database and, based on identification that was made by the CBIR System, assigns an ontological class to each one of them. Next, the request is send to the OntoPlay, to make changes in the ontology. During this process, the following situations may materialize.

- **Case 1.** Objects does not have any identifications – none of the classes from the ontology can be assigned to it. This case is being reported to the Administrator to deal with.
- **Case 2.** Object has only one identification type – if there exist class in the ontology, which has the same name it is assigned to the object; if not, logs are added with information about lack of the class in the ontology. (This is a good case to check if a specific class is not missing in the ontology).
- **Case 3.** Object has one or more identification types, but one of them is manual. In this case the process will behave like in Case 2, with only one identification type – manual.
- **Case 4.** Object has more than one identification type and it has no manual identification type. For each pattern, process counts the number of occurrences of it. The winner class is the one with the biggest count. In case of a tie, a random class is selected. Next, for this class, process acts like in Case 2.

5.2 SPARQL Queries

As a part of the "toolbox", the user interface for SPARQL queries was also created. Advanced users can search the ontology with the SPARQL syntax. Module provides also a few examples of queries such as: find all the triples in the ontology or find individuals that have property *hasColor* with value *Green*.

6 SCBIR – ILLUSTRATION HOW IT WORKS

While the main goal, of the current stage of the project, was to develop a hybrid CBIR system (the SCBIR), one of more specific sub-goals, was to facilitate GUI that would give the user a possibility to "construct a building" from the selected parts. Properties introduced to the ontological hierarchy (see, Section 2.2), allow organization of selected architectural elements so that they appear on the screen in a "natural order". This is achieved as a result of a three step process.

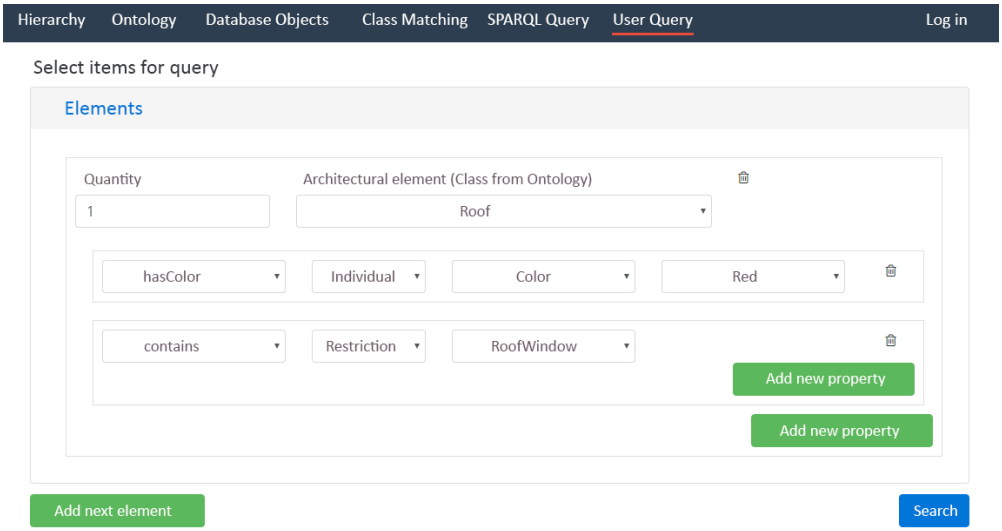

FIGURE 4. User Query Form – Step 1 – selection of architectural elements – constrains

The first step consists of selection of the part of the building. WEB Form facilitates construction of the User Query. The process is dynamically driven by the underlying ontology. Having selected an ontological class, one can add more complex restriction(s) to it. An example of constructing such a query is shown in Figure 4.

For example, let us assume that a roof is selected. However, the selection can go further, for example, a roof, which has color *Brown* and contains *Antenna* on it. In the ontology we have properties such as: *hasColor* and *contains*. As mentioned in Section 2.1, the Domain of the property *hasColor* is naturally any part of the building and the instances of the *Color* class are the Range. Any part of the building can contain any other part as its part. The user can also select quantity of an element (for instance, if (s)he wants to have more than one element – for instance: two windows in the building, one roof, *etc.*). The user can construct more complex restriction, such as: (s)he wants to have a wall, which contains a window, which has a white color. Having added all parts, the appropriate request is being made and send to the OntoPlay, which manages the ontology. OntoPlay temporally adds a new class expression to the ontology, in order to get individuals satisfying the query. The query results are sent back to the Ontobrain application. Before displaying the results, object images are retrieved from the database, and then we send them to the user – matching by an Individual Id – which is the same as database Id of an object. Depending on the imposed restrictions, one can obtain different number of retrieved images.

In the second step, user decides, which images will be incorporated in the building, as shown in Figure 5. User applies navigation (left/right) to select a specific part of the building (images), chosen in the previous step. For instance, if the user had selected that (s)he wants a red *Roof*, (s)he will be able to choose only these images that have class *Roof* assigned, and have property *hasColor* set to the *Red*. Choosing a specific part is done by clicking on the image.

In the last step, we sort the selected architectural elements for the user, by using ontology properties in order to organize the elements in a layout logical for human. We have developed an algorithm that spatially sorts architectural elements. Sorted elements are sent back to the user, based on their spatial properties. The algorithm iterates through classes, from the user query, sends request to the OntoPlay, to return all properties, Domain of which is set to the currently iterated class. From these properties, only these with prefix that starts with *isAboveOf* and the suffix contains name of class that user selected (there could be multiple) are included. Then the second request is sent to the OntoPlay, to return all classes that are set as a Range of chosen properties, from the previous step, and filter them to only these, which are selected by the user. The application checks if currently iterated class has bigger index than any element that is in Range of selected properties. If yes, those two classes have to be swapped – this means that both elements were in a wrong spatial order. The same reasoning is being made for the *isOn* property. Because of the way that the images exist in the database, we do not "place them one on another" – but parts that are in a specific spatial relationship to each other, are displayed accordingly. Example of the result of the sorting procedure is presented in Figure 6.

The two elements that are located side-by-side, suggest that there is an actual overlap, such as: in the top-left we

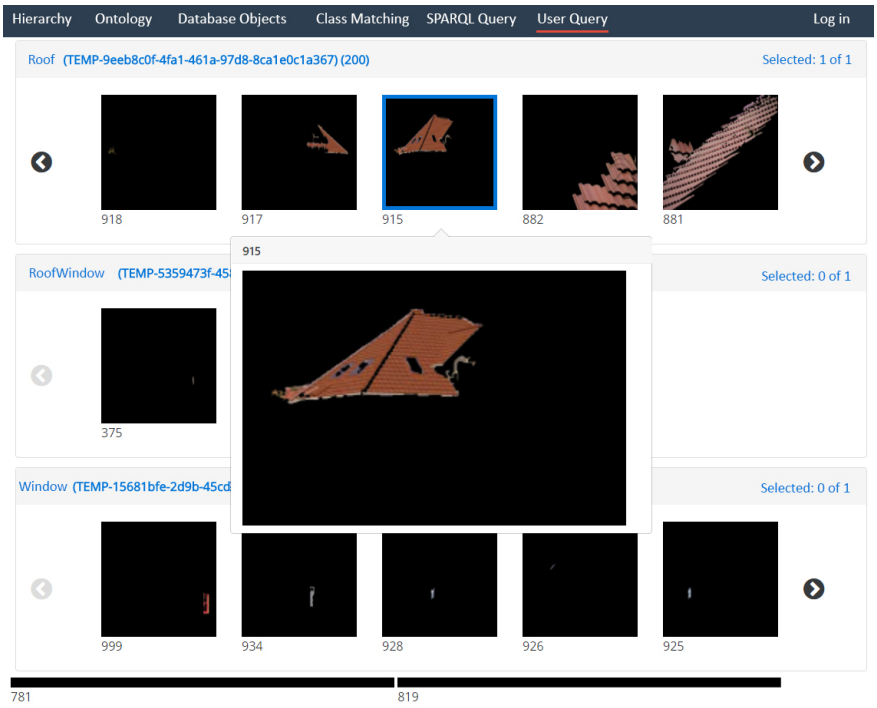

FIGURE 5. User Query Form – Step 2 – selecting specific parts of the building (images from the database)

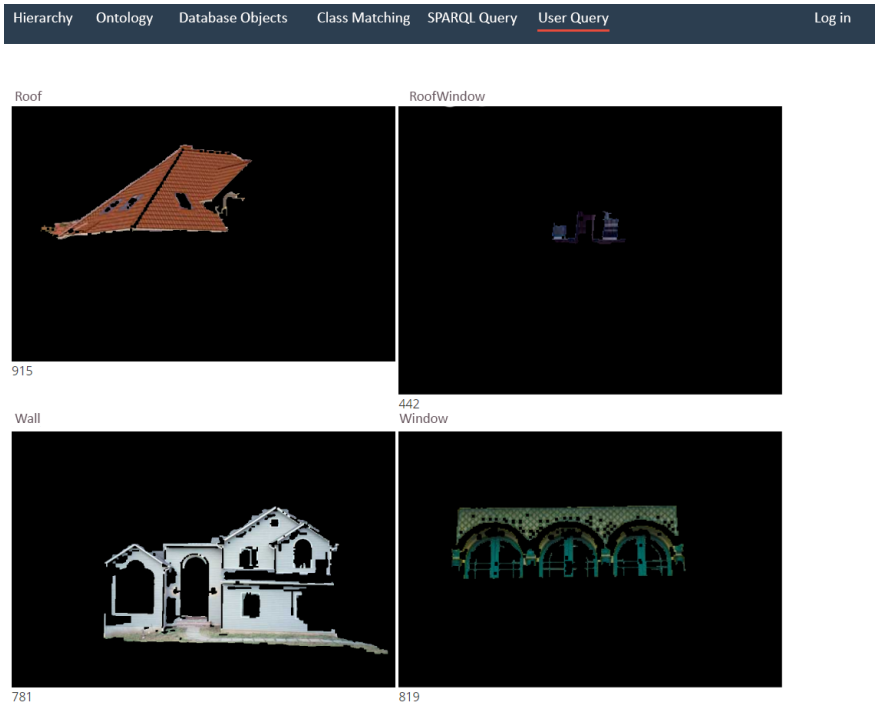

FIGURE 6. User Query Form – Step 3 – presenting the sorted result of the user query.

have *Roof*, in the top-right we have *Skylight*. Bottom-left presents *Wall* and on the *Wall* we have a "normal" *Window* visible on the bottom-right corner. The hierarchy between *Roof*, *Skylight*, *Wall* and *Window* is preserved in this figure. *Roof* is on top of the *Wall*, *Window* is within the *Wall* and *Skylight* is on the *Roof*.

CONCLUDING REMARKS

The, long-term, aim of our work is to combine advantages of CBIR systems and semantic technologies to develop a hybrid SCBIR system. To this effect, in this contribution, we have started from an existing CBIR system, which provided us with a database of segmented images of detached homes and their surroundings. For this database, we have instantiated an ontology, formally representing the domain. This ontology consists not only of elements captured by the image processing modules, but includes also information representing spatial relations between them. Next, we have developed a new module that, among others, facilitates the use of semantic technologies for retrieval of images from the database of the CBIR system, using semantic technologies. The developed module allows also retrieval of image segments and ordering them according to the way they exist in the real world.

Work, reported here, represents an initial step towards creation of an ultimate SCBIR. We will report on our progress in subsequent publications.

REFERENCES

[1] R. Kerry and K. R. Wood, "How do people manage their digital photographs?" in *SIGCHI Conference on Human Factors in Computing Systems* (Ft. Lauderdale, Florida, USA, April 510, 2003), pp. 409–416.

[2] D. Sagarmay, *Multimedia Systems and Content-Based Image Retrieval* (IDEA Group Publishing, 2004).

[3] R. Yong and H. T. S., "Relevance feedback techniques in image retrieval," in *Principal of Visual Information Retrieval*, edited by M. S. Lew (Springer, London, 2001), pp. 219–258.

[4] L. Fei-Fei and P. Perona, "A bayesian heirarcical model for learning natural scene categories," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition CVPR* (IEEE, Los Alamitos, CA, 2005), pp. 524–531.

[5] T. Kinnunen, J.-K. Kamarainen, L. Lensu, and H. Klviinen, "Unsupervised object discovery via selforganisation," in *Pattern Recognition Letters*, Vol.33 (Elsevier, Los Alamitos, CA, 2012), pp. 2102–2112.

[6] E. L. van den Broek, T. Kok, T. E. Schouten, and L. G. Vuurpijl, "Human-centered content-based image retrieval," in *Proceedings of XIII Conference on Human Vision and Electronic Imaging*, Vol.6806 (SPIE – The International Society for Optical Engineering, Feb. 14, 2008), pp. 1–12.

[7] A. Bursuc and T. Zaharia, "Artemis@ mediaeval 2013: A content-based image clustering method for public image repositories," in *ACM Multimedia* (ACM, Barcelona, Spain, 2013).

[8] P. William and S. Gerald, "Visualization and browsing of image databases," in *Multimedia Analysis, Processing and Communications*, edited by W. Lin, D. Tao, J. Kacprzyk, Z. Li, E. Izquierdo, and H. Wang (Springer, Berlin, 2011), pp. 3–57.

[9] T. Jaworska, "Image processing for cbir system," in *Proceedings of the ICINCO 4th International Conference on Informatics in Control, Automation and Robotics* (Angers, France, 2007), pp. 375–378.

[10] T. Jaworska, "Multi-criteria object indexing and graphical user query as an aspect of content-based image retrieval system," in *Information Systems Architecture and Technology*, edited by L. Borzemski and A. G. *et al* (Wydawnictwo Politechniki Wroclawskiej, Wrocaw, 2009), pp. 103–112.

[11] T. Jaworska, "The inner structure of database for the cbir system," in *Proceedings of International Conference on Computational Intelligence for Modelling, Control and Automation - CIMCA08*, edited by M. Mohammadian (Wien, Austria, 2008).

[12] T. Jaworska, "Cbir search engine for user designed query (udq)," in *Proceedings of the 7th International Join Conference on Knowledge Discovery, Knowledge Engineeing and Knowledge Management*, Vol. 1 (SCITEPRESS, Lisbon, Portugal, 2015), pp. 372–379.

[13] T. Gruber, *Ontology*, `http://tomgruber.org/writing/ontology-definition-2007.htm`, 2009.

[14] N. F. Noy and D. L. McGuinness, Ontology development 101: A guide to creating your first ontology, `http://protege.stanford.edu/publications/ontology_development/ontology101-noy-mcguinness.html`, 2017.

[15] DBpedia, `http://dbpedia.org/ontology/Building`, 2017.

[16] P. Buttigieg, *GAZ - Enviroment Ontology*, https://github.com/EnvironmentOntology/gaz, 2015.
[17] J. Hartz, Homepage, https://kogs-www.informatik.uni-hamburg.de/~hartz/, 2017.
[18] TRIMS, eTRIMS - E-Training for Interpreting Images of Man-Made Scenes, http://www.ipb.uni-bonn.de/projects/etrims/results.html, 2010.
[19] ResearchGate, Thomas J. Lampoltshammer page on ResearchGate, https://www.researchgate.net/profile/Thomas_Lampoltshammer, 2017.
[20] ONTOLOG collaborative work environment - historic archives, http://ontolog.cim3.net, 2017.
[21] C. Common, Swoogle - semantic web search, http://swoogle.umbc.edu/, 2007.
[22] M. Ganzha, M. Paprzycki, W. Pawowski, P. Szmeja, K. Wasielewska, and C. E. Palau, "From implicit semantics towards ontologiespractical considerations from the inter-iot perspective," in *Proceedings of the Globe-IoT Towards Global Interoperability among IoT Systems*, 2017. (in press)
[23] K. Goczya, *Ontologie w Systemach Informatycznych [Ontologies in Information Systems]* (Exit, 2011).
[24] P. Markiewicz, *Budownictwo Oglne dla Architektw [Civil Engineering for Architects]* (Archi-Plus, 2001).
[25] B. Wapinska and M. Popek, *Podstawy Budownictwa [Basics of Civil Engineering]* (2009).
[26] Angular, One framework. Mobile and Desktop, https://angular.io/ (2017).
[27] Protege, A free open-source ontology editor and framework for building intelligent systems, http://protege.stanford.edu/, 2017.
[28] M. Drozdowicz and M. Al-Wazir, *Ontoplay*, https://github.com/mdrozdo/OntoPlay (2017).
[29] M. Drozdowicz, M. Ganzha, M. Paprzycki, P. Szmeja, and K. Wasielewska, "Ontoplay – A flexible userinterface for ontology-based systems," in *bibinfo booktitle Proceedings of the First International Conference on Agreement Technologies, AT 2012, Dubrovnik, Croatia, October 15-16, 2012* (2012), pp. 86–100.
[30] dotNetRdf, An open source .net library for rdf, http://www.dotnetrdf.org/ (2017).