Exploiting Context and Semantics for UAV Path-finding in an Urban Setting

Marjan Alirezaie, Andrey Kiselev, Franziska Klügl, Martin Längkvist, and Amy Loutfi

Machine Perception and Interaction Lab, AASS
Örebro University, Örebro Sweden
{firstname.lastname}@oru.se

Abstract. In this paper we propose an ontology pattern that represents paths in a geo-representation model to be used in an aerial path planning processes. This pattern provides semantics related to constraints (i.e., flight forbidden zones) in a path planning problem in order to generate collision free paths. Our proposed approach has been applied on an ontology containing geo-regions extracted from satellite imagery data from a large urban city as an illustrative example.

Keywords: Semantic Web for Robotics, Representation and reasoning for Robotics, Ontology Design Pattern, Path Planning

1 Introduction

Path planning for unmanned aerial vehicles (UAVs) in urban environments is becoming of increasing importance as the usage of drones is increasing both for recreation as well as urban monitoring. Aerial path planners would have the potential to provide greater safety for operating these vehicles, especially when the line of sight for operators is not always available, or there are limitations in real-time feedback. For such path planners, flight elevation is a critical parameter, which depending on the scenario, can have varying requirements [1]. For instance, consider a scenario where a drone is expected to monitor a flooded area in order to acquire images useful for the decision making process of a rescue team. Keeping a close distance to the object of interest is an important constraint to maintain. In addition, a path planner should also consider contextual information about the environment. For example, another scenario may require that the same type of aerial vehicle avoids residential zones in order to preclude privacy breach of individual citizens. Thus, an automated path planner for aerial vehicles needs, not only the geometrical constraints but also the contextual information about the surrounding objects in order to generate an admissible path. This contextual information augmented to geographical objects can imply further characteristics including the structure and affordances of the objects in a given environment. The term "*kindergarten*", for example, assigned to a region

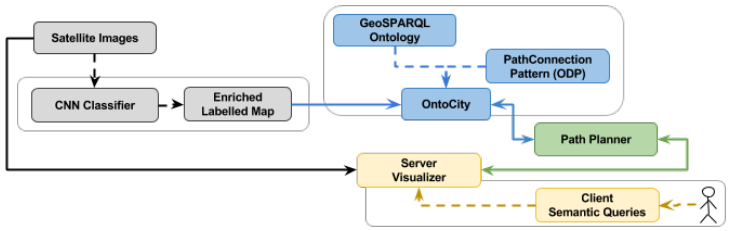


Fig. 1. Semantically Enabled Path Planning System includes: data layer in gray, semantic layer in blue, path planner in green and a visualizer in yellow.

can provide semantically meaningful information about the structure of the region e.g., "*the region is composed of at least a building and a playground*", or likewise, the term "*Bridge*" given to a region indicates that "*there is water area in the vicinity of the region*".

In this paper we exploit semantic information available in satellite images of urban maps in order to contribute in path finding and planning process for UAVs. Further and how they affect the complexity of a path planning process. Although a precise answer to this question depends on many factors including the planning method, we focus on those set of problems where the constraints increase the searching time and consequently the complexity of the path finding process. More specifically, in this paper, we show how exploiting the semantics of constraints given in a path planning problem can improve the time complexity of the planning process, which otherwise could be undesirably high.

Enabling path planners with semantics is not novel and has been studied in the literature. Many of the studies are targeting indoor environments where the semantic representation of the given objects either helps to interact with humans [13] who guide the robot to avoid the obstacles or provides a taxonomy of objects which contributes in generating a reachability graph [14]. Path planning in outdoor environments has also benefited from the semantics of objects to ease the interaction with humans during the navigation [15]. There are also other ideas of using semantics to define places as well as robot's actions related to navigation [16]. However, in the aforementioned related work, there is a semantic model which is very specific to the given environment or the navigation problem. In this paper, we emphasize more on the semantic model for outdoor environments and propose a generic representation of paths in the form of an ontology design pattern (ODP) [7] that can be reused in the design of other geo-related ontologies. More specifically, we will show how our proposed pattern applied on existing ontologies can be automatically queried by a path planner and result in information useful to avoid obstacles.

Figure. 1 illustrates the main structure of our system including a path planner enabled with semantics to better function in outdoor environments. As mentioned earlier, we use satellite images of urban maps (in this paper, the central part of Stockholm) provided by Vricon [10]. In order to be used by a path planner, this data is first processed by our CNN-based classifier which results in a

set of 2D segments each is assigned with a label (e.g., building, water, etc) and an elevation value. The details of the classification process is out of the scope of this paper and can be found in [11, 12]. As shown in Fig 1, given the labeled segments as the result of the classifier, the path planner (in green) together with the semantic model (in blue) will be able to respond the queries made by the user via a 3D visualizer (in yellow). These processes are going to be explained in the following order: In section 2 we briefly explain path planning problem and a specific path planning algorithm studied in this paper. The details of our proposed semantic model (ontology pattern) in conjunction with the reasoning process are also explained in section 3. In section 4 we show how using the proposed ontology pattern can affect the path planning time in practice. We end this paper with a discussion on the possible extension of the work in the future.

2 Path Planning Problem

Let $\mathrm{X} \subseteq \mathbb{R}^d$ ($\mathrm{d} \in \mathbb{N}$, $\mathrm{d} \geq 2$) be a d-dimensional configuration space whose members indicate all the states of a vehicle or a robot's body in terms of its joint angles. Let X_{obs} and X_{free} also represent the obstacles and obstacle-free space, respectively, where: $\mathrm{X}_{free} = \mathrm{X} \setminus \mathrm{X}_{obs}$. Assuming that the initial condition $\mathrm{x}_{init} \in \mathrm{X}_{free}$ and $\mathrm{X}_{goal} \subseteq \mathrm{X}_{free}$, a path planning problem is defined as a triplet (X_{free}, x_{init}, X_{goal}) [4], whose solution is a collision free path from x_{init} to X_{goal}. Path planning methods which are expected to find such solutions considering set of constraints, are categorized into 5 main groups including sampling based, node based optimal, mathematic model based, bio-inspired based and multi-fusion based algorithms [2]. In this paper, our focus is on sampling based algorithms as used in many robotic applications. They are on-line and reasonably quick in finding a possible path from a given source to a given destination.

2.1 Sampling Based Path Planning Method

A sampling-based path planner relies on generating a tree whose nodes represent samples that are randomly selected from a given configuration space X. The construction process of the tree continues until either a feasible path from a given x_{init} to a given goal is found or the searching time expires [3]. Rapidly-exploring Random Tree (RRT) and Probabilistic Roadmaps (PRM) are the important examples of sampling-based planners. Although RRT and PRM differ in details of the tree construction process, they are both mainly based on a random sampling process upon the configuration space X.

Algorithm 1 shows how the RRT random tree $\mathcal{T}$ rooted at the initial sample x_{init} is built [3]. Within a certain amount of steps (k times) the algorithm randomly fetches a configuration sample and adds it to the tree by linking it to the nearest node in the tree provided that the connection is collision free. In order to find a collision free path, these samples should necessarily be located in X_{free}. Although we do not necessarily need to construct a configuration space in the beginning, however, it is possible to end up with situations where no solution is

found due to the time limit for a collision-free path searching process in a highly constrained environment [3].

Algorithm 1 RRT Tree Construction

1: $\mathcal{T}$.init(x_{init})
2: **for** i = 1 to k **do**
3: $\mathcal{G}$.add(x_{init})
4: $x_{rand} \leftarrow RAND\_CONF()$
5: $x_{near} \leftarrow NEAREST\_VERTEX(x_{rand}, \mathcal{T})$
6: $x_{new} \leftarrow NEW\_CONF(x_{near})$
7: $\mathcal{T}.add_vertex(x_{new})$
8: $\mathcal{T}.add_edge(x_{near}, x_{new})$
9: **end for**
10: **return** $\mathcal{T}$

In case of an UAV, obstacles are understood as either elevation constraints or semantics constraints. In this paper, our goal is to show how using the semantics we can update the configuration space X and therefore reduce the time required for a collision-free path searching process. For the sake of simplicity, we consider a simple drone with 1-dimensional configuration, where each single configuration is represented as a 3D point in the space. In this case, the configuration space X is represented in 3D, where x, y and z coordinates are limited within a specific numeric range depending on the width, length and height size of the environment.

3 Semantics in Path Planning

Each labeled segment generated by the data classifier is represented in our ontology called OntoCity (see Fig 1). The OntoCity ontology relies on GeoSPARQL proposed by OGC as a standard vocabulary for geospatial data in RDF that enables qualitative spatial reasoning upon this type of data [5]. The concept ***Feature*** is a general concept in GeoSPARQL which defines any spatial object that has a geometry. By extending this concept in OntoCity, we create a taxonomy of geographical features. The main concept subsuming the ***Feature*** class is called **ontocity:*Region*** which is categorized into 2 main subclasses **ontocity:*ManmadeRegion*** and **ontocity:*NaturalRegion***. Due to the lack of space, we suffice to briefly mention another categorization of the class **ontocity:*Region*** indicating the type of regions on the ground such as **ontocity: *VegetationArea***, **ontocity: *WaterArea*** and **ontocity:*PavedArea***. Each of these classes is subsumed by more specific classes that are as such defined as either man-made (e.g., **ontocity:*Road*** $\sqsubseteq$ **ontocity:*PavedRegion***) or natural (e.g., **ontocity:*River*** $\sqsubseteq$ **ontocity: *WaterRegion***) regions. A region can also represent a way (i.e., a transportation route) located between two places. The class **ontocity: *Way*** $\sqsubseteq$ **ontocity:*Region*** is represented for this purpose and subsumes classes such as **ontocity:*Road*** (and subsequently **ontocity:*Street***, **ontocity:*HighWay***, etc) and **ontocity:*River***. Each generated segment by the classifier which is henceforth referred to as region, is represented as an instance of a subclass of the class

ontocity:*Region* in OntoCity equivalent to its label. This representation also includes the geometry of each segment as well as pairwise topological relations between any two segments.

3.1 The PathConnection Ontology Pattern

Although OntoCity extends the GeoSPARQL ontology by providing a taxonomy of geo-related concepts, it requires further semantics to contribute in path planning processes. In OntoCity, there is a representation of transportation routes in the form of the class **ontocity:*Way*** whose subclasses are **ontocity:Road**, **ontocity:Street**, **ontocity:River**, etc. To provide further details (e.g., the regions connected to the route) an extension is required. Representation of a path with its involved regions asks for a relation that is defined between more than two elements. More specifically, a path is seen as an n-ary relation semantics that can be represented via the generic n-ary[1] ontology pattern.

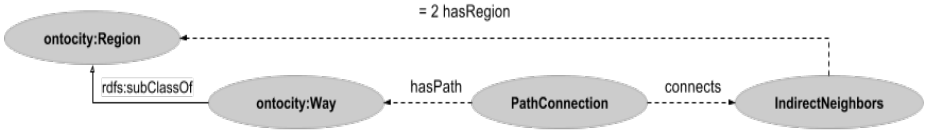


Fig. 2. The PathConnection Pattern representing a symmetric n-ary relation.

However, the n-ary path relation in our case is a symmetric n-ary (SNAry) relation. According to [9], if z is the value of an SNAry relation for the two elements x and y, then: SNAry(x,y) = z $\iff$ SNAry(y,x) = z. We likewise assume that the region x is connected to the region y via the path z, if and only if the region y is also connected to region x via the same path. As illustrated in Fig. 2, our ontology pattern called PathConnection relies on the SNAry relation and contains the two classes ***IndirectNeighbors*** and ***PathConnection***, where the former provides the link to the two regions which are represented in OntoCity and are connected via a path, and the latter indicates the path (as a **ontocity:*Way***) connecting these two regions together. The DL representation of the classes are as follows:

$$\textbf{\textit{IndirectNeighbors}} \sqsubseteq = \ 2 \ hasRegion.\textbf{ontocity:\textit{Region}}$$

$$\begin{aligned}\textbf{\textit{PathConnection}} \sqsubseteq \ & \exists \ connects.\textbf{\textit{IndirectNeighbors}} \sqcap \\ & = \ 1 \ hasPath.\textbf{ontocity:\textit{Way}}\end{aligned}$$

There are also two object properties, *hasPath*, *connects* and *hasRegion* with the following definitions that indicates their domain and range classes:

$\top \sqsubseteq \forall \ hasPath^{-}.\textbf{\textit{PathConnection}}$,$\top \sqsubseteq \forall \ connects^{-}.\textbf{\textit{PathConnection}}$,$\top \sqsubseteq \forall \ hasRegion^{-}.\textbf{\textit{IndirectNeighbors}}$
$\top \sqsubseteq \forall \ hasPath.\textbf{ontocity:\textit{Way}}$,$\top \sqsubseteq \forall \ connects.\textbf{\textit{IndirectNeighbors}}$,$\top \sqsubseteq \forall \ hasRegion.\textbf{ontocity:\textit{Region}}$

[1] https://www.w3.org/TR/swbp-n-aryRelations/

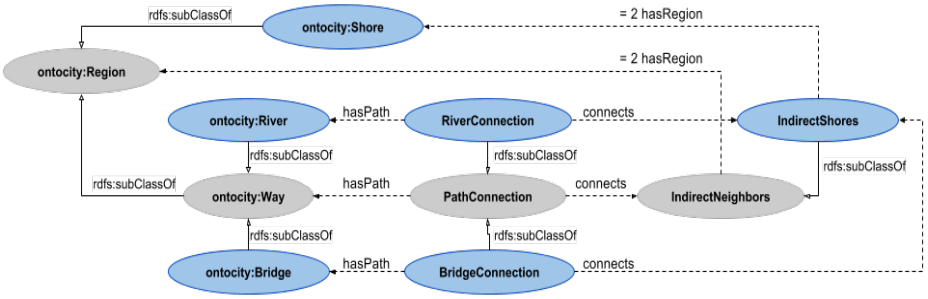


Fig. 3. The PathConnection Pattern - River and Bridge example.

By specializing the classes of the pattern, the ontology will contain more specific path connections. For instance, as shown in Fig. 3, the class ***PathConnection*** as the main class of the pattern subsumes the class ***RiverConnection*** whose path and region parts are defined as the subclasses of the class **ontocity:*River*** $\sqsubseteq$ **ontocity:*Way*** and the class **ontocity:*Shore*** $\sqsubseteq$ **ontocity:*Region***, respectively:

$$\begin{aligned} \textit{RiverConnection} \sqsubseteq\ & \textit{PathConnection} \sqcap \\ & \exists\ \textit{connects}.\textit{IndirectShores} \sqcap \\ & =\ 1\ \textit{hasPath}.\textbf{ontocity:}\textit{River} \end{aligned}$$

$$\begin{aligned} \textit{where}: \\ \textit{IndirectShores} \sqsubseteq\ & \textit{IndirectNeighbors} \sqcap \\ & =\ 2\ \textit{hasRegion}.\textbf{ontocity:}\textit{Shore} \end{aligned}$$

$$\begin{aligned} \textit{and}, \\ \textbf{ontocity:}\textit{Shore} \sqsubseteq\ & \textbf{ontocity:}\textit{GroundArea} \sqsubseteq\ \textbf{ontocity:}\textit{Region} \end{aligned}$$

Likewise, the class ***BridgeConnection*** is another specialization of the class ***PathConnection*** that defines bridges as paths over water areas as follows:

$$\begin{aligned} \textit{BridgeConnection} \sqsubseteq\ & \textit{PathConnection} \sqcap \\ & \exists\ \textit{connects}.\textit{IndirectShores} \sqcap \\ & =\ 1\ \textit{hasPath}.\textbf{ontocity:}\textit{Bridge} \end{aligned}$$

$$\begin{aligned} \textit{where}: \\ \textbf{ontocity:}\textit{Bridge} \sqsubseteq\ & \textbf{ontocity:}\textit{Way} \sqsubseteq\ \textbf{ontocity:}\textit{Region} \end{aligned}$$

Although path connections are defined as the sibling classes, they can be further related to each other due to the similarities that might exist in their types of *regions* or *ways* involved in their definitions. For instance, the two classes ***BridgeConnection*** and ***RiverConnection*** are similar in the sense that both connect shore areas. Finding these similarities might help a path planner to find better candidates or substitutes for the areas expected to be avoided.

3.2 Reasoning Upon the Ontology Pattern:

Given the populated ontology with all the regions and their connections, the system is asked to find an aerial path between two regions without passing upon

a specific region type $\mathcal{C}$ as a constraint. Assuming this region type is defined in the ontology, $\mathcal{C} \sqsubseteq$ **ontocity:*Region***, the following query given in DL-query syntax let the reasoner retrieve all the candidates of the region type $\mathcal{C}$ that are although different from $\mathcal{C}$ (and therefore legal for the sampling process of the path planner), they are similar to $\mathcal{C}$ in the sense that they play the same role as $\mathcal{C}$ in connecting specific region types:

```
ontocity:Region and (not C) and
inverse hasPath some (
        connects some (
                inverse connects some (
                        hasPath some C)))
```

This query is executed by the path planner during the tree construction process explained in Algorithm 1. In other words, involving the query process to the path planning algorithm results in adding extra steps between line 4 and 5. This extension has been shown in Algorithm 2 in red. Each time a random sample is fetched (line 4), the planner checks if it belongs the forbidden area $\mathcal{C}$ (line 5). If it is the case, the selected x_{rand} has to be replaced by a new sample taken from the alternative region $\mathcal{R}_{alt}$ retrieved by the reasoner as the result of the query.

Algorithm 2 Semantically Extended RRT Tree Construction

1: $\mathcal{T}$.init(x_{init})
2: **for** i = 1 to k **do**
3: $\mathcal{G}$.add(x_{init})
4: $x_{rand} \leftarrow RAND\_CONF()$
5: **if** $x_{rand} \in \mathcal{C}$ **then**
6: $\mathcal{R}_{alt} \leftarrow QUERY\_PATTERN(\mathcal{C})$
7: $x_{rand} \leftarrow RAND\_CONF(\mathcal{R}_{alt})$
8: **end if**
9: $x_{near} \leftarrow NEAREST\_VERTEX(x_{rand}, \mathcal{T})$
10: $x_{new} \leftarrow NEW\_CONF(x_{near})$
11: $\mathcal{T}.add_vertex(x_{new})$
12: $\mathcal{T}.add_edge(x_{near}, x_{new})$
13: **end for**
14: **return** $\mathcal{T}$

Although the regions retrieved by the running the query upon the ontology are authorized regions and therefore are considered as part of the X_{free} space of the path planner, it is worth finding these regions in order to emphasize on their existence and increase their chance to be chosen within the sampling phase. More specifically, without emphasizing on these alternative paths, there is the risk of not finding a path when the majority of the area is covered by the forbidden region $\mathcal{C}$. One may pose the question: why before running the sampling process, we do not exclude all the forbidden regions from the configuration space. The answer is that depending on type, size, number and the geometry of constraints (i.e., forbidden zones) this process can become highly time consuming. Furthermore, regardless of the location of forbidden zones, we have to always exclude

them from the configuration space. However, in our suggested approach we apply the reasoner only if an invalid sample is selected, which is perhaps compensated by an admissible alternative.

4 Scenario

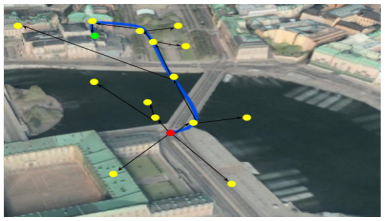
Using the populated OntoCity, the RRT path planner considers regions as obstacles. Each obstacle is represented in the form of a vertical 3D box whose base and height are equivalent to the bounding box of the region's boundary and the elevation average of the region, respectively. As shown in Fig. 1, the user is able to choose an initial and a goal point by clicking on the 3D map provided by the visualizer and asks for a collision free path. He/she can also mention specific region types as the constraints (i.e., forbidden zones) of the path finding problem.

However, avoiding regions due to their type or label is not always straightforward. Since the sampling based path planners are in theory incomplete, if the area is highly constrained, the process of finding valid samples and subsequently a collision free path (in terms of constraint satisfaction) is prone to fail. To illustrate, we use water as a constraint in the central part of Stockholm which is made of up of a number of islands. Given the water area as a constraint ($\mathcal{C} =$ **ontocity:*WaterArea***), the process of finding a path between two points located at two different sides of water will have a chance to successfully finish only if the sampling process takes advantage of the context of the scenario and samples from the bridges whose areas' size is unfortunately much smaller than the size of the environment.

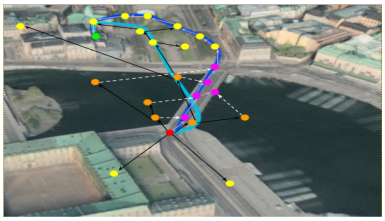
As explained in Section 3.1, our solution to this problem is to apply the extended RRT Algorithm 2 according to which, once a sample is taken from a forbidden area ($x_{rand} \in \mathcal{C}$), the reasoner runs the following query upon the ontology to find a substitute region ($\mathcal{R}_{alt}$) for water areas connecting two shores:

```
Region and (not WaterArea) and
inverse hasPath some (
       connects some (
              inverse connects some (
                     hasPath some WaterArea)))
```

Due to the given constraint, all the regions which are the instances of the subclasses of the class *ontocity:****WaterArea*** $\sqsubseteq$ ***Region*** (e.g., rivers) are seen as constraints. Given the forbidden region types in the form of constraints, the reasoner will retrieve all the connections (i.e., instances of the ***PathConnection*** class) whose path (i.e., a subclass of the class **ontocity:*Way***) refers to the instances of these regions. The reasoner can also find the types of regions (i.e., indirect neighbors) connected via a forbidden region (e.g., a river). Knowing these neighbors, the reasoner will be able to retrieve an alternative region ($\mathcal{R}_{alt}$) which is similarly connecting the neighbor regions in order to emphasize them during the sampling process ($x_{rand} \in \mathcal{R}_{alt}$).



(a) Constraint: Elevation of regions



(b) Constraint: Water Areas. Invalid and alternative samples are in orange and pink, respectively.

Fig. 4. Path Planning With Constraints. The initial, goal and valid samples are in red, green and yellow, respectively. The generated collision-free path is in dark blue.

Figure 4(a) shows a path that connects two points (x_{init} in red and $x_{goal}ingreen$) by crossing the river. Given water areas as forbidden zones, the planner would have difficulties in finding a path as the majority of samples are located in water areas due to their bigger area size. However, as depicted in Fig. 4(b), the path planner constructing the tree according to Algorithm 2, replaces samples taken from the river (x_{rand} in orange) with new samples taken from the bridge as an alternative (x_{alt} in pink) for the class **ontocity:*River***.

Table 1. Path Planner Performance

	Without Ontology Pattern	With Ontology Pattern
Success Rate	24.2%	91%
Average of execution time	8.4 sec	0.68 sec

We have run the algorithm for 70 different path problems with different initial and goal points located in the central part of Stockholm. Table 1 shows the success rates with and without using the ontology reasoning during the tree construction process of the planner. As we can see, the integration of semantics into the plan generation process increases the success rates from 24.2% (17 successful cases out of 70) to 91% (64 successful cases out of 70) within 10 seconds set as the time limit for path finding. Without using the ontology, the average time for generating a path approaches the set upper limit of the planning process.

5 Discussion & Future Work

In this paper, we explained our preliminary work on enabling RRT path planner with semantics to deal with the planning time complexity in highly constrained environments. We have proposed an ontology pattern to model path connections in the form of an n-ary relation. We also showed how involving the semantics of constraints can contribute in decreasing the time of a path finding process. Although in order to achieve such an improvement we need to first populate the ontology with all the path connections between regions, this population process needs to be done only once. It is worth mentioning that in this work, we were

only concern about path finding regardless of the length of the paths. As the next step, we will target path optimization with semantics as our objective function.

Acknowledgments. This work has been supported by the Swedish Knowledge Foundation under the research profile on Semantic Robots, contract number 20140033.

References

1. H.L. Wang, W.T. Lyu, P. Yao, et al. Three-dimensional path planning for unmanned aerial vehicle based on interfered fluid dynamical system. Chin. J. Aeronaut., 28 (1) (2015), pp. 229239
2. Liang Yang, Juntong Qi, J. Xiao and Xia Yong. A literature review of UAV 3D path planning. Proceeding of the 11th World Congress on Intelligent Control and Automation, Shenyang, 2014, pp. 2376-2381.
3. Steven M. Lavalle. Rapidly-Exploring Random Trees: A New Tool for Path Planning. 1998.
4. Karaman, Sertac and Frazzoli, Emilio. Sampling-based Algorithms for Optimal Motion Planning. Int. J. Rob. Res. 2011, pp. 846–894
5. Battle, R. and Kolas, D. Enabling the Geospatial Semantic Web with Parliament and GeoSPARQL. Int. J. Semant. web. 2012, pp. 355–370
6. K Janowicz. Observation-Driven Geo-Ontology Engineering. Int. J. Transactions in GIS. 2012.
7. Gangemi, A. and Presutti, V. Ontology Design Patterns. Handbook on Ontologies. International Handbooks on Information Systems, 2009.
8. I. Arpinar, A. Sheth, C. Ramakrishnan, E. Usery, M. Azami, and M. Kwan. Geospatial Ontology Development and Semantic Analytics. T. GIS 10 (4): 551-575 (2006)
9. Maria Poveda, Mari Carmen SuarezFigueroa. OntologyDesignPattern. `http://ontologydesignpatterns.org/wiki/Submissions:Symmetric_n-ary_relationship` (accessed on February 2017).
10. Vricon, Homepage. Available online: `http://www.vricon.com` (accessed on February 2017).
11. Längkvist, M. and Kiselev, A. and Alirezaie, M. and Loutfi, A. Classification and Segmentation of Satellite Orthoimagery Using Convolutional Neural Networks. J. Remote Sensing, 8 (4), (2016)
12. Alirezaie, M. and Längkvist, M. and Kiselev, A. and Loutfi, A. Open GeoSpatial Data as a Source of Ground Truth for Automated Labelling of Satellite Images. Proc. WS. on Spatial Data on the Web (SDW 2016), GIScience, pp. 5–8.
13. Borkowski, A. and Siemiatkowska, B. and Szklarski, J. Towards Semantic Navigation in Mobile Robotics. Recent Advances in Intelligent Information Systems, pp. 711-720
14. Uhl, K. and Roennau, A. and Dillmann, R. From Structure to Actions: Semantic Navigation Planning in Office Environments. IROS Workshop 2011
15. Drouilly, R. and Rives, P. and Morisset, B. Semantic Representation For Navigation In Large-Scale Environments. IEEE Int. Conf. on Robotics and Automation, ICRA 2015.
16. L. F. Posada, F. Hoffmann, and T. Bertram. Visual semantic robot navigation in indoor environments. ISR/Robotik 2014.