POLITECNICO DI MILANO
Corso di Laurea Magistrale in Ingegneria Informatica
Dipartimento di Elettronica, Informazione e Bioingegneria

Reasoning on the Whole Structure of Buildings using a Logical Relational Learning Framework

Relatore: Prof. Francesco Amigoni
Correlatore: Ing. Matteo Luperto

Tesi di Laurea Magistrale di:
Michele Launi, matricola 816933

Anno Accademico 2014-2015

Sommario

Il mapping semantico nell'ambito della robotica mobile autonoma coinvolge la classificazione di ambienti. Essa consiste nell'associare un'etichetta semantica (come 'corridoio' o 'ufficio') alle stanze visitate da un robot. Gli approcci più diffusi si basano su un ragionamento locale, in cui solo le caratteristiche di alcune stanze limitrofe sono considerate ai fini della classificazione di una stanza. In questa tesi, proponiamo un metodo che permette di ragionare *globalmente* sulle intere strutture degli edifici. Usiamo un framework di apprendimento logico e relazionale chiamato kLog. Rappresentiamo ogni piano di un edificio come un grafo, dove ogni nodo è una stanza, a cui sono associati un insieme di caratteristiche geometriche e una etichetta semantica, e ogni arco è un collegamento diretto tra due stanze. Forniamo degli strumenti software per creare dataset realistici, basati su planimetrie di edifici realmente esistenti. Valutiamo l'utilità del nostro approccio in tre applicazioni: l'etichettatura semantica di stanze, la classificazione di intere planimetrie di edifici e la validazione di dataset simulati. Le ultime due applicazioni non sono facilmente affrontabili utilizzando i metodi classici di mapping semantico, basati su un approccio locale. In questa tesi dimostriamo che i tre problemi sono risolvibili usando il nostro approccio globale.

Abstract

Semantic mapping for autonomous mobile robots includes the place classification task that associates semantic labels (like 'corridor' or 'office') to rooms perceived in an environment. The mainstream approaches are characterized by local reasoning, where only features relative to the neighbourhood of each room are considered in the process of classification. In this thesis, we propose a method that enables *global* reasoning on the whole structure of buildings. We use a statistical relational learning framework, called kLog. We represent each floor of a building as a graph, where each node is a room and is associated to a set of features and to a semantic label and edges represent doorways. We provide software tools to create realistic datasets, based on real-world floor plans. We assess the utility of our approach in three applications: semantic labeling of rooms, classification of entire floors of buildings, and validation of simulated datasets. The last two applications cannot be easily addressed by classical local semantic mapping methods. In this thesis we demonstrate that the three tasks are achievable using our global approach.

Ringraziamenti

Eccoci qui: un'altra fase della vita che volge al termine. E' il momento di fermarsi un attimo e abbandonarsi ai ricordi. Sono felice. Sono felice per il percorso fatto fino ad ora... ma soprattutto sono felice per le persone che mi hanno accompagnato lungo la via e che voglio qui ringraziare.

Desidero ringraziare Serena, primavera della mia vita, per l'immenso supporto ricevuto anche nei momenti più difficili.

Ringrazio i miei genitori, Daniela e Fernando, e i miei nonni, Ortensia, Paola e Giuseppe per tutto l'amore che mi hanno donato e per avere sempre creduto in me. Ringrazio l'altro mio nonno Giuseppe che è riuscito a trasmettermi molto, nonostante non lo abbia mai conosciuto. Ringrazio la mia bisnonna Rosa, con cui sono cresciuto e che mi ha lasciato inestimabili lezioni di vita. Ringrazio tutta la mia splendida famiglia e in particolare ringrazio i miei cugini Clara, Elena e Simone, senza i quali il mio naso starebbe meglio e senza i quali potrei avere delle pantofole decenti ai piedi.

Ringrazio tutti i miei amici e, siccome voi siete tanti e io sono sbadato, procederò per gruppi e non per singoli nomi... Ringrazio tutti NOI-la vendetta, per gli innumerevoli momenti di allegria e di festa, per le tante risate che ci siamo fatti assieme e per quelle che ci faremo. Ringrazio gli Amici ArcaCinemaGiovani e in particolare la Confraternita Goliardica Maranza, perché ogni Venezia potrebbe essere l'ultima... Ringrazio gli amici del gruppo di Milano (a cui un nome ben preciso non lo so dare) per le tante serate imbruttite. Gracias a todos mis amigos madrilenos, especialmente mis companeros de piso, para la experiencia irrepetible que he vivido. Ringrazio il Progetto VolT, per le interessantissime iniziative portate avanti assieme. Ringrazio il Gruppo Folkloristico Terra del Sole, per le tante belle esperienze vissute assieme e le diverse realtà che tramite voi ho conosciuto.

Un ringraziamento speciale va a tutti i miei compagni di università, con cui si è creata una grande amicizia oltre le mura del politecnico, senza i quali

non avrei mai mai mai potuto resistere tutti e cinque gli anni.

Ringrazio tutti i docenti con cui ho avuto a che fare nella mia carriera scolastica, dalle elementari fino all'università, che hanno contribuito nella mia formazione ben oltre i confini della didattica. Un ringraziamento particolare va al mio maestro di musica Giovanni Iuliano, per i tanti anni passati assieme e per il fondamentale metodo di studio che mi ha lasciato.

Infine ringrazio di cuore il professor Francesco Amigoni e Matteo Luperto, senza la cui guida, i cui consigli e i cui suggerimenti, non strettamente tecnici, non sarei potuto arrivare a conclusione di questo lavoro.

E poi ringrazio te. Sì proprio te, che mi sono sicuramente dimenticato di ringraziare prima ma che assieme a tutti gli altri hai contribuito a rendere unico questo viaggio.

Grazie a tutti.
Michele

Contents

Sommario **I**

Abstract **III**

Ringraziamenti **V**

1 Introduction **1**

2 State of the art **5**
2.1 Semantic mapping 6
2.1.1 The different concepts of map 6
2.1.2 Construction of a semantic map 9
2.1.3 Use of a semantic map 12
2.1.4 The role of semantic maps in our work 12
2.2 Graphs and kernel methods 13
2.2.1 Introduction to graphs 13
2.2.2 Kernel methods 14
2.3 Logical and relational learning 16
2.3.1 The learning problem 17
2.3.2 The logical language 18
2.3.3 The two learning settings 19
2.3.4 LRL tasks and applications 20
2.3.5 LRL and semantic mapping 21

3 Relational and logical representation of buildings **23**
3.1 The included information 24
3.2 The relational models 25
3.3 Labeling schemas 27
3.4 Logical representation 29

4 Knowledge representation in kLog **33**
4.1 kLog . . . 33
4.1.1 General overview . . . 34
4.1.2 kLog language . . . 34
4.1.3 Available tasks . . . 36
4.1.4 Kernel method . . . 37
4.1.5 Examples of application . . . 42
4.2 Our kLog representation . . . 42
4.2.1 kLog representation of the first model . . . 43
4.2.2 kLog representation of the second model . . . 45

5 Software tools for creating the datasets **49**
5.1 Floor Plans Editor (FPE) . . . 49
5.1.1 General schema . . . 50
5.1.2 Creation . . . 50
5.1.3 Editor . . . 53
5.1.4 Post-editing . . . 56
5.2 RoomsPy . . . 57
5.2.1 Input formats . . . 58
5.2.2 Feature extraction . . . 58
5.2.3 Output formats . . . 60

6 Experimental evaluation **63**
6.1 Task definition and evaluation methods . . . 64
6.2 Place classification . . . 67
6.2.1 Data and results . . . 68
6.2.2 Analysis of the results . . . 69
6.3 Building classification . . . 70
6.3.1 Data and results . . . 72
6.3.2 Analysis . . . 74
6.4 Dataset validation . . . 75
6.4.1 Data and results . . . 77
6.4.2 Analysis of the results . . . 77

7 Conclusions and future work **81**

Bibliography **85**

List of Figures

1.1 An example of a highly-symmetrical office building. 2

2.1 Example of a multilevel architecture, from [72]. The maps are represented from the bottom upwards from the less abstract to the more abstract. 7

2.2 Example of kernel trick. The objects of the original space are mapped in a new space where a linear separation is possible. Left: the original space. Right: the new space. 15

2.3 The usage of SRL techniques to obtain a semantic world starting from a simulated environment, from [53]. 22

3.1 Basic elements of an E/R model. 26

3.2 The E/R schema of MODEL1. 27

3.3 The E/R schema of MODEL2. 28

3.4 Top: set of semantic labels for SCHOOL building type. Bottom: set of semantic labels for OFFICE building type. 30

4.1 An example of graphicalized data. The relation *on_same_paper* was not contained in the original dataset but has been derived from an intensional signature. 38

4.2 An example of NSPDK, from [28]. Top: pairs of neighborhood subgraphs at fixed distance 6 for radii 0, 1, 2, and 3. Bottom: pairs of neighborhood subgraphs of fixed radius 1 at distances 0, 1, 2, and 3. 40

4.3 An example of soft matching, from [28]. Only features generated by a selected pair of vertices are represented: vertices A and B at distance 1 yield a multinomial distribution of the vertex labels in neighborhoods of radius 1. On the right it is shown the contribution to the kernel value by the represented features. 41
4.4 An example of a dataset represented with MODEL1. 44
4.5 A partial example of a dataset represented with MODEL2. . . . 46

5.1 A screenshot of Floor Plans Editor (FPE). The left side contains the working area, the top-right part the labels of the SCHOOL building type, and the bottom-right part the buttons for the editing operations. 51
5.2 An example of floor plan of a school, from [69]. 52
5.3 An example of room insertion. Left: the insertion of the node representing the room. Center: the insertion of the vertex of the polygon representing the shape boundary of the room (i.e., the corners of the room). Right: the final result of the insertion, containing 6 walls and a door. 53
5.4 The result of the creation of a map, starting from Figure 5.2, using FPE. 54
5.5 A portion of the XML file derived from the map presented in Figure 5.4. 55
5.6 An example of correction of a missing door. Left: the map where the door was missing. Right: the corrected map. 56
5.7 An example of label correction. Left: the central room labeled with the wrong label. Right: the same room with the correct label. 56
5.8 An example of endpoints automatic correction. Left: three separate points. Right: an unique point. 57
5.9 An example of automatic identification of connected rooms. Left: three rooms sharing doors with the corridor but still not connected in the topological graph. Right: the same rooms connected topologically to the corridor. 57
5.10 A portion of the XML file enriched with features derived from the XML file presented in Figure 5.5. 61

5.11 A portion of kLog ground atoms derived from the XML file presented in Figure 5.5 using MODEL1. 61

5.12 A portion of kLog ground atoms derived from the XML file presented in Figure 5.5 using MODEL2. 62

6.1 Examples of ROC curves. In red (A) it is represented the ROC curve of a perfect classifier, while in blue (D) it is represented the ROC curve of a random classifier. The orange (B) and the green (C) ROC curves belong to other two classifiers and it is possible to state that the orange curve represents a better classifier since it has a larger area under the curve than the green one. 66

6.2 Examples of ROC curves. In red (A) it is represented the ROC curve of a perfect classifier. The green (B) and the blue (C) ROC curves belong to other two classifiers and it is possible to state that the green curve represents a better classifier since it has a larger area under the curve than the blue one. 67

6.3 Results of place classification of SCHOOLs using MODEL1 and $\mathcal{L}_{R/C}$. Left: ROC curve, with an area under the curve of 0.96. Right: PR curve, with an area under the curve of 0.81. 69

6.4 Results of place classification of OFFICEs using MODEL1 and $\mathcal{L}_{R/C}$. Left: ROC curve, with an area under the curve of 0.95. Right: PR curve, with an area under the curve of 0.85. 70

6.5 Results of place classification of SCHOOLs using MODEL2 and $\mathcal{L}_{R/C}$. Left: ROC curve, with an area under the curve of 0.92. Right: PR curve, with an area under the curve of 0.74. 71

6.6 Results of place classification of OFFICEs using MODEL2 and $\mathcal{L}_{R/C}$. Left: ROC curve, with an area under the curve of 0.88. Right: PR curve, with an area under the curve of 0.66. 72

6.7 Results of building classification using $\mathcal{L}_{R/C}$. Left: ROC curve, with an area under the curve of 0.69. Right: PR curve, with an area under the curve of 0.71. 73

6.8 Results of building classification using $\mathcal{L}_{R/C/E/S}$. Left: ROC curve, with an area under the curve of 0.74. Right: PR curve, with an area under the curve of 0.73. 74

6.9 Results of building classification using $\mathcal{L}_{\text{SCHOOL}}$ and $\mathcal{L}_{\text{OFFICE}}$. Left: ROC curve representing a perfect classifier. Right: PR curve representing a perfect classifier. 75
6.10 A simulated office building generated using real-world data as inspiration (a) and a simulated office building generated without using real-world data as inspiration (b), from [4]. . . . 76
6.11 Results of dataset validation of RoboCub simulated offices using the labeling schema $\mathcal{L}_{R/C}$. Left: ROC curve representing a perfect classifier. Right: PR curve representing a perfect classifier. 77
6.12 Results of dataset validation of RoboCub simulated offices using the labeling schema $\mathcal{L}_{R/C/E/S}$. Left: ROC curve representing a perfect classifier. Right: PR curve representing a perfect classifier. 78
6.13 Results of dataset validation of RoboCub simulated offices using the labeling schema $\mathcal{L}_{\text{OFFICE}}$. Left: ROC curve representing a perfect classifier. Right: PR curve representing a perfect classifier. 79

Chapter 1

Introduction

This year I didn't go to school
This year I didn't work
This year I didn't die
This year I didn't bloom

Google Autocompletion

Semantic mapping for autonomous mobile robots involves the task of associating semantic labels (like 'room' and 'corridor') to spatial entities perceived within an environment by means of sensors.

A semantic label identifies an entity, indicates its function, and allows a more informed interaction of the robots with the environment. Most of the methods for semantic mapping follow an approach that, roughly, starts from the data perceived by sensors mounted on-board on the mobile robots (e.g., laser range scanners and cameras), extracts features from these data, and classifies the areas from which the data have been acquired using (supervised) machine learning techniques [38, 63, 71].

The mainstream approaches for semantic mapping are characterized by *local* reasoning: spaces are either considered independently (the semantic label of a space is determined only according to features relative to that space, as perceived by the robot) or in relation to their neighbourhoods (the semantic label of a space is determined according to the features relative to that space and to the labels of nearby spaces). However, when considering structured buildings, more complex relations between spatial entities clearly emerge.

An example is reported in Figure 1.1, which shows a floor plan of an office building. The building (A) is composed of two identical blocks of offices (B). Each block of offices is in turn composed of two specular sub-blocks (C), symmetrical with respect to the shared wall. These symmetry relations between parts of buildings are recurrent and can be easily found by looking at their global structure, but are difficult to detect by looking only at the local neighbourhood of each room. A global reasoning on the semantic maps of the buildings can thus provide new useful information, that could be integrated with the knowledge obtained by standard local methods. For instance, detecting symmetries within a building during an exploration task could help to transfer acquired knowledge to an unseen new part of the building, which is detected to be symmetrical to an already explored part.

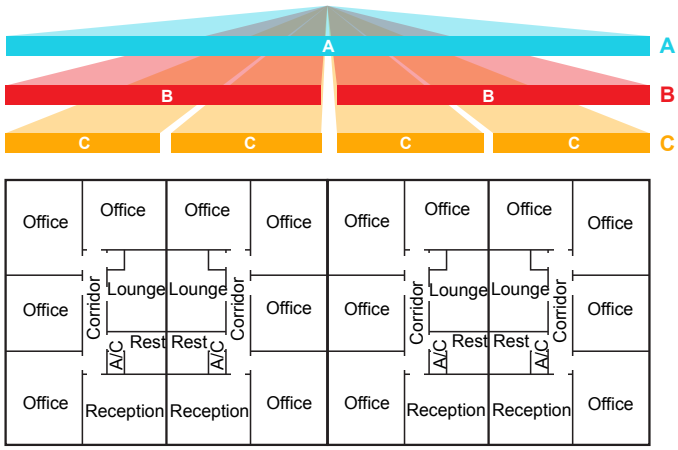


Figure 1.1: An example of a highly-symmetrical office building.

In this thesis, we propose a framework to reason about the global structure of buildings. Our method is based on Logical Relational Learning (LRL) [24], a subfield of Statistical Relational Learning (SRL) [35], which represents the meeting point of machine learning and logical knowledge representation. We show that the use of LRL techniques enables a *global* form of reasoning on the structure of the buildings, thus deriving new relations, regularities, and semantic features of environments.

In particular, we use a state-of-the-art LRL tool, *kLog* [28], originally applying it to the context of semantic mapping. Using this framework, we are able to perform some learning tasks already achieved by classical local methods, but we can also successfully carry out new learning tasks that could hardly be performed using local reasoning, such as validating a dataset or

classifying an entire building according to its function.

The thesis is structured as follows.

Chapter 2 contains the theoretical bases of our work. The first part of the chapter illustrates the state of the art of semantic mapping, presenting the approaches developed so far to build a semantic map. The second part of the chapter introduces briefly the concepts of graph, kernel method, learning algorithm, and kernel function. The third part of the chapter is about logical and relational learning, presenting the logical language used, the LRL representation of learning problems, and the available tasks.

Chapter 3 illustrates the three key factors needed in order to define a semantic map: which type of information to include, the level of detail of the model, and the set of semantic labels to employ. We describe in detail each factor, presenting the choices made in our models. In this chapter we also provide a first high-level translation of the elements of a floor plan into logical formulas.

Chapter 4 presents extensively the principal tool used in this thesis, kLog. We explain the logical relational learning setting used, the learning methods available, and the possible tasks that can be accomplished. In the end we present how we translated relational models in signatures written in the actual kLog language.

Chapter 5 is devoted to the creation of the datasets used in the learning tasks. We present the two tools we developed which allow to obtain kLog ground atoms representing a floor plan starting from an image. The first tool allows to build the metric map and the topological map of a floor plan, saving them in an XML file. The second tool is able to parse several input formats containing floor plans, compute geometrical features, and translate the result in kLog ground atoms.

Chapter 6 presents three experiments performed to evaluate our approach. The first experiment simulates the classical semantic mapping task to assign labels to the rooms of a floor plan, using different relational models and labeling schemas. The second experiment uses semantically labeled floorplans to establish their building type. The third experiment compares real-world floor plans and simulated floor plans in order evaluate if the virtual environments resemble to their real counterparts.

Chapter 7 contains our overall conclusions and some future work proposals based on our work.

Chapter 2

State of the art

You come from the sky.
The incarnate dream?
Then far from the light.
The incarnate dream?
Then there wasn't the sun,
the wind came cold.
The moon has no dream?
Magnet of the night
without stopping to me
quiet as the world I hear.

Ray Kurzweil's Cybernetic Poet

In this chapter some theoretical bases of our work are introduced, divided into three thematic areas: semantic mapping, graphs and kernel methods, and logical and relational learning. More specifically, in the first one, it is proposed a partial overview of the state of the art of semantic mapping, which introduces the concept of maps used in robotics up to the semantic maps and the presentation of the technology used to build and use them. In the second section, graph theory, kernel methods, learning algorithms, and kernel functions, are briefly introduced. In the last section, we present logical and relational learning, which represents the meeting point of machine learning and logical knowledge representation.

2.1 Semantic mapping

The ability of a robot to recognize the environment where it operates is of vital importance in order to carry out effectively its tasks. Furthermore, the more accurately a robot understands the environment, the more it can be efficient and autonomous. This is specifically true for those settings originally designed exclusively for human beings, but where there is anyway the need to insert a robotic element.

To allow the robot to operate effectively and interact with the environment, it becomes necessary to provide it with a map, namely, a representation of the surrounding space. This can be loaded into the machine or constructed by the robot itself using sensors, the robotic equivalent of our (human) sense organs. The robot should be able not only to build the map, but also to use its information in concrete tasks. Sensors are the instruments used to connect the real world with its representation in the map, for example locating the position of the robot. A reasoning system is needed to calculate the best way to accomplish a task and plan actions, for example selecting the fastest way to the goal.

2.1.1 The different concepts of map

The representation of the environment possessed by a robot can be composed of several maps, each one representing different features of the environment. Their level of abstraction depends on the tasks they are used for: from collision avoidance while moving into a room, to path planning between rooms, to exploration and search in a floorplan. The more the task to be achieved is complex, the more the robot has to have a detailed representation of the environment.

The maps are typically organized in multilevel systems, collections of maps ordered by an increasing level of abstraction [72]. These levels, also called layers, are strongly interconnected since a variation in the lower levels can have consequences on the higher levels and each one of them can store several maps, for example two maps representing the same features of the environment but obtained with two different processes. Figure 2.1 shows an example of a four-level architecture composed by a metric map, a navigation map, a topological map, and a conceptual map.

Metric map: in the lowest level ambient occupancy is represented, deter-

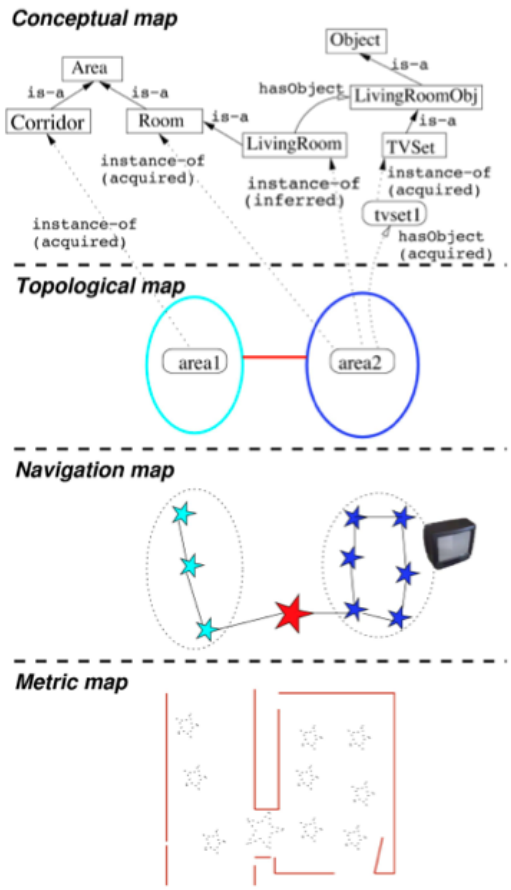


Figure 2.1: Example of a multilevel architecture, from [72]. The maps are represented from the bottom upwards from the less abstract to the more abstract.

mining the walkable portion of the space and the positions of the obstacles. When a robot is placed in an unknown environment, it does not have a metrical map of the place. It become thus necessary to obtain it using methods that take as input repeated sensor's scans from different points in the space. These methods take the name of Simultaneous Localization and Mapping (SLAM) and represent all the activities needed by a robot to reconstruct the metric map and correctly positioning itself in it. Some examples of SLAM can be found in [59, 60, 85].

SLAM is typically achieved employing laser scans, cameras, and, recently, Microsoft Kinect as sensorial input. These technologies can be used individually or more than one can be mounted on the same robot. Several representations are possible for a metric map such as a *grid*

map, where the space is divided in regular cells and each cell of the grid is tagged as free or occupied. Examples of grid maps can be found in [27, 62, 84]. A second type of representation is the one obtained converting sensor's space primitives into *points* and *line segments*. This map is incrementally built during exploration, clustering the points returned in lasers scans and approximating clusters with line segments as in [42, 51, 86]. A further step in this process consists in operating a second clustering among segments in order to obtain cleaner edges as in [66].

Navigation map: these maps represent the reachability of the space by the robot and are usually derived from the metric maps of the first layer. They contain the available paths, physical routes inside the environment, even inside a single room. Examples of these maps are described in [52, 68].

Topological map: at a higher abstraction layer topological maps can be found, where the spaces of the metric maps are clustered into interconnected sets of nodes. Typically each cluster represents a room or another closed space and the connections between them represent the doorways (physical, e.g., doors, or conceptual, e.g., arches). A label can be assigned to each room to represent information about its function. Labels can be introduced directly by the users or derived from an artificial reasoner according, for example, on geometrical features.

Conceptual map: at the top of the map layers, areas that form the basic spatial entities are coupled with finer semantical information derived from analyzing the data contained in the lower levels or inferred using a reasoning system. Conceptual maps are usually grounded to a commonsense ontology that helps the reasoning task by providing a taxonomy of the indoor spaces and of all the objects and characteristics that can be possibly found inside them.

A multilevel system as the one just described allows an incremental construction of the representation of the environment, typically building the higher layers on the lower ones, and is a widespread choice for semantic mapping.

2.1.2 Construction of a semantic map

A *semantic map* is built when semantic labels are associated to the spatial elements of a map, creating new information beyond the already existing. The presence of semantic labels helps the spatial knowledge of the robot and increases its performance especially on complex tasks and human interaction. *Semantic mapping* is the procedure through which a topological map is added to a basic metric map and to each element is given a semantic value. The first step, in order to construct a semantic map, is to obtain precise metric maps from which extract the features subsequently used to assign the labels. Two principal sources of features can be identified:

Raw sensors' data. With this approach, from an already defined metric map, the first step is the partition of the floorplan into well defined spaces representing the rooms. This process can use techniques like generalized Voronoi diagrams [9, 50] or hidden Markov models [76, 94]. The second step is to extract, for each identified space, several geometrical features that describe it such as its area, perimeter, or more complex ones as described in [58, 65].

Landmarks. A landmark is a recognizable feature used for navigation, a feature that stands out from the surrounding environment and that is often clearly distinguishable. This approach is usually employed when camera recordings are available and consists in identify univocally a portion of the space, in the form of a particular scene or object. The set of possible landmarks can be already part of the background knowledge of the system or can be incrementally expanded during exploration. Examples of the usage of landmarks are described in [48, 49, 77].

The next step in order to obtain a semantic map is the classification of the spaces, consisting in the attribution to each room of one or more semantic labels representing their functional role (e.g., office, corridor, kitchen). The possibilities for assigning the labels may vary greatly depending on the features resultant from the previous step.

A first possibility exploits the combination of the landmarks and a predefined ontology, under the assumption that a room can be labeled depending on the objects that contains. The robot tries to act as a human being, with its pre-acquired background knowledge, exploring the rooms and linking the

objects found to the elements of the ontology [19, 80, 97]. The assumption of this methodology is that the rooms contain an adequate number of landmarks, otherwise this system could not correctly work as reported in [73]. Another possibility is to link not a single object, but the whole appearance of the room itself to the ontology, using a photo or a frame from a video obtained with the camera [71]. The main limit of this approach is the size of the ontology since it must contain a vast variety of landmarks (i.e., images of rooms) to be effective and it can be difficult to keep it constantly updated [73]. Another problem raised in [70] is that the appearance of the room itself may change very much during a day, or with different lights, complicating the identification process.

A second possibility for assigning the labels is to use the results of the SLAM and the consequent features extracted, as described in [63]. The method is able to classify the lasers scans simultaneously while they are recorded and save the results in the semantic map incrementally. Once each Cartesian point of this two-dimensional map of the floor is classified, the map is divided in homogeneous portions with several methods. This can be done using a Bayesian approach as in [9, 30] or combining a clustering and a boosting phase [12, 63].

A common practice consists in the combination of the two presented sources of features, as in [30, 39], in order to obtain better performance since the two approaches are complementary: the laser classification constitutes a solid base and it is invariant to light conditions, while the camera observation returns more details. Recent works are mainly oriented in this direction, as in [73], where the authors present a system which exploits multiple sensors: three labels describing the environment are obtained independently by classifying the features with different methods and a final label is calculated by merging the three labels with a multimodal approach.

The idea of using data from the two different sources is used also in [97], where a general framework for semantic mapping robots is presented. It is based on a multi-layered spatial representation of the environment and allows different input data and classifiers to be easily integrated. A similar system, with a hierarchical semantic map built from laser range scanner and camera data is presented in [31]. The system in [64] classifies single laser range scans as belonging to rooms, corridors, hallways, or doorways using AdaBoost. A similar approach, which also uses features extracted from cameras, is reported in [65]. In the above works, a space is represented, in order to be classified, by

a vector of features obtained from the environment, following an attribute-value learning schema [24].

More recently, extensions of the these frameworks, that consider other sources of knowledge beyond the vectors of features, have been proposed. The system presented in [71] uses a probabilistic semantic mapping framework that applies semantic labels starting from six kinds of environment features (objects, doorways, room shapes, room size, appearance, and associated spaces). The semantic map is represented as a probabilistic chain graph model that generalizes Bayesian Networks and Markov Random Fields. The use of a chain graph allows to consider the uncertainties of the sensory models, to classify a newly perceived room also accordingly to the label of the rooms that are connected to it, and to predict the existence of a feature of a certain category (like a room and its label) in the unexplored space, extending the semantic map accordingly. Another approach, reported in [38], integrates metric, topological, and semantic representations with information derived from natural language. This is done using a factor graph formulation of the semantic properties and inferring these properties for unlabeled spaces by combining natural language descriptions with image- and laser-based scene classifications.

In the methods illustrated so far, the association of a semantic label to a perceived space is basically independent of the other semantic labels associated to previously seen spaces (with some relevant exceptions, like [71]). In this sense, we say that these approaches are *local*, since each space is classified using (almost) only the local features associated to the space itself, largely disregarding the structure of the whole map.

In [7], a rather different approach is introduced, focused entirely on the whole structure of a building. The authors consider a knowledge base of $38,000$ rooms (representing the MIT and KTH university campuses) and demonstrated that knowledge retrieved analyzing a set buildings with the same function (e.g., dormitory or research lab) can be transfered to an unexplored building of the same type. Differently from the previous methods, that of [7] operates at an higher level of abstraction, reasoning directly on maps of the highest layers without considering the underlying sensors' scans. Each floor of the buildings is represented as a graph, where nodes are rooms labeled according to their function (e.g., classroom, office).

2.1.3 Use of a semantic map

Once a semantic map is obtained, it can be used in several tasks, we list here briefly the most common:

Path planning: it consists in deciding which is the best path between two poses of the space. A semantic map can be used in the planning process to obtain faster or more secure paths that connects two rooms of a floor. Some examples are presented in [17, 50, 60]

Exploration: the objective is to chose an efficient exploration strategy for an unknown environment, for example chosing which is the best set of rooms to visit first. Having information about other floorplans, similar to the unexplored one, allows the robot to make better choices. During a single exploration phase it is important to store knowledge about the already visited portion of the space, expecially the typology of the rooms. Using a semantic map, the increase in performance can be remarkable, especially in the case of a coordinated exploration performed by a set of robots as shown in [3, 79, 81].

Search and rescue: it consists in a specific task operated under emergency conditions where the environment is potentially unknown. A high level of effectiveness is fundamental to reach the goal that may be, for istance, to find a victim of an accident and fastly lead him/her outside the building. An example is illustrated in [81], in which the semantic map results useful to improve both the efficiency and effectiveness of the search process.

Domotic: in a domestic environment, the map owned by the robot can be constructed with very accurate scans of the environment and then enriched, with the supervision of the householder, with semantic information. In this way the maps are constructed once and are constantly used by the robot in repetitive tasks such as object retrieval. A detailed description of such system can be found in [31].

2.1.4 The role of semantic maps in our work

To obtain a semantic map, we have seen how there can be various possibilities depending on the source of data and the consequent high-level reasoning.

Our model is based only on the laser scans of the environment and does not consider any camera as input. It is also similar to that of [7] since it uses graphs, representing topologically floorplans of buildings, as primary source of knowledge. We wanted to overcome the local reasoning on the single nodes and to focus on the global properties of a graph, exploiting the potential embedded in such structures. The detailed description of our data model will be presented in Chapter 3.

2.2 Graphs and kernel methods

Our approach is deeply grounded in a graph representation of a semantic map so in this section we will present some theoretical concepts about graphs.

2.2.1 Introduction to graphs

A *graph* is constituted by a set N of simple objects called nodes and by a set E of arcs (or edges) that connect them. Formally, a graph G is defined as a couple $G = (N, E)$ where:

$N = \{n_1, n_2, \ldots, n_n\}$ is the set of the nodes;

$E = \{(n_1, n_2), (n_1, n_3), \ldots, (n_{n-1}, n_n)\} \subseteq N \times N$ is the set of the arcs.

Specific information, for example a numerical value or a label, can be associated to both nodes and arcs, in this case the graph takes the name of *weighted graph*. The most general interpretation of the meaning of a graph is the representation of entities, the nodes, and the relations between them, the arcs.

When labels are used on the arcs of a graph with the intention to store information about many different relations between the elements, the graph takes the name of multirelational graph. The capability to represent more than one relation in a single data structure is very important in many contexts and graphs have been in fact proven to be more useful as data abstractions rather than intrinsic feature vectors when there is a need to capture several high order relationships between objects [36].

2.2.2 Kernel methods

One of the traditional approaches to perform data mining on graph-like structures (e.g., discover hidden patterns, or cluster nodes) consists in applying a pre-processing step to reorganize the graphs to a classical tabled representation in order to apply standard machine learning algorithms. For example, the type of a node can be represented by a categorical value in a column of the database or the edges may be transformed in one or more columns. On the contrary, other methods, such as *kernel methods*, are designed to reduce these efforts by considering learning from multi-relational data descriptions directly [22, 34, 95].

The key feature of most of these methods is the so-called *kernel trick* [32] that allows learning algorithms to operate in high-dimensional feature spaces without complexity issues. The trick consists in calculating the distances between elements in the new feature space, which are guaranteed to be easy to compute, without considering the mapping function between the original space and the new space, which can be complex or unknown. An example of kernel trick is showed in Figure 2.2.

A kernel learning method is characterized by two elements: the *learning algorithm*, that represents the scope of the learning process and how it is pursued, and the *kernel function*, that represents the measure of difference between elements.

The learning algorithm contains both the learning task and a solution strategy, that is, the way the search space is explored. Examples of learning algorithms are, among the many, kernel perceptrons [46], principal components analysis (PCA) [78], spectral clustering [26], and support vector machines (SVM) [11]. This last class of algorithms is the one used for our learning task so we will now briefly present its theoretical basis. In their simplest definition, *support vector machines* are a class of supervised linear binary classifiers that, given labeled training data, outputs a straight line which categorizes new examples. An important feature of these classifiers is their ability to find the optimal separating plane, the one which maximizes the margin (i.e., the lowest distance between elements and the plane) of the training data. The right part of Figure 2.2 represents an example of SVM. The interesting capacity of support vector machines is that they are able to classify samples with any N-dimensional plane if the kernel trick is used: a

non-linear function is learned by a linear support vector machine in a high-dimensional feature space because the system is controlled by a parameter that does not depend on the dimensionality of the space.

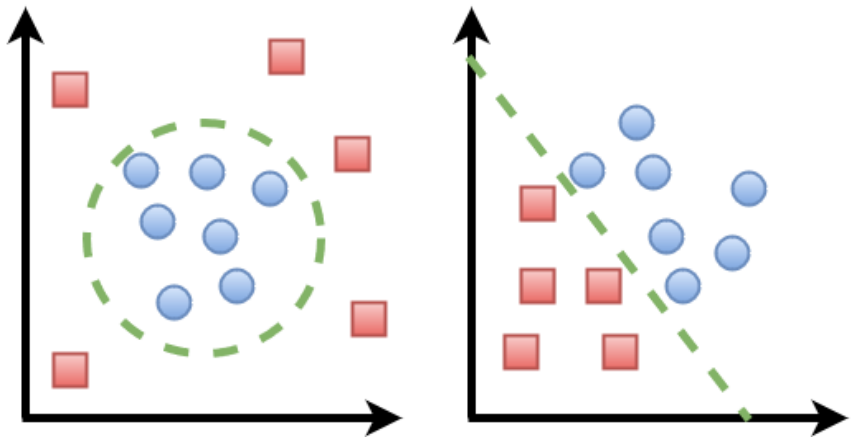


Figure 2.2: Example of kernel trick. The objects of the original space are mapped in a new space where a linear separation is possible. Left: the original space. Right: the new space.

The second element defining a kernel method is its kernel function. These functions can be classified upon the data they are intended to be used with and are usually build following two possible driving forces: model-driven or syntax driven [32].

In the first case, the kernels are designed in order to adapt to a specific context, to use some kind of background knowledge about the data. Examples of this category are the Fisher kernels [40,41] or the diffusion kernels [47].

In the second case, instead, the driving force is represented by the model where the data is encapsulated. Examples of this class are the kernels specifically designed for strings [43, 54, 92], trees [18, 44, 91], or graphs. In this last category, that is the one we are interested in, several different approaches are possible.

The random walks kernel takes in consideration two graphs (or subgraphs) and possible sequences of their nodes, called walks,reconstructed following the arcs [33, 45]. The similarity between the two structures is measured counting the number of identcal walks. The main problem of this kernel is its high complexity that has been partially solved by [90] with a fast compu-

tation algorithm. Another possible solution is the one that leads to shortest path kernel [10] where the concept of walk is substituted by that of path, where repetitions are avoided. In particular, similarity is measured counting the number of identical shortest paths connecting all the pairs of nodes of each structure. For potentially large graphs, this solution is still not convenient. Edit distance kernels [67] try to solve the problem of complexity not considering anymore entire graphs (or subgraphs) but only smaller portions of their structure. They count how many edit operations (e.g., insertion or deletion of a node) are needed to pass from a structure to the other. Finally, similar solutions are the *weighted decomposition kernels* [57] where the similarity is calculated on a smaller portion of the graph centered around a specific part of the structure (e.g., a node having a particular feature and its neighborhood, or a particularly meaningful arc).

In conclusion, it is important to highlight that in many cases a single learning algorithm could be used with several kernel functions. For this reason, it is the combination of a learning algorithm and a kernel function that identifies a kernel method.

The principal framework used in this work of thesis, kLog, implements a specific kernel method, based on a set of support vector machines (LIBSVM [15]) and on a decomposition kernel (NSPDK [20]), that is discussed in detail in Chapter 4.

2.3 Logical and relational learning

To better understand this work, it is important to start from the idea that learning tasks over complex, relational, and uncertain data structures, such as graphs, which are difficult if not impossible to represent as vectors of features without any loss of information, cannot be straightforwardly modeled with standard machine learning approaches.

The field of *Statistical Relational Learning* (SRL), introduced comprehensively in [35], aims to overcome this limitation. SRL provides new, more expressive, knowledge representation frameworks that exploit the use of relational models, able to represent a variable number of entities as well as relationships. A relational model can have several representations, when it is grounded in, or derived from, first-order logic it is called a logical representation. The interest in learning using these expressive languages and the results obtained with these formalisms soon resulted in the emergence of a

new subfield of artificial intelligence, and in particular of statistical relational learning, called Logical Relational Learning (LRL), exhaustively discussed in [24].

2.3.1 The learning problem

Broadly speaking, data mining is viewed as the process of computing the set of patterns $Th(Q, D, \mathcal{L})$ where $\mathcal{L}$ represent the search space of all possible patterns expressible within a language, D is the set of examples that need to be generalized and Q is the set of constraints that specify which patterns are of interest. The set $Th(Q, D, \mathcal{L})$ can therefore be defined as the set of all patterns $h \in \mathcal{L}$ that satisfy the constraint $Q(h, D)$ with respect to the data set D.

A slightly different perspective is given by the machine learning view, which is often formulated as that of finding a particular function h (again belonging to a language of possible functions $\mathcal{L}$) that minimizes a loss function $l(h, D)$ on the data. The machine learning and data mining views can be reconciled, for instance, by requiring that the constraint $Q(h, D)$ succeeds only when $l(h, D)$ is minimal. Central in the definition of the constraint $Q(h, D)$ and the loss function $l(h, D)$ is the *covers* relation c between the data and the rules that specifies when a rule covers an example, or, equivalently, when an example satisfies a particular rule.

Traditional machine learning methods employ the single-tuple single-table assumption, which assumes that the data can be represented using attribute-value pairs. Representations that can easily be mapped into this format are called propositional. For those data that doesn't fit in this restriction, techniques able to simplify the model and bring it to a propositional environment are available, but usually they need a background knowledge over the domain of the data to be applied. General-purpose machine learning systems able to cope with such representations directly, without any ontological knowledge of the data, are provided by logical relational learning since it naturally has the instruments to directly deal with structured data.

Once it is defined the scope of the mining process, a search strategy must be defined. The computation of the solutions proceeds typically by searching the space of possible patterns or hypotheses $\mathcal{L}$. Symbolic machine learning and data mining techniques typically structure the space $\mathcal{L}$ according to generality. One pattern or hypothesis is *more general* than another if all

examples that are covered by the latter pattern are also covered by the former.

Generality is a central concept during the search for solutions. The reason is that generality can often be used both to prune the search space and to guide the search towards the more promising parts of the space. The large majority of statistical relational learning systems often search the space in a general-to-specific fashion.

2.3.2 The logical language

Logical relational learning uses a logical description language as $\mathcal{L}$ which provides a very expressive representation.

The fundamental elements of the language are *ground atoms*, composed of predicate symbols p/n (where p is the name of the symbol and n is its arity), and terms t. A term t can be a costant (conventionally starting with a lowercase character), or a variables (conventionally starting with an uppercase character). A ground atom is thus a formula of the form

$p(t_1, ..., t_n)$

and possess a truth-value (i.e., can be either true or false).
Ground atoms represent relationships among the objects denoted by the terms appearing in the atoms for example:

married(john,mary)

A special case of relation can be represented by an unary relation which specifies the tipology of the object represented by its only term, for example:

man(john)
woman(mary)

A *clause* is a formula of the form

$h_1; ...; h_n \Leftarrow b_1, ..., b_m$

where h_i and b_i are atoms, the symbol "," stands for conjunction (the conditional operator AND), the symbol ";" stands for disjunction (the conditional

operator OR), and the symbol "⇐" (or sometimes ":-") stands for implication (the conditional operator IF). Furthermore, if atoms contain variables, they are universally quantified. For example:

female(X);male(X)← human(X)

specifies that for all X, when X is human, X is also male or female.

2.3.3 The two learning settings

In a LRL environment two settings are possible accordingly to the choice of the language and of a *covers* relation. The language $\mathcal{L}$ that we have introduced so far could be considered as coposed by two different languages. The first one is a language of examples $\mathcal{L}_e$, whose elements are descriptions of examples, observations, or data, and the second one is a language of hypotheses $\mathcal{L}_h$, whose elements describe hypotheses about (or regularities within) the examples, observations or data. With these premises, the *covers* relation can be defined as a relation $c : \mathcal{L}_e \times \mathcal{L}_h$ that determines whether a hypothesis matches an example. Based on the choice of $\mathcal{L}_e$, $\mathcal{L}_h$, and c, two main settings are possible:

Learning from entailment: $\mathcal{L}_e$ and $\mathcal{L}_h$ are logical formulas, typically subsets of clausal logic, and the relation c corresponds to logical entailment.

Learning from interpretations: $\mathcal{L}_h$ is a set of logical formulas, most often subsets of clausal logic, $\mathcal{L}_e$ is a set of interpretations (i.e., possible scenarios) and the covers relation c corresponds to logical satisfiability.

More formally, an *interpretation* is a set of ground atoms all belonging to the same domain. When an interpretation contains all and only the ground atoms true in the domain (i.e. it is under the closed world assumption) it takes the name of *Herbrand interpretation*. For this reason all ground atoms in an Herbrand interpretation are assumed to be true and any possible other ground atom is assumed to be false.

Since the framework used in this work of thesis (kLog), and in general the context of LRL, are under the closed world assumption, for the sake of simplicity, when we refer to interpretations we always refer to Herbrand interpretations.

2.3.4 LRL tasks and applications

Independently from the setting used, the tasks that could be performed using LRL techniques are the standard machine learning tasks enriched with some particular possibilities.

Classification: the classical classification task (the problem of assigning new observations to a set of categories, also called supervised learning) can naturally be executed. In some statistical relational learning formulations, it is possible to perform collective classification, the (simultaneous) prediction of the class of several objects given objects' attributes and their relations. Collective classification is useful in those scenarios where the category of an element depends not only on its features, but also on the other elements in the dataset.

Clustering: the purpose is to group objects that share common features, without a training phase (unsupervised learning). With LRL it is possible to implement a link-based clustering where the clusters of similar objects are determined according to the relations a single element has with the others and, more in general, to the relations present in its neighborhood.

Filtering: it consists in the selection of information that is relevant to an entity, that will be kept, and useless information, that will be discarded. Using LRL techniques, collaborative filtering can be performed, where the lack of data owned by a single entity is overcame using the information brought by its neighbors.

Link prediction: the scope is to determine if a relation between two or more objects exists or not. This is a supervised learning problem, so examples of the particular relation we want to detect are needed as training set. If we see a relation as an object of the model, link prediction can be interpreted as a binary classification determining if it is actually present or not.

Concrete examples of the application of SRL and LRL are available in many different fields of study:

Chemistry and biology: logical relational learning is applied in structure activity relationship prediction (SAR), an essential phase in drug design and discovery, molecular analysis, and toxicology. LRL is used

to distinguish the active moleculas from the inactive ones using the structural alert, a particular substructure (subgraph) that can present interesting patterns [37, 82].

Web mining: we find examples in particular in the subfields of link mining and latent relations discovery. Its purpose is to analyze a single, enormous graph or network, retrieving those smaller subgraphs that share a common pattern [8, 14, 21, 55].

Natural language processing: it represents another important field where LRL techniques are being developed. In this case, the data are constituted by natural language phrases and queries. The scope is to enable machines to derive meaning from human language, or to use human language to generate new sentences. This field has been pioneered by LRL and many interesting works can be found in literature [23, 61, 88, 96].

To implement these solutions, many SRL and LRL techniques have been used. Markov logic networks (MLN) [74], probabilistic relational models [29], and generalizations of conditional random fields to arbitrary relations [83] have been proven to be successful.

2.3.5 LRL and semantic mapping

For what concerns our work, while some of the methods presented in Section 2.1.3 attempt to extend the classical local semantic mapping approach, none of them considers all the structure of the semantic map as a single entity, but they consider each element of the map as a separate instance, reasoning locally. With this work of thesis, we aim to contribute to overcome this situation.

The use of SRL techniques in semantic mapping has been pioneered by [53], which proposes a method for obtaining semantic maps of indoor environments using a framework based on a MLN and data driven Markov chain Montecarlo (MCMC) sampling. Using MCMC, the system samples many possible semantic worlds (i.e., istances of plausible semantic maps) from a metric map and selects the most likely semantic world as the one that better fits the input data. The use of a MLN allows to evaluate the plausibility of each semantic world sampled from the MCMC step, considering all the rooms at the same time, thus adopting a global approach.

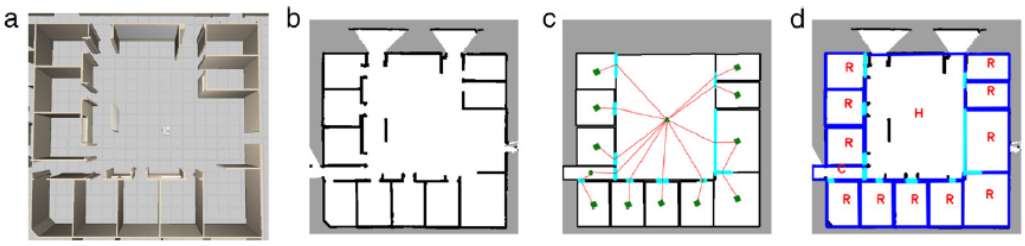


Figure 2.3: The usage of SRL techniques to obtain a semantic world starting from a simulated environment, from [53].

kLog [28], the LRL framework used in this thesis and introduced in detail in Chapter 4, provides a logical language and a framework for relational learning based on support vector machines and graph kernels. kLog, unlike typical statistical relational learning frameworks, does not employ a probabilistic framework but is based on linear modeling in a feature space constructed by a graph kernel, where nodes of the graph correspond to entities and relations, some given in the input data and some (expressing background knowledge) defined declaratively in Prolog. In a sense, our approach is similar to the one presented in [53], but in our case LRL techniques are used not for recreating the environment, but for classification purposes.

Chapter 3

Relational and logical representation of buildings

Anger, desolation, and work.
Ooh, exhaustion!
Why does the rain talk?
Why does the love fight?

Keith's Think-Zone Poem Generator

One of the more typical representations for a semantic map of a floor plan of a building is a graph where each node represents a room of the floor plan and labels are assigned to rooms, reflecting their role in the building (e.g., corridor, classroom, kitchen).

We decided to enrich the standard state of the art model proposing a more complex and complete representation which contains also elements from the metric map. In particular, we extract geometrical objects (e.g., doors, walls) and features (e.g., areas, perimeters) from the two-dimensional physical representation of the environment to store a larger quantity of data in the model.

We identified three key factors in order to define a semantic map:

The information to include: which concepts are useful to represent in a semantic map and why;

The level of detail of the model: which elements of the metric map are embedded into the higher-level representations and how, since different granularities are possible;

The label set employed: which labels are available in the classification of the rooms; the higher the number of labels the more detailed the classification.

In each section of this chaper we present one of these factors, specifying the decisions made to obtain the models used in this thesis. Finally, in the last section, we present an example of how our models are translated into logical formulas, keeping a high level of abstraction (i.e., without using a specific syntax such as Prolog syntax).

3.1 The included information

We decided to take inspiration from the research of the urban planner Kevin Lynch in order to select which information to represent. In his most famous work [56], Lynch studied how human observers gain information about a city and which are the elements that help them in this task. We make a parallel with our context and import similar elements in our semantic map model. Lynch identifies five types of elements (nodes, paths, edges, landmarks, and districts) that we adapted as follows:

Nodes: represent the spaces where the observer can enter, the origins and the destinations of their movements. They do not include only those places where an action is performed but also those that are mere conjunctions between spaces. In the original work of [56], the concept of node includes buildings of the city, roads, squares, or in general any location of interest that an explorer would visit. In our context nodes are the *rooms* of the floorplan, the nodes of the topological graph.

Paths: are the routes along which the observer moves in the environment. Each path can be composed of several segments and reflects the mental process that rises in a person when he/she thinks how to reach a node starting from another one. In [56], paths are all the possible connections between nodes; they can be a sequence of streets, pedestrian itinerary, or public transport lines. In our setting, paths are represented by the *connections* between the rooms, the sequences of arcs in the topological graph. We consider only topological paths over the map, without considering the physical navigation through spaces.

Edges: are defined as the linear elements that shape the borders between different spaces. In the urban perspective of [56], edges are exterior walls of the buildings, riversides, or even railway borders. In our representation, edges are the *line segments* defining the shapes of the polygons representing the boundaries of the rooms. They can belong to two types: barriers or sutures. The formers are impenetrable parts of the environment and divide the rooms, like the *walls* in our setting. Sutures are instead borders that connect different spaces and that are crossed by paths; in a floorplan they are represented by the *doors*.

Landmarks: are reference points of the space used for navigation. They are usually symbolic, clearly visible, and simple to be identified by the observer. In the cities, landmarks are tall buildings, signs, shops, or natural elements of the landscape. In our case we decided not to use signs as landmarks because we use only data from laser scans and not camera images. We used as landmarks entrances, stairs, and elevators since they are places where the observer must pass in order to enter in the environment represented by the map.

Districts: are groups of nodes that share common features and are typically close to each other. In the city are usually denoted by buildings with a same architecture or scope (e.g., old town, financial district). The observer can use them to choose paths at a higher level of abstraction. In our case, districts can be, for istance, represented by clusters of rooms all connected to the same corridor.

3.2 The relational models

In this thesis, we decided to model (represent) the data with entity-relationship models [16]; they represent a high-level knowledge disregarding how data are physically implemented in the database. E/R models are the most common choice in LRL due to their expressiveness, graph-like structure, and facility to be translated into logical formulas [24].

An E/R model is composed of three classes of components:

Entities: objects or concepts that represent important data. They are typically denoted by nouns (e.g., professor, student, course) and are rep-

resented as rectangles.

Relations: relevant associations between entities. They are usually denoted by verbs (e.g., teach, attend) and represented as arcs with a diamond in the middle.

Attributes: characteristics of either an entity or a relation (e.g., name of a person, or starting date of a relation). The set of attributes that each entity or relation has is fixed while their values change. They are represented by circles or ellipses connected to the other elements of the model.

Figure 3.1 shows the graphical representation of the components of a E/R model.

We developed two different models with different specificities of representation. The first model is more abstract and information from the metric map is used to enrich the content of a topological map. The second model, instead, is more concrete and, in addition to using the elements from the metric map to compute geometrical features, represents them directly as entities.

The E/R diagram of the first model (MODEL1) contains nodes, paths, and landmarks, while edges are used to create the model but are not directly represented. Nodes are the fundamental entities of the model, under the name of *rooms*, and their attributes are a set of geometrical features and a semantic label. We selected 8 geometrical features, calculated from the polygon that represents the shape boundary of the room. These features are a subset of those used in [63]: area, perimeter, area divided by perimeter, mean distance between the centroid and the shape boundary, form factor, circularity, normalized circularity, average normalized distance between the centroid and the shape boundary. The detailed description of how these features are calculated can be found in Section 5.2. A *connected* relation is used to represent when two rooms have a door in between, and a series of such

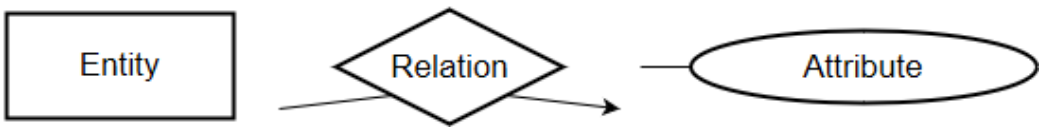


Figure 3.1: Basic elements of an E/R model.

connections represents a path. A path between two rooms is not memorized explicitly but can be derived from the graph-like structure of the data, following chains of arcs between origin node and destination node. Landmarks are represented by those nodes that represent (according to their semantic labels) the entryway of the floor, namely entrances, stairs, and elevators.

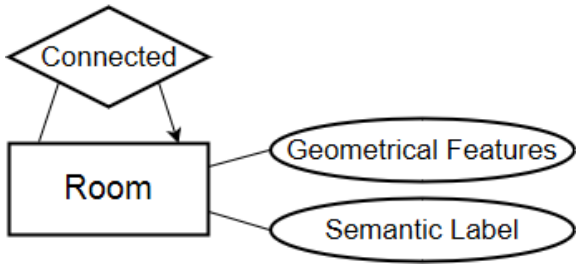


Figure 3.2: The E/R schema of MODEL1.

The E/R diagram of the second model (MODEL2) keeps the structure of the previous one, but enriches it with a more strict connection to the metric map. This model contains, in fact, *line segments* which are the borders that constitute the polygonal contour of each room. They represent edges and can be divided in two entities: *walls* and *doors*, respectively barriers and sutures according to [56]. Following the approach used in [93], we divided doors in explicit and implicit. In the first case, spaces are physically separated by a solid element (e.g., the classical domestic doors) while in the case of implicit doors the two areas are not physically divided but there are elements of the environment that express the division (e.g., a different set of tiles on the floor). In this model, the *connected* relation is no more present explicitly, but is implicitly represented by the doors (i.e., when two rooms have a door in between, the two rooms are connected). Finally, the *intersect* relation is introduced to relate two line segments that share at least one point; checking which rooms have line sgments that intersect, it is possible to deduce which rooms are adjacent.

3.3 Labeling schemas

In the E/R models previously described, a semantic label is considered as an attribute given to a node to express its functional value. In the case of

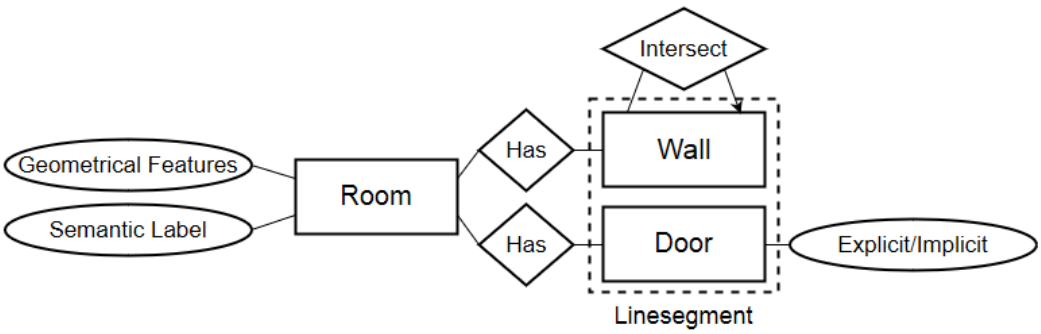


Figure 3.3: The E/R schema of MODEL2.

semantic mapping, a label contains the type of the room, chosen among a set of available categories. The set of possible labels, the *labeling schema*, depends on the type of the building. Building type is denoted by the use that human beings make of the building (e.g., school, office) which imposes shape and structure of the building [75].

According to [20], the definition of the label set is critical for the performance of place classification. In our thesis, we used a hierarchical labeling schema to evaluate the impact of different labels on the classification. The two upper and more general levels are common to all building types, while a specific set of labels characterizes each building type at the bottom level.

At the top and most general level, the labeling schema is called $\mathcal{L}_{R/C}$ and contains only two general categories:

ROOM**:** a space in which an activity is performed;

CORRIDOR**:** a space used to connect different spaces together.

We define a second set of labels, $\mathcal{L}_{R/C/E/S}$={FUNCTIONAL ROOM, CORRIDOR, ENTRANCE, SERVICE ROOM}, where CORRIDOR and ENTRANCE specialize CORRIDOR and ROOM is divided in:

FUNCTIONAL ROOM**:** a space in which the core activity of the building is performed (e.g., a classroom in a school, an office or a meeting room in an office building);

SERVICE ROOM**:** a space used to support the core activities of the building (e.g. administrative rooms, bathrooms or canteens).

These general labeling schemas are shared by all building types and constitute a basis for the specific building type schemas $\mathcal{L}_{type}$. The set of labels relative to a building type has been extracted from architectural sources [23], [24], Figure 3.4 contains two examples of such detailed schemas: $\mathcal{L}_{\text{SCHOOL}}$, and $\mathcal{L}_{\text{OFFICE}}$.

The rows of the schemas represent the already discussed label sets, $\mathcal{L}_{R/C}$ and $\mathcal{L}_{R/C/E/S}$. Along the columns, it is instead represented a categorization of spaces depending on their dimensions (Small, Medium, Big), the characteristic to be used by many users simultaneously (Collective) or the irrelevance of their size (ND). In Figure 3.4(a) it is represented $\mathcal{L}_{\text{SCHOOL}}$, its most significant labels are CLASSROOM and LAB which represent didactic rooms. In Figure 3.4(b) it is instead represented $\mathcal{L}_{\text{OFFICE}}$, whose most significant labels are CUBICLE and OFFICE. Some labels are shared by both schemas, since the rooms they represent are commonly found both in schools and in offices; it is the case, for example, of CLOSET, BATHROOM, and CAFETERIA.

3.4 Logical representation

Once the relational models and the label sets have been defined, we translate them into logical formulas. The following formulas are written using the notation introduced in Chapter 2, their actual format, written in kLog language, will be presented in Chapter 4, after having introduced kLog itself.

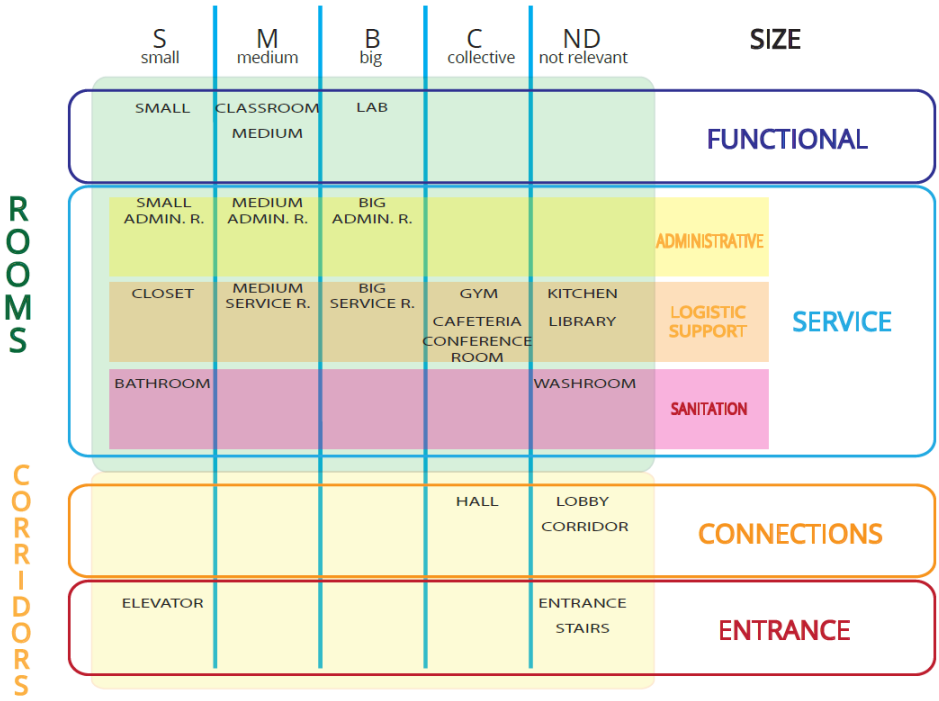
We decided to assign to each object of the world a unique identifier and state with a single logical formula to which entity (e.g., room, door, wall) it belongs to. In this way we obtain formulas of the type:

Entity(id)

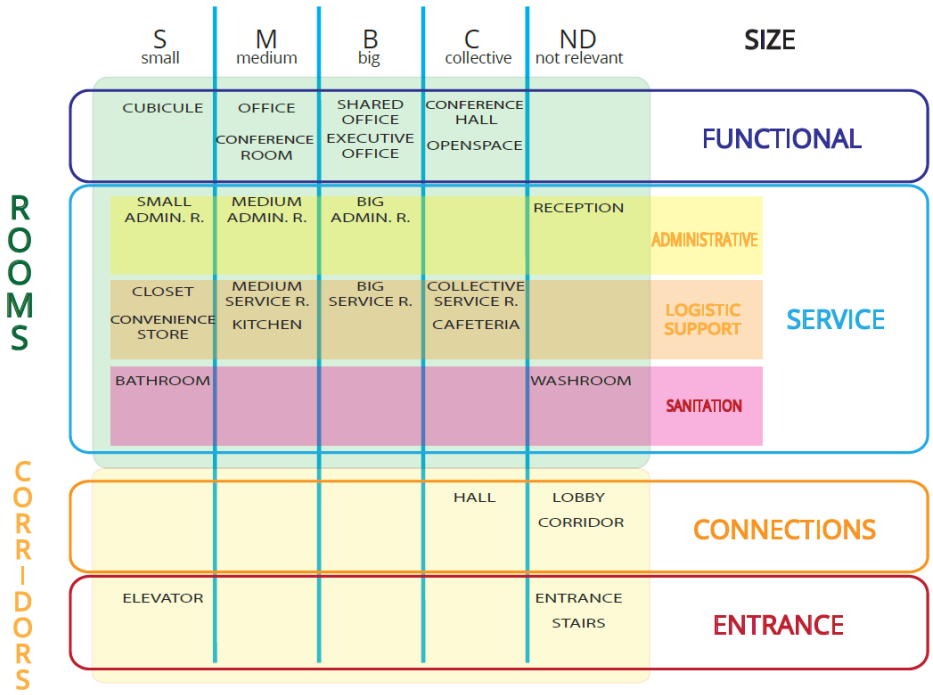
Each relation is represented with a formula named after the relation type and has two arguments (since we employ only binary relations) representing the identifiers of the involved entities:

Relation(id1,id2)

Each property is represented with a formula named after the attribute and has two arguments that contain the identifier of the element having such



(a) SCHOOL.



(b) OFFICE.

Figure 3.4: Top: set of semantic labels for SCHOOL *building type. Bottom: set of semantic labels for* OFFICE *building type.*

feature and the value of the feature itself (that can be both categorical or numeric):

Attribute(id,value)

Combining this notation with the first relational model we obtain the following predicates:

Room(room_id)
Connected(room_id,room_id)
Label(room_id,semantic_label)
1st*_geometrical_feature(room_id,numerical_value)*
2nd*_geometrical_feature(room_id,numerical_value)*
...
n^{th}*_geometrical_feature(room_id,numerical_value)*

where *semantic_label* belongs to the selected set of labels ($\mathcal{L}_{R/C}$, $\mathcal{L}_{R/C/E/S}$, or $\mathcal{L}_{type}$).

For the second relational model, instead, we obtain the following predicates:

Room(room_id)
Wall(wall_id)
Door(door_id)
Intersect(wall_id,wall_id)
Has_wall(room_id,wall_id)
Has_door(room_id,door_id)
Doortype(door_id,door_type)
Label(room_id,semantic_label)
1st*_geometrical_feature(room_id,numerical_value)*
2nd*_geometrical_feature(room_id,numerical_value)*
...
n^{th}*_geometrical_feature(room_id,numerical_value)*

where *door_type* is either *implicit* or *explicit* and *semantic_label* belongs to the selected set of labels ($\mathcal{L}_{R/C}$, $\mathcal{L}_{R/C/E/S}$, or $\mathcal{L}_{type}$).

Chapter 4

Knowledge representation in kLog

This quiet moment's
despairingly beautiful.
Did it matter?

JanusNode

In this chapter, we present the principal tool used in this thesis, kLog. In the first part of the chapter we present a general overview of the tool, its language, the tasks it can perform, the methods used to solve the tasks, and some examples of its usage in several fields. In the second part, we describe how our models can be represented in kLog, with examples of simple E/R structures translated in its language.

4.1 kLog

In this work of thesis we use kLog [28] for performing different learning tasks, exploiting global reasoning on the structure of buildings.

kLog is embedded in Prolog, from which it takes the name, and allows users to specify in a declarative way different types of logical and relational learning problems, keeping a high level of abstraction. kLog is based, as many other logical and relational learning frameworks, on the learning from interpretations setting where objects (entities) and their relationships can be naturally represented. The main difference from the other statistical

relational learning frameworks is that kLog does not employ a probabilistic framework but is rather based on a kernel-defined feature space, where multi-relational data are considered directly, without being reconducted to a classical table representation.

The strengths of kLog are the same of logical and relational learning approaches and can be summarized in two main advantages. First, once structured data are represented using the kLog language, kLog can be easily used for different tasks such as classification, regression, multitask learning, and collective classifications. Second, kLog feature space is built using a graph kernel on a graphical representation of the data. Features spaces built by graph kernels are particularly suited for learning tasks on complex data structures, like semantic maps, that can be naturally represented as graphs, namely, as networks of relations.

4.1.1 General overview

In order to learn, kLog essentially describes learning problems at three different levels. The first level specifies the logical and relational learning problem and consists of logical formulas, representing the structure of the data and the data itself. At the second level, the data are graphicalized, that is, are transformed into graphs leading to the specification of a graph learning problem. Graphicalization is the equivalent of knowledge-based model construction. Indeed, SRL systems such as probabilistic relational models [29] and Markov logic networks [74] also produce graphs, even if these graphs represent *probabilistic* graphical models. Finally, the graphs produced by kLog are turned into feature vectors using a graph kernel, which leads to a statistical learning problem at the third level.

4.1.2 kLog language

A kLog program consists of:

- **a set of ground facts**, embedded in a standard Prolog dataset, representing the data of the learning problem;

- **a set of logical signatures**, describing the elements (entities and relations) of the relational model;

- **a set of logical predicates**, associated with intensional signatures, representing background knowledge;
- **a standard Prolog program**, which specifies the learning problem and makes calls to kLog library predicates.

In order to specify the semantics of the language, it is convenient to formalize the domain of the learning problem as a set of constants (objects) $\mathcal{C}$ and a finite set of relations $\mathcal{R}$. Constants are themselves partitioned into a set of entity identifiers $\mathcal{E}$ and set of property values $\mathcal{V}$. A *ground atom* $r(c_1, ..., c_n)$ is composed by a relational symbol $r \in \mathcal{R}$ of arity n followed by an n-tuple of constant symbols $c_i \in \mathcal{C}$. An *interpretation* is a finite set of true ground atoms. In kLog, interpretations are always considered complete, following the closed world assumption: if an entity or a relation is not directly affirmed, that specific element does not exist in the interpretation.
The signature for a relation $r/n \in \mathcal{R}$ is an expression of the form

$$r(name_1 :: type_1, ..., name_n :: type_n) :: level.$$

where the relational symbol r represents the name of the predicate itself, $name_{1...n}$ are the columns of r and $type_{1...n}$ can both be entities from $\mathcal{E}$ or values from $\mathcal{V}$. The *level* of a signature is either *extensional* or *intensional*. In the extensional case, all the ground atoms which contribute to every interpretation are those and only those listed in the data. In the intensional case, ground atoms are those which result from a Prolog reasoning process. In this way, users may take advantage of many of the extensions of definite clause logic that are built into Prolog. The ability to specify intensional predicates through clauses is useful for introducing background knowledge in the learning process, as it is common practice in inductive logic programming. Once the graphicalization step occurs, there will be no difference between the facts contained in the original dataset and the facts obtained through intensional predicates: the two will contribute equally to the learning task.
Some examples of extensional signatures are:

student(student_id::self)::extensional.
professor(professor_id::self)::extensional.
pubblication(pub_id::self,author::property)::extensional.
advised_by(p1::student,p2::professor)::extensional.

An example of intensional signature is:

on_same_paper(s::student,p::professor)::intensional.
on_same_paper(S,P):-
student(S),
professor(P),
pubblication(PUB,S),
pubblication(PUB,P).

In order to ensure that the learning process is well-defined, two additional assumptions on the dataset are necessary. First, the primary key of every relation should not contain relevant information since it will not contribute in any way to the learning phase. We can say that it merely act as a placeholder. Second, a relation whose arity of elements (i.e., entities from $\mathcal{E}$) is one must describe every entity of an interpretation and is named E-relation. This does not prohibit the presence, in an E-relation, of features describing the element itself (i.e., values from $\mathcal{V}$) , which will be used in the learning process. Once an entity has been specified in this special kind of relation, it can be involved in other relations between entities of arbitrary arity called R-relations.
Some examples of E-relations are:

student(s015)
professor(p009)
pubblication(pub077,s015)

An example of R-relation is:

$advised_by(s015, p009)$

4.1.3 Available tasks

A supervised learning job in kLog is specified as a set of relations. We begin defining the semantics of a learning job consisting of a single relation. Without loss of generality, let us assume that this relation has signature

$r(name_1 :: \mathcal{E}, ..., name_n :: \mathcal{E}, name_{n+1} :: \mathcal{V}, ..., name_{n+m} :: \mathcal{V})$

Several situations may arise depending on the relational arity n and the number of properties m in the target relation r, as summarized in Table 4.1. When $n = 0$, the learning job consists of predicting properties of an entire interpretation, when $n = 1$ properties of a specific type of entities, when $n = 2$ properties of couples of entities, and so on. When $m = 0$ we have a binary classification task (where positive cases are ground atoms that belong to the interpretation and no value is predicted). When $m = 1$ it is the case of multiclass classification, if the property is categorical, or regression, when the property is numeric. Property types (numerical or categorical) are not need to be specified since they are automatically detected by kLog inspecting the dataset. When $m > 1$, two or more properties have to be predicted at the same time. kLog recognizes that such a declaration defines a multitask learning job, as in the case where the learning job consists of several target relations.

4.1.4 Kernel method

The first operation performed by kLog over the dataset is its graphicalization. This needs to be intended as a pre-processing step, needed to transform the logical statements into a graph. To each interpretation z corresponds a graph $G_z = (V_z \cup F_z, E_z)$:

Nodes: there is a vertex in V_z for every ground atom of every E-relation

Table 4.1: The tasks available in kLog, from [28]. n represents relational arity, m represents the number of properties.

m	n		
	0	1	2
0	Binary classification of interpretations	Binary classification of entities	Link prediction
1	Multiclass/regression on interpretations	Multiclass/regression on entities	Attributed link prediction
> 1	Multitask predictions on interpretations	Multitask predictions on entities	Multitask attributed link prediction

and there is a vertex in F_z for every ground atom of every R-relation.

Arcs: an edge $(n1, n2) \in E_z$ if and only if $n1 \in V_z$, $n2 \in F_z$ and the identifier of $n1$ belongs to the set of the identifiers of $n2$, which means that $n1$ must be an entity of the E/R diagram, $n2$ must represent a relation and that $n1$ is involved in $n2$.

For example, given the previously presented signatures and the following ground atoms:

professor(p009)
student(s015)
student(s030)
student(s053)
pubblication(pub077,s015)
pubblication(pub077,p009)
pubblication(pub120,s053)
advised_by(s015,p009)
advised_by(s030,p009)
advised_by(s053,p009)

the graphicalization process will produce the graph contained in Figure 4.1.

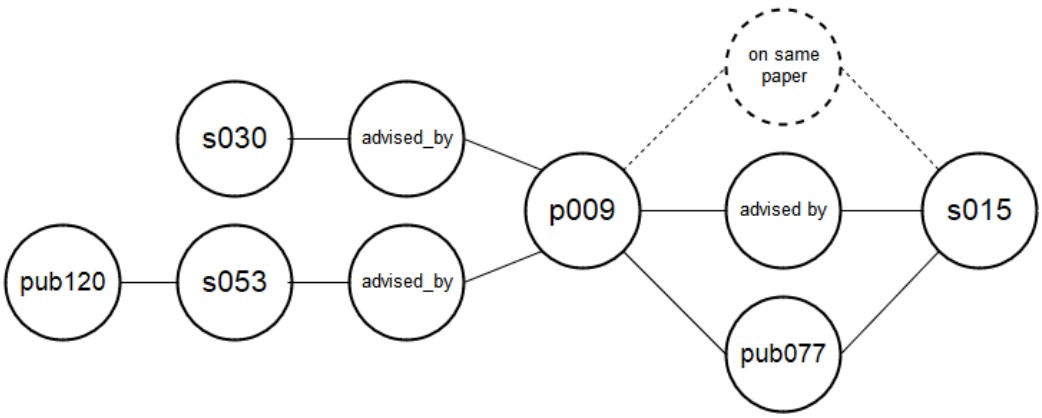


Figure 4.1: An example of graphicalized data. The relation on_same_paper *was not contained in the original dataset but has been derived from an intensional signature.*

Once the logic formulas are unfolded into the graph, a suitable graph kernel is applied. The current implementation of kLog uses an extension of

the fast Neighborhood Subgraph Pairwise Distance Kernel (NSPDK) [20]. NSPDK is a decomposition kernel that compares and scores pairs of subgraphs. These subgraphs are rooted in a vertex v that represents the origin of a topological ball (i.e., a set of nodes surrounding a central element) called *neighborhood*. Two important parameters control the function of the kernel:

- r, that represents the topological radius of the neighbourhood, starting from the root;

- d, that represents the exact distance between the two roots of each pair of subgraphs that will be compared with the kernel.

Actually, in the implementation of NSPDK, parameters r and d are substituted by r^* and d^* which represent the maximum values of radius and distance, respectively, upon which the kernel operates. An example of these parameters is represented in Figure 4.2.

The original NSPDK is suitable for sparse graphs with discrete vertex and edge labels, it performs an *hard match* between couples of subgraphs, using their exact topological structure to calculate the kernel distance. In kLog NSPDK is extended to deal with *soft matches*, useful when the graphs are not sparse or some vertices exhibit large degrees. Soft matching relaxes the all-or-nothing type of match, avoiding overfitting and allowing subgraphs to match partially, in a soft way. Given a subgraph, only the multinomial distribution (i.e., the histogram) of the labels is considered, discarding all structural information. The kernel is then computed as the dot product of the corresponding histograms. An example of soft matching is presented in Figure 4.3.

To reduce computational complexity two constraints are introduced:

- A root v must be a *kernel point*. Kernel points are typically vertices that are believed to represent information of high importance for the task, declared by the user as a list of domain relations. All vertices that correspond to ground atoms of these relations become kernel points while vertices that are not kernel points contribute to the computation only when they occur in the neighborhoods of kernel points.

- Only neighborhoods centered on the same type of vertex (a precise entity or relation) will be compared.

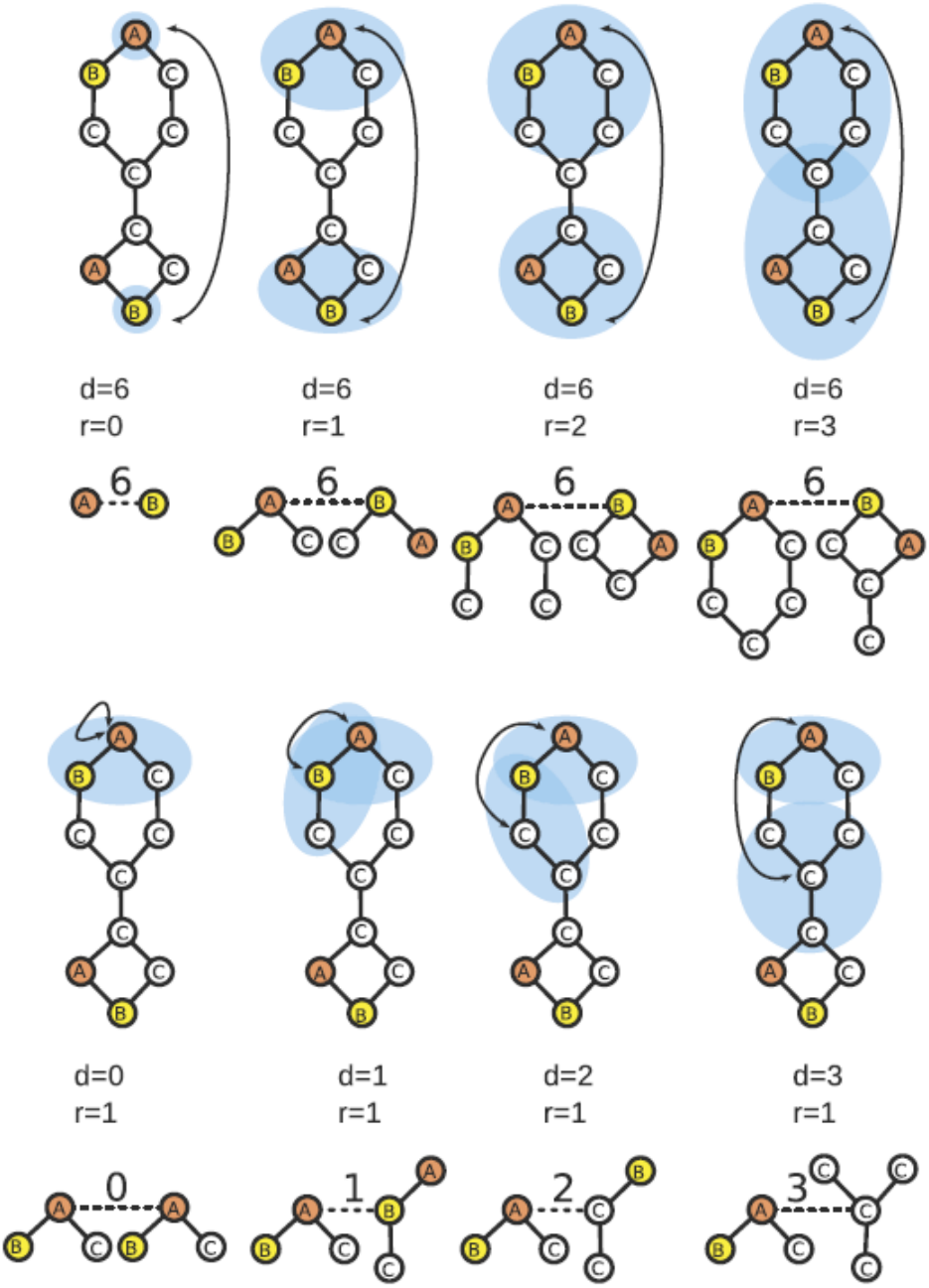


Figure 4.2: An example of NSPDK, from [28]. Top: pairs of neighborhood subgraphs at fixed distance 6 for radii 0, 1, 2, and 3. Bottom: pairs of neighborhood subgraphs of fixed radius 1 at distances 0, 1, 2, and 3.

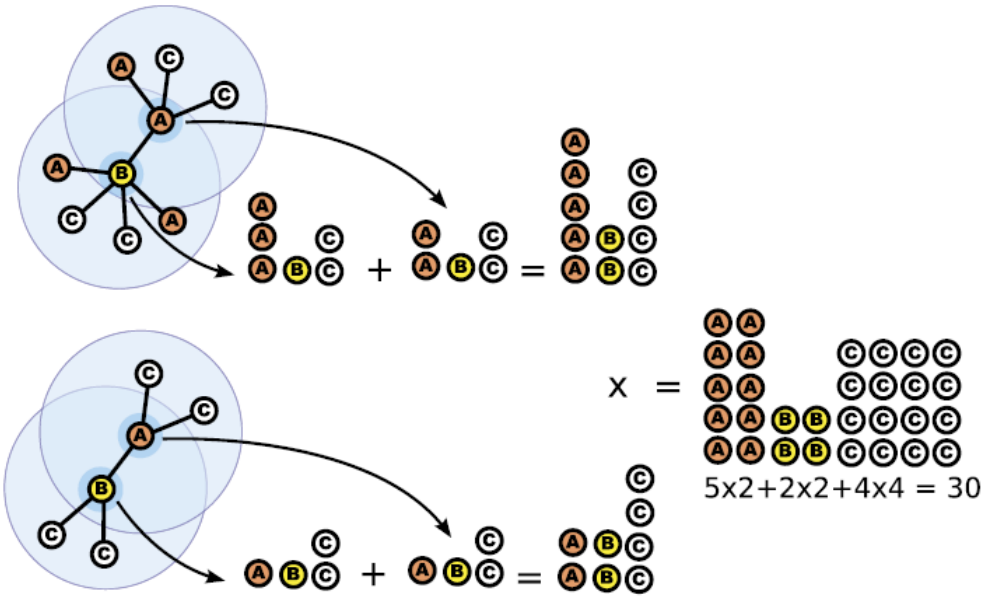


Figure 4.3: An example of soft matching, from [28]. Only features generated by a selected pair of vertices are represented: vertices A and B at distance 1 yield a multinomial distribution of the vertex labels in neighborhoods of radius 1. On the right it is shown the contribution to the kernel value by the represented features.

To complete the definition of the kernel method, a learning algorithm must be specified. kLog uses a collection of support vector machine implementations called LIBSVM [15] among which it is possible to choose the correct SVM depending on the task: binary classification, multiclass classification, or regression. The link prediction, not natively available in LIBSVM, is also possible because those which were relations in the original logical formulas become nodes of the graph during the graphicalization stage and the problem is reconducted to binary or multiclass classification.

4.1.5 Examples of application

kLog has already been used in several fields to accomplish many of the tasks listed in Table 4.1. In biology, it has been used for classification and regression over molecules: each interpretation represents a molecule to which kLog assigns a numerical value, for example biodegradability factor (regression over interpretations), or a class, for example mutagenic or not (classification over interpretations) [89]. In [28], kLog has been shown to be effective in determining the relations between students and professors in an academic environment (link prediction) and in classifying the movies contained in a large, high-dimensional graph as blockbusters or not (entity classification). Other applications are natural language processing [88], images understanding [5], robotic grasping [6], and even football-related predictions [87]. In this thesis we originally apply kLog to semantic mapping.

4.2 Our kLog representation

In this section we discuss how we represent our data in kLog language. We provide the signatures of the extensional predicates (i.e., those predicates explicitly affirmed in the dataset), the signatures of the intensional predicates (i.e., those predicates not explicitly stated but logically derivable from the extensional predicates), and some examples of the actual ground facts for each of the two models we use, MODEL1 and MODEL2 presented in Chapter 3.

4.2.1 kLog representation of the first model

Starting from generic logical formulas of Section 3.4 we write the signatures for each element of the model.
We translate entities and relations to the following kLog signatures:

room(room_id::self)::extensional.
connected(r1::room,r2::room)::extensional.

The geometrical features are represented by:

1st_geometrical_feature(r::room,1stgf::value)::extensional.
...
nth_geometrical_feature(r::room,nthgf::value)::extensional.

Labels' signatures can be represented in two ways, depending on the labeling schema adopted. In the case of $\mathcal{L}_{R/C}$ a binary relation has been adopted that states if a space is a corridor, otherwise it is considered a room; in this way it is possible to set up the problem of predicting a label as a binary classification on entites. The signature is therefore:

corridor(r::room)::extensional.

In the cases of $\mathcal{L}_{R/C/E/S}$ and $\mathcal{L}_{type}$ the label is instead represented as a categorical value coupled with a room identifier:

label(r::room,label::property)::extensional.

The intensional signature used to represent districts is:

on_same_corridor(r1::room,r2::room)::intensional.
on_same_corridor(R1,R2):-
connected(R1,R3), connected(R2,R3), corr(R3). (in the case of $\mathcal{L}_{R/C}$)
or
label(R3,corridor). (in the cases of $\mathcal{L}_{R/C/E/S}$ and $\mathcal{L}_{type}$)

Figure 4.4 contains a very simple example of a dataset, represented with the

first E/R model.

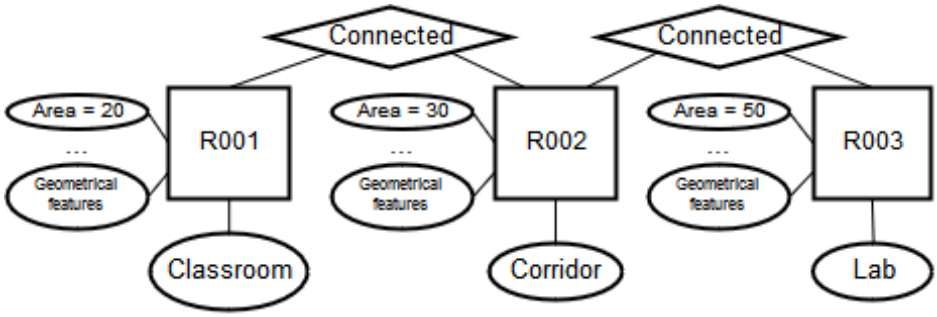


Figure 4.4: An example of a dataset represented with MODEL1.

The kLog representation of the example contained in Figure 4.4 is:

room(r001)
room(r002)
room(r003)
connected(r001, r002)
connected(r002, r003)
area(r001, 20)
area(r002; 30)
area(r003; 50)
...
8th*_geometrical_feature*

In the case of the labeling schema $\mathcal{L}_{R/C}$:

corridor(r002)

In the case of the labeling schema $\mathcal{L}_{\text{SCHOOL}}$:

label(r001,classroom)
label(r002,corridor)
label(r003, lab)

The result of kLog reasoning over intensional predicates is:

on_same_corridor(r001,r003)

4.2.2 kLog representation of the second model

For the second relational model, we translate entities listed in Section 3.2 with the following kLog signatures:

room(room_id::self)::extensional.
wall(wall_id::self)::extensional.
door(door_id::self)::extensional.

The relations among these entities are:

has_wall(r::room,w::wall)::extensional.
has_door(r::room,d::door)::extensional.
intersect(w1::wall,w2::wall)::extensional.

The attributes are translated with the same signatures of MODEL1 with the addiction of doortype:

1st*_geometrical_feature(r::room,*1st*gf::value)::extensional.*
...
n^{th}*_geometrical_feature(r::room,*n^{th}*gf::value)::extensional.*
doortype(d::door,type::property)::extensional.

The semantic labels are translated with the same assumptions of the first model. In the case of $\mathcal{L}_{R/C}$ we have:

corridor(r::room)::extensional.

In the cases of $\mathcal{L}_{R/C/E/S}$ and $\mathcal{L}_{type}$ we have:

label(r::room,label::property)::extensional.

In this model we designed two intensional signatures representing two useful relations among rooms: connection and adjacency. Two rooms are connected if they share a door. This definition is the same used to construct the connection graph in MODEL1 but in this case it is not a feature directly expressed in the model but it is derived using kLog reasoning. The actual signature is:

connected(r1::room,r2::room)::intensional.
connected(R1,R2):-
has_door(R1,D),
has_door(R2,D).

We assume that two rooms are adjacent if they share a wall or if they have intersecting walls. The signature for this relation is:

adjacent(r1::room,r2::room)::intensional.
adjacent(R1,R2):-
has_wall(R1,W),
has_wall(R2,W).
adjacent(R1,R2):-
has_wall(R1,W1),
has_wall(R2,W2),
intersect(W1,W2).

Figure 4.5 contains an example of a dataset, represented with MODEL2. For the sake of simplicity, this E/R model is not complete, but contains only examples of the features of the second model.

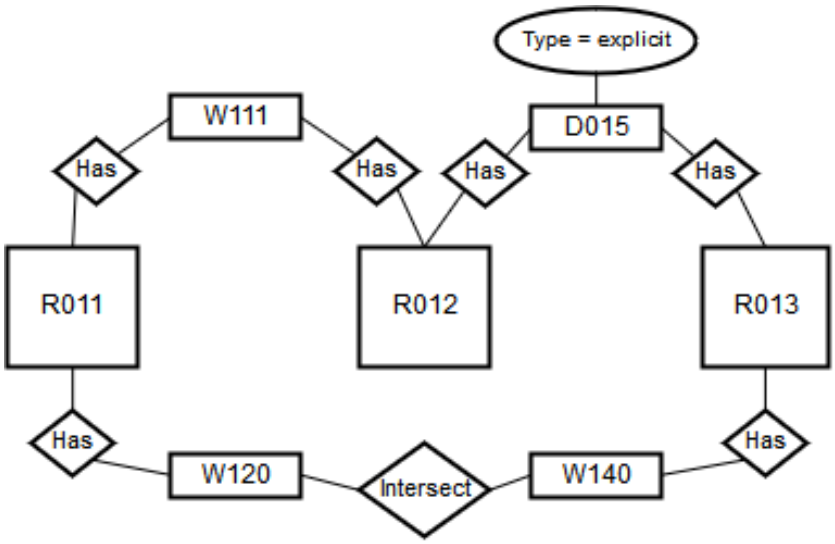


Figure 4.5: A partial example of a dataset represented with MODEL2.

The kLog representation of the example contained in Figure 4.5 is:

room(*r*011)
room(r012)
room(r013)
wall(w111)
wall(w120)
wall(w140)
door(d015)
has_door(r012,d015)
has_door(r013,d015)
doortype(d015,explicit)
has_wall(r011,w111)
has_wall(r012,w111)
has_wall(r011,w120)
has_wall(r013,w140)
intersect(w120,w140)

The relations resulting from kLog reasoning are:

connected(r012,r013)
adjacent(r011,r012)
adjacent(r012,r013)

Chapter 5

Software tools for creating the datasets

Blissful quiet
the rocking of a recent love
is both repose and anguish
in my fainting dreams

Racter

In this chapter, we present the two software tools developed to obtain the datasets that are provided to kLog. The input of the process is an image representing a two-dimensional view of a floor plan of a building, while the desired output is a list of ground atoms represented in kLog language. The first tool is the Floor Plans Editor (FPE) and allows to create an XML file that represents geometry and topology of a floor plan. The second instrument is a translator called RoomsPy, that allows to parse several input file formats (including the above mentioned XML), to extract geometrical features, and to export the results in several output formats (including kLog language).

5.1 Floor Plans Editor (FPE)

The first component developed is FPE, a tool used to create a dataset or to refine an existing one. A dataset, in our case, is composed of a set of floor plans from different buildings, stored in XML files. FPE allows to create a

floor plan from scratch, to visualize an existing one and, possibly, to modify it.

A similar tool was already proposed in [25], allowing users to design a topological map of the environment; we move a step further since our tool permits also to model the metric representation of a floor plan and to automatically extract its topology. Furthermore, with the geometrical representation of the space, we allow users to extract (with other tools) geometrical features of the environment.

The implementation of FPE is motivated by the need to obtain, in order to test semantic mapping approaches, many data all belonging to the same building type (e.g., schools, offices) in a faster way than using a robot: while a robot exploration may take hours to obtain the map of a single floor plan, with FPE an human being can obtain it in minutes. Furthermore, the data obtained are of a higher quality than the usual results of robotic exploration and this allow us to better stress high-level reasoning capabilities, without the shortcomings of imprecise data.

5.1.1 General schema

FPE is an application developed in Processing [1], a Java dialect, using its libraries for graphical representation. Figure 5.1 contains a screenshot of the software.

The interface is very simple and user-friendly. The left part of the screen contains the working area, where the user can manage the metric floor plan and the consequent topological graph. In this area, the tool allows to plot simultaneously three elements: a background image containing a sketch of the floor plan, the metric representation of the environment, and the topological graph of rooms and their connections. In the right part of the screen, instead, there is a column containing the set of available labels and some buttons used to load a floor plan, perform changes, and save the results.

5.1.2 Creation

The starting point, in this scenario, is an image containing the floor plan of a building, as the one presented in Figure 5.2. FPE permits the creation from scratch of a semantic map containing all the elements discussed in Chapter 3 (i.e.,nodes,paths,edges,landmarks, and districts). This is an essential func-

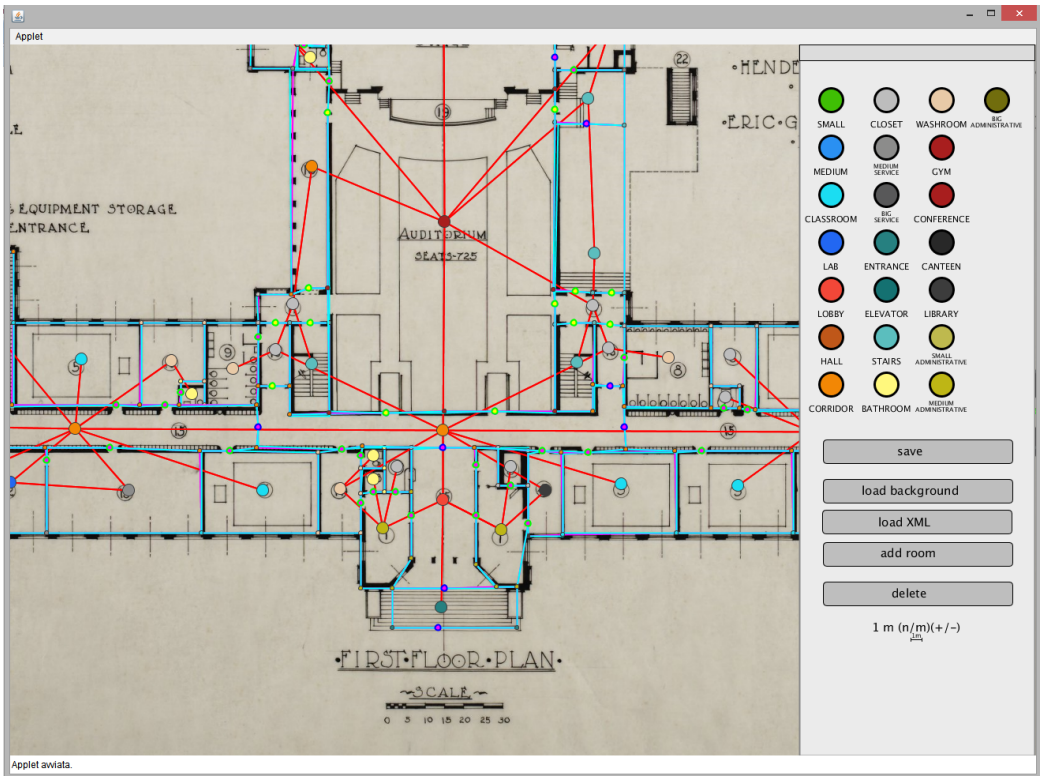


Figure 5.1: A screenshot of Floor Plans Editor (FPE). The left side contains the working area, the top-right part the labels of the SCHOOL *building type, and the bottom-right part the buttons for the editing operations.*

tionality for all those cases where an actual perception of the environment is missing, unfeasible or even impossible.

The first step in the process consists in loading into the application an image containing the floor plan. Once the image is loaded, it is necessary to correctly set up the scale factor that represents the correspondence between pixels in the image and meters in the real world. This is a very important passage since images may come from different sources and there is no guarantee that they have the same scale factor. The procedure is performed using a graduation tick that it is possible to lengthen and to shorten until it represents 90 cm of the real world (a standard measure for a door).

Subsequently, it is possible to add rooms to the map. After having pressed the *add a room* button, it is necessary to choose a label from the list in the top-right part of the screen; then, the user can click on the center of a room to place a node of the topological graph. The next step consists in specifying the

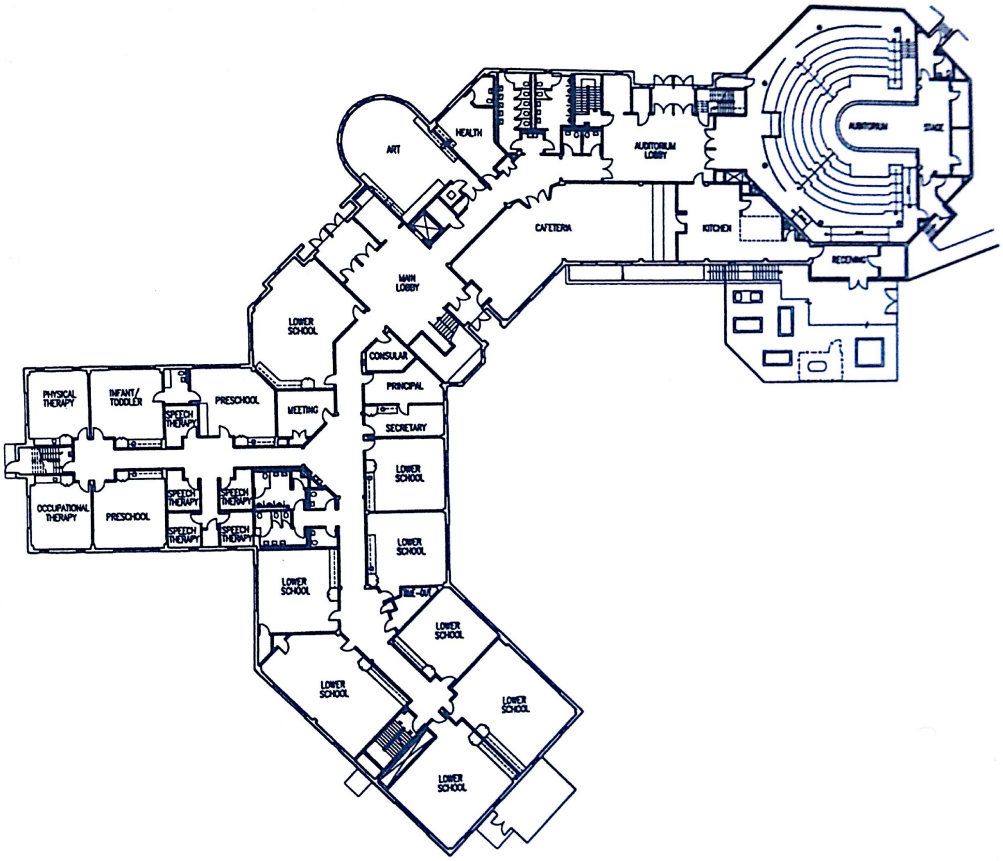


Figure 5.2: An example of floor plan of a school, from [69].

points of the bounding polygon delimiting the room. Each side of the polygon is represented with a line segment which has a specific type: wall, implicit door, or explicit door. The default option is that a linsegment represents a wall, while a keyboard input is used to specify a different type. An example of room insertion is presented in Figure 5.3.

This process is repeated for each room in the floor plans until the whole floor is sketched. The connections between nodes of the topological graph are automatically detected by the software: when two rooms share a door, a connection between them is added. The result obtained starting from the previously presented background image is shown in Figure 5.4.

In the end, it is possible, pressing the corresponding button, to save the graph in an XML file that contains the fully detailed description of the

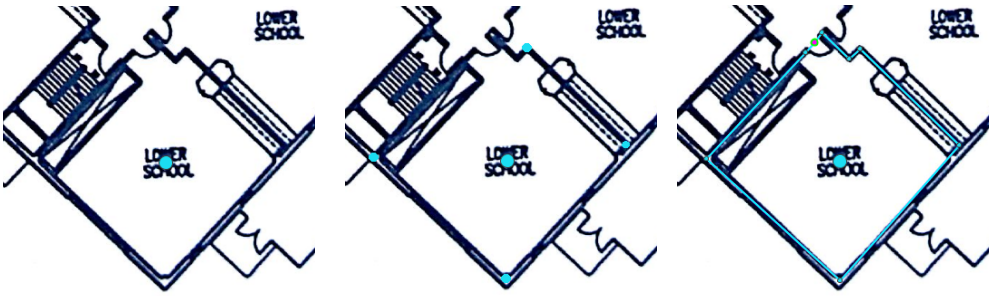


Figure 5.3: An example of room insertion. Left: the insertion of the node representing the room. Center: the insertion of the vertex of the polygon representing the shape boundary of the room (i.e., the corners of the room). Right: the final result of the insertion, containing 6 *walls and a door.*

environment. An example of output XML file is shown in Figure 5.5

5.1.3 Editor

With our editor, it is possible to keep an image of the floor plan in background and load a previously created XML referred to the same floor plan, that will be plotted in the foreground. In this way, it is easy to see errors in the sketch and promptly fix them.

The basic operations are the insertion and the deletion of a room. For the insertion the procedure is the same as in the case of the creation of a room from scratch. To delete a room the procedure is very simple, it is just needed to select a room clicking its node on the graph and subsequently click the deletion button. Another possible operation is the change of the label of a room, it is just needed to select a room and then click on a new label in the list on the right.

The scenarios where our editor may result useful are multiple and we want to highlight some of them. First, there is the possibility to correct mistakes occurred in the creation phase, in order to refine the data. A second scenario may arise in the case of a change in the label set. With our editing system, it would be easy to update the labels of an already existing floor plan, for example dividing rooms in more specific subsets. Finally, we have to consider that buildings are subject to refurbishment and a user may want to reflect the modifications of a floor plan into the dataset.

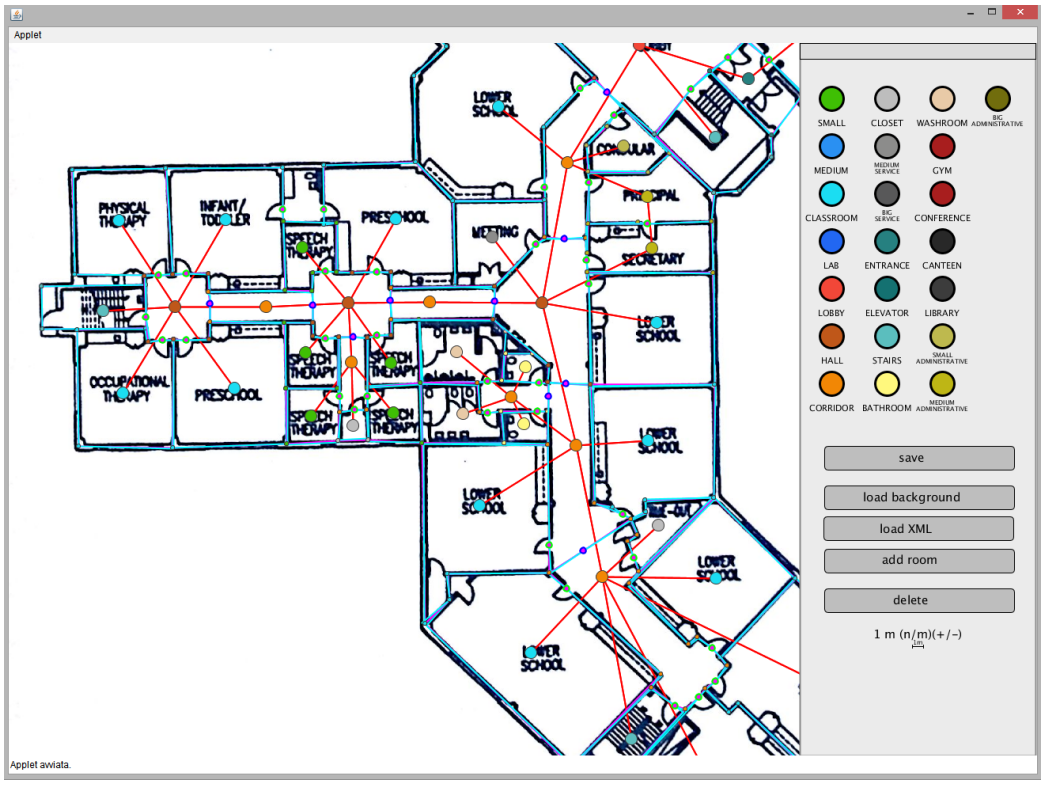


Figure 5.4: The result of the creation of a map, starting from Figure 5.2, using FPE.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<building id="3f0b8784-33ee-4082-9e96-61faa44ba0e3">
  <scale>
    <represented_distance>
      <value>7</value>
      <um>pixel</um>
    </represented_distance>
    <real_distance>
      <value>90</value>
      <um>cm</um>
    </real_distance>
  </scale>
  <building_type>
    <main_type>SCHOOL</main_type>
  </building_type>
  <floor>
    <spaces>
      <space id="bfeb6869-3c8a-471d-8fee-6c4e51121457">
        <labels>
          <type>R</type>
          <label>CLASSROOM</label>
        </labels>
        <centroid>
          <point x="450" y="779"/>
        </centroid>
        <bounding_box>
          <maxx>
            <point x="516" y="769"/>
          </maxx>
          <maxy>
            <point x="450" y="844"/>
          </maxy>
          <minx>
            <point x="375" y="777"/>
          </minx>
          <miny>
            <point x="440" y="707"/>
          </miny>
        </bounding_box>
        <bounding_polygon>
          <point x="461" y="717"/>
          <point x="516" y="769"/>
          <point x="450" y="844"/>
          <point x="375" y="777"/>
          <point x="427" y="720"/>
          <point x="436" y="713"/>
          <point x="440" y="707"/>
          <point x="455" y="722"/>
          <point x="461" y="717"/>
        </bounding_polygon>
```

```xml
        </bounding_polygon>
        <space_representation>
          <linesegment id="a37c25da-1833-455e-af08-715e42b5bd3b">
            <point x="461" y="717"/>
            <point x="515" y="770"/>
            <class>WALL</class>
            <type>EXPLICIT</type>
            <features>NORMAL</features>
          </linesegment>
          <linesegment id="4d22142f-128b-4328-bc6e-b2f2ed9db99f">
            <point x="515" y="770"/>
            <point x="450" y="844"/>
            <class>WALL</class>
            <type>EXPLICIT</type>
            <features>NORMAL</features>
          </linesegment>
          <linesegment id="d8774b66-2228-4a3a-9f7b-4e8f11867522">
            <point x="450" y="844"/>
            <point x="376" y="777"/>
            <class>WALL</class>
            <type>EXPLICIT</type>
            <features>NORMAL</features>
          </linesegment>
          <linesegment id="05140dea-363f-46ef-a5e5-10d555e98595">
            <point x="376" y="777"/>
            <point x="431" y="719"/>
            <class>WALL</class>
            <type>EXPLICIT</type>
            <features>NORMAL</features>
          </linesegment>
          <linesegment id="e6f891b7-b169-4e5c-aafe-41f6e2c8b8b6">
            <point x="431" y="719"/>
            <point x="436" y="713"/>
            <class>WALL</class>
            <type>EXPLICIT</type>
            <features>NORMAL</features>
          </linesegment>
          <linesegment id="b7a2fa62-ae6e-4d3c-9462-ee3afb57d154">
            <point x="436" y="713"/>
            <point x="436" y="713"/>
            <class>PORTAL</class>
            <type>EXPLICIT</type>
            <features>NORMAL</features>
          </linesegment>
          <linesegment id="be05b745-6681-4813-9297-d27371eb27d5">
            <point x="436" y="713"/>
            <point x="440" y="708"/>
            <class>WALL</class>
            <type>EXPLICIT</type>
            <features>NORMAL</features>
          </linesegment>
```


Figure 5.5: A portion of the XML file derived from the map presented in Figure 5.4.

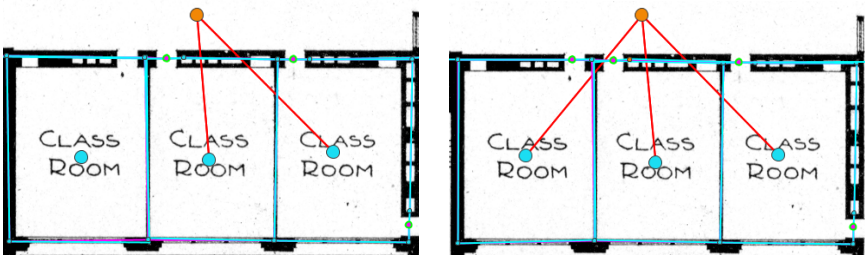


Figure 5.6: An example of correction of a missing door. Left: the map where the door was missing. Right: the corrected map.

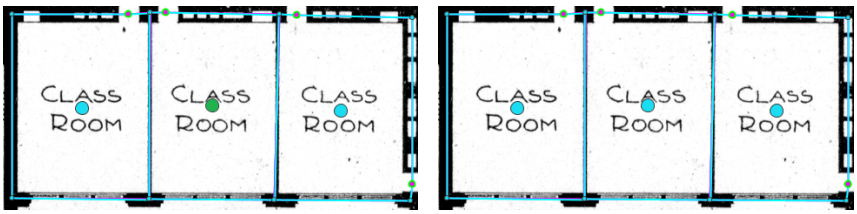


Figure 5.7: An example of label correction. Left: the central room labeled with the wrong label. Right: the same room with the correct label.

5.1.4 Post-editing

Once the dataset is completed, several refinement methods are automatically applied. First, points that should coincide but are separated (due to representation errors) are grouped. An example of this correction is represented in Figure 5.8: suppose to have a corner shared, in the real world, by three walls but that the line segments representing them in the metric map have their endpoints at a negligible distance. With a simple correction, these points will be grouped all in the very same location.

Another improvement is the automatic identification of connected rooms. This is done checking whether two rooms share a door and, in the case, adding a topological connection between them. Figure 5.9 shows an example of this procedure.

Finally, there is a function that unifies the identifiers of the elements of the map. It is in fact possible that in the creation phase the same door or wall has different identifiers in the different rooms where it appears, in this post-editing phase these errors are corrected. Once two objects of the metric map are found to coincide, their identifiers (and all their references contained in other objects) are unified in a unique new id.

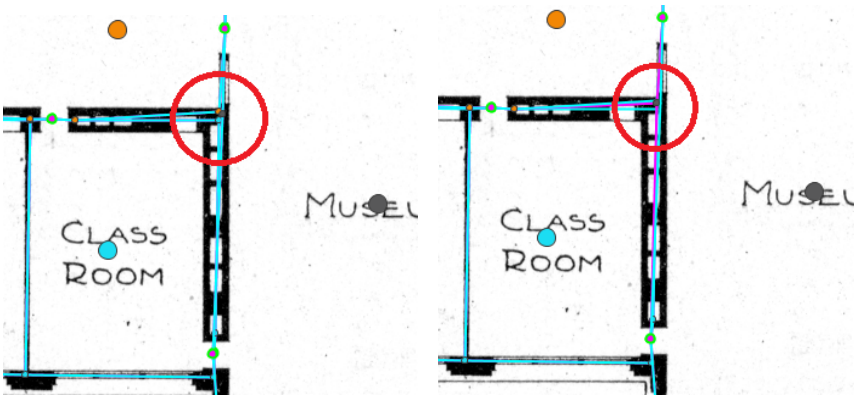


Figure 5.8: An example of endpoints automatic correction. Left: three separate points. Right: an unique point.

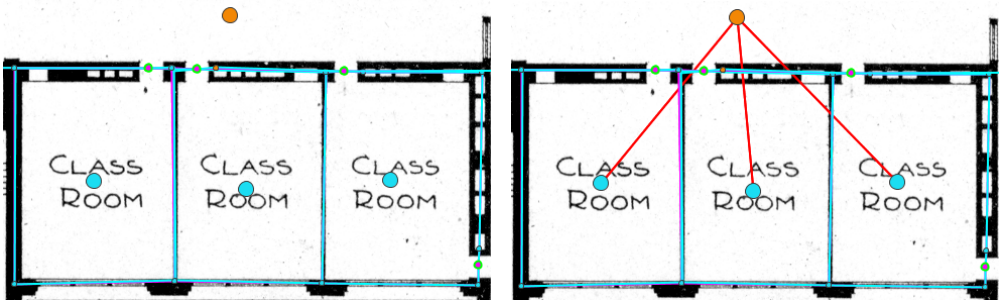


Figure 5.9: An example of automatic identification of connected rooms. Left: three rooms sharing doors with the corridor but still not connected in the topological graph. Right: the same rooms connected topologically to the corridor.

5.2 RoomsPy

The second software component developed in this thesis is RoomsPy which is a collection of python scripts able to parse input data representing one or more floor plans, extract the desired features, and write the results in one of the available formats. It was conceived as an interface to be used to integrate several other scripts operating on different data formats in order to reuse the extracted information. We developed only the functionalities related to this thesis but the system is designed to facilitate its extension by future users who can add new input/output formats or extract different features from the data.

5.2.1 Input formats

The first functionality of RoomsPy is to read several data formats and transform floor plans into its internal representation. For our purposes we manage two data formats:

FPE XML: it is the output format of FPE, shown in Figure 5.5;

MIT XML: in this fomat were saved the data presented in [7], containing the representation of MIT campus.

5.2.2 Feature extraction

We extract from the data eight different features, choosen among the ones presented in [63]. These features are calculated using the bounding polygon $P(r\ )$of each room r:

$$P(r) = \{v_0, v_1, ..., v_{N-1}, v_N \equiv v_0\}$$

where each v_i is a vertex of the polygon having coordinates x_i, y_i.
An example of $P(r)$ is shown in the right part of Figure 5.3 where the vertex $\{v_0, v_1, ..., v_8\}$ are the blue points at the corners of the room.

The features we extract are:

Area: the area of the polygon $P(r)$, given by

$$f_{area} = \frac{1}{2} \sum_{i=0}^{N-1} (x_i y_{i+1} - x_{i+1} y_i)$$

Perimeter: the perimeter of the polygon $P(r)$, given by

$$f_{perimeter} = \sum_{i=0}^{N-1} dist(v_i, v_{i+1})$$

where $dist(v_i, v_{i+1}) = \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2}$

The ratio between area and perimeter: the area of $P(r)$ divided by its perimeter defined as

$f_{AP} = \frac{f_{area}}{f_{perimeter}}$

Mean distance between the centroid and the shape boundary: calculated as

$$f_{mean_shape} = \frac{1}{N} \sum_{i=0}^{N-1} dist(v_i, c)$$

where $dist(v_i, c) = \sqrt{(x_i - c_x)^2 + (y_i - c_y)^2}$

given that c_x and c_y are the coordinates of the centroid $c = (c_x, c_y)$ defined as

$$c_x = \frac{1}{6f_{area}} \sum_{i=0}^{N-1} (x_i + x_{i+1})(x_i y_{i+1} - x_{i+1} y_i)$$

$$c_y = \frac{1}{6f_{area}} \sum_{i=0}^{N-1} (y_i + y_{i+1})(x_i y_{i+1} - x_{i+1} y_i)$$

Form factor: the form factor of polygon $P(r)$, defined as

$$f_{form_factor} = \frac{4\pi f_{area}}{\sqrt{f_{perimeter}}}$$

Circularity: the circularity of polygon $P(r)$, defined as

$$f_{circularity} = \frac{f_{perimeter}^2}{f_{area}}$$

Normalized circularity: the normalized circularity of $P(r)$, defined as

$$f_{ncirc} = \frac{4\pi f_{area}}{f_{perimeter}^2}$$

Average normalized distance between the centroid and the shapeboundary: given the previous definitions of centroid and distance from centroid,

can be computed as

$$f_{n_dist_shape} = \frac{1}{N} \sum_{i=0}^{N-1} \overline{dist}(v_i, c)$$

where $\overline{dist}(v_i, c) = \frac{dist(v_i,c)}{max_dist(v_i,c)\forall i}$

5.2.3 Output formats

We manage two possible output formats for the data:

Enriched XML: it is an XML file of the same type saved by the tool FPE but enriched with the extracted features. The XML file with the features extracted from the previously presented example can be found in Figure 5.10;

kLog language: it is the format needed by kLog as input data. It contains the dataset translated in kLog language, where the environment is represented using logical formulas.

A kLog file contains more floor plans, each one representing an interpretation, that is, the content a single XML file. For this reason the formulas contained in this data format become:

$$interpretation(interpretation_id, predicate).$$

where *predicate* is a logical formula as the ones presented in Chapter 4 and *interpretation_id* is an identifier that represents univocally each interpretation. As *interpretation_id*, RoomsPy assigns to each parsed XML its own name. The translation in kLog language of the previous example is shown in Figure 5.11 and Figure 5.12, representing the two relational models employed in this thesis, respectively MODEL1 and MODEL2 presented in Chapter 3.

```
<features>
  <area value="165" />
  <perimeter value="52.0" />
  <aoverp value="3.17307692308" />
  <adcs value="27.8123766619" />
  <ff value="287.535940132" />
  <circularity value="16.3878787879" />
  <normalcirc value="0.766808857755" />
  <andcs value="0.794963874496" />
</features>
```

Figure 5.10: A portion of the XML file enriched with features derived from the XML file presented in Figure 5.5.

```
interpretation(millneckmanor,room(r141e82)).
interpretation(millneckmanor,area(r141e82,183)).
interpretation(millneckmanor,perimeter(r141e82,56.0)).
interpretation(millneckmanor,aoverp(r141e82,3.26785714286)).
interpretation(millneckmanor,adcs(r141e82,16.9002311334)).
interpretation(millneckmanor,ff(r141e82,307.30309923)).
interpretation(millneckmanor,circularity(r141e82,17.1366120219)).
interpretation(millneckmanor,normalcirc(r141e82,0.73330542807)).
interpretation(millneckmanor,andcs(r141e82,0.634701502828)).
interpretation(millneckmanor,connected(r141e82,r55fba5)).
interpretation(millneckmanor,room(r3ee56c)).
interpretation(millneckmanor,area(r3ee56c,165)).
interpretation(millneckmanor,perimeter(r3ee56c,52.0)).
interpretation(millneckmanor,aoverp(r3ee56c,3.17307692308)).
interpretation(millneckmanor,adcs(r3ee56c,27.8123766619)).
interpretation(millneckmanor,ff(r3ee56c,287.535940132)).
interpretation(millneckmanor,circularity(r3ee56c,16.3878787879)).
interpretation(millneckmanor,normalcirc(r3ee56c,0.766808857755)).
interpretation(millneckmanor,andcs(r3ee56c,0.794963874496)).
interpretation(millneckmanor,connected(r3ee56c,r55fba5)).
interpretation(millneckmanor,room(r01f42d)).
interpretation(millneckmanor,area(r01f42d,93)).
interpretation(millneckmanor,perimeter(r01f42d,39.0453610172)).
interpretation(millneckmanor,aoverp(r01f42d,2.38184505348)).
interpretation(millneckmanor,adcs(r01f42d,36.6618538448)).
interpretation(millneckmanor,ff(r01f42d,187.028629393)).
interpretation(millneckmanor,circularity(r01f42d,16.3929055587)).
interpretation(millneckmanor,normalcirc(r01f42d,0.766573721134)).
interpretation(millneckmanor,andcs(r01f42d,0.871913277988)).
interpretation(millneckmanor,connected(r01f42d,r55fba5)).
interpretation(millneckmanor,room(re09699)).
interpretation(millneckmanor,area(re09699,88)).
interpretation(millneckmanor,perimeter(re09699,38.0)).
interpretation(millneckmanor,aoverp(re09699,2.31578947368)).
interpretation(millneckmanor,adcs(re09699,55.8204105775)).
interpretation(millneckmanor,ff(re09699,179.391035958)).
interpretation(millneckmanor,circularity(re09699,16.4090909091)).
interpretation(millneckmanor,normalcirc(re09699,0.765817599767)).
interpretation(millneckmanor,andcs(re09699,0.88503538985)).
interpretation(millneckmanor,connected(re09699,r480c3c)).
interpretation(millneckmanor,room(r1eba10)).
interpretation(millneckmanor,area(r1eba10,88)).
interpretation(millneckmanor,perimeter(r1eba10,38.0)).
interpretation(millneckmanor,aoverp(r1eba10,2.31578947368)).
interpretation(millneckmanor,adcs(r1eba10,59.5423998111)).
interpretation(millneckmanor,ff(r1eba10,179.391035958)).
interpretation(millneckmanor,circularity(r1eba10,16.4090909091)).
interpretation(millneckmanor,normalcirc(r1eba10,0.765817599767)).
interpretation(millneckmanor,andcs(r1eba10,0.890181049636)).
interpretation(millneckmanor,connected(r1eba10,r480c3c)).
interpretation(millneckmanor,room(r3e601b)).
```

Figure 5.11: A portion of kLog ground atoms derived from the XML file presented in Figure 5.5 using MODEL1.

```
interpretation(millneckmanor,room(r141e82)).
interpretation(millneckmanor,wall(ls70407055)).
interpretation(millneckmanor,haswall(r141e82,ls70407055)).
interpretation(millneckmanor,wall(ls70555755)).
interpretation(millneckmanor,haswall(r141e82,ls70555755)).
interpretation(millneckmanor,wall(ls575557144)).
interpretation(millneckmanor,haswall(r141e82,ls575557144)).
interpretation(millneckmanor,wall(ls5714472144)).
interpretation(millneckmanor,haswall(r141e82,ls5714472144)).
interpretation(millneckmanor,wall(ls7214472153)).
interpretation(millneckmanor,haswall(r141e82,ls7214472153)).
interpretation(millneckmanor,wall(ls72153195153)).
interpretation(millneckmanor,haswall(r141e82,ls72153195153)).
interpretation(millneckmanor,wall(ls195153206153)).
interpretation(millneckmanor,haswall(r141e82,ls195153206153)).
interpretation(millneckmanor,portal(ls206153206153)).
interpretation(millneckmanor,hasportal(r141e82,ls206153206153)).
interpretation(millneckmanor,portaltype(r141e82,explicit)).
interpretation(millneckmanor,wall(ls206153227153)).
interpretation(millneckmanor,haswall(r141e82,ls206153227153)).
interpretation(millneckmanor,wall(ls22715322742)).
interpretation(millneckmanor,haswall(r141e82,ls22715322742)).
interpretation(millneckmanor,wall(ls227427042)).
interpretation(millneckmanor,haswall(r141e82,ls227427042)).
interpretation(millneckmanor,area(r141e82,183)).
interpretation(millneckmanor,perimeter(r141e82,56.0)).
interpretation(millneckmanor,aoverp(r141e82,3.26785714286)).
interpretation(millneckmanor,adcs(r141e82,16.9002311334)).
interpretation(millneckmanor,ff(r141e82,307.30309923)).
interpretation(millneckmanor,circularity(r141e82,17.1366120219)).
interpretation(millneckmanor,normalcirc(r141e82,0.73330542807)).
interpretation(millneckmanor,andcs(r141e82,0.634701502828)).
interpretation(millneckmanor,room(r3ee56c)).
interpretation(millneckmanor,wall(ls7215372302)).
interpretation(millneckmanor,haswall(r3ee56c,ls7215372302)).
interpretation(millneckmanor,wall(ls72302189302)).
interpretation(millneckmanor,haswall(r3ee56c,ls72302189302)).
interpretation(millneckmanor,wall(ls189302189284)).
interpretation(millneckmanor,haswall(r3ee56c,ls189302189284)).
interpretation(millneckmanor,portal(ls189284189284)).
interpretation(millneckmanor,hasportal(r3ee56c,ls189284189284)).
interpretation(millneckmanor,portaltype(r3ee56c,explicit)).
interpretation(millneckmanor,wall(ls189284189153)).
interpretation(millneckmanor,haswall(r3ee56c,ls189284189153)).
interpretation(millneckmanor,wall(ls18915372153)).
interpretation(millneckmanor,haswall(r3ee56c,ls18915372153)).
interpretation(millneckmanor,area(r3ee56c,165)).
interpretation(millneckmanor,perimeter(r3ee56c,52.0)).
interpretation(millneckmanor,aoverp(r3ee56c,3.17307692308)).
interpretation(millneckmanor,adcs(r3ee56c,27.8123766619)).
interpretation(millneckmanor,ff(r3ee56c,287.535940132)).
```

Figure 5.12: A portion of kLog ground atoms derived from the XML file presented in Figure 5.5 using MODEL2.

Chapter 6

Experimental evaluation

You broke my soul,
the juice of eternity,
the spirit of my lips.

Ray Kurzweil's Cybernetic Poet

This chapter discusses the experimental evaluation of our framework in various settings.

In the first part of the chapter, we explain how specific learning tasks can be defined in kLog, providing the most relevant predicates needed in order to set up learning problems. We then illustrate how the results of our experimental evaluations are measured, explaining some common machine learning assessment methods.

In the second part of the chapter we present in detail the three classes of experiments that we performed:

Place classification: the reproduction of the classical semantic mapping task of assigning a label to each room of a floor plan;

Building classification: the classification of an entire floor plan, achieved considering its whole structure and aiming to classify it with the correct building type;

Dataset validation: the binary classification between real floor plans and simulated environments.

For each one of the listed experiments we first describe the learning task, than we explain which datasets we use, providing raw results, and finally we analyze the results showing streghts and shortcomings of our approach.

In the first experiment the novelty is represented by our approach, in contrast to the usual *local* approach, we use a *global* approach which assigns labels considering the whole structure of the floor plan. In the second and the third experiment, the novelty is per se the capability of our global approach to address them, since they cannot be easily addressed by classical local semantic mapping methods.

6.1 Task definition and evaluation methods

In order to define a learning task in kLog it is necessary to write a Prolog program which calls the kLog libraries and contains the formulation of the learning problem. The first part of the program contains the *domain* of the problem, that is the list of extensional signatures (describing entities, relations, and attributes of the relational model) and the intensional signatures (representing the background knowledge) that will be transformed into ground atoms using Prolog reasoning capabilities. The second part of the program contains a list of predicates describing the learning problem and setting kLog parameters.
The principal predicates are:

***klog_flag(nspdk,distance,VALUE)*:** sets the distance parameter d^*, the maximum distance between pairs of subgraphs compared by NSPDK, as explained in Chapter 4.

***klog_flag(nspdk,radius,VALUE)*:** sets the radius parameter r^*, the maximum radius of expansion of each subgraph compared by NSPDK, as explained in Chapter 4.

It is important to highlight that the parameters d^* and r^* are the maximum values for the parameters d and r of the kernel: NSPDK iteratively applies all the values $r = 1,2,...,r^*$ and $d = 1,2,...,d^*$. The choice of d^* and r^* depends on the relational model used and on the density of the graph derived from ground atoms, and it is not computable a priori. Running several times the tests it is possible to identify the optimal values D and R that maximize effectiveness and efficiency: values lower than D and R would

penalize the quality of the results while values higher than D and R would produce a higher computational overhead without a corresponding increase in performance. In our case, we discovered that for both our models, using floor plans of schools and offices, $D = 4$ and $R = 5$.

***klog_flag(nspdk,match_type,TYPE)*:** indicates the matching type of the kernel. TYPE assumes the value *soft* for the soft-matching version of NSPDK and *hard* for the hard-matching version.

***attach(FILE)*:** indicates the name of the external file containing the ground atoms to be used as dataset for the task; the included file must be a .pl file and must be in the same location of the principal program. The dataset can also be splitted in more than one file, in this case a list of Prolog file names can be specified.

***kfold(RELATION,PARAM)*:** indicates the scope of the learning task, i.e., the relation to be predicted. kLog natively implements *k-fold cross-validation* and the value of PARAM defines the quantity of folds (k).

k-fold cross-validation is a common technique for estimating the performance of a classifier. In k-fold cross-validation, the original set of n samples (in our case n interpretations) is randomly partitioned into k subsamples (each one containing approximately n/k interpreatations). Of the k subsamples, a single fold is selected as the validation set for testing the model, and the remaining $k - 1$ folds are used as training set. The cross-validation process is then repeated k times, with each of the k subsamples used exactly once as the validation set. In kLog, the k results from the folds are then combined to resemble the result of a single classification.

In all our experiments, we used *leave-one-out cross-validation* that is a particular case of k-fold cross-validation where $k = n$; therefore each fold is composed of exactly one interpretation.

The experimental evaluation depends on the learning task. In the context of multiclass classification, we use as a metric to evaluate the performance of our method the number of the correctly classified labels over the total number of assigned labels.

In the case of binary classification we use two graphical instruments to evaluate the performance of our method: *receiver operating characteristic*

curve (ROC curve) and *precision-recall curve* (PR curve). The first one computes and plots true-positive rate (i.e., true facts correctly classified over the total amount of true facts) against false-positive rate (i.e., false facts uncorrectly classified as true over the total amount of false facts) at different values of a threshold parameter. The performance of a classifier is better as the area under its ROC curve is larger. The ROC curve of a random binary classifier is the bisector of the first quadrant and has an area equal to 1/2 while a perfect classifier has a ROC curve that starts from $(0, 0)$, moves vertically up to $(0, 1)$, and then horizontally to $(1, 1)$, with an area equal to 1. Figure 6.1 represents some examples of ROC curves.

The precision-recall curve is a graph which has on the y-axis precision (i.e., true facts correctly classified over the total facts classified as true) and on the x-axis recall (that is equivalent to true-positive rate) at different values of a threshold parameter. As for the ROC curve, the larger the area under the curve, the better the classifier. The perfect classifier has an area under the PR curve equal to 1. Figure 6.2 represents some examples of PR curves.

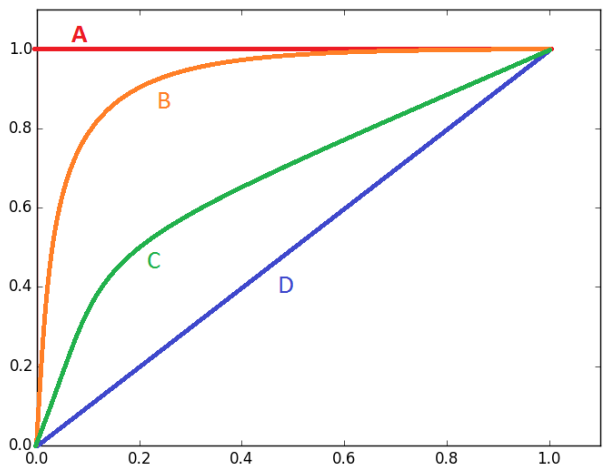


Figure 6.1: Examples of ROC curves. In red (A) it is represented the ROC curve of a perfect classifier, while in blue (D) it is represented the ROC curve of a random classifier. The orange (B) and the green (C) ROC curves belong to other two classifiers and it is possible to state that the orange curve represents a better classifier since it has a larger area under the curve than the green one.

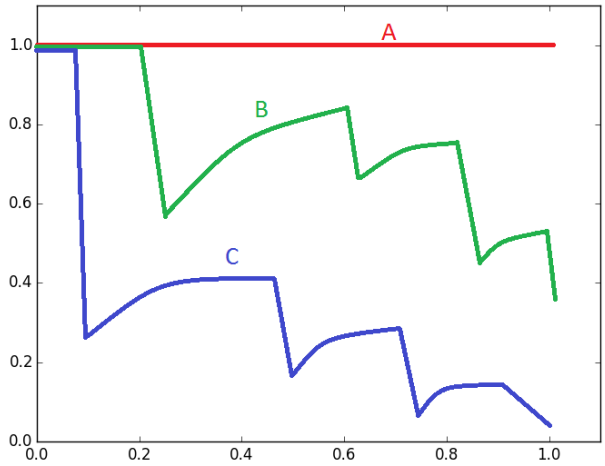


Figure 6.2: Examples of ROC curves. In red (A) it is represented the ROC curve of a perfect classifier. The green (B) and the blue (C) ROC curves belong to other two classifiers and it is possible to state that the green curve represents a better classifier since it has a larger area under the curve than the blue one.

6.2 Place classification

The first learning task we implemented is a place classification task. This test mimics the classical task of semantic mapping of assigning semantic labels to spaces of an already explored environment, knowing only the metric map of these spaces and, more generally, data obtained from sensor readings. The novelty of the approach used in this thesis is the possibility to reason on the whole structure of a floor plan: even if the labels are assigned to one room at a time, the classifier considers not only the characteristics of the current room, but also the characteristics of the other rooms of the floor plan and its overall structure.

The nature of the task in kLog depends principally on the labeling schema adopted. When the labeling schema is $\mathcal{L}_{R/C}$, kLog performs a binary classification, deciding whether each room of the floor plan is either a ROOM or a CORRIDOR. When the labeling schema is $\mathcal{L}_{R/C/E/S}$ or $\mathcal{L}_{type}$, kLog performs a multiclass classification task for each room of the floor plan, predicting the associated label. In the case of $\mathcal{L}_{R/C/E/S}$ the labels are generic and describe

the functionality of a room at an abstract level. In the case of $\mathcal{L}_{type}$ the classification process assigns to each space an higly specialized label, as the ones presented in Figure 3.4. To perform place classification in kLog, no additional signature needs to be specified, it is just needed to specify the scope of learning task:

kfold(corridor,PARAM)

when the labeling schema is $\mathcal{L}_{R/C}$, or

kfold(label,PARAM)

for the other labeling schemas.

6.2.1 Data and results

We performed place classification using the data obtained with our tools (30 plans of entire floors of school buildings and 20 floor plans of office buildings). Each building is represented both with the more abstract formalism of MODEL1 and with the more concrete formalism of MODEL2. For these two settings, each one of the four labeling schemas ($\mathcal{L}_{R/C}$,$\mathcal{L}_{R/C/E/S}$, $\mathcal{L}_{\text{SCHOOL}}$, and $\mathcal{L}_{\text{OFFICE}}$) is used.

Using MODEL1 (i.e., using the topological graph of *connected* relations and the extracted geometrical features) we obtain the following results:

$\mathcal{L}_{R/C}$: kLog is able to perform an excellent classification for both building types (SCHOOLs and OFFICEs), with areas under the ROC curve of 0.96 and 0.95, and areas under the PR curve of 0.81 and 0.85, as it can be seen in Figure 6.3 and Figure 6.4, respectively.

$\mathcal{L}_{R/C/E/S}$: kLog is able to perform multiclass classification with an average classification error of 47% for the SCHOOL building type and 46% for the OFFICE building type.

$\mathcal{L}_{SCHOOL}$ and $\mathcal{L}_{OFFICE}$: kLog is able to perform multiclass classification with an average classification error of 66% for the SCHOOL building type and 62% for the OFFICE building type.

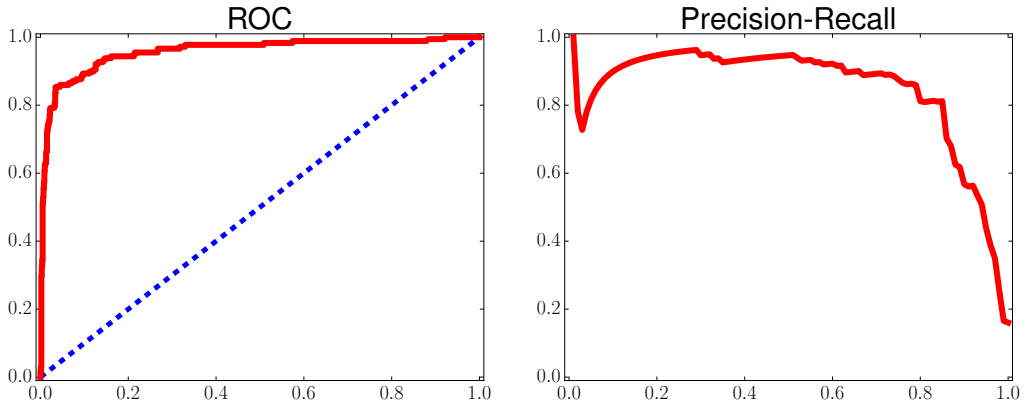


Figure 6.3: Results of place classification of SCHOOLs *using* MODEL1 *and* $\mathcal{L}_{R/C}$*. Left: ROC curve, with an area under the curve of* 0.96*. Right: PR curve, with an area under the curve of* 0.81*.*

Using MODEL2 (i.e., representing directly walls and doors, and deriving the *adjacent* relations between spaces) we obtain the following results:

$\mathcal{L}_{R/C}$**:** kLog is able to perform an excellent classification for both building types (SCHOOLs and OFFICEs), with areas under the ROC curve of 0.92 and 0.88, and areas under the PR curve of 0.74 and 0.66, as it can be seen in Figure 6.5 and Figure 6.6, respectively.

$\mathcal{L}_{R/C/E/S}$**:** kLog is able to perform multiclass classification with an average classification error of 61% for the SCHOOL building type and 60% for the OFFICE building type.

$\mathcal{L}_{SCHOOL}$ **and** $\mathcal{L}_{OFFICE}$**:** kLog is able to perform multiclass classification with an average classification error of 75% for the SCHOOL building type and 73% for the OFFICE building type.

6.2.2 Analysis of the results

If we evaluate the results altogether, we can outline two different behaviors: when it is asked to determine a label between ROOM and CORRIDOR, kLog achieves almost perfect results, while, when the learning task is to distinguish the functionality of a room, using more than one label (even in the high-level classification between FUNCTIONAL ROOM and SERVICE ROOM with only four

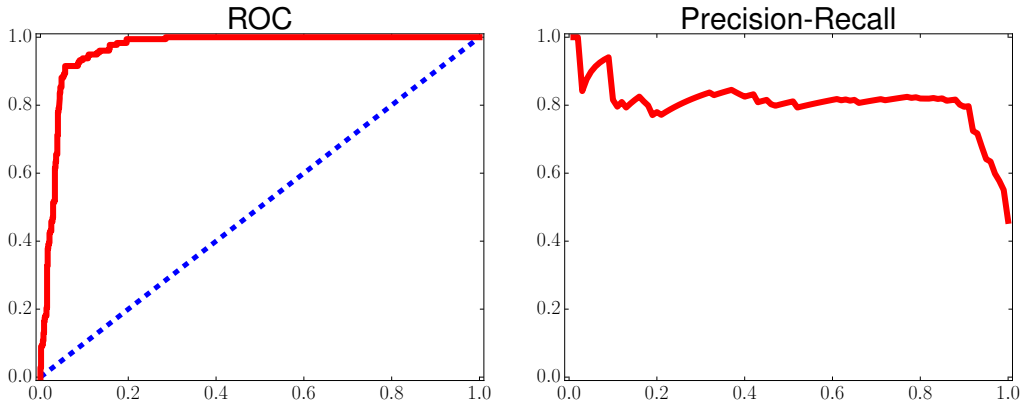


Figure 6.4: Results of place classification of OFFICEs *using* MODEL1 *and* $\mathcal{L}_{R/C}$*. Left: ROC curve, with an area under the curve of* 0.95*. Right: PR curve, with an area under the curve of* 0.85*.*

labels) the quality of the results considerably decreases. This suggests that, in order to classify correctly spaces according to very specific labels, global knowledge about geometry and connections of the spaces is not sufficient, and additional information about spaces (e.g., their appearance obtained from a camera, as in [72]) should be considered.

Comparing the two models, it can be noticed that the results obtained with MODEL1 slightly overcome the results obtained with MODEL2 in all the presented learning tasks. This can be explained by the fact that the extracted geometrical features well summarize the metric representation of the environment. Thus, the direct insertion of physical elements (e.g., doors, walls) in the model does not increase the performance of the classification task. Another possible explanation may be that topological information is much more relevant than geometrical features in discriminating between ROOMs and CORRIDORs. This explanation validates our approach, which considers the whole structure of a floor plan (global approach) and not only the geometrical features of each single room (local approaches in the state of the art of semantic mapping [7, 38, 64]).

6.3 Building classification

Global reasoning enables learning tasks that go beyond what can be done with classical local approaches, like classifying an entire building as a whole,

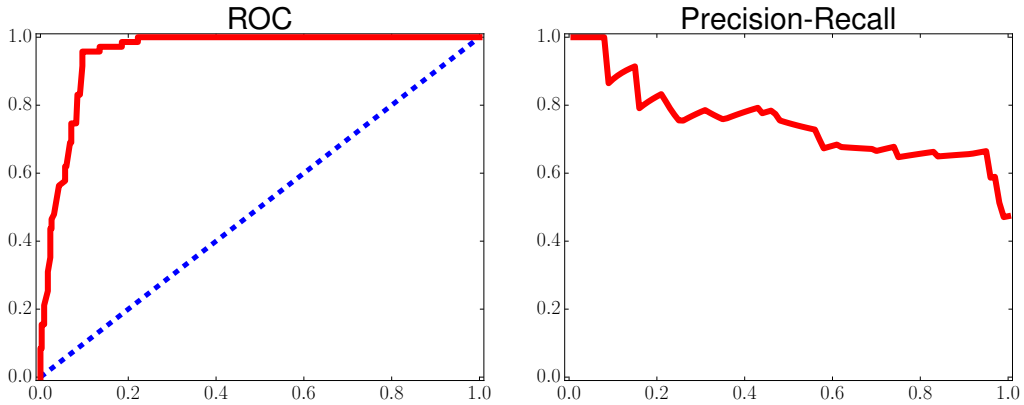


Figure 6.5: Results of place classification of SCHOOLs *using* MODEL2 *and* $\mathcal{L}_{R/C}$*. Left: ROC curve, with an area under the curve of* 0.92*. Right: PR curve, with an area under the curve of* 0.74*.*

without separating it in smaller instances (rooms) as in place classification. We refer to this task as *building classification*: while performing building classification we assume to know the structure and the geometry of a floor of a building, and we ask kLog to perform multiclass classification categorizing the entire building as one of the available building types. In our experiments we consider two different building types, OFFICE and SCHOOL. In kLog we add the following extensional signature:

building_type(bt::property)::extensional.

and in each interpretation one of following mutual exclusive ground atoms:

building_type(office)
building_type(school)

Finally, it is needed to specify the scope of the learning task:

kfold(building_type,PARAM)

The ability of classifying buildings could be useful for a robot moving in large heterogeneous structured environments, like a university campus, and needing to detect whether the floor in which it is currently navigating hosts

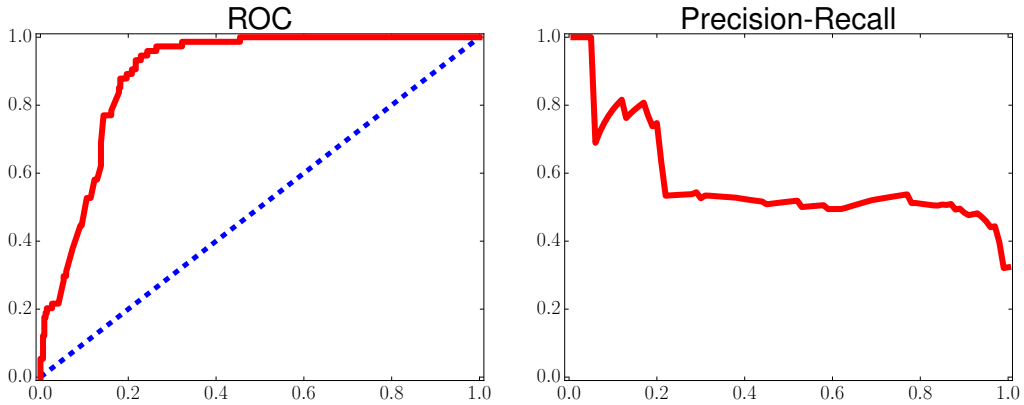


Figure 6.6: Results of place classification of OFFICEs *using* MODEL2 *and* $\mathcal{L}_{R/C}$*. Left: ROC curve, with an area under the curve of* 0.88*. Right: PR curve, with an area under the curve of* 0.66*.*

teaching activities (i.e., it is composed of classrooms and lecture halls) or the administrative section of the campus (i.e., it is composed of faculty's offices, reception,...).

We test this ability with other experiments, using the publicly available datasets of the MIT campus [14]. In this experiments we classify floor plans between DOORMs, OFFICEs, and RESEARCH LABs building types. In kLog we added the following extensional signatures:

building_type(bt::property)::extensional.

and in each interpretation one of following mutual exclusive ground atoms:

building_type(doorm)
building_type(office)
building_type(research lab)

6.3.1 Data and results

We perform building classification using the data obtained with our tools (10 floor plans of school buildings along with 10 floor plans of offices, all represented with MODEL1) with the $\mathcal{L}_{R/C}$, the $\mathcal{L}_{R/C/E/S}$, and the combination

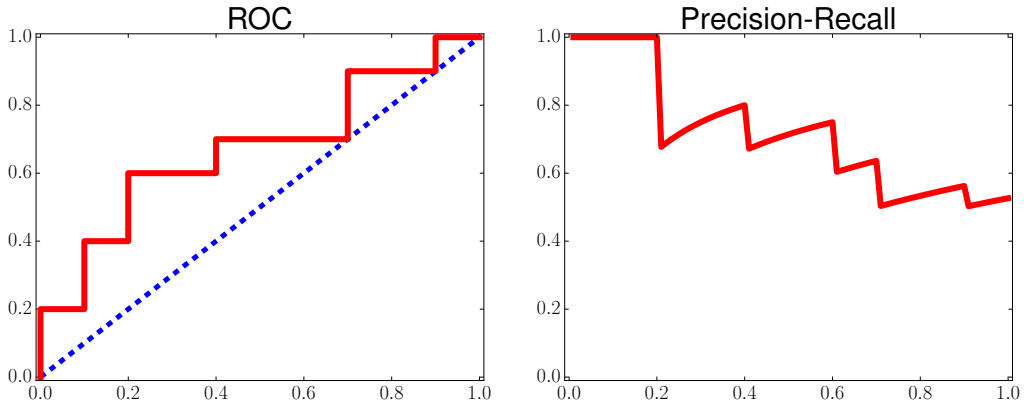


Figure 6.7: Results of building classification using $\mathcal{L}_{R/C}$. Left: ROC curve, with an area under the curve of 0.69*. Right: PR curve, with an area under the curve of* 0.71*.*

of $\mathcal{L}_{\text{SCHOOL}}$ and $\mathcal{L}_{\text{OFFICE}}$ labeling schemas, obtaining the following results:

$\mathcal{L}_{R/C}$: kLog is able to perform a good classification of the building type, with an area under the ROC curve of 0.69 and an area under the PR curve of 0.71, as it can be seen in Figure 6.7. It is important to highlight how, in this case, only a limited knowledge on the function of the spaces (ROOM or CORRIDOR) is present, thus the classifier relies primarily on the structure of the environment.

$\mathcal{L}_{R/C/E/s}$: kLog is able to perform a slighty better classification of the building type, with an area under the ROC curve of 0.74 and an area under the PR curve of 0.73, as it can be seen in Figure 6.8. It is important to highlight how a more specific knowledge on the functionalities of the spaces increases the quality of the classification.

$\mathcal{L}_{SCHOOL}$ and $\mathcal{L}_{OFFICE}$: with the complete label sets available, the system behaves as a perfect classifier, without failing the identification of any interpretation; the corresponding curves are represented in Figure 6.9. In this case, with a very fine grained knowledge about the semantic labels of the floor plan, the result is almost trivial since both $\mathcal{L}_{\text{SCHOOL}}$ and $\mathcal{L}_{\text{OFFICE}}$ contain exclusive labels (e.g., CUBICLE, CLASSROOM) which clearly characterise the building type.

We tested our building classification approach on the MIT campus, selecting a sample of 33 floorplans belonging to the 3 different types of buildings

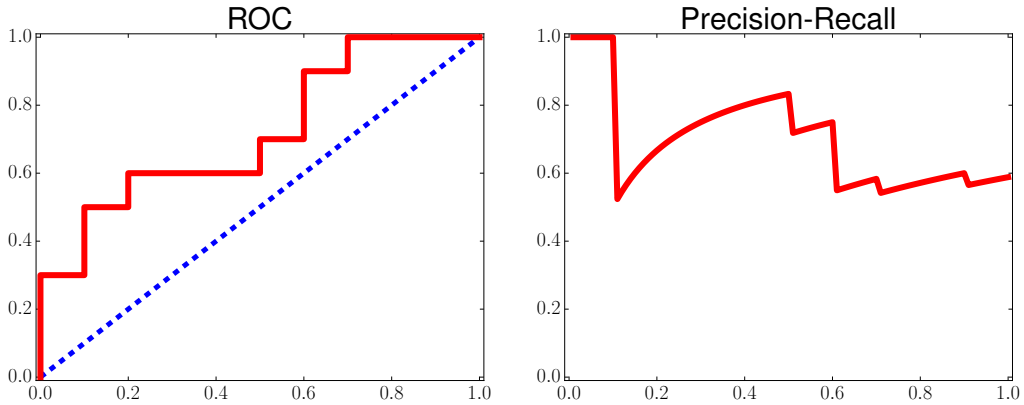


Figure 6.8: Results of building classification using $\mathcal{L}_{R/C/E/S}$. Left: ROC curve, with an area under the curve of 0.74*. Right: PR curve, with an area under the curve of* 0.73*.*

(DOORM, OFFICE, and RESEARCH LAB) and using their own specific labeling schema composed of 24 semantic labels. kLog obtained very good results, with 94% of accuracy in predicting the building type.

6.3.2 Analysis

We demonstrated that with our method it is possible to perform a learning task that is not possible using a local approach. The results of building classification are excellent given a highly detailed labeling schema ($\mathcal{L}_{type}$) but have been shown to be good also using the less informative $\mathcal{L}_{R/C}$ labeling schema and exploiting topological and metrical information. Reasoning on the global structure of a building can compensate the limited amount of information about the functions of the spaces, when the task is the classification of the whole building. These findings are confirmed by the results obtained with $\mathcal{L}_{R/C/E/S}$ that, as expected, are in between those obtained with the less informative labeling schema $\mathcal{L}_{R/C}$ and the perfect classification obtained with the more specific and informative labeling schemas $\mathcal{L}_{SCHOOL}$ and $\mathcal{L}_{OFFICE}$. MIT dataset further showed that our approach is effective when applied in a university campus where all the structures shared a single labeling schema. The classifier is able to identify building types analyzing their global topological structure, enriched with geometrical features and semantic labels.

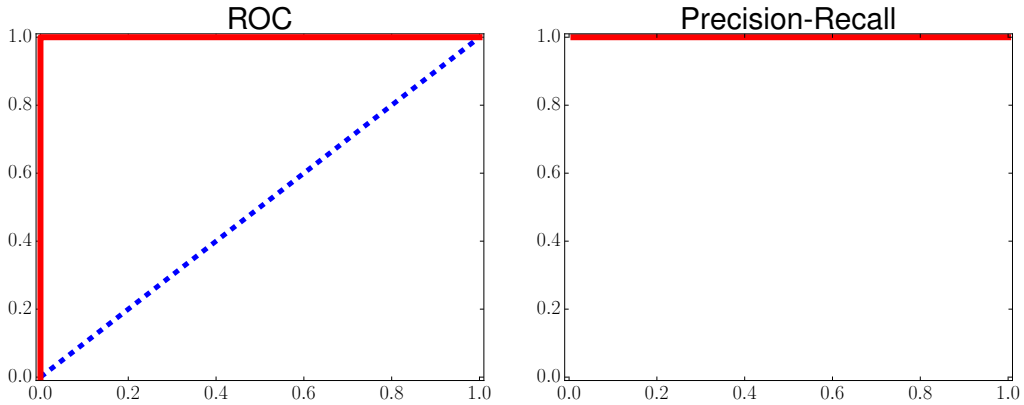


Figure 6.9: Results of building classification using $\mathcal{L}_{\text{SCHOOL}}$ and $\mathcal{L}_{\text{OFFICE}}$. Left: ROC curve representing a perfect classifier. Right: PR curve representing a perfect classifier.

6.4 Dataset validation

The use of simulation in robotics has become more and more relevant in the last years, especially as a way to preliminarily test systems that will be later assessed in real-world settings [4]. If simulation tools have become more and more physically realistic [13], the design of simulated environments is usually receiving less attention, with the consequence that not all simulated environments are realistic (i.e., are similar to their real counterparts) from a structural point of view.

For example, Figure 6.10 shows two simulated offices. The first one (Figure 6.10a) is based on a real-world environment while the second one (Figure 6.10b) is randomly generated without considering the common topology of an office. Starting from the top-left corner (A), the paths to reach two close locations (B and C) in the realistic office are very similar, while in the other office they are completely different. This situation can hinder the transfer of results from simulation to the real world.

With the following experiment we apply our approach to test if a simulated environment presents the same structural characteristics of a real-world environment. In the context of our approach, this can be done by evaluating if the topological and geometrical properties of spaces of a floor of a target building are coherent to those of real-world floors of buildings of the same type. In this way, we can assess if a simulated environment is structurally realistic.

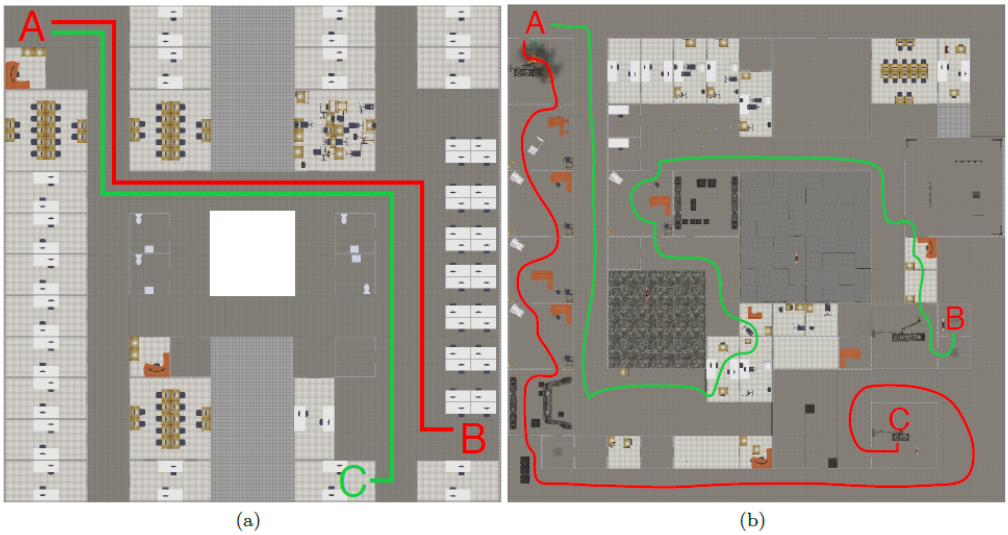


Figure 6.10: A simulated office building generated using real-world data as inspiration (a) and a simulated office building generated without using real-world data as inspiration (b), from [4].

In kLog, dataset validation can be represented as a binary classification of interpretations, in which each interpretation is categorized as a positive instance of OFFICE or as negative example of an unrealistic simulation. In kLog we added the following extensional signatures:

validation(v::property)::extensional.

and in each interpretation one of following mutual exclusive ground atoms:

validation(simulated)
validation(real)

Finally, it is needed to specify the scope of the learning task:

kfold(validation,PARAM)

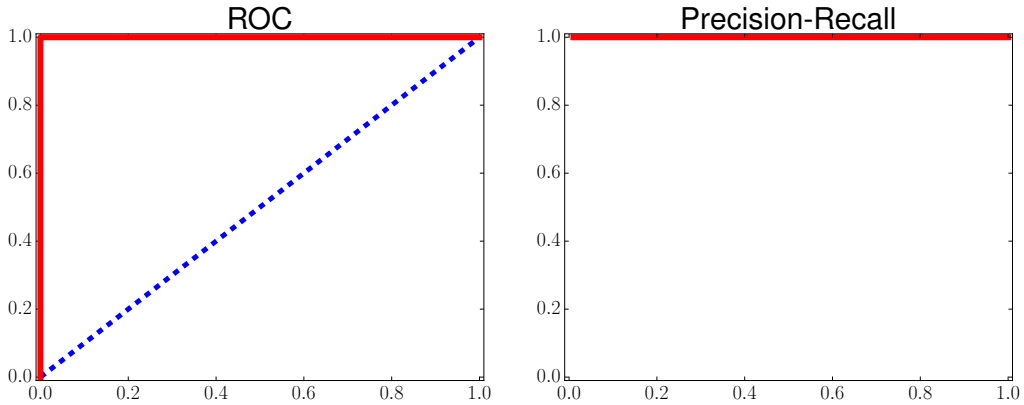


Figure 6.11: Results of dataset validation of RoboCub simulated offices using the labeling schema $\mathcal{L}_{R/C}$. Left: ROC curve representing a perfect classifier. Right: PR curve representing a perfect classifier.

6.4.1 Data and results

We built a dataset of all the office environments used during the finals of the 2012 and 2013 editions of the Virtual Robot Competition of the RoboCup Rescue Simulation League [2], which represent 6 simulated environments designed to test the behavior of robots in an Urban Search and Rescue task. Figure 6.11, Figure 6.12, and Figure 6.13 show that using our approach we are always able to distinguish a real-world office building from an office building simulated in the competition, no matter which label set is used ($\mathcal{L}_{R/C}$, $\mathcal{L}_{R/C/S/E}$, $\mathcal{L}_{\text{OFFICE}}$).

6.4.2 Analysis of the results

According to [4], simulated experimental settings provide valuable results about the general performance of a robot system if the settings are enough similar to the real settings where the system can possibly operate.

In many virtual settings, the physical realism of the environment (e.g., the precise reproduction of the physical laws) is carefully taken into account while the structural plausibility (e.g., how much the simulated environment resembles to a real environment) is not considered.

Our results suggest that the structural realism of environments used in the Virtual Robot Competition could be improved in order to make the transfer of the results from simulation to the real-world easier. More generally,

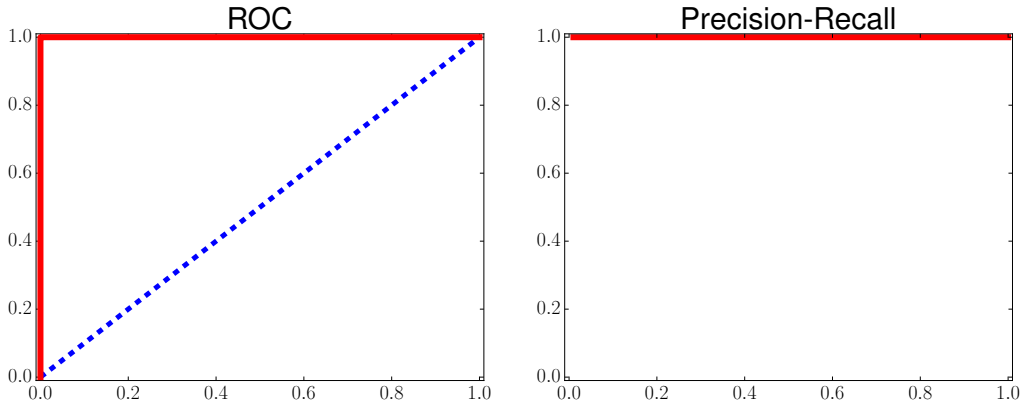


Figure 6.12: Results of dataset validation of RoboCub simulated offices using the labeling schema $\mathcal{L}_{R/C/E/S}$. Left: ROC curve representing a perfect classifier. Right: PR curve representing a perfect classifier.

our system can be used effectively in different situations to distinguish realistic and unrealistic simulated environments, thus providing a method for validating their suitability as an experimental testbeds.

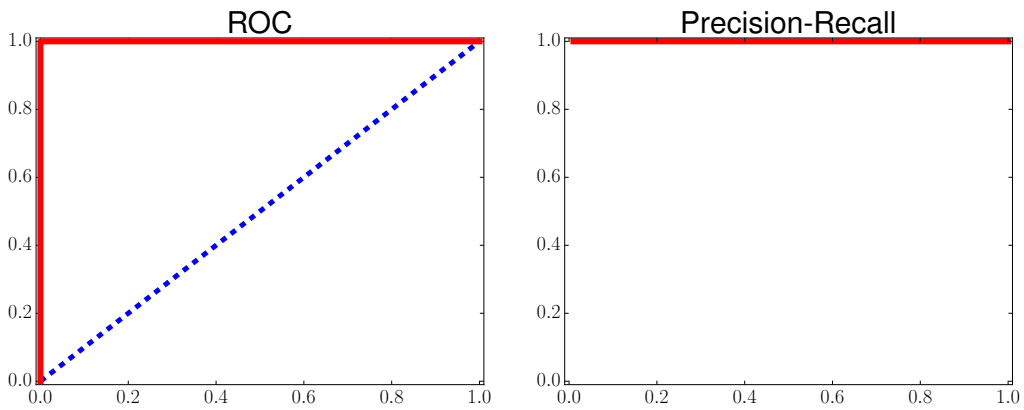


Figure 6.13: Results of dataset validation of RoboCub simulated offices using the labeling schema $\mathcal{L}_{\text{OFFICE}}$. Left: ROC curve representing a perfect classifier. Right: PR curve representing a perfect classifier.

Chapter 7

Conclusions and future work

I have loved you for a thousand years
I have loved you with an everlasting love
I have loved to the point of madness
I have low self esteem

Google Autocompletion

In this thesis, a logical and relational learning framework, kLog, has been applied to the field of semantic mapping, in the context of mobile autonomous robotics.

We started from analyzing the information included in a semantic map, identifying, thanks to city planning sources, five fundamental elements: nodes, paths, edges, landmarks, and districts. We adapted these fundamental elements to our context, floor plans of buildings, obtaining two relational models. We created two hierarchical labeling schemas, for schools and offices, where each label represents the functional role of a room in the building, at several levels of abstraction.

We translated the relational models in logical models, more specifically, in the dialect used by kLog, the framework employed in our work. We used a learning from interpretations statistical relational learning approach where each interpretation is represented by a set of logical ground atoms describing an entire floor plan of a building. Each interpretation is graphicalized (i.e., translated into a graph-like structure) in order to solve learning tasks using a graph kernel method over the whole structure. The novelty we introduce with our approach is in fact to consider the global structure of the floor

plans in order to perform learning tasks related to semantic mapping, unlike classical approaches, which consider rooms individually.

We also developed two software tools which permit to obtain a logical representation of a floor plan of a building starting from an image representing the floor plan. The first one, Floor Plans Editor (FPE), allows to build the metrical map of a floor plan and to automatically obtain the topological map, saving both as XML files. The second one, RoomsPy, allows to parse several input file formats (including the above mentioned XML), in order to extract geometrical features and to export the results in several output formats (including kLog logical ground atoms).

We evaluated our approach with three experiments, two of which are not possible using a local approach:

Place classification: this experiment reproduced the classical semantic mapping task of assigning a label to each room of a floor plan. We obtained excellent results in determining the structural role of a room in a floor plan (i.e., distinguishing between rooms and corridors), while more specific functional classifications obtained worse results. This led us to the conclusion that additional information about spaces (e.g., their appearance obtained from a camera) should be considered in order to precisely classify their function in the building.

Building classification: this experiment aimed at classifying an entire floor plan of a building with the correct building type, which is not possible using a local approach. We obtained good results in classifying among two building types (i.e., schools and offices) with the less informative labeling schemas and excellent results with the most informative ones.

Dataset validation: this experiment aimed at identifying which environments were real and which simulated, and is not possible using a local approach. We obtained excellent results even with the least informative labeling schema, suggesting that virtual environments used in some robot simulations should be improved (from a structural point of view) in order to make the transfer of the results from simulation to the real-world easier.

We present now some possible future developments of our work:

Enable the employement of partial information: our framework requires complete information about floor plans in order to work. A possible enhancement would be to design a framework able to take as input also a portion of a floor plan, for example the portion of environment already mapped by a robot during exploration. This would allow to implement several tasks as the assignment of temporary labels to the explored portion of the environment and the prediction over the unexplored portion both in terms of structure and of expected room types. This would require a radical change in the framework since it is needed to switch from a learning from interpretations setting to a learning from entailment setting, where closed world assumption does not apply.

Extend the framework to cope with other data sources: we demonstrated how structural information about the environment, even when combined with geometrical features, is not enough informative to permit a precise classification of the function of a room. A possible future work could try to overcome this limitation by using as input data more sources of information like, for example, camera images.

Extend the analysis to more building types: we experimented our approach using two building types: schools and offices. It would be interesting to analyze the behavior of the system in different environments, both public (e.g., malls, theatres) and domestic (e.g., houses, flats).

Change the size of the analyzed environment: we reasoned on structures representing floor plans of buildings, it would be interesting to apply the same analysis to graphs representing smaller and larger environments. A possibility is to represent with a graph the space of a single room. Nodes could represent objects and landmarks of the environment and arcs could represent spatial distances between elements. A possible learning task could be to classify the functions that each part of the space has (e.g., in a kitchen we can identify a cooking area, identified by the burners, and a dining area, identified by the table). Another possibility is to represent in the graph an urban context improving navigation capabilities of a robot in an unknown city reasoning on the graphs representing already explored cities.

Bibliography

[1] Processing. `https://processing.org/`. Accessed: 2015-15-11.

[2] Robocup rescue league. `http://www.robocuprescue.org/`. Accessed: 2015-27-11.

[3] Philipp Althaus and Henrik Christensen. Behavior coordination in structured environments. *Advanced Robotics*, 17(7):657–674, 2003.

[4] Francesco Amigoni, Matteo Luperto, and Viola Schiaffonati. Towards generalization of experimental results for autonomous robots. Submitted to Robotics and Autonomous Systems, 2015.

[5] Laura Antanas, Paolo Frasconi, Fabrizio Costa, Tinne Tuytelaars, and Luc De Raedt. A relational kernel-based framework for hierarchical image understanding. In *Structural, Syntactic, and Statistical Pattern Recognition*, pages 171–180. Springer, 2012.

[6] Laura Antanas, Plinio Moreno, and Luc De Raedt. Relational kernel-based grasping with numerical features. In *Proceedings of International Conference on Inductive Logic Programming*, 2015.

[7] Alper Aydemir, Patric Jensfelt, and John Folkesson. What can we learn from 38,000 rooms? reasoning about unexplored space in indoor environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4675–4682, 2012.

[8] Pierre Baldi, Paolo Frasconi, and Padhraic Smyth. *Modeling the Internet and the Web*. Wiley, 2003.

[9] Patrick Beeson, Nicholas K Jong, and Benjamin Kuipers. Towards autonomous topological place detection using the extended voronoi graph.

In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4373–4379, 2005.

[10] Karsten M Borgwardt and Hans-Peter Kriegel. Shortest-path kernels on graphs. In *IEEE International Conference on Data Mining*, 2005.

[11] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

[12] Emma Brunskill, Thomas Kollar, and Nicholas Roy. Topological mapping using spectral clustering and classification. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3491–3496, 2007.

[13] Stefano Carpin, Mike Lewis, Jijun Wang, Stephen Balakirsky, and Chris Scrapper. Usarsim: a robot simulator for research and education. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1400–1405, 2007.

[14] Soumen Chakrabarti. *Mining the Web: Discovering knowledge from hypertext data*. Elsevier, 2002.

[15] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):1–27, 2011.

[16] Peter Pin-Shan Chen. The entity-relationship model—toward a unified view of data. *ACM Transactions on Database Systems (TODS)*, 1, 1976.

[17] Howie Choset and Keiji Nagatani. Topological simultaneous localization and mapping (slam): toward exact localization without explicit localization. *IEEE Transactions on Robotics and Automation*, 17(2):125–137, 2001.

[18] Michael Collins and Nigel Duffy. Convolution kernels for natural language. In *Advances in neural information processing systems*, pages 625–632, 2001.

[19] Fabiano Corrêa, Rodrigo Belizia Polastro, Fabio Gagliardi Cozman, and Jun Okamoto Jr. Dealing with semantic knowledge in robotics with a probabilistic description logic. In *Proceedings of the Argentine Symposium on Artificial Intelligence*, 2010.

[20] Fabrizio Costa and Kurt De Grave. Fast neighborhood subgraph pairwise distance kernel. In *Proceedings of the International Conference on Machine Learning*, pages 255–262, 2010.

[21] Mark Craven and Seán Slattery. Relational learning with statistical predicate invention: Better models for hypertext. *Machine Learning*, 43(1-2):97–119, 2001.

[22] Chad Cumby and Dan Roth. On kernel methods for relational learning. In *Proceeding of IMLS International Conference on Machine Learning (ICML)*, pages 107–114, 2003.

[23] James Cussens and Saso Dzeroski. *Learning language in logic*. Number 1925. Springer, 2000.

[24] Luc De Raedt. *Logical and relational learning*. Springer, 2008.

[25] Leone D'Emilio. Un modello generativo spettrale basato su grafi per il mapping semantico di edifici. Master's thesis, Politecnico di Milano, 2014.

[26] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. Kernel k-means: spectral clustering and normalized cuts. In *Proceedings of the international conference on Knowledge discovery and data mining*, pages 551–556. ACM, 2004.

[27] Elisabetta Fabrizi and Alessandro Saffiotti. Extracting topology-based maps from gridmaps. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2972–2978, 2000.

[28] Paolo Frasconi, Fabrizio Costa, Luc De Raedt, and Kurt De Grave. klog: A language for logical and relational learning with kernels. *Artificial Intelligence*, 217:117–143, 2014.

[29] Nir Friedman, Lise Getoor, Daphne Koller, and Avi Pfeffer. Learning probabilistic relational models. In *Proceedings of International Join Conference on Artificial Intelligence (IJCAI)*, pages 1300–1309, 1999.

[30] Cipriano Galindo, Juan-Antonio Fernández-Madrigal, Javier González, and Alessandro Saffiotti. Robot task planning using semantic maps. *Robotics and Autonomous Systems*, 56(11):955–966, 2008.

[31] Cipriano Galindo, Alessandro Saffiotti, Silvia Coradeschi, Pär Buschka, J Fernández-Madrigal, Javier Gonzalez, et al. Multi-hierarchical semantic maps for mobile robotics. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2278–2283, 2005.

[32] Thomas Gärtner. A survey of kernels for structured data. *ACM SIGKDD Explorations Newsletter*, 5(1):49–58, 2003.

[33] Thomas Gärtner, Peter Flach, and Stefan Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Learning Theory and Kernel Machines*, pages 129–143. Springer, 2003.

[34] Peter V Gehler, Bernhard Schölkopf, Camps-Valls L Bruzzone, et al. *An introduction to Kernel learning algorithms*. Wiley, 2009.

[35] Lise Getoor. *Introduction to statistical relational learning*. MIT press, 2007.

[36] Edwin R Hancock and Richard C Wilson. Pattern analysis with graphs: Parallel work at bern and york. *Pattern Recognition Letters*, 33(7):833–841, 2012.

[37] Christoph Helma. *Predictive toxicology*. CRC Press, 2005.

[38] Sachithra Hemachandra, Matthew R Walter, Stefanie Tellex, and Seth Teller. Learning spatial-semantic representations from natural language descriptions and scene classifications. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2623–2630, 2014.

[39] Stephen C Hirtle and John Jonides. Evidence of hierarchies in cognitive maps. *Memory & cognition*, 13(3):208–217, 1985.

[40] Tommi Jaakkola and David Haussler. Probabilistic kernel regression models. In *Proceedings of the International Conference on AI and Statistics*, volume 126, pages 00–04, 1999.

[41] Tommi Jaakkola, David Haussler, et al. *Exploiting generative models in discriminative classifiers.* MIT Pres, 1999.

[42] Jeong-Hun Jang and Ki-Sang Hong. Fast line segment grouping method for finding globally more favorable line segments. *Pattern Recognition*, 35(10):2235–2247, 2002.

[43] Thorsten Joachims. *Learning to classify text using support vector machines: Methods, theory and algorithms.* Kluwer Academic Publishers, 2002.

[44] Hisashi Kashima and Teruo Koyanagi. Kernels for semi-structured data. In *Proceeding of IMLS International Conference on Machine Learning (ICML)*, volume 2, pages 291–298, 2002.

[45] Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. Marginalized kernels between labeled graphs. In *Proceeding of IMLS International Conference on Machine Learning (ICML)*, volume 3, pages 321–328, 2003.

[46] Roni Khardon, Dan Roth, and Rocco A Servedio. Efficiency versus convergence of boolean kernels for on-line learning algorithms. In *Proceedings of Advances in Neural Information Processing Systems Conference*, pages 423–430, 2001.

[47] Risi Imre Kondor and John Lafferty. Diffusion kernels on graphs and other discrete structures. In *Proceedings of the 19th international conference on machine learning*, pages 315–322, 2002.

[48] Jana Košecká, Fayin Li, and Xialong Yang. Global localization and relative positioning based on scale-invariant keypoints. *Robotics and Autonomous Systems*, 52(1):27–38, 2005.

[49] Jana Košecká, Liang Zhou, Philip Barber, and Zoran Duric. Qualitative image based localization in indoors environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, 2003.

[50] Benjamin Kuipers and Yung-Tai Byun. A robust, qualitative method for robot spatial learning. In *Proceedig of the Association for the Advancement of Artificial Intelligence Conference*, pages 774–779, 1988.

[51] Rolf Lakaemper. Simultaneous multi-line-segment merging for robot mapping using mean shift clustering. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1654–1660, 2009.

[52] Jean-Claude Latombe. *Robot motion planning.* Springer, 2012.

[53] Ziyuan Liu and Georg von Wichert. A generalizable knowledge framework for semantic indoor mapping based on markov logic networks and data driven mcmc. *Future Generation Computer Systems*, 36:42–56, 2014.

[54] Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. *The Journal of Machine Learning Research*, 2:419–444, 2002.

[55] Qing Lu and Lise Getoor. Link-based classification. In *Proceeding of IMLS International Conference on Machine Learning (ICML)*, volume 3, pages 496–503, 2003.

[56] Kevin Lynch. *The image of the city.* MIT press, 1960.

[57] Sauro Menchetti, Fabrizio Costa, and Paolo Frasconi. Weighted decomposition kernels. In *Proceedings of the international conference on Machine learning*, pages 585–592. ACM, 2005.

[58] Joseph Modayil, Patrick Beeson, and Benjamin Kuipers. Using the topological skeleton for scalable global metrical map-building. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 2, pages 1530–1536, 2004.

[59] Michael Montemerlo and Sebastian Thrun. Fastslam 2.0. *FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics*, pages 63–90, 2007.

[60] Michael Montemerlo, Sebastian Thrun, Daphne Koller, Ben Wegbreit, et al. Fastslam: A factored solution to the simultaneous localization and mapping problem. In *Proceedig of the Association for the Advancement of Artificial Intelligence Conference*, pages 593–598, 2002.

[61] Raymond J Mooney and Loriene Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries*, pages 195–204, 2000.

[62] Hans Moravec. Sensor fusion in certainty grids for mobile robots. *AI magazine*, 9(2), 1988.

[63] Óscar Martínez Mozos. *Semantic labeling of places with mobile robots*. Springer, 2010.

[64] Oscar Martinez Mozos, Cyrill Stachniss, and Wolfram Burgard. Supervised learning of places from range data using adaboost. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1730–1735. IEEE, 2005.

[65] Oscar Martinez Mozos, Rudolph Triebel, Patric Jensfelt, Axel Rottmann, and Wolfram Burgard. Supervised semantic labeling of places using information extracted from sensor data. *Robotics and Autonomous Systems*, 55(5):391–402, 2007.

[66] Claudio Mura, Oliver Mattausch, Alberto Jaspe Villanueva, Enrico Gobbetti, and Renato Pajarola. Automatic room detection and reconstruction in cluttered indoor environments with complex room layouts. *Computers & Graphics*, 44:20–32, 2014.

[67] Michel Neuhaus and Horst Bunke. Edit distance-based kernel functions for structural pattern classification. *Pattern Recognition*, 39(10):1852–1863, 2006.

[68] Paul Newman, John Leonard, JD Tardó, and José Neira. Explore and return: Experimental validation of real-time concurrent mapping and localization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1802–1809, 2002.

[69] Bradford Perkins. *Building type basics for elementary and secondary schools*. Wiley, 2010.

[70] Andrzej Pronobis. *Semantic mapping with mobile robots*. KTH Royal Institute of Technology, 2011.

[71] Andrzej Pronobis and Patric Jensfelt. Large-scale semantic mapping and reasoning with heterogeneous modalities. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3515–3522. IEEE, 2012.

[72] Andrzej Pronobis, Patric Jensfelt, Kristoffer Sjöö, Hendrik Zender, Geert-Jan M Kruijff, Oscar Martinez Mozos, and Wolfram Burgard. Semantic modelling of space. In *Cognitive Systems*, pages 165–221. Springer, 2010.

[73] Andrzej Pronobis, O Martinez Mozos, Barbara Caputo, and Patric Jensfelt. Multi-modal semantic place classification. *The International Journal of Robotics Research*, 2009.

[74] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine learning*, 62(1-2):107–136, 2006.

[75] Aldo Rossi. *L'architettura della città*. Marsilio, Padova, 1966.

[76] Axel Rottmann, Óscar Martínez Mozos, Cyrill Stachniss, and Wolfram Burgard. Semantic place classification of indoor environments with mobile robots using boosting. In *Proceedig of the Association for the Advancement of Artificial Intelligence Conference*, pages 1306–1311, 2005.

[77] Alberto Sanfeliu and King-Sun Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man and Cybernetics*, (3):353–362, 1983.

[78] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *Artificial Neural Networks*, pages 583–588. Springer, 1997.

[79] Reid Simmons, David Apfelbaum, Wolfram Burgard, Dieter Fox, Mark Moors, Sebastian Thrun, and Håkan Younes. Coordination for multi-robot exploration and mapping. In *Proceedig of the Association for the Advancement of Artificial Intelligence Conference*, pages 852–858, 2000.

[80] Lauro Snidaro and Gian Luca Foresti. Knowledge representation for ambient security. *Expert Systems*, 24(5):321–333, 2007.

[81] Cyrill Stachniss, Oscar Martinez Mozos, and Wolfram Burgard. Speeding-up multi-robot exploration by considering semantic place information. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1692–1697. IEEE, 2006.

[82] Michael JE Sternberg, Ross D King, Ashwin Srinivasan, and Stephen H Muggleton. Drug design by machine learning. In *Machine Intelligence 15, Intelligent Agents [St. Catherine's College, Oxford, July 1995]*, pages 328–338. Oxford University, 1999.

[83] Charles Sutton and Andrew McCallum. *An introduction to conditional random fields for relational learning*. MIT press, 2006.

[84] Sebastian Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, (1), 1998.

[85] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT Press, 2005.

[86] Koji Tsuda, Michihiko Minoh, and Katsuo Ikeda. Extracting straight lines by sequential fuzzy clustering. *Pattern Recognition Letters*, 17(6):643–649, 1996.

[87] Jan Van Haaren and Guy Van den Broeck. Relational learning for football-related predictions. *Preliminary Papers ILP*, pages 1–6, 2011.

[88] Mathias Verbeke, Paolo Frasconi, Kurt De Grave, Fabrizio Costa, and Luc De Raedt. klognlp: Graph kernel–based relational learning of natural language. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 85–90, 2014.

[89] Mathias Verbeke, Vincent Van Asch, Roser Morante, Paolo Frasconi, Walter Daelemans, and Luc De Raedt. A statistical relational learning approach to identifying evidence based medicine categories. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 579–589, 2012.

[90] SVN Vishwanathan, Nicol N Schraudolph, Mark W Schmidt, and Kevin P Murphy. Accelerated training of conditional random fields

with stochastic gradient methods. In *Proceedings of the international conference on Machine learning*, pages 969–976. ACM, 2006.

[91] SVN Vishwanathan and Alexander Johannes Smola. Fast kernels for string and tree matching. *Kernel methods in computational biology*, pages 113–130, 2004.

[92] Chris Watkins. Kernels from matching operations. *Computer Science Dept. Technical Report*, 98(07), 1999.

[93] Emily Whiting, Jonathan Battat, and Seth Teller. Topology of urban environments. In *Computer-Aided Architectural Design Futures*. Springer, 2007.

[94] Denis F Wolf and Gaurav S Sukhatme. Semantic mapping using mobile robots. *IEEE Transactions on Robotics*, 24(2):245–258, 2008.

[95] Brian J Worton. Kernel methods for estimating the utilization distribution in home-range studies. *Ecology*, 70(1):164–168, 1989.

[96] John M Zelle and Raymond J Mooney. Learning to parse database queries using inductive logic programming. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1050–1055, 1996.

[97] Hendrik Zender, O Martínez Mozos, Patric Jensfelt, G-JM Kruijff, and Wolfram Burgard. Conceptual spatial representations for indoor mobile robots. *Robotics and Autonomous Systems*, 56(6):493–502, 2008.