UAB
Universitat Autònoma
de Barcelona

Relational Models for Visual Understanding of Graphical Documents. Application to Architectural Drawings.

A dissertation submitted by **Lluís-Pere de las Heras** at Universitat Autònoma de Barcelona to fulfil the degree of **Doctor of Philosophy**.

Bellaterra, September 29, 2014

This document was typeset by the author using $\LaTeX\,2_\varepsilon$.

The research described in this book was carried out at the Centre de Visió per Computador, Universitat Autònoma de Barcelona.

ISBN: 978-84-940902-4-0

Printed by Ediciones Gráficas Rey, S.L.

A la Sandra, l Alba i mons pares...

De vegades,
improvitzar és la millor manera
de tenir-ho tot sota control.
Lluís-Pere de las Heras

Agraıments

Magradaria agrair a certa gent i institucions que han fet possible la realització daquesta tesi. Sense ells no hagués estat possible.

Primer de tot magradaria agrair a la Universitat Autònoma de Barcelona per donar-me loportunitat, ja no només de doctorar-me, sinó també de llicenciar-me com a enginyer i màster en informàtica. Onze anys fa ja que em vaig incorporar a la institució i per tant, es podria dir que sóc de la cantera. A més, voldria també agrair al Centre de Visió per Computador per acollir-me aquests últims 5 anys i on he realitzat el Màster i aquesta tesi doctoral.

Magradaria agrair a la meva directora de tesi, la Dra. Gemma Sánchez, per donar-me loportunitat de desenvolupar aquesta tesi doctoral, per la seva orientació i consell durant tots aquests anys.

Magradaria agrair especialment a tres persones que també han estat clau pel desenvolupament daquesta tesi. Voldria primer de tot agrair al Dr. Oriol Ramos per la seva col·laboració, paciència, pedagogia i amistat. Gracies Oriol, de veritat. Voldria també donar les gràcies al Dr. Ernest Valveny per orientar-me en els sempre complexos primers passos de doctorat. Finalment expressar enèrgicament la meva gratitud al Dr. Josep Lladós per tot el que m ha ajudat a arribar fins aquí. Gràcies Josep.

I would like to thank Prof. Dr. Prof. h.c. Andreas Dengel and Dr. Marcus Liwicki who gave the opportunity of spending three profitable months of my research at the German Research Center for Artificial Intelligence in Kaiserslautern. There I met great professionals and even better people. Thanks to Dr. Stephan Baunmann, Dr. Ludger van Elst, Heinz Kirchmann, Klaus Broelemann, and Ahmed Sheraz.

Durant la persecució daquesta tesi doctoral he format part dun grup de gent genial pel que fa tant essers humans com a investigadors. Vull agrair al Dr. Dimosthenis Karatzas, Dr. Mara̧l Rossinyol, Dra. Alicia Fornés, Dr. Albert Gordo, Lluís Gómez, Núria Cirera, Hongxing Gao, Christophe Rigaud i Antonio Clavelli per les grans reunions de grup que tant mhan il·lustrat en lanàlisi de documents.

Evidentment vull agrair a tots els meus amics sense els quals la vida seria tan avorrida que seria impossible realitzar una tesi doctoral. Gràcies Riu, Gómez, Llergo, Mireia, Marta, Ade, Roger, Jon, David, Anjan, Toni, Carles, Fran, Rubén, Ekain, Camp, Ivet, Alejandro, Joan, Yainuvis i Jorge. Merci companys, de veritat.

I would like to thank somebody very special to me. Thank you so much, Martin. I spent part of my best lifetime in Kaiserslautern and it was thanks to you. I am sure that even better times will come together. Thanks also to Christian, Anika, Christine, and Sylvia for the great time we have spent together.

También vull agrair moltíssim a la meva família. Sé que sona a tòpic però tinc una família que no me la mereixo. I quan parlo de família parlo de la *Cuadri*, evidentment. Moltes gràcies a tots per ser com sou.

Gràcies a aquelles persones més pròximes i que més estimo. Vull agrair de tot el cor a la iaia, l avi, la Rosa Marta, la Silvia, la Fany, el Sergi, el Fran, la Carla, la Yolanda i l Emilio. Grácies.

Que puc dir de vosaltres, Albeta, mare i pare. Ja sabeu tot el que us arribo a estimar, com us admiro i us respecto. Estic tant agraıt per tot el que heu fet per mi, TANT, que sóc sincer quan us dic que em salten les llàgrimes quan escric, rectifico i torno a escriure aquesta línia mirant de trobar les paraules correctes. Moltíssimes gràcies.

Sandreta, GRÀCIES en majúscules per ser com ets i per fer-me a mi com sóc. Pel teu amor, paciència, i sacrifici en els moments més difícils daquesta tesi. Gràcies per fer tan durs els matins i tan boniques les tardes i les nits de cada dia laborable. Gràcies pels caps de setmana, pels viatges junts, per les aventures. Gràcies per tot amore, testimo més que res en aquest món. 87 30

Finalment, voldria agrair-te especialment a tu tieta, ja no només per tot allò que has fet per mi sempre (que n estic segur que és més del que ha fet qualsevol altra tieta pel seu nebot), sinó per mostrar-me lesperit de lluita necessari per tal de sortir-sen en situacions difícils. Ets un exemple per tots nosaltres. Ens en sortirem, nestic segur. Testimo molt.

Abstract

Graphical documents express complex concepts using a visual language. This language consists of a vocabulary (symbols) and a syntax (structural relations among symbols) that articulate a semantic meaning in a certain context. Therefore, the automatic interpretation of these sort of documents by computers entails three main steps: the detection of the symbols, the extraction of the structural relations among these symbols, and the modeling of the knowledge that permits the extraction of the semantics. Different domains in graphical documents include: architectural and engineering drawings, maps, flowcharts, etc.

Graphics Recognition in particular and Document Image Analysis in general are born from the industrial need of interpreting a massive amount of digitalized documents after the emergence of the scanner. Although many years have passed, the graphical document understanding problem still seems to be far from being solved. The main reason is that the vast majority of the systems in the literature focus on a very specific problems, where the domain of the document dictates the implementation of the interpretation. As a result, it is difficult to reuse these strategies on different data and on different contexts, hindering thus the natural progress in the field.

In this thesis, we face the graphical document understanding problem by proposing several relational models at different levels that are designed from a generic perspective. Firstly, we introduce three different strategies for the detection of symbols. The first method tackles the problem structurally, wherein general knowledge of the domain guides the detection. The second is a statistical method that learns the graphical appearance of the symbols and easily adapts to the big variability of the problem. The third method is a combination of the previous two inheriting their respective strengths, i.e. copes the big variability and does not need of annotated data. Secondly, we present two relational strategies that tackle the problem of the visual context extraction. The first one is a full bottom up method that heuristically searches in a graph representation the contextual relations among symbols. Contrarily, the second is syntactic method that models probabilistically the structure of the documents. It automatically learns the model, which guides the inference algorithm to counter the best structural representation for a given input. Finally, we construct a knowledge-based model consisting of an ontological definition of the domain and real data. This model permits to perform contextual reasoning and to detect semantic inconsistencies within the data. We evaluate the suitability of the proposed contributions in the framework of floor plan interpretation. Since there is no standard in the modeling of these documents, there exists an enormous notation variability and the sort of information included in the documents also varies from plan to plan. Therefore, floor plan understanding is a relevant task in the graphical document understanding problem. It is also worth to mention that, we make freely available all the resources used in this thesis (the data, the tool used to generate the data, and the evaluation scripts) aiming at fostering the research in graphical document understanding task.

Resum

Els documents gráfics són documents que expressen continguts semántics utilitzant majoritáriament un llenguatge visual. Aquest llenguatge está format per un vocabulari (símbols) i una sintaxi (relacions estructurals entre els símbols) que conjuntament manifesten certs conceptes en un context determinat. Per tant, la interpretació dun document gráfic per part dun ordinador implica tres fases. (1) Ha de ser capaçde detectar automáticament els símbols del document. (2) Ha de ser capaçdextreure les relacions estructurals entre aquests símbols. I (3), ha de tenir un model del domini per tal poder extreure la semántica. Exemples de documents gráfics de diferents dominis són els planells darquitectural i d enginyeria, mapes, diagrames de flux, etc.

El Reconeixement de Gráfics, dintre de lárea de recerca de Análisi de Documents, neix de la necessitat de la indústria dinterpretar la gran quantitat de documents gráfics digitalitzats a partir de laparició de lescáner. Tot i que molts anys han passat daquests inicis, el problema de la interpretació automática de documents sembla encara estar lluny de ser solucionat. Básicament, aquest procés sha alentit per una raó principal: la majoria dels sistemes dinterpretació que han estat presentats per la comunitat són molt centrats en una problemática específica, en el que el domini del document marca clarament la implementació del mètode. Per tant, aquests mètodes són difícils de ser reutilitzats en daltres dades i marcs daplicació, estancant així la seva adopció i evolució en favor del progrés.

En aquesta tesi afrontem el problema de la interpretació automática de documents gráfics a partir dun seguit de models relacionals que treballen a tots els nivells del problema, i que han estat dissenyats des dun punt de vista genèric per tal de que puguin ser adaptats a diferents dominis. Per una part, presentem 3 mètodes diferents per a lextracció dels símbols en un document. El primer tracta el problema des dun punt de vista estructural, en el que el coneixement general de lestructura dels símbols permet trobar-los independentment de la seva aparença. El segon és un mètode estadístic que aprèn laparença dels símbols automáticament i que, per tant, sadapta a la gran variabilitat del problema. Finalment, el tercer mètode és una combinació dambdós, heretant els beneficis de cadascun dels mètodes. Aquesta tercera implementació no necessita de un aprenentatge previ i a més sadapta fácilment a múltiples notacions gráfiques. D altra banda, presentem dos mètodes per a la extracció del context visuals. El primer mètode segueix una estratègia *bottom-up* que cerca les relacions estructurals en una representació de graf mitjançant algorismes dintel·ligència artificial. La segona en canvi, és un mètode basat en una gramática que mitjançant un model probabilístic aprèn automáticament lestructura dels planells. Aquest model guia la interpretació del document amb certa independència de la implementació algorísmica. Finalment, hem definit una base del coneixement fent confluir una definició ontològica del domini amb dades reals. Aquest model ens permet raonar les dades des dun punt de vista contextual i trobar inconsistències semántiques entre les dades. Leficiència daquetes contribucions han estat provades en la interpretació de planells darquitectura. Aquest documents no tenen un estándard establert i la seva notació gráfica i inclusió dinformació varia de planell a planell. Per tant, és un marc rellevant del problema de reconeixement gráfic. A més, per tal de promoure la recerca en termes de interpretació de documents gráfics, fem públics tant les dades, leina per generar les dades i els evaluadors del rendiment.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Nowadays, despite the dramatic increase in the creation and processing of digital data, paper-based storage capacity is still growing [56]. This has currently fostered the research in Document Image Analysis (DIA) systems, which aim to automatically store, extract, and process the information contained in documents. DIA is at the intersection of the research fields of image processing, pattern recognition, artificial intelligence, linguistics, and storage systems [74], and encloses several research lines, such as type and hand-written text recognition, document categorization and understanding, and graphics recognition.

Graphics Recognition is the sub-domain of DIA that analyzes the content of *graphical documents*. A graphical document is either a physical or a virtual matter created to convey and communicate a piece of information that, part or all, is manifested in a pictorial or diagrammatic form, agreeing to a graphical language. Examples of graphical documents are maps, engineering drawings, flowcharts, diagrams, etc.

Graphical languages are human-made systems of communication that allow to express complex semantical concepts using graphical information. Making the analogy to natural languages, lexical data (e.g. words) is arranged in a specific order according to syntactical rules defined by a language to give a complete meaning to a sentence. In graphical languages, one can think that the vocabulary consists a set of symbols, and the grammatical rules are contextual relations among these symbols that give the complete meaning to a graphic. Thus, graphical document *understanding* entails the parsing by a reading system that recognizes both, the isolated symbols in the document and their contextual relations. Yet, graphical languages, as natural languages do, vary depending on several aspects, e.g. the nature of *reader* (infants or adults), the sort of information enclosed (technical, complex, informal), creativity and design matters, etc. For instance, we show in Figure 1.1 four graphical documents that represent the same concept using different visual languages. Therefore, in the same manner as humans do, computers need to *speak* the different variations of graphical languages in order to understand the information conveyed in these documents.

This work tackles the problem of graphical document interpretation. It presents different proposals to automatically learn, extract, and analyze the contextual information among the graphical symbols in a document, with the final objective of understanding its content. We describe the issues that these tasks entail, how they have been dealt in the literature, and propose a generic approximation oriented to solve a real problem: floor plan interpretation.

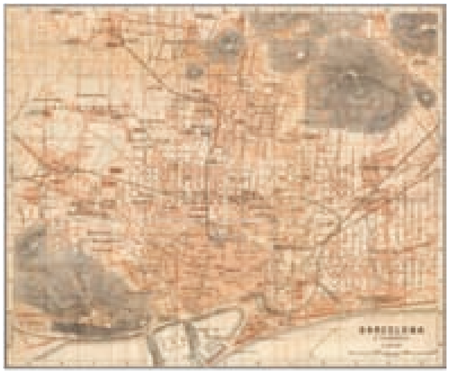
1.1.1 Graphics recognition

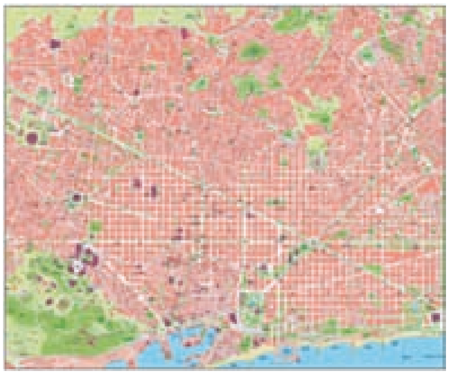
Graphics Recognition (GR) is the domain in Pattern Recognition and Document Image Analysis that analyses the documents that are rich in graphical information. The GR community has its own technical committee [11] in the International Association for Pattern Recognition [6] and organizes the International Workshop on Graphics Recognition [1] every 2 years. This workshop leads the community to present the latest advances in the field and to put in common the future challenges. Some of these challenges include symbol recognition and spotting, on-line and off-line recognition, low-level image processing, text/graphics separation, image retrieval, document understanding, and performance evaluation.

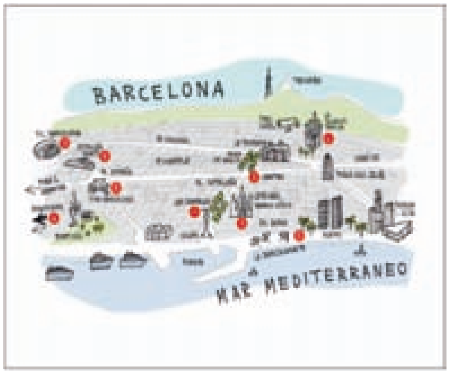
In this dissertation, we focus on document understanding, which embraces an important part of the GR tasks. It aims at overcoming the semantic gap between the visual appearance of the images and the semantic meaning they convey. In other words, this whole process of understanding can be seen as the *translation* of the instances generated by the visual language into a well-structured representation that expresses in a suitable format for computers the whole meaning of a document. This process consists of three main subtasks explained in the following: (i) symbol recognition, (ii) context extraction, and (iii) semantic understanding. We graphically show in Figure 1.2 an example on flowchart recognition to illustrate this process.

- Graphical symbols conform the vocabulary of the visual language of a document. In Figure 1.2, the vocabulary consists of by boxes of various kinds, arrows, and text. Historically, symbol recognition has been one of the main challenges for the GR community. In its first stages, isolated symbols have been detected either by applying several image transformations, such as mathematical morphology, or by analyzing and grouping graphical primitives obtained after vectorization. Despite the good results achieved in traditional databases, the large variability in the symbol vocabularies has delimited the success of these techniques into a small set of symbol variations. Therefore, the ongoing challenges in symbol recognition fall into designing methodologies that are able to generalize for multiple symbol representations and to deal with large image collections.

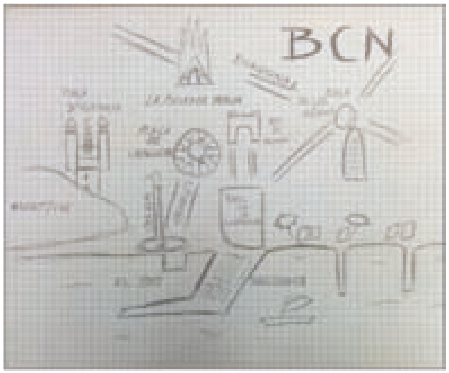
- The graphical vocabulary defines a graphical syntax that augments the expressiveness of the visual language. This syntax allows to describe complex semantics in terms of contextual relations among the document entities. Again, in the flowchart representation in Figure 1.2, text is contained inside the shape of the boxes that, at the same time, are connected by arrows. In fact, the use of this contextual information in object recognition tasks is a growing trend in


(a) Barcelona 1901.


(b) Barcelona currently.



(c) Barcelona touristic.



(d) My Barcelona.

Figure 1.1: Maps of Barcelona. Despite aiming at modeling similar concepts, graphical documents may vary the representation of the information depending on temporal, social, cultural, and functional contexts. For instance, (a) and (b) are faithful representations of the streets of Barcelona, but with more than a century of difference. Differently (c) is a sketch representation of the hot spots in the city. It focuses on giving a fast and specific picture using a very simple representation. Finally, (d) is a hand-made drawing that encloses the author's own subjective idea of Barcelona. All these documents have different styles, involving distinct information complexities, resolutions, 3 of them are colored whereas the last is not, etc. With all, the human receptor is able extract the information from them and *understand* their meaning.

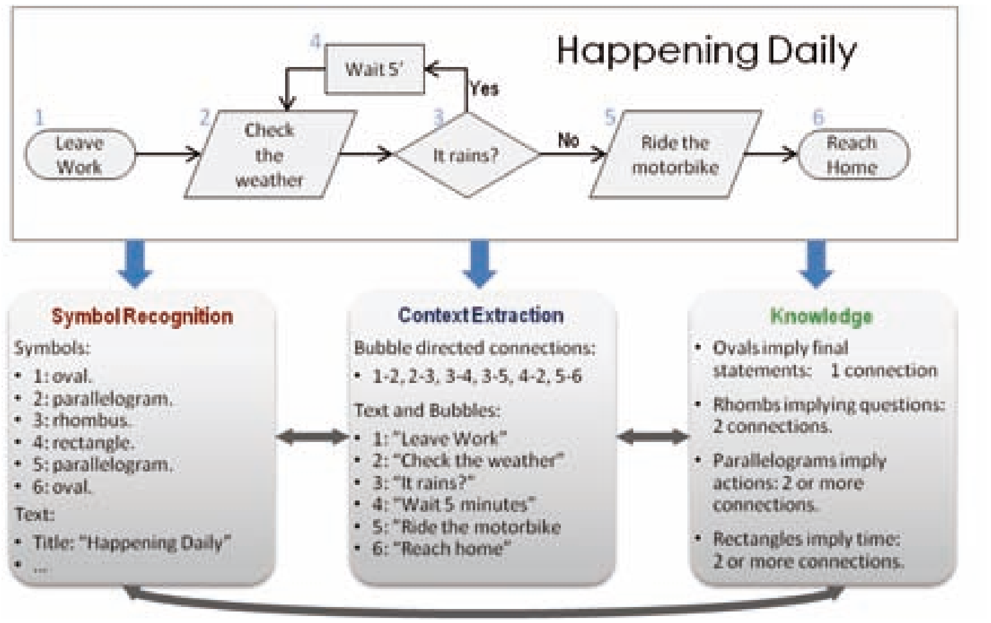


Figure 1.2: Document understanding process.

Computer Vision [29] since it increases the discriminability in this process: it is very probable to find text in an already detected box.

- Visual vocabularies and syntaxes represent complex semantic concepts. Therefore, the domain of a document should be known in order extract and process the information conveyed. For instance, in Figure 1.2, the flowchart defines a workflow of sequential actions defined by the arrows that connect the boxes. Moreover, each shape of the boxes defines a different sort of action (initial or final action, process, time, etc), which is semantically correlated with the text that describes each specific action. Thus, this semantic knowledge not only allows to reveal the meaning of the document, but can also guide the recognition process in the detection of instances and contextual relations.

Despite the huge research in this topic, the nature of graphical of documents producing a high variability in terms of visual languages has lead the researches to focus on a controlled set of document domains. For this reason, the big challenge on document understanding falls into the creation of reading systems that are able to generalize for multiple documents written in different visual languages.

1.1.2 Floor plan interpretation

In architecture and building engineering, a floor plan is drawing to scale, showing a view from above of the relationships between rooms, spaces and other physical features at one level of a structure [95]. The different steps in the building design generate multiple floor plans with distinct sort of information: structural dimensions, construction materials, furniture, water pumps, gas, air and electrical distribution, etc. Recently these documents were handmade whereas they are currently generated by CAD software; tools that aid architects in the complete design of a building.

In other words, floor plans are graphical documents that visually model the structure of buildings. Here, the graphical vocabulary consists of elements that are semantically associated to real concepts, such as walls, doors, windows, etc. They are basically defined by simple graphical primitives such as lines, arcs, textures, and colors. These symbols have strong structural relations defined by the visual syntax of the domain, that allow to describe more complex elements: such as rooms, corridors, and terraces. One can think, for instance, that a room is a closed environment surrounded by walls, doors, and windows that delimit its space. Contrarily, it is very unprovable to find an isolated door in a middle of a room. Each of these symbols, either simple or complex, has an associated semantic meaning that characterizes the concept of a building. For example, the symbol door represents an object that gives access between two rooms, rooms might have different functionalities living-room, bedroom, kitchen , there are different building intends dwellings, fabrics, theaters , etc. Therefore, as in every sort of graphical document, the task of floor plan understanding entails the extraction of the symbols and their contextual relations in order to comprehend its meaning.

Indeed, automatic floor plan understanding is a hot topic. On the one hand, architects tend to re-utilize old not-digitalized designs in order to cut designing costs. Therefore, they usually need to convert manually old drawings into new aiding computer tools. This tends to be a tedious task that is expensive in human resources. On the other hand, despite their main architectural design purpose, nowadays floor plans are spreading their usability into other different areas. New tools help non-expert users to virtually create and modify their own house by simply drawing its floor plan on an on-line application, such as for instance, Autodesk Homestyler [3] and Floorplanner [4]. These tools can automatically generate the 3D view of a building to get an idea of how it would finally look like. More recently, Google has introduced more than 10,000 indoor floor plans in Google Maps Indoor to facilitate the mobile user navigation inside large buildings, usually airports, stations, and malls [5]. In addition, state agents with large number of properties may index floor plans by some structural information extracted from them, as individual room size of each building. This kind of indexing system would be of a great help when customers ask for specific requirements, like holding a conference or organizing musical shows.

Even though the great effort of the community, automatic floor plan interpretation is far from being solved. Firstly, there is a lack of a standard notation for the design of a floor plan[1]. Thus, they must face a high variability in the visual representation of a building, as the real examples shown in Fig 1.3. For instance, a wall can be depicted as a thick line, as a specific textured pattern, or as two thin parallel

[1]In Germany, a DIN-standard exists (DIN 1356-1), but is rarely used. Furthermore, standards vary from country to country and often even from one architecture company to another. Depending on the visual appealing, the architects within the same office decide to use different representation.

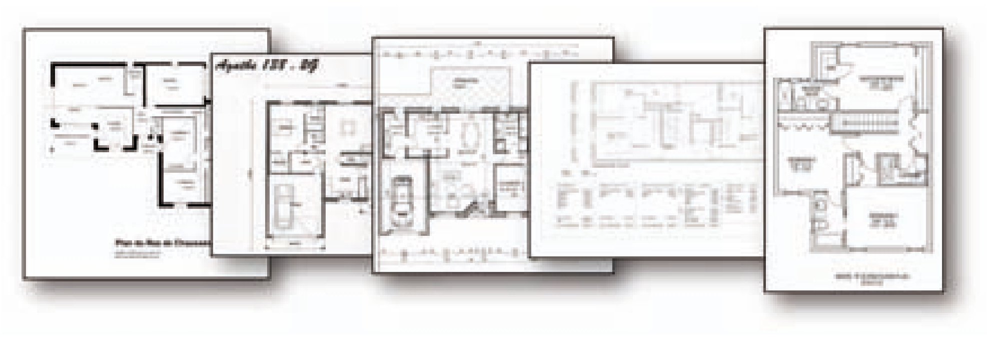


Figure 1.3: Real floor plans.

lines, etc.). Secondly, even these documents model the structure of a building, they intend different functionalities. There are floor plans for strict architectural finalities, for commercial purposes, for showcasing in design magazines, to show the emergency exits, etc. All of them contain different sort of information drawn in an appropriate visual language. Therefore, the problem of floor plan interpretation embraces the challenges of graphical document understanding: be able to deal with a huge variability in both, visual vocabularies and visual syntaxes, to extract the semantic content in the documents.

1.2 Objectives and Contributions

The ultimate aim of this thesis is to create a system architecture that is able to learn and understand the semantic content in graphical documents.

To achieve this goal, and following the steps that this process entails, we have defined the three following atomic objectives.

A. **To learn and recognize the lexicons**. We need to propose methods that are able learn and extract meaningful graphical objects from images. These methods have to be general enough to automatically adapt to different graphical vocabularies and elude the inherent difficulties of symbol recognition in document images.

B. **To learn and recognize the language syntax**. We require methods that allow to learn, extract, and analyze the contextual information among the graphical symbols. The aim is to end up with a suitable representation that provides the explanation of the document content making specific the structural and semantic relationships between the extracted symbols.

C. **To extract the semantic content**. Formally capture the complete meaning conveyed in the documents. To do so, the knowledge in the domain should be known to analyze and understand the visual representation in the document.

D. **To facilitate the learning of a language**. Hard coding the huge variability on graphical languages is unfeasible. Therefore, we need to provide to the computers generic tools that grant an automatic learning of the existing visual languages.

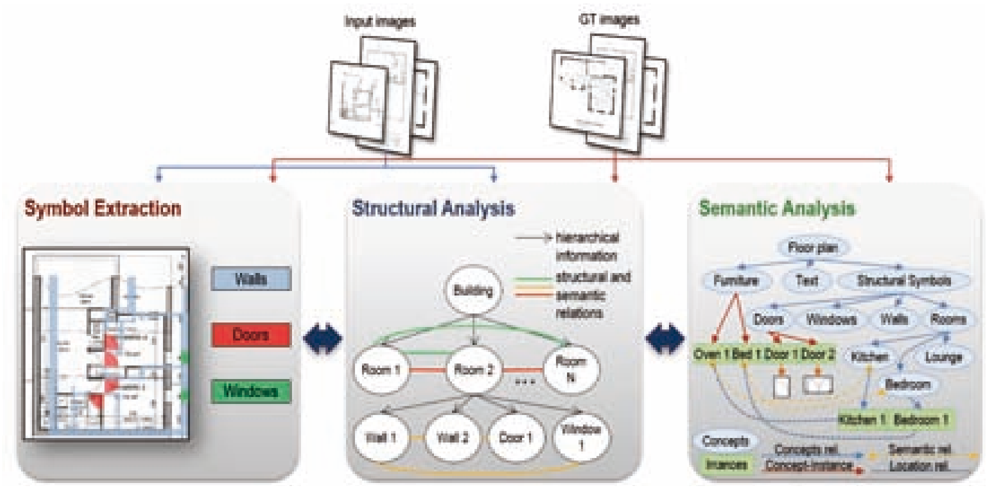


Figure 1.4: System architecture for floor plan understanding.

To achieve these objectives, we focused on the real framework of floor plan understanding. We have designed the system architecture shown in Figure 1.4 that composed by three main modules aiming at different tasks in the graphical understanding scenario: Symbol Extraction, Structural Analysis, and Semantic Analysis. This three modules produced 7 contributions separated in 4 chapters:

A. **Symbol Extraction.**

1. **Graphical object detection based on domain specific knowledge**. We present a methodology for extracting significant graphical content regardless its notation by combining general knowledge assumptions on the document domain. These assumptions are combined fuzzily to generate multiple segmentation candidates and select the ones that better characterize the element of interest.

2. **Graphical object detection based on appearance learning**. We propose a patch-based object segmentation method that relies on the automatic construction of a visual vocabulary to cluster the appearance of multiple object classes. This method just needs a small set of annotated images to work with different notations.

3. **Graphical object detection based on domain specific and appearance knowledge**. To overcome the rigidity of 1 and the need of annotated images to learn new notations of 2, we propose to combine sequentially both approaches to end up robust object detection method. This methods starts by proposing several object hypothesis using 1. Subsequently, similarly than in 2, a graphical vocabulary is created on the appearance hypothesis and used to spot lost instances in the first step.

B. **Structural and Syntactic Analysis.**

4. **Contextual extraction based on heuristic search**. From the symbols detected from 1, 2, and 3, we propose to extract the contextual information embedded in graphical documents by searching heuristically the best alignments among these objects. We have implemented a version of the A* graph traversal algorithm with a suitable monotonic heuristic to optimize this search. At the end, we construct a structural representation of the document that relates the graphical objects and their context.

5. **Syntactic model for graphical document understanding**. We propose a grammar-based model to either represent, learn, and recognize graphical documents. This syntactic model represents the content of documents hierarchically; where complex objects recursively derive into simpler parts that can be structurally and semantically constrained. Moreover, the probabilistic model embedded allows to learn those object derivations that are more common from a small set of annotated documents. This probabilistic model guides a bottom-up/top-down parser strategy to end up on the best document representation for a given instance.

C. **Semantic Analysis.**

6. **Ontology-based knowledge management**. Ontologies are widely used to make machine understandable some sort of knowledge in a certain domain. In this stream, we propose the use of ontologies to specifically and formally express graphical vocabularies and languages. Our final aim is to allow external agents to process, learn, and enhance this graphical knowledge. Actually, since our syntactic representation in 5 naturally transcribes to the ontology taxonomy, we can iteratively enhance the graphical knowledge based on automatic document understanding.

D. **Experimental Framework.**

7. **Structured database**. To conduct and test the mentioned contributions we needed to create an annotated collection of graphical documents. We have created a floor plan database labeled in terms of graphical objects and their contextual relations. Thus, it allows both, to perform the contextual learning for the grammar based method in 5 and a natural knowledge transcription to the ontology in 6. Furthermore, the lack of available tools to construct this structural ground-truth has led us to create a new image annotation tool. This is a general-purpose grondtruthing software that allows to define own object classes and properties, multiple labeling options are possible, grants the cooperative work, and provides user and version control.

1.3 Organization of the dissertation

The rest of this dissertation is organized in 7 chapters:

- In Chapter 2 we review the state-of-the-art in the domain. We report the recent approaches for graphical object recognition, structural interpretation, and semantic understanding. Additionally, we look into the most recent proposals in floor plan understanding, making emphasis in their strong points and lacks.

- In Chapter 3 we propose three different approaches for graphical object detection. All these three strategies are applied for the detection of walls for two main reasons. Firstly, this element gives crucial information concerning the structure of a floor plan. Secondly, it suffers a great graphical variability from plan to plan, which imposes the need of systems that are able to generalize for different graphical vocabularies.
- In Chapter 4 we explain the three different systems for extracting the contextual information from graphical documents. We evaluate the suitability of the methodologies for room detection in floor plans. Rooms are *white spaces* structurally composed by walls, doors, windows, and furniture symbols that enclose their environment. Hence, discovering the rooms not only entails the detection of the surrounding symbols, but also the extraction and the analysis of their structural and semantic dependencies.
- Chapter 5 is devoted to explain the definition of the domain knowledge: The Floor Plan Ontology. We show its appropriateness to formally store labeled floor plans, to validate the interpretations obtained in Chapter 4 according to some structural and semantic definitions, and to further classify and retrieve interpretations according to some semantic conceptions.
- In Chapter 6 we present the floor plan database and the groundtruthing tool designed to structurally and semantically annotate these documents. We extensively explain the four datasets of real floor plans, which include documents for multiple intends and different graphical notations. Hence, this database allows us to evaluate our methodologies for different graphical vocabularies and syntaxes.
- Finally, in Chapter 7 we conclude this dissertation, highlighting the suitability of the presented architecture for graphical document understanding. We additionally propose some future research lines to improve the obtained performance in order to construct real applications in the domain.

Chapter 2

State-of-the-Art in Graphical Document Recognition

2.1 Introduction

DIA tasks in general and GR in particular are born due to the industrial necessity not only of storing digitalized documents, but also to extract, classify, and index them according to the sort of information they convey. This industrial need arose because of the emergence of scanners that incremented massively the amount of digitalized documents to be managed. The research in GR became very popular in the 1990 s. The first complex systems for automatic graphical document interpretation appeared in this decade. From the very beginning, GR tasks have been applied in a set of very representative domains such as, electrical diagrams, engineering drawings, mathematical and chemical formulae, maps, and architectural plans.

Even though the initial interest, two main difficulties have hindered the evolution of fully automatic analysis systems. Firstly, it is not clearly defined how an interpretation result must be evaluated, i.e. how to obtain a numerical score out of that. Secondly, most of the existing architectures have tackled the problem under a very domain specific point of view, abandoning the generality to the benefit of performance on specific data. This fact has obstructed the re-usability of the existing approaches on different contexts and domains. Hereby, the researchers attracted by other GR tasks that ease the practical validation and comparison of their contributions, e.g. symbol spotting.

In this chapter, we firstly study the different methods for symbol recognition since it is a fundamental step in the graphical document understanding pipeline. Secondly, we review some of the existing the techniques on graphical document understanding from a more general perspective, focusing on how the structural approaches and knowledge models participate in this process. Finally, in order to contextualize our work, we overview in more detail the recent advances in architectural drawing understanding. We refer the interested reader on other GR tasks, such as symbol spotting and low-level processing, to [36].

2.2 Symbol recognition

Symbol recognition is a mature field of pattern recognition and several surveys and book chapters address the advances on this area [28, 32, 36, 69]. Symbols are patterns composed of visual primitives that belong to different classes in a graphical context. Therefore, one of the multiple possible categorization of symbol recognition methods is following the traditional pattern recognition classification: statistical and structural methods:

2.2.1 Statistical symbol recognition

In statistical recognition, each symbol is described as an n-dimensional feature vector $x = (x_1 \; x_2 \qquad x_n) \qquad \mathbb{R}^n$ of n measures. This feature vector is commonly called symbol *descriptor*. The classification of these descriptors entails the partitioning of the n-dimensional feature space into the different categories, one for each symbol. Therefore, the selection of appropriate feature descriptors and classification techniques are extremely important to maximize the discriminability among symbol classes.

Several surveys and book chapters review the advances on feature descriptors in the literature [36, 106]. The main goal of these description techniques is threefold. To minimize the distance among symbols of the same class while maximizing it to the rest of the classes. To minimize the spatial dimensionality to enhance the classification efficiency. And to to deal with affine transformations, noise, and distortions of the image. Some popular techniques involve the adoption or adaption of existing descriptors from other mainstream fields in pattern recognition, e.g SIFT [70], SURF [19], and HOG [33]. Yet, sometimes the description of the symbol characteristics requires of domain specific strategies accounting for these particular features, e.g. geometric moments [41, 59], zoning [43], and histogram-based [104]. Finally, classification techniques on statistical recognition methods benefit from the strong mathematical foundation of vectorial spaces, i.e. the computation of distances, products, and sums are well defined. Thus, several efficient algorithms for classification, such as KNN, boosting, SVM, and neural networks have been widely in the literature.

2.2.2 Structural symbol recognition

Structural methods describe the symbols as a set of logically related parts. These parts tend to be visual primitives such as lines, vectors, and arcs or combinations of them that in a certain context define a specific symbol instance. Then, structural-based description methods are appropriate when the structure itself is a representative and differentiable feature. Graph representations are suitable tools to describe the structure of symbols. Nodes tend to be primitives and edges spatial and geometrical relations among them. For every symbol, a graph modeling its structure is constructed. Then, the recognition process consists on finding isomorphic substructures in a graph representation. This matching process tends to be of a big complexity in high order structures. Moreover, some representations may be distorted due to the instance nature, e.g hand-drawn documents, or because of some artifacts are introduced in previous document transformations, e.g vectorization. Therefore, the current tendency in structural recognition methods is to design high efficiency algorithms that are able to deal with both, high order and error tolerant graph matching [40, 75]. In this stream, it is also noteworthy that graph-embedding techniques based on the combination of structural descriptors with statistical classification algorithms aim also at overcoming these problems by taking advantage of the mathematical foundations of statistical classification strategies [25, 50].

Syntactic methods are also a traditional manner to tackle symbol recognition in a structural manner. A grammatical formalism defines by means of a set of production rules the structure of the possible symbols. Therefore, the recognition of an input image entails the parsing of its representation to check whether the language of the grammar can generate it. The first syntactic approaches defined the structure of symbols in terms of their hierarchy parts, subparts, etc. . Obviously, these limited languages were not sufficient to express the existing variability and complexity in symbol structures. Therefore, the language expressiveness has been enhanced by introducing attributes and structural constraints in the rules [24, 45]. Moreover, probabilistic models has been incorporated into the grammatical formulation which, combined with appropriate parsing techniques, are able to deal with inexact recognition scenarios [16].

2.3 Interpretation of graphical documents

As introduced already in the introduction of this thesis, the process of graphical document understanding encompasses the translation of visual vocabularies into a higher-level conceptualization of the content. This process generally involves the extraction of the symbols (vocabulary), its structure (the syntax), and its meaning (the semantics). Each of these steps forms part of the complete pipeline of interpretation, and sometimes the manner they tackle each respective problem strongly depend on the domain of the document.

Since we have already seen in the previous section some of the symbol recognition paradigms, here we want to focus on the contextual analysis part, i.e. how the knowledge can be modeled and actually, how it may contribute in the interpretation either by guiding the contextual analysis or by detecting inconsistencies at the semantic level. In [36], the authors classify the graphical interpretation techniques in two different groups regarding the form of taking advantage of the domain knowledge in the interpretation: bottom-up and knowledge-based strategies.

Bottom-up

Bottom up interpretation, also called *ad-hoc* interpretation, consists on decomposing the problem hierarchically. The pipeline of the process is the result of combining multiple strategies in a pseudo-sequential order, where the output of one process is the input of the next. In these approaches, the knowledge of the domain is, in the most of the cases, defined on the implementation of the solution. Therefore, this approaches tend to focus on a small set of domains and their re-usability on other contexts usually leads to a considerable re-engineering. Some interpretation techniques using this methodology are [13,39,72,84]. In fact, as we report in the next chapter, the vast majority of the contributions for architectural drawing understanding are bottom-up methods.

Knowledge-based

In contrast with bottom up strategies, knowledge-based methods define the context of the documents with a certain independence to the algorithmic implementation. This fact increases the relevance of the domain knowledge definition while relaxes the algorithms specificity, i.e. the contextualization of the domain allows to guide and verify the every step involved in the interpretation. Ideally, different knowledge models may guide similar implementations in multiple domains. Yet, even if there exist some proposals in this stream, such as structural, syntactic and ontology-based analysis methods [76,79,88], the practical solution for multiple real scenarios seems to be far. We broadly address the problem of domain definition by the use of syntactical methods and ontologies in Chapters 4 and 5 respectively.

2.4 Floor plan analysis

Researchers from document analysis community has already put many efforts to analyze and transfer data from paper or on-line input to digital form, Architectural floor plans are one example of application. The conversion of these diagrams, printed or hand drawn, from paper to digital form usually needs vectorization and document preprocessing, while the on-line input needs to manage hand drawn strokes and distortions. The analysis of these diagrams allows the recognition of different structural elements (doors, windows, walls, etc.), recognition of furniture or decoration (tables, sofas, etc.), generation of corresponding CAD format, 3D reconstruction, or finding the overall structure and semantic relationship between elements.

The work of Tombre s group in [12], [38]. and [39] tackle the problem of floor plan 3D reconstruction. In these works, they have as input scanned printed plans. First a preprocess separates text and graphics information. In the graphical layer thick and thin lines are separated and vectorized. Walls are detected from thick lines whereas the rest of the symbols, including doors and windows, are detected from the thin ones. In this process, they consider two kinds of walls: ones represented by parallel thick lines and others by a single thick line. Doors are seek by detecting arcs, windows by finding small loops, and rooms are composed by even bigger loops. At the end, they can perform 3D reconstruction of a single level [12], or put in correspondence several floors of the same building by finding special symbols as staircases, pipes, and bearing walls [38]. Either in [38] and [39] it is indicated the need of human feedback when dealing with complex plans. Moreover, the symbol detection strategies implemented are oriented to one specific notation. A hypothetical change of the floor plan drawing style might imply the reconsideration of part of the method.

Or et al. in [80] focus on 3D model generation from a 2D plan. Using QGAR tools [89], they preprocess the image by separating graphics from text and vectorizing the graphical layer. Subsequently, they manually delete the graphical symbols as cupboards, sinks, etc. and other lines disturbing the detection of the plan structure. Once the remaining lines belong to walls, doors, and windows, a set of polygons is generated using each polyline of the vectorized image. At the end, each polygon represents a particular block; walls are represented by thick lines, windows by rectangles inside walls, and doors by arcs, which simplify their final detection. This system is able to generate a 3D model of one-story buildings for plans of a predefined notation. Again, the modification of the drawing style leads to the redefinition of the method.

Cherneff in [27] presents a knowledge-based interpretation method for architectural drawings: KBIAD. His aim is to extract the structure of the plan; this means walls, doors, windows, rooms, and the relations between them. The input is an already vectorized plan with vectors, arcs, and text that is preprocessed to obtain special symbols as doors. The system has two models: the semantic and the structural one. The semantic model represents the plan with building components as walls, doors, and windows, and their relations that arrange in composite structures as rooms. The structural one represents the geometry of the plan, including two-dimensional spatial indexing of primitives. A predefined Drawing Grammar represents the drawing syntax of a plan describing its symbols and components as a set of primitives and their geometrical relationships. The rules have to be general enough to accept all the variations in a symbol but specific enough to distinguish between symbols. For example, they define walls as parallel segments that can have windows or doors at the end. This fact strongly restricts the interpretation possibilities, since walls in real floor plans can be curved or even not be modeled by parallel lines.

The work presented by Ryall in [93] focus on segmenting rooms in a building. They propose a semi-automatic method for finding regions in the machine printed floor plan image, using a proximity metric based on a proximity map. This method is an extension of the area-filling approach that is able to split rooms when there is a lack of physical separation. Nevertheless, the method retrieves many false positives given by objects that are also drawn by closed boundaries, such as tables, doors, and staircases. Once more, the method works on a single notation.

Macé in [72] also focused on the extraction of thestructure from scanned plans. As in [12], [38], and [39] a text/graphic separation is done followed by a thin/thick separation from graphic components. In that way the authors look for walls among the set of thick lines. Then, they look for parallel lines extracted from contours, expecting walls to be formed by very thick lines. Afterwards, they find doors and windows to finally detect rooms based on a recursive decomposition of images until convex regions are found. The wall detector strongly depends on the wall notation, and should be re-designed to be able to cope with different floor plans.

Ahmed in [13, 14] starts with the classical text/graphics separation to later separate graphic components into thin, thick lines and, as a novelty, medium lines. Lines forming walls are extracted from thick and medium ones while thin lines are considered forming symbols. Then symbol spotting is applied using SURF to detect doors and windows and extract the rooms from the plan. At the end, text inside the rooms allows to label each of them. This method is further enhanced by the same authors in [14] by splitting rooms in as many parts as labels are inside them, just splitting them vertically or horizontally according to the distribution of their labels. These works take into account some structural and semantic information, as they are labeling rooms with their name and are verifying their composition using the position of their doors and windows. However, as before, the method might have to be revisited when dealing with floor plans of different graphical conventions.

Some works have as an input a CAD file format that contains the real non-distorted original polylines and lines. This is the case of the work of Lu in [71], where 3D reconstruction is performed from CAD floor plans. First they extract parallel overlapped lines to find T, X, and L shapes. Later they find their connections to construct walls and then the 3D reconstruction of the structure. After extracting the structure, they delete the lines in order to segment graphical symbols as furniture or stairs. Their method bases the recognition on typical features as geometrical ones, attributes of the lines, relational attributes among components, etc. They reconstruct 3D building model based on the integration of the recognition results and are specific for a single CAD file notation.

Also, the work of Zhi et al. in [107] takes as input a CAD file. It extracts automatically the geometrical and topological information from a 2D architectural drawing and transforms it into a building evacuation simulator. Firstly, they semi-automatically filter out redundant information such as furniture, text, specification notes, and dimensions, and only keep the essential entities: walls, doors, windows, lifts, etc. Then, they transform the plan into an attributed graph and look for loops, which accordingly to their attributes, are classified into different types: spatial loops (rooms, corridors), physical loops (walls, columns), door loops, window loops, and unidentified loops. Even this procedure is easy to use, it leads to some classification errors and further reasoning is needed. Finally, they identify the plan units (compartments) and the system is integrated in a model that simulates emergency evacuations from complex buildings.

Works like [18] and [67] analyze hand-sketched floor plans. In [18] a hand-sketched analyzer transforms floor plan into a CAD file. They extract the lines that model the building structure, which are sketches on a preprinted paper with a grid of lines in drop out color. The method describes line elements, such as walls and windows, and closed region elements, such as doors. On the other hand, [67] uses subgraph isomorphism and Hough transform to recognize different building elements and their topological properties. Subgraph isomorphism allows to recognize symbols and Hough transform to detect walls made by hatched patterns. It is worth to mention that in both, [18] and [67], the drawing conventions are set beforehand.

Floor plan structural retrieval is one of the recent interests for architects. The works of Weber et al. in [15, 101] and Wessel et al. in [102] are two examples in this domain. In the case of [101], the query is a sketch drawn on-line by the user. Their system allows the user to sketch a schematic abstraction of floor plan and searches for floor plans that are structurally similar. The sketch is translated into a graph enclosing the structure of the plan and it is compared with the graphs representing plans in a repository using subgraph matching algorithms. In [102] the input is a polygon soup representing a 3D plan, so they do not need to vectorize the plan. From this polygon soup, the authors extract the structural polygons of each floor stage by grouping that ones that are parallel to the floor at a determined height. The rest are considered furniture. Then, the rooms, doors, and windows are detected by cutting the horizontal plane of each floor. Finally, they construct a graph where attributed nodes are rooms and attributed edges are connections between them: doors or windows. Based on this connectivity graph, fast and efficient shape retrieval from an architectural database can be achieved.

2.5 Conclusions

In his chapter, we have explained the current issues on graphical document interpretation. The vast majority of the methods presented in the literature center their understanding on a small set of document domains. They tend to embed the knowledge model in the implementation, abandoning the generality and thus hindering their re-usability in other contexts. Moreover, they are often evaluated on private data using self-defined evaluation techniques. Hereby, the progress on systems to fully understand of graphical models has been obstructed in comparison to other GR tasks.

In order to overwhelm these concerns, in this thesis we propose several contributions that aim at generalize at much as possible. Furthermore, all the data, the evaluation protocols, and the relational models proposed are deeply explain and shared freely for research purposes.

Chapter 3

Symbol detection

3.1 Introduction

As mentioned in the introduction of this dissertation, to understand natural language sentences we need firstly to be able to extract the primitives or tokens, these are letters or words. Then, the analysis of syntax of the sentence leads to its semantic interpretation. Analogously, to interpret graphical documents we need to extract the graphical primitives. These primitives are usually formed by simple structures as lines or textures that convey meaningful information regarding structures that are more complex. Nevertheless, given the nature of the graphical documents, these primitives may vary a lot from document to document, even when they belong to the same topic.

This is the case of floor plans. In these documents *walls* are considered to be the main primitive symbol; they conform the structure of the building and convey inherent information concerning the rest of structural elements, such as doors, windows, and rooms. Not coincidentally, most of the state-of-the-art strategies on floor plan interpretation has put their first effort on wall extraction [12, 38, 39, 72, 80]. Yet, the nonexistence of a standard graphical notation that produces a large variability in walls modeling, see Figure 3.1, has lead the wall detection problem to be a challenging task. In fact, most of traditional strategies have focused only on a reduced range of similar notations, which in turn, lead to the nonexistence of robust floor plan interpretation systems.

We try to overcome the wall detection problem by proposing three different strategies of handling multiple graphical notations. We call our first approach *structural segmentation.* It settles the segmentation on intrinsic attributes of walls: *they are a repetitive element, naturally distributed within the plan, and commonly modeled by parallel straight lines.* Although most of the instances are detected, these structural assumptions are too restrictive and certain instances such as curved walls are missed. The second strategy, called *statistical segmentation*, is a patch-based approach that learns, from a small set of annotated images, the graphical notation of walls. We propose multiple alternatives regarding the patch topology, description and learning/classification framework and study their impact in the overall system performance. The generality of this approach allows to successfully find other structural elements such as doors and windows. Finally, we have additionally pipelined these two approaches to end up with a notation invariant approach that does not need labeled data to learn the graphical appearance, and it is robust to encounter curved instances. We call this method *combined segmentation*.

In this chapter, we firstly review the preprocessing applied to the images in order to enhance the performance of the proposed methods. Then, we explain these 3 wall segmentation approaches and present a contrasted summary of the results on floor plans of different notations.

3.2 Document preprocessing

A good preprocessing transforms the original image into a more suitable input where the information of interest remains accurate to the original but is easier to be extracted and manipulated. In floor plans, as it is the case of graphical documents, the different formats, qualities, and amount and type of the information included forces the analysis methods to be very flexible to deal with this huge variability. Preprocessing techniques then are used to relax this input variability and let methods with strict input needs to generalize better.

We propose the use of 4 iterative preprocessing tasks that have been applied on the images for all the methods presented in this dissertation on symbol detection. This preprocessing tasks can be applied iteratively and they are described in the following:

Binarization

Floor plans may contain color for different purposes: as additional information on the structural arrangement of the objects in the building, i.e. rooms, to highlight some relevant information in the document, or just for design matters. But floor plans can be also found in gray-scale and binary formats. Therefore, color can not be considered as a key piece of information to base the subsequent analysis methods. Moreover, floor plans images tend to have a high resolution, which combined with the RGB information leads to an excessive dimensionality to deal with. Therefore, to omit the color information and reduce the document dimensionality we binarize all the original images using the Otsu method.

Text/Graphic separation

In real floor plans textual information such as dimensions and annotations might appear or not. It strongly depends on the floor plan type. In the case of documents with commercial purposes, the functionality and area of each room is frequently written in. In floor plans for construction guidance, textual information is combined with graphical symbols to indicate the location and installation of different equipments such as the electric and the heating. Contrarily, in modern drawings aiming at enhancing the beauty of the designs, minimalistic information is included and therefore, text hardly appears. Hence, likewise the color information, text is omitted and filtered out using the method proposed by Tombre et al. [97]

Deskewing

We also tend to assume that a floor plans are correctly oriented, which means that most of the lines are perfectly horizontal and vertical. Not coincidentally, there are analysis methods in the literature [13, 14] that base the detection of some structural symbols by just considering the lines in these directions. Nevertheless, the orientation of a floor plan is another fact that strongly depends on the document digitalization procedure. We dispose of documents that are slightly *disoriented* as a result of a miss-placing at scanning time. To solve this issue, we detect and correct possible deviations in floor plan orientation by adapting the approach for hand-written text deskewing [81].

Rescaling

Floor plan documents tend to be large images. Therefore, for efficiency issues we downscale all those images that are of a higher resolution than 4000 × 4000 pixels. We use a bicubic interpolation to minimize the side-effects of this process.

3.3 A structural approach for wall segmentation

Even without being architectural expert, it is easy for humans to identify the walls in a floor plan. They have graphical characteristics regarding their structural functionality. Not by chance, most of the existing wall detection methods have been focused on encountering these characteristics to spot walls: straightness [72], orthogonality [39], saliency [13], etc. Nevertheless, these approaches focused on strict wall features observed in their images, which may not be satisfied in other collections. On the contrary, the underlying idea of our method is to base the segmentation on a flexible combination of 5 structural premises for characterizing walls in a general means, called *wall-assumptions*:

Wall-assumption 1. *Walls are modeled by parallel lines.*

Wall-assumption 2. *They are rectangular; longer than thicker.*

Wall-assumption 3. *Walls appear in orthogonal directions.*

Wall-assumption 4. *Different thickness is used for external and internal walls.*

Wall-assumption 5. *They appear repetitively and naturally distributed among the plan.*

As it is shown in Figure 3.1, this set of *wall-assumptions* are far from being a collection of unbreakable statements that perfectly define walls in their graphic composition. For example, there are floor plans with diagonal or curved walls, buildings with the same thickness for interior and exterior walls, etc. Nevertheless, a relaxed combination of them enhances the flexibility of the system, leading it to a good final segmentation independently of the building or document complexity.

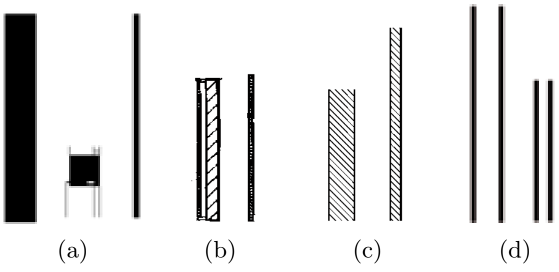


Figure 3.1: Real graphical examples of vertical walls for *Dataset Black* in (a), *Dataset Textured* in (b), *Dataset Textured2* in (c), and *Dataset Parallel* in (d). These datasets are introduced in Chapter 6

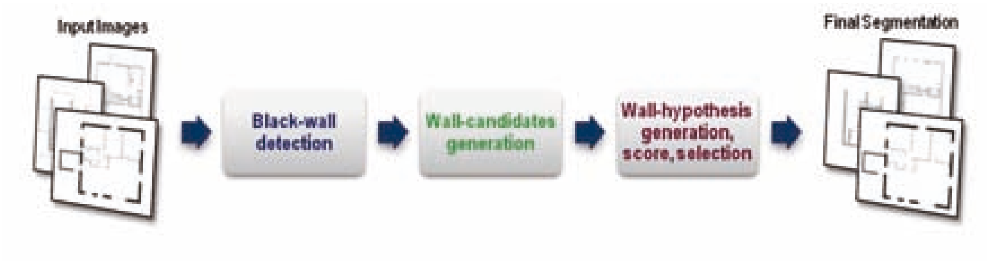

Input Images
Black-wall detection
Wall-candidates generation
Wall-hypothesis generation, score, selection
Final Segmentation

Figure 3.2: Structural segmentation pipeline.

Figure 3.2 shows the pipeline of our approach. Firstly, the input image is preprocessed to filter out unnecessary information. After that, the preprocessed document is transformed into its edge image when it contains black thick walls. Then, using run length analysis, the parallel lines in the plan are detected and the distances between them are quantized in a histogram. Outstanding values of the histogram correspond to frequent runs likely to define wall segments. Finally, the final wall segmentation is given by the combination of wall image candidates according to the *wall-assumptions* postulated above.

3.3.1 Black-wall detection

Despite walls are usually drawn by parallel lines with a repetitive graphical pattern, texture, or emptiness between them, there are floor plans that include walls graphically composed by black thick lines. An example of these sort of models, that we commonly call *black walls*, is shown in Figure 3.1a. Since, agreeing to *wall-assumption 1*, we base the structural segmentation on detecting parallel lines, we need to automatically identify those floor plans with black walls and transform them into a more suitable input; this is the edge image.

To detect the existence of this type of walls, run lengths over the foreground pixels in the horizontal and vertical directions are quantized in a histogram. Floor plans with black walls present more sparse frequencies with significant out-layers in high positions. Contrarily, the runs in images with a lack of black walls are distributed more normal-like in the lower bins. We fit a mixture of Gaussians into the 1D data using the EM algorithm, and set relaxed boundary on the *sigma* parameter η^{thw} to detect those plans containing black walls. Those positive instances are transformed into their corresponding edge image using the Canny edge detector.

3.3.2 Wall-candidates generation

Wall segment candidates of different thickness are generated in this step according to the *wall-assumption 1*. Firstly, the parallel lines at different image orientations α are detected by foreground runs of a certain minimum length rl^b_{min}. Then, the distance between each couple of parallel candidates is calculated by background runs in their orthogonal orientations. The runs are quantized into a histogram hist_{RL}, where high frequencies stand for repetitive runs among black lines, and thus, possible wall thicknesses. On the other hand, lower frequencies, which are the vast majority, are produced by other infrequent objects modeled by parallel lines. The hist_{RL} is smoothed and those bins with highest frequencies according to a predefined threshold are grouped into a set of adjacent runs. This is done to reduce the noise dependency on poor quality plans. Finally, a segmentation image is generated by retrieving the foreground lines involved in each of the thickness clusters. They are considered as segments which possibly belong to walls, or part of them; from now on, called wall-candidates. The different steps implicated in this process are illustrated in Figure 3.3.

3.3.3 Wall-hypothesis generation, score, and selection

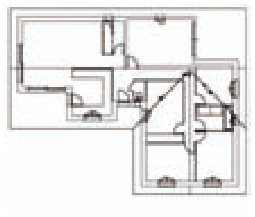
Wall-candidates are combined generating multiple wall segmentation hypothesis. The resulting hypothesis are ranked according to the properties involved in the *wall-assumptions.* The final segmentation adopted is the one with the highest score.

Generation

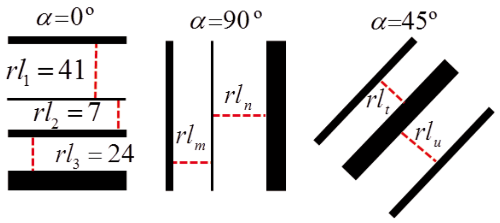
Multiple segmentation hypothesis are generated from the set of wall-candidates because generally, in floor plans, interior and exterior walls have different widths. There are also some interior walls which usually are slightly thicker than the rest, mainly due to their structural purposes in the building architecture. Moreover, some walls are graphically modeled by more than two single parallel lines. Therefore, as *wall-assumption 4* states, there may be more than one wall-candidate that lead to the correct segmentation. A visual example of this situation is shown in Figure 3.3d, where walls have three different thicknesses.

The k-combinations for the n wall-candidates for all possible k subsets, except for the empty set, are generated spreading into $2^n - 1$ final combination subsets. The final segmentation hypothesis set S is given by the logical disjunction function over the wall-candidates w_i in every subset:

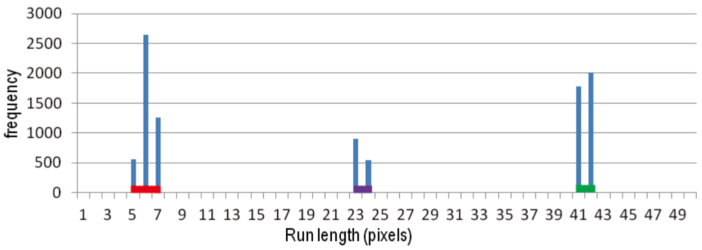
$$S = \quad w_1 \;;\quad w_n \;;\quad w_1 \quad w_n \;;\quad w_1 \qquad\qquad w_n \tag{3.1}$$



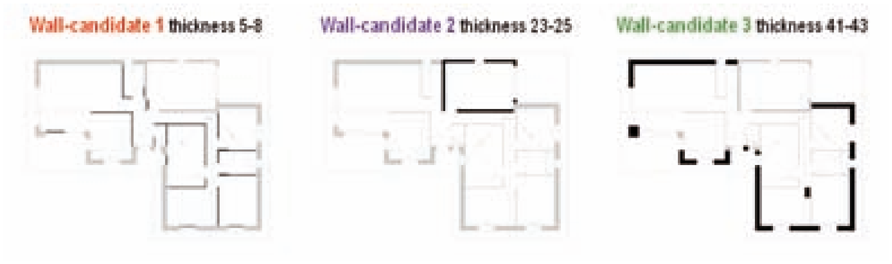
(a) Edge image



(b) Zoom-in for the run length calculation in 3 α orientations



(c) hist$_{RL}$



Wall-candidate 1 thickness 5-8
Wall-candidate 2 thickness 23-25
Wall-candidate 3 thickness 41-43

(d) Wall-candidates

Figure 3.3: In (a) we show the input image of the system. Since the original has black walls, the edge image is considered instead. In (b) we zoom-in the calculation of the background run lengths from the input image. The runs are calculated in α orientations, which agreeing to *wall-assumption 3*, preferably include 0° and 90°. This runs are quantized into the hist$_{\text{RL}}$ histogram shown in (c). The adjacent bins are clustered into sets, represented in the image by different colors. Finally, each set generates its corresponding wall-candidate in (d).

renamed as,

$$S = \{h_1, ..., h_i, ..., h_I\}, \tag{3.2}$$

where h_i is a final segmentation image hypothesis.

Score

For each one of the final segmentation hypothesis in S, four different normalized attribute scores are calculated to account their agreement to the *wall-assumptions*:

- **SH** is frequency in hist_{RL} for those thickness involved in each the segmentation hypothesis:
$$\text{SH}_{h_i} = \text{hist}_{\text{RL}}(h_i) \tag{3.3}$$
SH benefits those hypothesis formed by several wall-candidates segmentations with multiple thickness , which agrees with *wall-assumption 4*.

- **CC**: is the summation of the number of individual connected components in each wall-candidate involved in the segmentation hypothesis:
$$\text{CC}_{h_i} = \#\text{cc}(h_i) \tag{3.4}$$
where $\#\text{cc}(h_i)$ is the number of connected components in the hypothesis h_i. This attribute score avails segmentations with multiple components, which agrees with *wall-assumption 5* when mentioning that walls should appear *repetitively.*

- **AR** states for the mean longness aspect ratio (longitude / thickness) of the connected components in each of the wall-candidates:
$$\text{AR}_{h_i} = \overline{\text{long}(\text{cc}_j(h_i)) \;\; \text{thick}(\text{cc}_j(h_i))} \quad j \; cc_j \quad h_i \tag{3.5}$$
According *wall-assumption 2*, walls are longer than thicker, and then, longer aspect ratios are favored in the final segmentation.

- **DiffD** accounts on the difference of black pixel distribution between the input image and each of the segmentation hypothesis. This difference is calculated locally for the regions r from a rectangular grid placed on both images:
$$\text{DiffD}_{h_i} = \sum_{n=1}^{r} \sum_{m=1}^{r} p_{nm} - p^{i}_{nm} \tag{3.6}$$
where p_{mn} and p^{i}_{mn} are the percentage of the black pixels in the mn^{th} region of the original image and h_i respectively. DiffD enforces segmentations distributed similarly to the input image throughout the plan, agreeing with *wall-assumption 5* in terms of walls location, and allows to filter dispersedly located elements.

Selection

Finally, to determine which segmentation hypothesis agrees better to the *wall-assumtions*, we calculate their global score as:

$$W(h_i) = \text{SH}_{h_i} + \text{CC}_{h_i} + \text{AR}_{h_i} + \text{DiffD}_{h_i} \tag{3.7}$$

That segmentation hypothesis with the higher score will be taken as the final wall segmentation for the given input image.

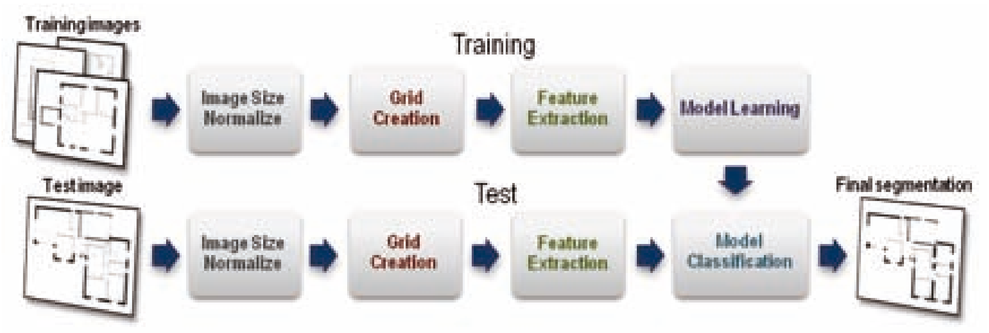


Figure 3.4: Statistical segmentation pipeline.

3.4 A statistical approach for object segmentation

We have implemented a patch-based approach for object detection and recognition in architectural floorplans. Even if it was thought to segment walls, it is also able to segment other structural elements such as walls and windows as we report in Section 3.6.3. The pipeline of the process is shown in Figure 3.4. For both, training and test, there is a common step for image size normalization, for defining the grid topology, and for a subsequent feature extraction. We define three different grids to characterize the composition of the patches in the images. Then, for every patch, a descriptor of its representative features is extracted. We propose two different strategies for learning and classifying these features. The first approach constructs a vocabulary of representative patches where each word has assigned a probability of belonging to every class of objects. In the testing, each patch is assigned to the nearest word in the dictionary, inheriting the class probabilities of the word. In the second approach, we perform the training and testing by means of a support vector machine classifier SVM. Each step of method is explained extensively in the following.

3.4.1 Images size normalization

Some works in the literature on floor plan interpretation assumed that all floor plan images have the same resolution and line thickness. However, this is not necessarily true. Resolution and line thickness will strongly depend on the device used to capture the images (scanner or camera) and on the resolution of acquisition. This can result in a larger variability that can be a problem for approaches working at patch level. Therefore, images in a dataset are automatically normalized regarding their line thickness.

This process is based on regularizing the resolution of the floor plans regarding the most basic structural element: the thinnest line. A histogram based on counting the consecutive black pixels in both vertical and horizontal directions, is created for each floor plan. Assuming that the thinnest line is the most common type of structural element, the histograms maxima should indicate the width of the thinnest lines in each image. Finally, all plans are resized taking the plan with the thinnest lines as a reference and using bilinear interpolation. In this way we get more similar symbol representations for all the plans.

3.4.2 Grid creation

Our main objective is to perform a pixel-level object segmentation. However, using pixels as elementary units involves a high computational cost, sometimes making the problem infeasible in terms of speed and memory. Thus, considering patches of neighboring pixels not only increases the speed of the proposed method, but also allows to encapsulate local redundancy which could be used as feature statistics. Nevertheless, these techniques have the drawback of abandoning pixel accuracy. For that reason, we have defined three different grid topologies to study which is the one that leans better to the final solution.

- **Non-overlapped regular grid:** This grid is composed of squared non-overlapped patches directly defined over the image. The main advantage of this topology is its simplicity and its cheap computation cost. However, since each pixel of the image belongs to only one patch, final pixel class assignment will be only affected by its patch label, while sometimes one patch can contain pixels from different categories. Moreover final assignment of pixel category will strongly depend on how patches fall into the image.

- **Overlapped regular grid:** In order to avoid the strong dependence on the grid location over the image, we have also defined a squared patched grid, but with overlapping. In this grid, each pixel belongs to several patches according to the parameter σ_{ov}, which specifies in pixels, the separation between patch neighbor centers. Therefore, final class assignment of a pixel is weighted up between the class probabilities of all its patches. This process is explained in section 3.4.4. The main advantage of this topology is that images are defined by more patches and thus, object boundaries would be better segmented. On the other hand, for the same reason, pixel-level classification is more costly with respect to a non-overlapped grid.

- **Deformable grid:** With this topology we aim at adapting the grid to center the cells on the objects. We have defined a deformable squared patched grid which follows the concept of deformable model presented in [64]. By the time the regular grid is constructed, for each of its cells, we move its center (within a deformation area) to the point that maximizes the total amount of intensity of pixels in the 9-neighboring patches and the patch itself.

3.4.3 Feature extraction

Once the desired grid is created, a patch-descriptor is calculated to represent every patch that contains at least one black pixel. Hence, since white patches are considered as *background*, they are ruled out in the learning step due to computational reasons. We have used four patch-descriptors to analyze the impact of feature extraction in the global performance of the system.

- **Pixel Intensity Descriptor:** called PID for ease, is a simple descriptor formed by concatenating the raw pixels of the patch in a row-wise manner.
- **Principal Component Analysis:** PCA is calculated over the row-wise vectors of all patches. The 95% of the discriminative information of the patches is maintained meanwhile the dimensionality is highly reduced.
- **Blurred Shape Model:** BSM is a shape descriptor introduced by Escalera et al. in [43] that has been successfully applied to different graphics recognition applications. The patch is divided in $n \times n$ equal-sized subregions (*BSMreg*) where each subregion receives votes from the points in it, and also from the points in the neighboring subregions. Each point contributes with a weight according to the distance between the point and the subregion centroid. The final description is a vector formed by concatenating the number of weighted votes received by each subregion.
- **SIFT, Scale Invariant Feature Transform:** is a descriptor presented by Lowe in [70] that describes an image patch as the accumulation of the local gradient orientations. It is invariant to rotations and scaling transformations, and has been used in multiple domains in computer vision such as object recognition [47], biometrics [21], and robotics [96].

3.4.4 Model learning and classification

Up to now the training and the test images are described by means of patch-descriptors. We propose two different strategies to learn these descriptors and classify them into object classes. The first approach trains and classifies using SVM whereas the second method is bag-of-words model. In both methods, the output labels for the patch-descriptors are combined to obtain the final classification at pixel-level.

Support-vector machine approach

We use the SVM implementation of LIBSVM [26]. N patch-descriptors for each object class C are selected randomly to train the classifier. We use a Radial Basis Function kernel (RBF) defined as:

$$K(pd_i \ \ pd_j) = e^{-\gamma \ \ pd_i - pd_j \ \ ^2} \tag{3.8}$$

where pd_i and pd_j are patch-descriptors and $\gamma \quad \mathbb{R}_+$ is the RBF width parameter selected by cross-validation.

At testing time, each patch-descriptor is classified by the SVM outputting the probability estimates for each one of the object classes.

Bag-of-patches approach

In the learning phase we cluster all the labeled patch-descriptors from the learning-set into a vocabulary of K representative words. We use the fast version of K-Means proposed in [42]. Then, the probability for each word of belonging to each object class is calculated. Every patch-descriptor pd, which has ground-truth label to the class $c_i \quad C = \quad c_1 \qquad c_N$ Background , is assigned to its closest word in the dictionary w_j. Then, the conditional probability for a word of belonging to each object class is given by:

$$p(c_i \; w_j) = \frac{\#(pd_{w_j} \;\; c_i)}{\#pd_{w_j}} \quad i \;\; j \tag{3.9}$$

Where $\#(pd_{w_j} \;\; c_i)$ states for the number of patch descriptors with the label c_i assigned to codeword w_j, and $\#pd_{w_j}$ is the total number of patch-descriptors assigned to w_j. The summation of the probabilities of a codeword for all the classes is one:

$$\sum_{i=1}^{M+1} p(c_i \; w_j) = 1 \quad j \tag{3.10}$$

The classification phase is performed by a Nearest Neighbor (NN) classifier in the Euclidean space. Each patch-descriptor is hard-assigned to the closest word in the vocabulary inheriting its class probabilities.

Final pixel assignation

The two classification strategies explicated above output the object class probabilities for descriptors in the test images. Nevertheless, since we desire a pixel-level segmentation, the final pixel assignation would depend on the grid topology used at testing time.

In the case a non-overlapped grid, one pixel px just belong to one patch and therefore, the final pixel categorization is straightforward:

$$c(px) = \arg\max_i (p(c_i \; pd)) \tag{3.11}$$

On the other hand, in deformable and overlapped grids pixels may be contained in several patches. Since every patch has its own probability of belonging to every class, pixels would acquire a definite number of classification probabilities per object category. This issue can be seen as a combination of classifiers problem, where different classification results are obtained for a single pixel. We adapt the Mean Rule presented in the theoretical framework for combining classifiers of Kittler *et al.* [65] to obtain the final pixel classification:

$$c(px) = \arg\max_i mean(P(c_i \; pd)) \quad pd \quad px \qquad pd \tag{3.12}$$

3.5 Combining the structural and statistical methods for wall segmentation

As a third contribution for wall segmentation in floor plans, we propose an approach that is able to overtake the singular drawbacks for both methods presented above, while maintaining their graphical notation invariability. On the one side, this new method is not limited to detect straight objects as the structural-based approach is, but is able segment curved and squared instances. On the other side, contrarily to

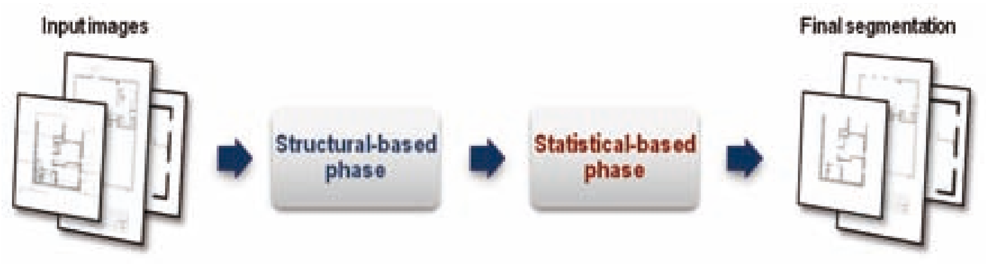


Figure 3.5: Combined segmentation pipeline.

the statistical-based approach, it does not need any pre-annotated data for learning the objects graphical appearance.

This method is the result of serializing slight modifications of both strategies. Firstly, potential straight wall segments are extracted in an unsupervised way similar to the structural-based method explicated in Section3.3, but restricting even more the wall candidates considered in the original approach. Then, based on the statistical-based method presented in Section 3.4, these segments are used to learn the texture pattern of walls at different scales and spot lost instances. The pipeline of the resulting approach is shown in Figure 3.5.

In this section we only focus on those aspects of the method that have been modified from the original approaches and so, avoid redundant explanations.

3.5.1 Structural-based phase

All the steps in this phase but the last one are the same as the structural-based approach. In the original approach, after preprocessing those documents with black walls and generating the wall-segment candidates from the runlength histogram, the wall-candidates are combined into a higher number of wall-hypothesis. The hypothesis with the highest attribute score is the one taken as the final segmentation. Contrarily, here we calculate the attributed values defined in Section 3.3.3 for every wall-candidate w_i^{str}. Then, we rank the wall-candidates according to their score and only the top n are taken into consideration for the subsequent statistical-based phase.

The benefit of the adopted strategy is twofold. We only keep those segmentations with a higher confidence score, which tend to be instances from exterior and interior walls. We can automatically adapt the subsequent statistical phase to the specific characteristics of every w_i^{str} selected.

3.5.2 Statistical-based phase

At this point, we have a set of n wall-candidates $W^{\mathrm{str}} = \{w_1^{\mathrm{str}}, ..., w_n^{\mathrm{str}}\}$, each of them containing straight segments of one specific thickness. The aim of this phase is to learn their own graphical appearance to refine the segmentation obtained in the previous phase. To do so, we have adapted the bag-of-patches model from the original statistical-based method in Section 3.4.

To obtain the statistical segmentation we perform the following process of learning and classification n times; one for each $w_i^{\text{str}} \quad W^{\text{str}}$. For ease, let us consider we want to obtain the statistical segmentation associated to w_i^{str}. We start by splitting the input images into squared and overlapped patches of size $s_{w_i}^{\text{str}}$, which is strongly related to the thickness of the segments in w_i^{str}. This procedure is repeated for the image rotations 45°, 90° and 135° with two purposes: to get more learning instances and to achieve rotation-invariableness. Patches falling into segmented regions in w_i^{str} are labeled as positive examples $c =$ Wall whereas the rest are labeled as negative $c =$ Background instances. Completely white patches are filtered out. The image descriptor selected to describe the patches is the BSM [43]. Then, following the same strategy as in Section 3.5.2, an equal number p of positive and negative patch descriptors are clustered into a vocabulary of K visual words . Finally, the classification step is performed in the same manner as the bag-of-patches classification in Section 3.5.2, producing an output segmentation called w_i^{stat}.

3.5.3 Combining both segmentations

In the end, we need to combine the n wall segments obtained in the structural phase and their corresponding segments extracted in statistical phase. Therefore, the final segmentation is obtained from the binary sum of all these segmentations:

$$W = \sum_{i=1}^{n} w_i^{str} + w_i^{stat} \tag{3.13}$$

3.6 Experimental evaluation

In this section, we evaluate the methods for graphical object detection presented in this chapter in Sections 3.3, 3.4, and 3.5. The evaluation and comparison among the 3 strategies is pursued for wall detection, as walls are the main structural element in floor plans and they suffer a substantial variability from plan to plan. Yet, to demonstrate the graphical adaptability of the statistical approach, we also present some qualitative results on door and window detection. Therefore, we start by explaining the assessment protocol defined for object detection. Then we evaluate the 3 methods on the database explained in Chapter 6 taking special attention to their different configurations and parameter influences. And we finally compare quantitatively and qualitatively these strategies.

3.6.1 Wall evaluation protocol

The evaluation protocol proposed for wall detection works at pixel level. It is calculated over the three images that the system handle for every floor plan: the result or output image, the ground-truth image, that contains the labeled pixels for the different classes, and the original image. The use of the original image is justified because we only consider in the score those pixels that are black in the original-image, since only black pixels convey relevant information for segmentation.

Table 3.1: Definition of True-positives (TP), False-positives (FP) and False-Negatives (FN) for Wall segmentation evaluation. Black pixels in an image are considered as 1 and white ones as 0

	Original image	Ground-truth image	Output-image
TP	1	1	1
FP	1	0	1
FN	1	1	0

The results of our experiments in wall segmentation are expressed using the Jaccard Index (JI). JI is currently popular in Computer Vision since it is used in the well-known Pascal Voc segmentation challenge [44] as evaluation index. It is an objective manner of presenting the results because it takes into account both, false positives and negatives, experimented by the system. It is compressed in the interval [0,1] and the closer to 1, the better is the segmentation. JI is calculated as:

$$JI = \frac{TP}{TP + FP + FN} \tag{3.14}$$

where TP, FP, and FN are defined in Table 3.1 regarding the three images used in the evaluation.

In addition to that, since this method is thought as an initial step for a complete floor plan interpretation system, Recall is also taken into account; it is more straightforward and effortless to post-process an over-segmented result, than finding some lost walls in later processes of a global floor plan analysis system. The recall is calculated as follows:

$$recall = \frac{TP}{TP + FN} \tag{3.15}$$

3.6.2 Evaluation of the structural approach

The structural-based method is influenced by four parameters: (rl^b_{min}, α, η^{thw} and r). They are set experimentally in a very relaxed way for the multiple plans tested. The parameter rl^b_{min} states for the minimum run length in the black horizontal line generation for being considered as a possible line. rl^b_{min} is set to 10 pixels, which is sufficiently small to cope with low resolution documents, and adequately high for efficiency issues. The angle interval α specifies in which rotation of the input image lines can be detected. It has a strong impact when diagonal walls occur in the image. Yet, the lower α, the more image-lines are generated and thus, the slower is the global performance. Experimentally, we set α increment in 15°, which is a good trade-off between performance and speed. The sensitivity boundary over the estimated η^{thw} is used to detect plans with black-walls. The results obtained for the 4 different datasets have demonstrated that η^{thw} values for plans with black-walls are, at least, 75 times higher than in plans without this kind of walls. Therefore, in a very relaxed way, we decided that floor plans with η^{thw} estimation values over 25, are classified as documents containing black-walls. The last parameter to be set is the number of equal-size regions r used to calculate the black pixels distribution difference DiffD. Experimental tests have shown that the performance for $r = \ 9 \ 16 \ 25 \ $ varies at most 0.02 in terms of JI. For other close values to them, the rates drop significantly. $r = 9$ is adopted since is the configuration with the best global performance.

3.6.3 Evaluation of the statistical approach

We have 18 possible configurations for the statistical-based method when we consider the 3 grid topologies, the 3 different patch-descriptors, and the 2 learning/classification strategies. Moreover, each configuration has its own intrinsic parameters that are affected by the type of images is dealing with. Therefore, to evaluate this method smartly, we perform all our preliminary experiments on the most challenging dataset: the *Texturedset*. On this dataset, we show quantitatively the influence of the modules by switching one at a time among the multiple proposals. Once the best configuration is selected, we discuss the impact of the different parameters in the global performance.

- **Grid topology**. Among the 3 alternatives, the best results are obtained when patches are defined either by an overlapped or a deformable grid, see Tab.3.2. The main reason is that, unlike the non-overlapped grid, these topologies are able to incorporate in the classification process contextual information contained in neighbor patches. In the case of the overlapping grid, each pixel is influenced by several overlapped patches according to σ_{ov}. Conversely, in deformable grid, patches are adapted to objects; every patch center is moved regarding the pixel intensity of its neighbors. This contextual information allows, for instance, to increase the classification rate on pixels that are located in the borders of the walls. A correct classification of these pixels using a non-overlapped grid would depend on how patches fall into the image.

  When comparing quantitatively and qualitatively the performance using an overlapped and a deformable grid, the differences are minimal. Yet, the denser image representation of the overlapped grid when a high overlapping ratio σ_o is selected leads to better results in crowded areas. There, the deformable grid centers may be influenced into many orientations, leading in some cases to a similar representation to the non-overlapped grid. For this reason, we have selected the overlapped grid with an appropriate $\sigma_o v$ for our final method configuration. As we explain below, the σ_{ov} value is a trade-off between performance and speed.

Table 3.2: Results regarding the different grid compositions.

grid topology	descriptor	learn./class.	*Dataset*	*JI*
non-overlapped	BSM	SVM_{RBF}	*Textured*	0.74
deformable	BSM	SVM_{RBF}	*Textured*	0.80
overlapped	BSM	SVM_{RBF}	*Textured*	0.82

Table 3.3: Results regarding the different patch-descriptors.

grid topology	descriptor	learn./class.	*Dataset*	*JI*
overlapped	**PID**	SVM_{RBF}	*Textured*	0.75
overlapped	**PCA**	SVM_{RBF}	*Textured*	0.80
overlapped	**SIFT**	SVM_{RBF}	*Textured*	0.76
overlapped	**BSM**	SVM_{RBF}	*Textured*	0.82

Table 3.4: Results regarding the different learning and classification strategies.

grid topology	descriptor	learn./class.	*Dataset*	*JI*
overlapped	BSM	$\textbf{SVM}_{\text{linear}}$	*Textured*	0.76
overlapped	BSM	$\textbf{SVM}_{\text{RBF}}$	*Textured*	0.82
overlapped	BSM	**Kmeans+NN**	*Textured*	0.85

- **Patch descriptor**. Tab. 3.2 shows the performance of the system for the different strategies of describing the information in patches. PCA, SIFT, and BSM encapsulate better the patch information while decreasing the description dimensionality. In the case of PCA, we have selected experimentally to maintain the 95% of the original information as a trade-off between performance and dimensionality. For the SIFT descriptor, the orientation is fixed and the scale has been set to 2 experimentally. In the case of BSM we selected by cross-validation the number of subregions a patch is divided to, being $BSM_{reg} = 3$ for a patch-size of 18×18. BSM outperforms PCA and SIFT in all datasets, and it characterizes better the high intra-class variability existent in some images, as is the case of the *Texturedset*.

- **Learning/classification**. The results for the two learning and classification strategies using patches extracted from an overlapping grid and described using BSM are shown in Tab. 3.4. For the SVM classifier, we show additionally the results using both, a linear and a RBF kernel. The number of patch-descriptors p to train the SVM classifier is learned by cross-validation, leading 7500 patches of every class to obtain the best results in the *Texturedset*. Similarly, the number of words in the vocabulary is strongly influenced by the images. Images with objects with a high intraclass variability need of larger vocabularies that encapsulate the multiple variations. For the *Texturedset*, the best performance is obtained by $K = 2000$.

  From the table we can conclude that the vocabulary based approach highly outperforms the SVM-based method in challenging dataset *Texturedset*. Is it worth to mention though that this difference is not that high on more uniform datasets. In the *Blackset* dataset for instance, where walls are more uniformly drawn, the performance for the Kmeans+NN is only 1% higher than the SVM. In addition to that, the SVM classification is 3 times faster in average than the vocabulary-based approach. This leads the SVM-based method to be a

Table 3.5: Results regarding the normalization of the images in terms of line thickness.

grid topology	descriptor	learn./class.	*Dataset*	*JI*
overlapped	BSM	SVM$_{\text{RBF}}$	**Texdtured**	0.82
overlapped	BSM	SVM$_{\text{RBF}}$	**Texdtured**$_{\text{NORM}}$	0.83
overlapped	BSM	KMeans+NN	**Texdtured**	0.85
overlapped	BSM	KMeans+NN	**Texdtured**$_{\text{NORM}}$	0.86

plausible option in *easy* images.

- **Image normalization**. To conclude the study on how the different modules affect the global behavior of the method, we would also like to show the results comparison when the images of *Texturedset* dataset are normalized as explained in Section 3.4.1 or not. Table 3.5 shows the results, making clear that the preprocess of the images in terms of line thickness has a positive impact independently on the learning/classification strategy.

Finally, once we have selected the configuration that leads to the best results on the *Texturedset*, it is time to explain how we have set its intrinsic parameters and which impact they have in the overall method behavior.

The system is only influenced by 3 parameters, the patch-size, the overlapping-factor σ, and the size of the vocabulary, all 3 learned in validation time. For *Blackset*, 30 images are used for validation following a 5-fold strategy, while the 60 remaining are used for testing, with a 10-fold strategy. This procedure is repeated by exchanging some of the validation images for testing ones until all the 90 images in the dataset are tested. Similarly, the parameter validation in *Texturedset2* has been performed using 6 images following a Leave-One-Out strategy. The rest are used for learning and testing using a 3-fold procedure. On the other hand, regarding the *Texturedset* and *Parallelset*, due to the low number of instances, all of the images are used at once for parameter validation and testing following a Leave-One-Out strategy.

When we analyze the influence of the parameters in the different datasets it turns out that three aspects require the addition of more context to the final classification: a low resolution, a big intraclass variability, and a hight similarity with other floor plan elements. These are the cases of *Texturedset, Texturedset2*, and *Parallelset* respectively, in which bigger patches and more overlapped among them are used to deal with their respective problems. Moreover, a bigger vocabulary is needed to represent accurately the different textures existing for modeling exterior and interior walls in the *Texturedset.* In contrast, in *Blackset* a small vocabulary constructed from small patches is able to cope with the regularity of the black walls contained in this dataset. In Table 3.6, the parameters used in each dataset are shown numerically.

Additionally, we want to show the suitability of this method to also detect doors and windows. In quantitative terms, the method is able to detect up to the 91% of the windows in the *Blackset* and the 60% in the *Texturedset.* Regarding the doors, the method detects the 71% of the doors in the *Blackset* and the 63% in the *Texturedset.* We show qualitatively this performance for door and window detection on one instance

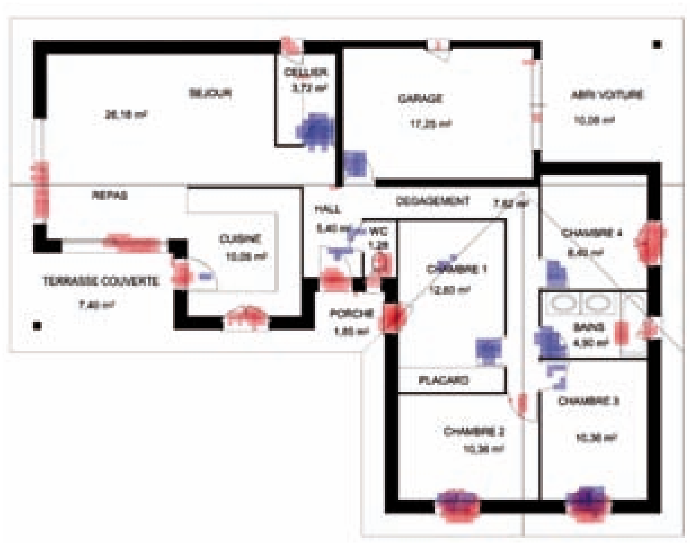


Figure 3.6: Windows and doors heatmap for a *Blackset* image. We show in blueish and reddish the most probable areas to encounter doors and windows respectively.

Table 3.6: Quantitative parameters on wall detection

	Patch-size	Voc. Size	Overlapping
BlackSet	10×10	100	5
TexturedSet	18×18	2000	3
TexturedSet2	20×20	1000	5
ParallelSet	42×42	1000	12

of the *Blackset* in Figure 3.6.

3.6.4 Evaluation of the combinational approach

Our method is inherently affected by the same parameters than the structural and statistical approaches. In the first step of our method, the parameter values considered in Section 3.6.2 are also adopted here. Hence, rl^b_{min} = 10pixlels, $\alpha = 15°$, $\sigma^{thw} = 25$, and r = 9pixels.

On the other hand, the parameters in the second step have been restudied and recalculated experimentally since the learning origin is completely different from Section 3.4. The parameters that affect the behavior of our method are three inherited from the original approach: the patch size, the overlapping factor, and the vocabulary size; and a fourth one generated by the new learning framework, that accounts the amount of patches used for creating the vocabulary p. Regarding patch size, only proportional values to the highest wall thickness in the wall-candidate have been tested, adopting finally 0 5 times the size of the thickest segment. For σ^{ov}, several proportional values to patch size have been tested being 1 $2 \times$ patchsize the value that leads to the best performance. In terms of the vocabulary size, smaller dictionaries proved to generalize better. Thus, just 300 words are enough to learn the wall texture. Finally, the experiments have shown that the more learning data, the better results. Therefore, high p values are preferred. Yet, we have detected a saturation point over the 75.000 patch-descriptors.

3.6.5 Discussion of the results

Once reviewed all the approaches and their parameters it is time to put them into context in both, numerically and graphically. The Tab. 3.7 shows the comparison among the three wall detection strategies presented in this thesis on our 4 datasets of real floor plans. Notice that we included the proposal of Sheraz et al. [13] but, since this method is strictly thought to work on black walls, it can only be applied on the *Black* dataset.

At a glance, the three methodologies behave similarly on each dataset. As it can be seen, their best results are obtained on the *Blackset*, achieving almost a perfect segmentation. On the other hand, wall segmentation on the rest o the datasets is a big deal more challenging and therefore, not that accurate. For the *Texturedset*, the lower resolution and the slightly different notation for exterior and interior walls increase the false positives rate, mainly given by the detection of symbols that are modeled similarly to interior walls. Again, the lack of texture in the *Parallelset* leads to wrongly segment other symbols that are also modeled by parallel lines. Finally, in the case of the *Texturedset2*, the unavoidable downscaling images are 5671×7383pixels introduces undesirable noise and brakes the original regularity of the hatched pattern, producing multiple textural possibilities for a single wall.

When comparing the performance of the three approaches, the statistical method outperforms the rest on all datasets. Its JI score is over 80% in the challenging *Texturedset* and *Texturedset2*, and up to 71% in the *Parallelset*. Nevertheless, when we compare the performance between the statistical method with the structural and combined methods, we must take into account that these latter do not need labeled data and they are adapted singularly to every image. With all, they obtain significantly close results on all datasets in both scores. In this stream, we also want to note the difference between the structural and combined performances. In the combined method, the statistical step permits to effectively spot lost instances from the structural phase without decreasing the precision. This leads to increase the recall on all datasets the JI score on the *Textured2* and *Parallel* datasets.

We finally show the qualitative results obtained by the different approaches on images of the four datasets in Figures 3.7, 3.8, 3.9, and 3.10. As it can be seen, the results for the three methods are satisfactory independently to the image notation, being the statistical approach the one that achieves the best results in all images. Furthermore, while the structural method only detects straight walls, the combined is able to retrieve curved and square instances, as it can be contrasted between Figures 3.7b and 3.7d. To further demonstrate the good performance of this method when labeled data is not available, we show in Figure 3.11 the results on some images

Table 3.7: Global results for wall detection

	Black		*Textured*		*Textured2*		*Parallel*	
	JI	recall	JI	recall	JI	recall	JI	recall
[13]	0.90	0.92						
Structural Sect.3.3	0.93	0.97	0.82	0.97	0.77	0.91	0.66	0.98
Statistical Sect.3.4	0.97	0.99	0.86	0.99	0.82	0.81	0.71	0.86
Combined Sect.3.5	0.95	0.99	0.82	0.98	0.79	0.96	0.67	1

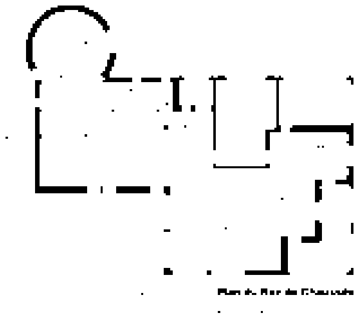
randomly downloaded from the Internet[1].

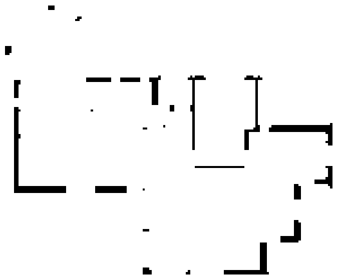
3.7 Conclusion

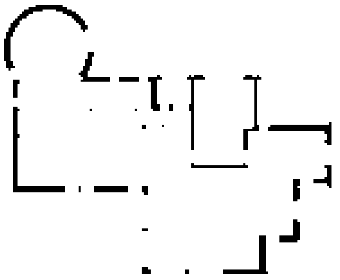
Graphics understanding entails a first step of symbol recognition. This process is hard given the nature of these documents and the huge variability existent in graphical symbols. In floor plans for instance, the lack of a graphical standard lead walls, doors, and windows to be modeled differently from document to document. Therefore, the creation of methodologies to cope with all these variations is a challenging task.

In this chapter, we have proposed three different strategies to detect walls in floor plans. These techniques, contrarily to the existing approaches, have demonstrated their adaptability to different graphical notations and floor plan complexities. Moreover, the generality of statistical-based approach allows to detect other architectural symbols such as doors and windows in their multiple graphical forms. With all, to further prove the convenience of these contributions, in the next chapter we will use them a key step in the floor plan understanding pipeline.

[1]Google Images: https://www.google.es/search?q=floor+plan


(a) Original image.


(b) Structural segmentation.


(c) Statistical segmentation.

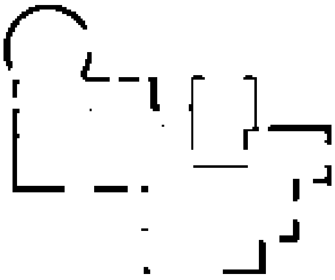
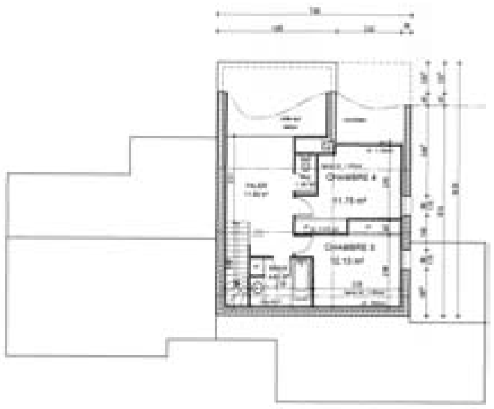
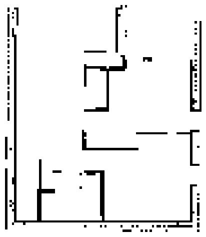
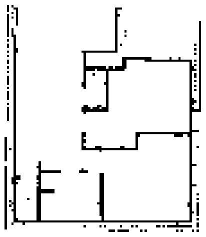

(d) Combination segmentation.

Figure 3.7: Wall segmentation results on the four for the *Blackset*.


(a) Original image.


(b) Structural segmentation.


(c) Statistical segmentation.

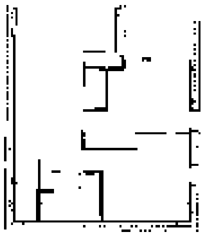
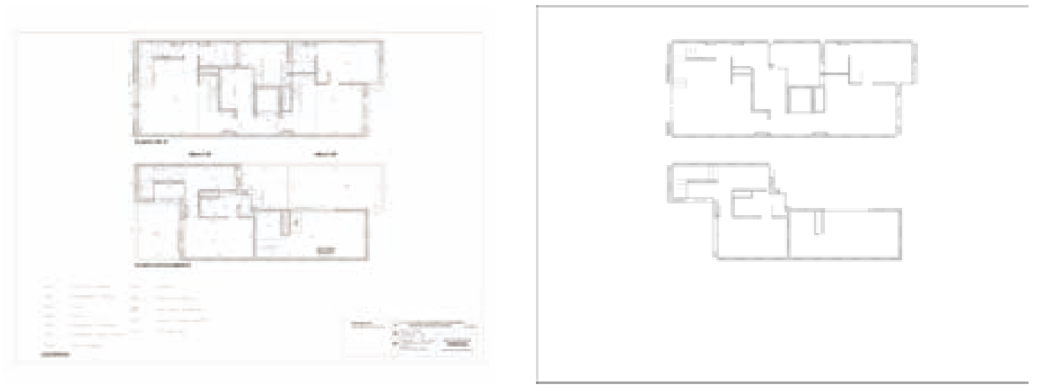

(d) Combination segmentation.

Figure 3.8: Wall segmentation results on the four for the *Texturedset*.


(a) Original image. (b) Structural segmentation.

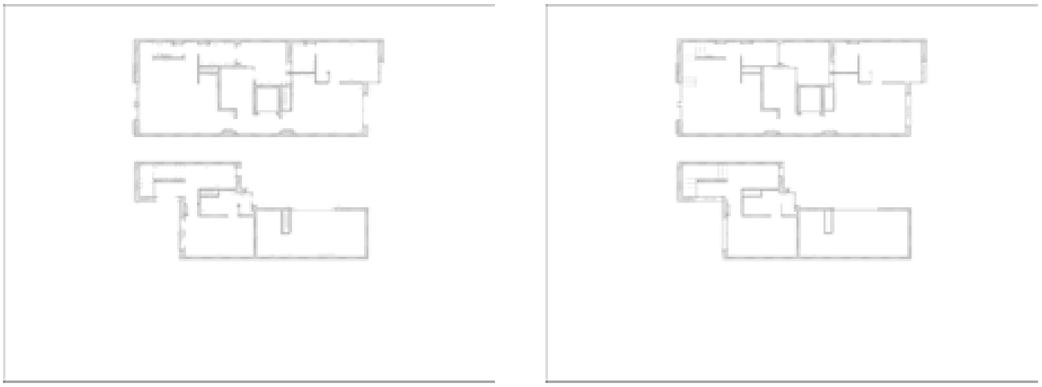
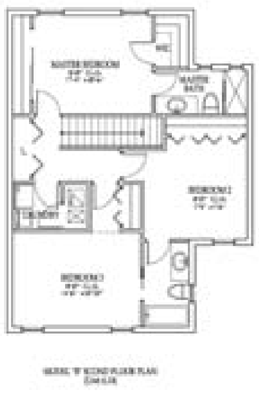
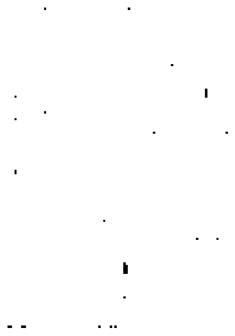
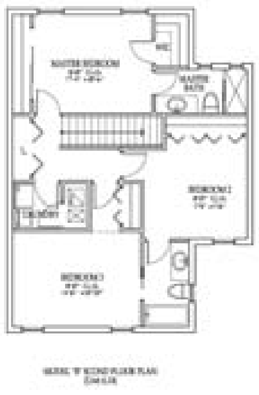

(c) Statistical segmentation. (d) Combination segmentation.

Figure 3.9: Wall segmentation results on the four for the *Textured2set*.


(a) Original image.


(b) Structural segmentation.


(c) Statistical segmentation.

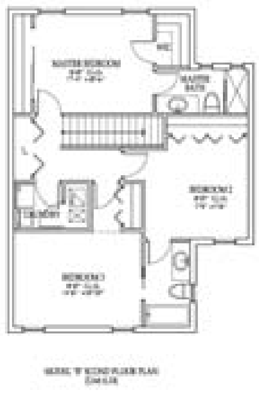
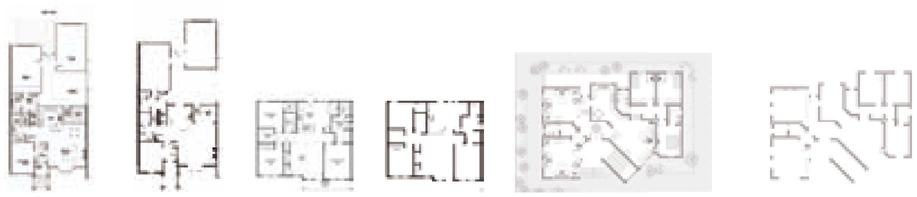

(d) Combination segmentation.

Figure 3.10: Wall segmentation results on the four for the *Parallel*.



(a) (b) (c) (d) (e) (f)

Figure 3.11: Wall segmentation for three images downloaded from the Internet.

Chapter 4

Structural and syntactic recognition

4.1 Introduction

A visual language expresses complex semantical concepts by means of a graphical vocabulary that is structured according to a visual syntax. In other words, a visual syntax defines contextual relations among the graphical objects that permit to augment their conveyed meaning w.r.t. considering these graphical items isolated. Therefore, the extraction of the contextual relations among the graphical symbols is a crucial step within the document understanding pipeline.

Contextual relations among graphical objects indicate different types of mutual dependences that can be categorized according to the sort of information they express. Generally they can be hierarchical, structural, or semantical. Hierarchical relations allow to define complex objects as an indefinite set of simpler items. Structural relations may determine, among others, object co-occurrences and relative localizations. And semantic relations carry higher-level conceptual information. Yet, in the same manner that each natural language has its own grammatical rules, each graphical language is governed by its own visual syntax. This fact implies that the contextual information conveyed in a document strongly depends on its language, i.e. the contextual relationships encountered in a city map are totally different from those in a flowchart. Therefore, as language models assist text recognition methods [46], the contextual knowledge in graphical documents is a substantial clue to guide and validate the recognition procedure. For instance, arrow symbols *connect* boxes in their extremities in flowchart drawings. When either arrows or boxes are recognized, the structural relation *connection* can be used to discover the missing parts. Hereby, the automatic structural analysis of graphical document entails 3 main challenges. (i) We need to find appropriate structures to represent the contextual information in a document. (ii) The graphical language should be known or learned in order to guide the recognition. (iii) The recognition procedure not only extracts the contextual relations among graphical elements, but it is also guided and validated by the domain knowledge.

In this chapter we present two different contributions to analyze the structure of graphical documents with the pretext of evaluating different strategies on representing, learning and recognizing their contextual information. Both strategies adopt graph formalisms of different complexities for context representation. Graphs are a natural manner for structuring this sort of information, where nodes represent graphical objects or concepts and attributed edges describe the types of relations among them. The first contribution, called *structural analysis of graphical documents* has as starting point the early detected symbols by the method explained in Section 3.4. Then, artificial intelligence search algorithms with heuristic knowledge of the syntax recognize the contextual information on the graph representation. The second called *syntactical analysis of graphical documents* is totally different in spirit. Here, a syntactical method is used to either represent, learn, and recognize graphical documents. This is a stochastic grammar over an attributed graph that models the hierarchical, structural, and semantic relations in a document. The probabilistic model embed in the grammar allows to automatically learn the visual syntax from annotated data. It is combined with a bottom-up/top-down parsing strategy that allows to extract the most probable contextual graph for a given input. For this latter approach we propose two contributions with different strategies for learning the contextual relations and two others for parsing.

The effectiveness of the presented methods is evaluated for the detection of rooms in floor plans. On the one hand, floor plans are representative instances of graphical documents given their variability in vocabulary and syntaxes, as explained in Section 1.1.2,. On the other hand, rooms are the elements that better represent structure of a building. They are conceptual entities, not symbols, that need to be extracted from the contextual analysis. At the end of this chapter we explain the adopted evaluation protocol for room detection and present a deep analysis of the obtained results.

4.2 A structural approach for graphical document analysis

The *a priori* knowledge in a certain document domain has been widely used to design adhoc recognition approaches in different scenarios, i.e. office documents, chemical structures, flowcharts, and floor plans. Information such as the type of the graphical items and their structural characteristics allow to create informed recognition techniques that demonstrated to work successfully in a set of controlled collections. Nevertheless, it has been proved that the more expertized an interpretation technique is, the worse it generalizes to other recognition scopes. For instance, most of the existing floor plan techniques are oriented to specifically work in a constrained set of graphical notations, they assume that building models are well-aligned w.r.t image frames, walls are straight, etc.

In this section we present a floor plan analysis method that its main contribution w.r.t. the existing approaches is its generality; it is not only able to interpret floor plans regardless their graphical notation, but it also has been successfully adapted to interpret graphical documents of a different domain flowcharts in [92]. As it can be seen in Figure 4.1, this system is the result of combining a statistical method to segment graphical primitives in the images, and a structural method to recognize their contextual dependences. Firstly, the symbol extraction methodology presented

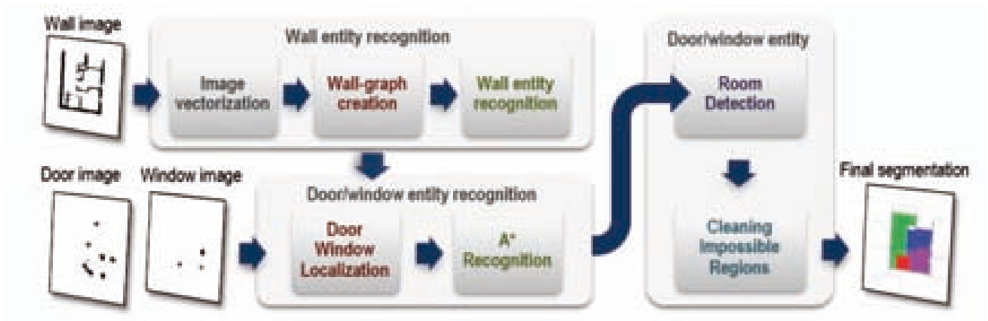


Figure 4.1: Pipeline of the method.

in Section 3.4 is used to extract the walls, the doors, and the windows from the document images. Therefore, no restrictions are made in terms of object shape and appearance. Secondly, a linear graph is constructed from the input image and it is optimally analyzed by an A* search algorithm to extract the contextual information among the early detected objects. Predefined domain knowledge such as *walls, doors, and windows tend to be incident* and *rooms are empty spaces surrounded by these elements* guide this graph recognition procedure. At the end, the method outputs a graph representation incorporating the complete structure of building modeled in the input floor plan.

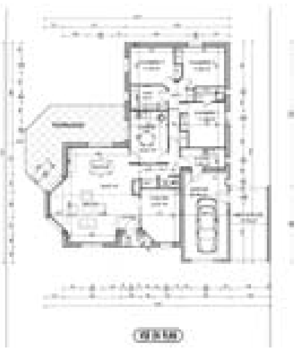
Since the segmentation of walls, doors, and windows is previously explained in Section 3.4, we focus on the structural analysis of the document, which entail the following steps: wall entity recognition, door and windows entity recognition, and room recognition.

4.2.1 Wall entity recognition

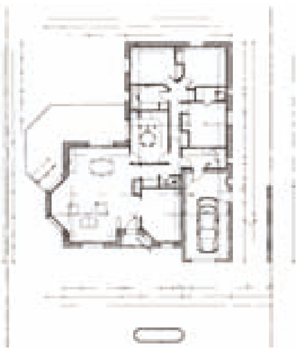
Up to here, the pixel-based approach has segmented and labeled the image pixels as belonging to walls, doors and windows. The structural-based recognition firstly groups the basic graphical segmentation into these three types of structural entities – walls, doors, and windows entities–. Then, rooms are detected by finding cycles in a plane graph of entities.

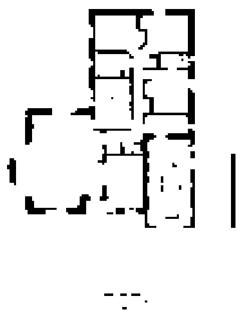
A wall-entity is the semantic definition of a real *wall* in a building: a continuous structure that is used to divide the space into different areas. It is usually delimited by windows, doors, and intersections with other walls. Thus, in order to extract a realistic structure of a floor plan, the system should be able to detect these entities from the wall-images obtained after Section 3.4.

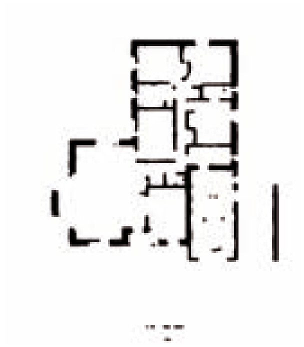
The reader might wonder at this point why wall entities are sought before door and window entities. There are mainly two reasons for this. Firstly, walls –and rooms– are the elements which mainly define the structure of a building. Almost all the rest of elements can be easily located using semantic assumptions based on wall location, e.g. usually doors and windows are placed between walls. This will lead to an easier door and window entity recognition afterwards. Secondly, walls are usually modeled


(a) Original Floor plan.

(b) Text-layer after Text/-Graphics segmentation.


(c) Graphic-layer after Text/Graphics segmentation.


(d) Segmented wall image.


(e) Logical AND between the original image and segmented wall image.

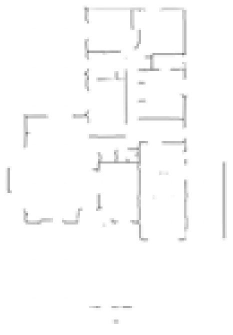

(f) Graph containing the wall-entities.

Figure 4.2: Complete flow of wall recognition process.

by a highlighted uniform texture, which makes them a big deal easier to detect than doors and windows.

This process is divided in three different stages. Firstly, wall-images are vectorized and post-processed to reduce the noise. Secondly, a planar graph is built out from the vectorization. Finally, wall-entities are extracted after analyzing the wall-graph.

Wall-image vectorization

In this step we want to extract a vectorial representation of the wall-image in Figure 4.2d. Since this image is obtained after classifying squared structures patches to detect linear elements walls , a raw vectorization of the image leads to encounter multiple corners and small unaligned segments for completely straight walls. This issue is solved by applying a morphological opening after closing the wall-image, which allows to delete small noise and join unconnected pixels, and a logical AND with the original opened image to make borders straighter. The result is shown in 4.2e. This modified wall-image is vectorized over its skeleton using QGAR Tools [89].

Wall-segment-graph creation

After vectorization, an attributed graph of line segments is created using the open source graph library called JGraphT[1], which is based on JAVA and includes a sort of complete modules for graph management already implemented.

In this attributed graph, the nodes are the segments obtained from the vectorization, and the edges represent binary junctions among connected nodes. The attributes of the nodes are the thickness of the line segment extracted from the skeletonization and the geometrical coordinates of the end-points of the segment. In this way, geometric computations among nodes such as distances or angles can be performed easily. On the other hand, edges contain two attributes: the coordinate of the junction point between the two segments, and the relative angle between them. We cal this graph wall-segment-graph (*wsg*).

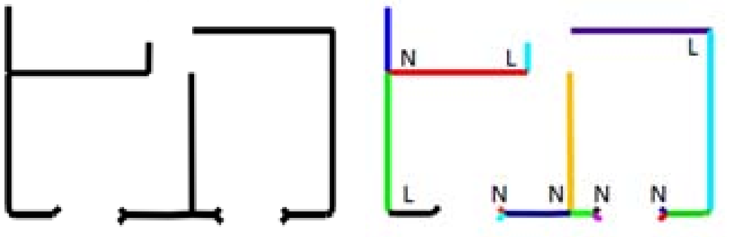
wsg traversing for wall entity recognition

The final task for wall entity recognition is based on the grouping of nodes that presumably belong to the same wall in the *wsg*. With this aim, three different kind of junctions within nodes are considered as being natural borders among walls:

1. N-junctions for $N > 2$: The intersection of three or more different wall-segments at a certain point can be considered as the intersection of N different walls.

2. L-junctions: Two wall-segments that are connected by a rectangle angle with a certain tolerance margin are considered to belong to two different walls.

3. 0-junctions: Any wall-segment which is not connected to any other in one of its end-points is considered as a natural delimiter for a wall.



(a) Vectorization of sub-wall entities in a part of a real input image. (b) Indicated N and L junctions with each of the Wall entities in different color.

Figure 4.3: Wall entity recognition.

The algorithm for wall-entity recognition firstly deletes the edges from the wall-segment-graph which are involved in N-junctions and L-junctions. N-junctions are easily found by consulting the degree of connectivity of the nodes at their ending points. If the connectivity degree is higher than 2, then that point is a N-junction. Regarding L-junctions, the process performed is the same but, this time, the degree has to be equal to 1, and also the *angle* attribute of the edge has to be close to 90°. Finally, the disconnected sub-graphs are found using the Depth First Search (DFS) algorithm. The complete process is shown in *algorithm* 1 and the result is visually shown in Figure 4.3.

We call the graph obtained after this process wall-graph (wg). Here, nodes are wall-entities, which can be seen as groups of connected wall-segments, attributed with the geometric coordinates of their end-points. Edges are connections among walls at these end-points.

4.2.2 Door and Window entity recognition

It is hard to imagine in the real world that a door or a window is not located between, at least, two walls. In floor plan documents, door and window symbols are modeled by lines that are incident with wall lines. If we take a look at the graph obtained after vectorizing the original floor plan image, and we focus our attention on a window or a door and the surrounding walls, it exists at least one path that only contains window or door line-nodes connecting one terminal of each wall, see Figure 4.4. Hence, we can take advantage out from this assumption in order to enhance the detection of these entities. Here, graph connections between walls are explored in the locations where doors and windows have been found after Section 3.4. This search is driven by the algorithm A*. Lately, a post-process heuristically seeks for windows and doors between well-aligned walls.

[1]http://jgrapht.org/

Algorithm 1 Wall-entity recognition

```
auxWallGraph := wg
interestEdges    searchEdges(auxWallGraph,Njuctions    Ljuctions)
delete(auxWallGraph,interestEdges)
**for all** 0junction    auxWallGraph **do**
  **if** notContains(visitedNodes,0junction) **then**
    **var** newWall :=
    **while** DFSiterator.hasNextNode **do**
      add(visitedNodes, nextNode)
      add(newWall, nextNode)
    **end while**
    createWall(wg,newWall))
  **end if**
**end for**
```

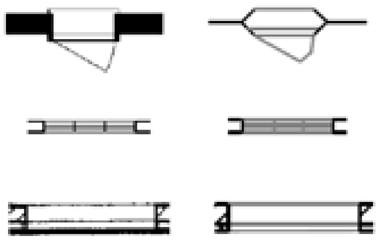
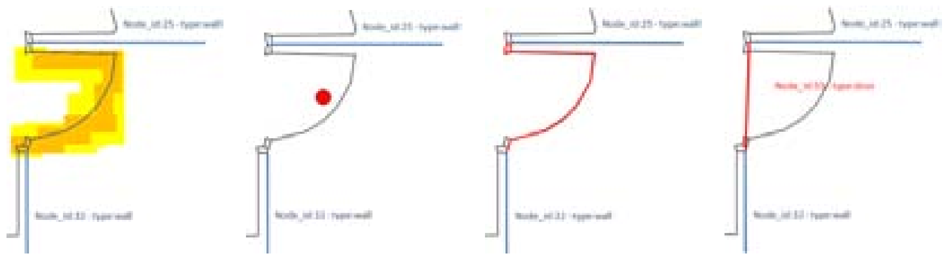


Figure 4.4: Left: three different windows from real floor plans with dissimilar notations. Right: the respective vectorization. Black vectors belong to walls and gray to windows.



(a) (b) (c) (d)

Figure 4.5: Process for finding a door entity. In (a) shows the detection of the door by the statistical approach presented in Section 3.4 over the real image. The centroid of the area where the door is found is shown as a red point in (b). The node expansion by A* for finding the path between the two walls in the graph is shown in red in (c). Finally, both walls are connected in (d) by means of a door node.

A* for Door and Window detection

As it is shown in Figure 4.5a, every vector in the original vectorized image which overlaps a region classified as door[2], by the system explained in Section 3.4, is considered as candidate door entity vector. For each of these regions, the center of mass is computed, and taken as a reference point, see Figure 4.5b. Then, the closest wall entities to the centroid of the region in the *wg* are retrieved. For each couple of walls close to a centroid, the respective lines in the original image graph, obtained after vectorization, are found. Then, a path between the two walls is optimally searched using A*, see Figure 4.5c.

There are mainly two reasons that explain the use of A*. First, we need an efficient search algorithm under the consideration that multiple paths between two wall nodes are possible, but only few of them are of real interest. A* is a path finder algorithm which is optimal when an appropriate monotonic heuristic is used. Second, we need to define an extra-cost of traversing nodes which are not candidates of being door vectors according to the areas of interest. This extra-cost can be easily added to the already traversed path at a certain point in A*.

Assume that we have detected two walls which are sufficiently close to a centroid that defines an area of interest. We consider arbitrarily one wall to be the starting node s and the other to be the goal node q. Then, the heuristic considered as the expected path distance from any node n to q is the Euclidean Distance:

$$h_q(n) = d(n\ \ q) \tag{4.1}$$

Since the distance from a node m to itself is 0, then the triangular inequality is

[2]In the rest of this section, all the process explained for *door* detection is also valid for *window* detection. However, we will only refer to doors for clarity and to avoid unnecessary repetitions.

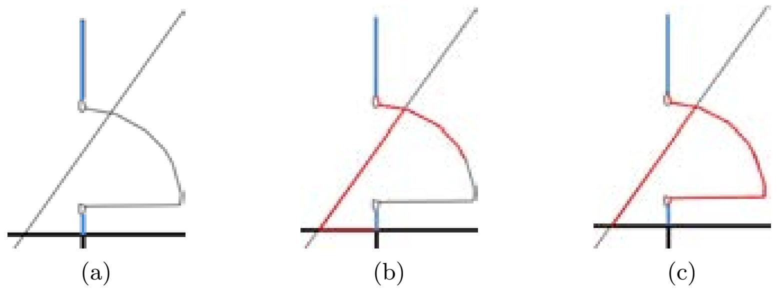


Figure 4.6: A problematic situation is shown in (a) for finding door lines between the blue wall candidates. In this case a ceiling line traverses the door symbol. The nodes expanded (red) by a pure implementation of A* algorithm in (b) shows that the final retrieved path does not traverses the complete door lines. Contrarily, in (c), additional cost for traversing wall nodes is added, and the final retrieved path is correct.

fulfilled:

$$h_q(n) \leq d(n\ \ m) + h_q(m) \tag{4.2}$$

where m is any adjacent node to the actual node already explored n. The equation 4.2 implies that h is monotonic and thus, the search is optimal.

The goal function to be minimized at each certain node n in the search is defined by the summation of the real cost of the traversed path $g(n)$ and the expected distance to the goal $h_q(n)$:

$$f_q(n) = g(n) + h_q(n) \tag{4.3}$$

The cost function $g(n)$ is given by summation of the cost traversed till its father p, and its own length $\ n\ $. Nevertheless, an extra-cost is given when crossing over those nodes which are not in the area of interest or are already labeled as walls. We define it recursively as:

$$g(n) = \begin{cases} g(p) + (\ n\ * W) & \text{if } n \quad N_{interestarea} \quad N_{wall} \\ \\ g(p) +\ n & \text{otherwise,} \end{cases}$$

where W is an heuristically defined cost. This extra-cost pushes the algorithm to prioritize the search on nodes which are door candidates and allows to avoid problematic situations as the one shown in Figure 4.6. In addition to that, a experimentally defined threshold allows a maximum number of node expansions to keep the memory use under control. This is of a great importance when there is not a real path between two walls.

Finally, for each resulting connection between walls, a virtual node is added to the wg with the respective attribute; door or window. This process is shown graphically in Figure 4.5d. The resulting graph, since it contains nodes attributed as walls, doors, and windows, is now called wdw-graph.

Wall well-aligned connections

The loss of any door or window entity at this point is a critical issue for the later room detection. Rooms are detected by finding closed regions in a wdw-graph. Therefore, when a door or a window is lost, the supposed room formed by these elements is lost. For this reason, a post-process to reduce the impact of losing any of these elements is carried out.

This process firstly looks for couples of walls that have a geometric gap between them and are sensibly well-aligned in orientation; the tolerance on both, gap distance and orientation angle, is experimentally learned from the ground-truth. A path among each of these couples of walls is searched using the A*, as explained in 4.2.2, but with a slightly modified cost function $g(n)$. Now, an extra-cost is given only to that nodes which already belong to walls. It is defined recursively as:

$$g(n) = \begin{cases} g(p) + (\ n\ * W) & \text{if } n \quad N_{wall} \\ \\ g(p) +\ n & \text{otherwise} \end{cases} \tag{4.4}$$

If a path exists, a new node of type *connection* is added to the graph and connected to the two correspondent wall terminals. The use of this technique not only results in a better room detection, but also helps on finding abstract boundaries between rooms that have no physical separation. The final graph is called *wdwcg*.

Room detection

Finally, closed regions are found from the plane *wdwcg* using the optimal algorithm from Jiang et al. in [61]. Before applying the algorithm, all the terminals of the graph are erased recursively. This leads to a better computation of the closed regions as unnecessary terminal paths are not taken into account when searching for closed regions. After obtaining the regions, their area is calculated and used to rule out impossible rooms regarding their absolute size, such as small regions representing holes for pipes in the plan.

The room information is introduced into the *wdwcg* to create the output representation of the analyzed floor plan. Hence, the attributed graph incorporates both, the complete set of architectural elements encountered and their structural dependences. This high-level document representation carries on information that could be of special interest for many applications. Techniques such as graph traversing can be used to extract neighbor or accessible rooms, and graph matching to retrieve similar floor plans regarding their structure.

4.3 Syntactic analysis of graphical documents

In the previous Section 4.2 we have presented a method for the extraction of the structure of floor plans that is able to cope with multiple graphical notations. Yet, as in most of the literature approaches dealing on this framework, the domain knowledge (the syntax) of the graphical documents is specifically defined by the recognition task.

This fact produces that, even if it can easily adapt to recognize other linear documents such as flowcharts, the contextual information of interest needs to be redefined for every new visual syntax.

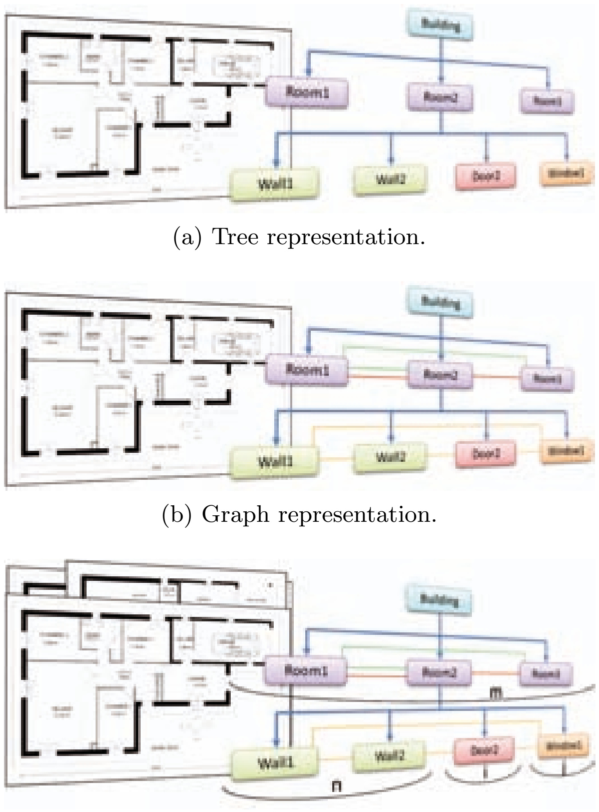
In this section we propose the adoption of a syntactic model to overcome this problem. In Section 2.3 we have already explained how the syntactic recognition has been used in the literature to solve multiple CV problems. Here we adopt an attribute graph grammar as unified framework for representing the information conveyed in documents, to automatically learn their visual syntaxes, and to use this knowledge to favor a better recognition.

Attribute graph grammars have a high representative power. They not only allow to represent images or documents hierarchically, but they include structural attributes and dependences among the existing elements. This complex representation can be learned from annotated data by a probabilistic model embed in the grammatical formalism. Then, when the model learning is combined with appropriate parsing techniques, the system retrieves that representation of the input that betters fits into the syntax.

In this section we firstly introduce the grammatical formalism used to represent and interpret architectural documents. This definition is adapted into a preliminary syntactic model that allowed us to study the feasibility of interpreting floor plans by means of a grammatical formalism. It is composed of two production rules, unary object attributes, and predefined contextual relations. Finally, we increase the original model complexity into a grammar that models the contextual relations using Probabilistic Graphical Models (PGMs). It worth to remark that, for a given input, the proposed contributions initialize the graph representation in order to relax the complexity of the inference process. This initialization entails the use of appropriate object recognition methods to break the semantic gap between image pixels and the grammatical symbols. In the first approach, a very ad-hoc extraction aims at almost perfect graph initialization. Contrarily, the second method initializes the graph using a more naive hypothesis of the complete structure of the document, giving more responsibility to the inference process. At the end, both methodologies output the floor plan contextual representation in an attributed graph that includes the hierarchical, structural, and semantic relations among the architectural elements.

4.3.1 The floor plan attributed graph grammar

The first point to discuss is how to represent the structure of the floor plans. The main architectural elements in these documents are those that model the structure of a building, viz. rooms, walls, doors, and windows. Therefore, we can model the structure of a floor plan using a hierarchical composition of those elements, see Figure 4.7a. In this model, a *building* is composed by a definite number of *rooms* and, at the same time, a *room* consists of a set of terminals that enclose their space, this are *walls, doors*, and *windows*. The expressiveness of this tree representation can be augmented by converting it into an attributed graph. In this graph, the nodes contain attributes that enclose specific features of the different architectural objects, and the attributed edges allow to constrain structural and semantic relations among them, see Figure 4.7b. Therefore, it is easy to see that every floor plan can be converted



(a) Tree representation.

(b) Graph representation.

(c) Graph grammar representation.

Figure 4.7: Representation models for floor plans.

into a unique attributed graph representation, where every input has a different set of nodes and edges with specific attribute values. All these graph representations can be generated by an attribute graph grammar in a similar manner as the And-Or graph representation is used to represent natural images [108], see Figure 4.7c. Here, And nodes are the architectural elements whereas the Or nodes represent the possible configurations of this elements; the unknown numbers *m,n,i* and *j*.

Our attribute graph grammar for floor plan interpretation $\mathcal{G}$ is composed by the following 5*tuple*:

$$\mathcal{G} = (V_N, V_T, R, S, P), \tag{4.5}$$

where:

- V_N is a finite set of non-terminal symbols, conventionally denoted by capital letters $V_N = \{A_1, A_2, ..., A_m\}$. In our case the building and the rooms.
- V_T is a finite set of terminal symbols, conventionally denoted by lowercase letters $V_T = \{w_1, w_2, ..., w_m\}$. These are the walls, doors, and windows.
- S is the starting symbol $\in V_N$: the building.

- R is the set of production rules that allow to derive a set of elements in their sub-elements: $R = \quad r_1 \; r_2 \qquad r_N$ and are of the form $r = \alpha - \quad \beta$, where $\alpha \; \beta \qquad V_T \bigcup V_N \;\; ^+$.

Additionally to this definition, each node V in G has a set of attributes $X(V)$ that can make reference to relative spatial, photometric, and functional characteristics. The attributes are divided in two groups: synthesized and inherited. Inherited attributes down from the parent nodes of the parse tree, meanwhile, synthesized are the result of the attribute evaluation rules:

$$r : \alpha \qquad \beta \tag{4.6}$$

$$f_i\big(X(\beta)\big) = X(\alpha) \tag{4.7}$$

$$g_i\big(X(\alpha)\big) = X(\beta) \tag{4.8}$$

Definition 1. *The parse graph G generated by $\mathcal{G}$ is the graph-structured representation of a possible derivation from the root node S by a sequence of production rules $r \subseteq R$.*

Definition 2. *A configuration of the grammar, denoted as $\mathcal{C}$, is the set of terminal nodes deterministically generated by a parse graph G:*

$$\mathcal{C} = \quad (w_i \;\; x(w_i)) : w_i \quad V_T \;\; i = 1 \;\; 2 \qquad K \tag{4.9}$$

We call the grammar to be ambiguous if one configuration can be derived from multiple parse trees.

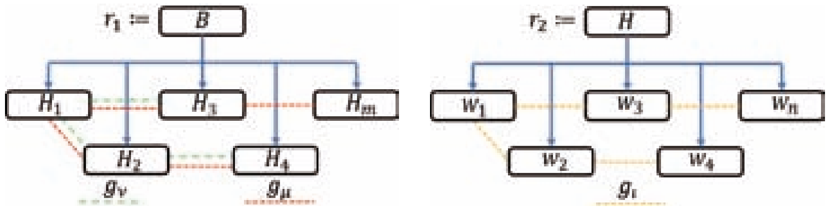
Definition 3. *The language of the grammar, denoted as $\mathbf{L}(\mathcal{G})$, is the set of all valid configurations that can be derived in a finite number of steps from S.*

$$\mathbf{L}(\mathcal{G}) = \quad \mathcal{C} : S \overset{R^*}{-} \quad \mathcal{C} \;\; n = 1 \;\; 2 \qquad N \tag{4.10}$$

We subsequently overview two adoptions of $\mathcal{G}$ to extract the structure of floor plans.

4.3.2 Defining the structural context: Syntactic-0

This adaptation of the graph grammar is composed by only two production rules and a probabilistic model that accounts for both, element frequencies and local attribute information. The parsing strategy is a two step bottom-up and top-down strategy. The bottom-up step constructs a probable And-graph configuration of the plan from a combination of methods for the object extracting the primitives. Then, the top-down step prunes that graph according to the grammar rules and outputs the most probable interpretation according to the probabilistic models embed. In the following, we first define the model, these are the production rules and attributes that are defined in the grammar. We secondly explain how we learn the probabilistic model in $\mathcal{G}$. And we finally overview the parsing strategy.



(a) $r1$ (b) $r2$

Figure 4.8: Rules for Syntactic-0. Rule $r1$ derives a Building into M rooms. Rule $r2$ derives a Room into N primitives.

Model definition

The graph grammar is composed by two production rules that allow to derive hierarchically and structurally a floor plan:

- r_1 defines a building B as an arrangement of rooms H that are contextually related in terms of neighborhood and accessibility, see Figure 4.8a. The neighborhood graph $g_\nu = (H, E_\nu)$ and the access graph $g_\mu = (H, E_\mu^B)$ define the structural connectivity among these rooms. When two rooms H_i and H_j share at least one primitive in their decomposition, they are called to be *neighbors* $(H_i, H_j) \in E_\nu$. Additionally, when at least one of the shared primitives is a door or a separation, they are *accessible* $(H_i, H_j) \in E_\mu^B$. Figure 4.8b shows the room connectivity for a possible floor plan derivation. This rule imposes that both, g_μ and g_ν are connected graphs –all the nodes are reachable from any other node in a finite number of traversing steps–.
- r_2 defines H as a closed environment of a set of architectural primitives w; these are walls, doors, and windows. These primitives are related in terms of incidence according to the undirected graph $g_\iota = (w, E_\iota)$. Two primitives are *incident* $(w_i, w_j) \in E_\iota$ whether their segments intersect at some point of the image. This rule is visually illustrated in Figure 4.8b.

We define two attributes for the rooms that will be used in the recognition to asses the confidence of the detections. These are their area $\varepsilon(H)$ and a perimeter $\rho(H)$, and are calculated from the planar configuration of the graph $g_\iota = (w, E_\iota)$ that composes every room.

Model learning

The probabilistic model for this grammar is defined on the parse graph derivations. It accounts for the frequency of the elements and the consistency of the attributes in G:

$$p(G) = p_{\mathrm{or}}(G)p_{\mathrm{x}}(G), \tag{4.11}$$

where:

- p_{or} models the frequency of the production derivations. This is the number of rooms that a building derives into, and the number of primitives that surround every room:
$$p_{\text{or}}(G) = p\big(\eta(B)\big)\frac{\sum_{H_i \quad G} p\big(\eta(H_i)\big)}{\eta(H_i)} \tag{4.12}$$
Where $\eta(A)$ is the number of children of A. Since there is a strong relation between $p\big(\eta(A)\big)$ w.r.t. the floor plan area, this model is normalized by the plan area to adapt to different building sizes and types.

- p_{x} models the room areas $\quad(H)$ and $\nu(H)$:
$$p_{\text{x}}(G) = \frac{1}{2\eta(B)} \sum_{H_i \quad G} p\big(\quad(H_i)\big) + p\big(\nu(H_i)\big) \tag{4.13}$$
All, $p\big(\quad(H_i)\big)$, $p\big(\nu(H_i)\big)$, and $p\big(\eta(A)\big)$ are pdfs modeled by Gaussian Mixtures estimated by the EM algorithm.

Model recognition

The recognition of an input document consists of finding that G that better fits into the model learned, i.e. that G that maximizes $p(G)$:
$$\mathbf{G} = \underset{G}{\operatorname{argmax}}\, p(G) \tag{4.14}$$

We have designed a parsing strategy that constructs a preliminary representation of the floor plan from already extracted primitives. This construction is set in a bottom-up manner; the contextual relations among the symbols are synthesized from the early detected nodes and relations. Then, in a *top-down* manner, multiple parse graphs are generated by considering different configurations agreeing with the grammar definition. Finally, we keep that parse graph that maximizes the probabilistic model.

Bottom-up parsing

The bottom-up parsing process constructs a graph representation of the floor plan from the terminal symbols to S. At each step, the parser constructs an abstraction level of the tree representation. Then, their contextual graphs are synthesized to construct the upper level. In the following we explain the process of constructing each abstraction level.

1. **From pixels to V_T:** This process of initializing the grammar primitives is performed by three different symbol recognition strategies, one for each possible class $c =$ wall, door, window . The reason of using such a specialized methods aims at obtaining and accurate primitive detection that facilitates the subsequent inference steps. For the detection of walls and doors we have adopted the same strategies as in [72]. Walls are detected on the coupling of Hough

transform and the image vectorization and doors are encountered by identifying arcs in the image [93]. Very differently, due to existing graphical variability in windows symbols, we have adopted the patch-based recognition strategy explicated in Section 3.4. This approach allows us to learn automatically the different window symbols instead of designing multiple ad-hoc strategies for the different variations. Moreover, this process has been enhanced a posteriori by taking advantage the contextual dependences between windows and walls. We introduce the wall information into the pact-based window segmentation to create a graph of neighbor patches. The Conditional Random Field CRF implementation of [49] is defined over this graph to enforce the spatial consistency between walls and windows. This graphical model computes the best graph representation by minimizing the energy of the graph class assignments c using the graph-cuts algorithm [23]:

$$- \log(p(c \; g; k)) = \sum_{s_i} \quad (c_i \; s_i) + k \sum_{(s_i \; s_j) \;\; g} \sigma(c_i \;\; c_j \; s_i \;\; s_j) \tag{4.15}$$

where   is a factor accounting for the unary potentials of the patch class assignments, and σ is a factor that bears the spatial consistency between neighbor patches and their assignments. Both factors are learned from training instances. Once the primitives are detected, we construct the graph g_ι by analyzing those objects that overlap at some point or that are close enough to consider them as incident.

2. **From V_T to H**: Before starting to find closed environments of walls, doors, and windows in g_ι to find possible rooms, we introduce to the graph an abstract component named *separation*. These elements connect relatively close and well aligned walls. Our intention is two-fold. On the one hand we want to over-recognize rooms; some of them may be lost when windows and doors have not been correctly detected. On the other hand, we want our system to detect rooms that are not physically separated, e.g kitchenettes: where the kitchen and the living-room share a common space. Then, we adopt an optimal algorithm to find regions in a planar graph [61] to detect the rooms in g_ι. This algorithm not only permits to recognize the rooms but we can also extract important structural information, e.g. the elements that are part of the external borders of building and its entrance rooms. Finally, the extracted room connectivity gives rise to the neighbor and accessibility graphs $g_m u$ and $g_n u$.

3. **From H to B**: r_1 is used to synthesize the building structure from the room configuration. At the end of this step a preliminary representation of the plan is constructed, including hierarchical, structural and semantic information of the floor plan.

Top-down parsing

A top-down parsing strategy generates, from the bottom-up process, multiple graph representations that are consistent to the grammar. A *structural pruning* is applied on those rooms that do not satisfy the structural conditions in r_1. Furthermore, a *probabilistic analysis* generates multiple graph instances according to probabilistic scores. At the end, the parser selects that valid graph representation with highest probability:

- *Structural pruning*: Some of the building representations constructed while parsing the document are highly inconsistent according to the grammar specification. In this stream, r_1 specifies that all the rooms should be connected in terms of neighborhood and accessibility. To fulfill this rule, the parser checks the graphs g_ν and g_μ connectivity using the Dijkstra shortest-path algorithm. From each unconnected component a new graph instance is created by keeping the parent and the offspring of the rooms.
- *Probabilistic pruning*: For each one of the rooms $H \quad G$ we calculate its p_x. When this probability is close to 0, we generate a new G that is the result ruling out that room from G and all its single parent offspring. This process is repeated for every room in G.

When the inference process finishes, the floor plan representation obtained includes the architectural objects, their mutual dependences, and a probability associated.

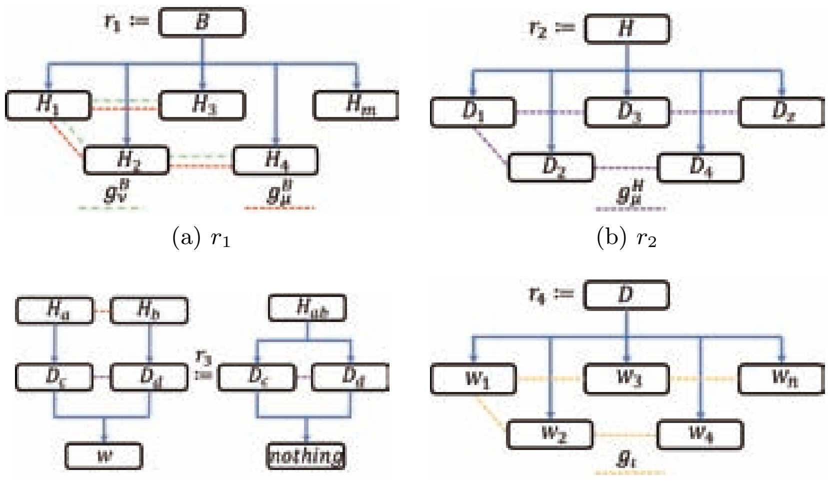
4.3.3 Learning the context: Syntactic

The previous methodology uses the structural information in three different manners. Firstly, it is defined at the grammatical rules, e.g a building is composed of accessible rooms. Secondly, attributes allow to assess the reliability of the room hypothesis. Finally spatial dependences are learned by a graphical model to enhance the initialization of the graph. Still, that syntactic approach does not learn any structural relation among the architectural symbols that is considered at inference time. In this following method we do learn structural context to favor a better interpretation. This fact permits to initialize the graph in a more naive fashion, giving more reliability to the recognition process. In the following we explain each module in grammatical formalism separately.

Model definition

This grammar represents hierarchically and structurally the architecture of a floor plan in similar manner to the previous syntactic approach in Section 4.3.2. Here, the root node is the *building* and the terminals symbols are *walls*, *doors*, *separations*, and *nothings* we explain below why a nothing element is added into the model . The grammar definition has two extra production rules that give rise to the concept of *domain*. The domains are physical regions of the image that enclose the different spaces of the rooms. Our intention is to permit the grammar to atomize the composition of rooms and predict which image spaces belong to a certain room and which not. Let s review the rules, which are visually represented in Figure 4.9:

- r_1 is very similar to the first production in the previous Section, which allows to derive building B into an arrangement of rooms H. Here, the rooms are also



(a) r_1 (b) r_2

(c) r_3 (d) r_4

Figure 4.9: Rules for Syntactic.

related in terms of neighborhood and accessibility according to the graphs g_ν^B and g_μ^B. Yet, this connectivity is not set in the rule definition but it is learned from the GT.

- A room can be divided into multiple domains D by r_2. The domains in G are also connected in terms of accessibility g_ν^H. When D_i and D_j share a *door* and a *separation* in their derivations they are considered *accessible* $(D_i, D_j) \in E_\nu^H$. Moreover and differently to r_1, two domains are also accessible when they share a *nothing* among their offspring.

- Two accessible rooms H_1 and H_2 consisting of two or more accessible domains D_1 and D_2 can be merged together into a new room. This rule can be applied recursively to merge several domains into a single room. As it will be explained later, r_3 allows to infer the final spatial configuration of the rooms. It is worth to remark that when this production is applied, the label of the shared primitive relabeled into *nothing*.

- Finally, r_4 is similar to r_2 in the previous approach. It defines how D is enclosed into a set of architectural primitives w. These primitives are related in terms of incidence according to the undirected graph $g_\iota = (w, E_\iota)$. Two of them are considered *incident* $(w_i, w_j) \in E_\iota$ whether their segments intersect at some point of the image.

The list of attributes for the different nonterminal nodes and the associate equations to the rules they participate in are summarized in Tab. 4.1. All the nonterminal nodes have a common attribute $\eta(A)$ stating for their number of children. Additionally, each nonterminal node has its own attributes accounting for specific structural characteristics. For the building, $\nu(B)$ and $\mu(B)$ are defined on the neighborhood and

Attributes	Nodes	Rules	Equations
η	Δ_N	$r_1\ r_2\ r_4$	$\eta(A) = \#\mathrm{Ch}(A)$
κ	H	r_3	$\kappa(H_i\ H_j) = \begin{cases} 1 & \text{if } (H_i\ H_j) \quad E_\nu^B \\ 0 & \text{otherwise} \end{cases}$
ν	B	r_1	$\nu(B) = \begin{cases} 1 & \text{if } g_\nu^B \text{ is connected} \\ 0 & \text{otherwise} \end{cases}$
μ	$B\ H$	$r_1\ r_2$	$\mu(A) = \begin{cases} 1 & \text{if } g_\mu^A \text{ is connected} \\ 0 & \text{otherwise} \end{cases}$
	$B\ H$	$r_1\ r_2$	$(A) = \sum_{A \leftarrow \beta} \ (\beta)$
ι	D	r_3	$\iota(D) = \sum\limits_{(w_i,w_j)\in E_\iota} \sum\limits_{s,t\in L} f_s(w_i) f_t(w_j);\ f_s = 1_{\{w_i = s\}}$

Table 4.1: Attributes, nodes, rules and equations

access connectivity of the graphs g_ν^B and g_μ^B respectively. Similarly for the rooms, $\mu(H)$ is calculated over the connectivity of the graph g_μ^H. Both, the building and the rooms have a common attribute that captures the area they cover in the image: (B) and (H). A domain is associated to the attribute $\iota(D)$ which bears the pairwise suitability for the node labellings in g_ι. This is, for instance, how likely to find a door incidentally related with a wall or a separation is.

The attributes for the terminal nodes $X(w) = \ \gamma(w)\ \delta(w)$ include their longest component of their bounding box $\gamma(w)$, and the output of the primitive detection method $\delta(w)$. Both, $\gamma(w)$ and $\delta(w)$ are extracted from the image.

Model learning

This grammatical formalism not only encodes the hierarchical configuration of the objects and their attributes, but it also learns the object contextual dependences and bears the output of the primitive detectors in the model. Thus, the probability of a parse graph G given a floor plan $\mathcal{I}$ is defined as:

$$p(G\ \mathcal{I}) = p(\mathcal{I}\ G)p(G) \tag{4.16}$$

where $p(G)$ is the prior probability of a parse graph and $p(\mathcal{I}\ G)$ is the likelihood probability of the given image $\mathcal{I}$. We have modeled both probabilities by PGMs.

Prior probability

According to Definition 1, G consists of non-terminal nodes, the production rules used in its derivation, and the attribute functions associated to the rules. Thus, we can define the prior probability of G as $p(G) \triangleq p(r\ X\ \Delta_N)$ where, r, X, and Δ_N are the set of all production rules, the attribute functions, and non-terminal nodes in G, respectively. Nevertheless, since the attributes already model the frequencies of the productions $\;\nu$ for r_1, r_2, and r_4, and κ for $r_3\;$, the prior probability is defined on the nonterminal nodes and their associated attribute functions $p(X\;\;\Delta_N)$. Thus, the parse graph models a PGM, where the attribute functions provide structural dependences among the architectural symbols in G. Considering the Hammersley-Clifford theorem [53], which states that a PGM factorizes by the product of factor functions defined over maximal cliques, we define the prior probability as:

$$p(X\Delta_N) = \frac{1}{Z}\prod_{c\;\;C}\sigma_c(X_c\;\;A_c) \tag{4.17}$$

where Z is the partition function, C is the set of maximal cliques in G, and $\sigma_c(X_c\;\;A_c)$ accounts for the interactions between the non-terminal nodes and the attribute functions in the maximal cliques c. When we extend this equation for all the attribute functions in G, the prior probability is defined as:

$$\begin{aligned} p(\mathrm{G}) = \frac{1}{Z}\;&\sigma_\eta(\eta\;\;B)\sigma_\nu(\nu\;\;B)\sigma_\mu(\mu\;\;B)\sigma\;(\quad B)\cdot\\ &\prod_{H_i}\sigma_\eta(\eta\;\;H_i)\sigma_\mu(\mu\;\;H_i)\sigma\;(\quad H_i)\cdot\\ &\prod_{D_j}\sigma_\eta(\eta\;\;D_j)\sigma_\iota(\iota\;\;D_j)\end{aligned} \tag{4.18}$$

The reader may notice that the factor for κ does not appear in the prior model. This is because, as we explain in details below, the probability of merging every pair of accessible rooms is equiprobable. The parsing strategy is the responsible of choosing which pair of rooms are merged at every step. Moreover, the factor $\sigma_\iota(\iota\;\;D_j)$ is the pairwise domain suitability between the connected primitives in D_j, and can be rewritten as:

$$\sigma_\iota(\iota\;\;D_j) = \prod_{(w_t\;\;w_u)\;\;E_\iota^{D_j}}\sigma(w_t\;\;w_u) \tag{4.19}$$

Then, we estimate the marginals and the partition function in (4.18) by the approximation of Kikuchi [55] and the prior probability of the graph is:

$$\begin{aligned} p(\mathrm{G}) \approx\;& \frac{p(\eta\;\;B)p(\nu\;\;B)p(\mu\;\;B)p(\quad B)}{p(B)^3}\cdot\\ &\prod_{H_i\;\;G}\frac{p(\eta\;\;H_i)p(\mu\;\;H_i)p(\quad H_i)}{p(H_i)^2}\cdot\\ &\prod_{D_j}p(\eta\;\;D_j)\frac{\prod_{w_t\;\;w_u}p(w_t\;\;w_u)}{\prod_{w_t}p(w_t)^{\#\mathrm{nb}-1}}\end{aligned} \tag{4.20}$$

where #nb is the number of neighbors of w_t. Finally, the approximated prior probability for G is:

$$\begin{aligned} p(\mathrm{G}) \approx\ & p(\eta \ B)p(\nu \ B)p(\mu \ B)p(\ \ B)p(B)\cdot \\ & \prod_{H_i \ \ G} p(\eta \ H_i)p(\mu \ H_i)p(\ \ H_i)p(H_i)\cdot \\ & \prod_{D_j} p(\eta \ D_j)p(D_j)\frac{\prod_{w_t \ w_u} p(w_t \ \ w_u)}{\prod_{w_t} p(w_t)^{\#\mathrm{nb}-1}} \end{aligned} \tag{4.21}$$

All the model parameters has been learned using the ontological knowledge explicated in Chapter 5. In the following we will explain how they have been estimated:

- $p(\eta \ A)$ is the likelihood for a Poisson pdf on the number of children for each node:
$$p(\eta = k \ A) = \frac{\lambda_A^k e^{-k}}{k!} = \frac{1}{k!} \tag{4.22}$$
where λ_A is the children average for a node type A, $k = \eta(A)$.
- $p(\nu \ B)$, $p(\mu \ B)$, and $p(\mu \ H)$ are learned from sampling over the global graphs connectivity of the children of B and H.
- $p(\ \ A)$ is a Gaussian mixture pdf, learned employing the EM algorithm, on the area of each type of node A.
- $p(w_t \ \ w_u)$ is learned from the GT.

Likelihood probability

The likelihood probability $p(\mathcal{I} \ G)$ is formally defined as the conditional probability of an object image $\mathcal{I}$ of being generated by a parse graph G. In practice, $\mathcal{I}$ is composed of a set of attributes x directly extracted from an image and $p(\mathcal{I} \ G)$ is a pdf linking these features with the terminal nodes of the parse graph. Thus, we define $p(\mathcal{I} \ G) \triangleq p(x \ w)$ as:

$$p(x \ w) = \frac{p(w \ x)p(x)}{p(w)} \tag{4.23}$$

where $p(x)$ is constant in the graph parsing, and $p(w \ x)$ estimates the domain of w given the attribute values $\gamma(w)$ and $\delta(w)$. It is learned using a multinomial logistic regression.

Model recognition

Again, we address this problem from a probabilistic point of view. Thus, the *understanding* problem is defined as a MAP inference given the input image.

$$G = \underset{G \ \ \mathcal{G}}{\mathrm{argmax}}\, p(G \ \mathcal{I}) = \underset{G \ \ \mathcal{G}}{\mathrm{argmax}}\, p(\mathcal{I} \ G)p(G) \tag{4.24}$$

In order to find this graph we propose a greedy search algorithm that involves three consecutive steps: a first bottom-up graph initialization, a top-down verification

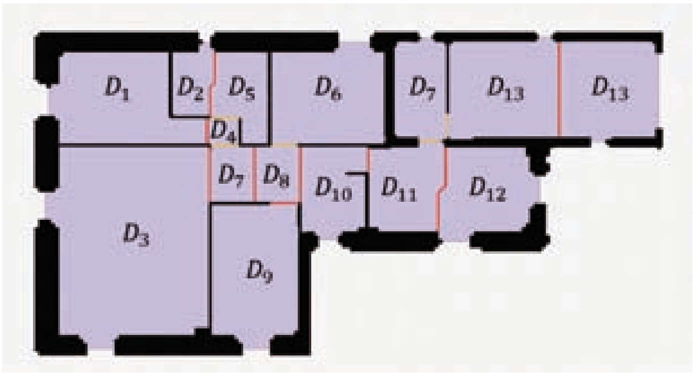


Figure 4.10: Watershed applied on the room structure. Rooms are in black, doors in red, and separations in orange. The original image is shown in Figure 4.11a

using r_4, and an iterative application of r_3 and r_2 to find the best room configuration. We explain these steps separately.

Bottom-up recognition

In this approach we initialize the graph using a more uninformed method than in Section 4.3. Here, the wall detector explicated in Section 3.5 is used to extract the walls from the image. This methodology uses general wall characteristics in the recognition and does not need any learning step to be adapted to every graphical notation. Once the wall symbols are initialized, the image is oversegmented into multiple domains using the watershed image transformation. The building can be seen as a topographic relief consisting of high mountains (walls) and basins (white spaces in between them). The negative distance transformation on the wall image produces several local minima at the farthest pixels of the segments. We settle the initial markers at this minima and apply the watershed transformation, which produces an excessive set of connected regions with different labels, see Figure 4.10. These regions are separated by the *wall* segments and watershed lines.

Top-down recognition

The production r_4 states that the domains consists of as set of walls, doors, and separators incidentally connected. Therefore, we use an explicit process for the detection of walls and doors on the watershed lines that separate two domains. The recognition of walls is realized by decreasing the sensitivity thresholding on the results obtained by the wall detector at the bottom-up step. And doors are detected by finding arcs using the Hough transform over the original image. When no walls and doors are detected, the watershed lines are considered *separations*. At the end of this top-down recognition, the incident graph g_ι over the primitive symbols is generated.

Iterative search for the best representation

The iterative search starts by synthesizing the attributes up to the starting symbol B. This procedure involves r_4, r_2, and r_1 for initializing the constrains in the nodes and a complete parse graph representation G for the input image. Then, all the matchings of $r3$ LHS on G are encountered and stored in a list M_{r_3} of merging candidates. For all these matchings, we only keep that graph with the highest posterior probability after the derivation. If this probability is higher than $p(G\ \mathcal{I})$, the corresponding derivation is adopted as new G. If not, the algorithm concludes. This process is detailed in the Algorithm 2.

Algorithm 2 Iterative graph search

```
$P'$    0
$P$    $p(G\ \mathcal{I})$
$M_{r3}$    matching(LHS($r3$) $G$)
while $size(M_{r3}) > 0$ do
   $P'$    $P$
   list$_P$        , list$_G$
   for all $m$    $M_{r3}$ do
      $G'$    $r3(G\ m)$
      add(list$_G$ $G'$)
      add(list$_P$ $p(G'\ \mathcal{I})$)
   end for
   $P$    max(list$_P$)
   if $P > P'$ then
      $G$    list$_G$max(list$_P$)
      $M_{r3}$    matching(LHS($r3$) $G$)
   else
      break
   end if
end while
$G$    $G$
```

4.4 Experimental Evaluation

In this section, we analyze the performance of the presented methods for room detection in floor plans. As explained before, the rooms carry the structure of the buildings and, since they are not graphical symbols but concepts, they need to be recognized after a structural analysis. We firstly explain the evaluation protocol adopted. Then, we analyze the results obtained for the three contributions proposed in this chapter. We finally discuss and contextualize the results by comparing them with recent methods on part of the same database.

4.4.1 Evaluation Method for Room detection

We based the performance evaluation for room detection on the protocol of Phillips and Chhabra [83], which was first introduced in this framework in [72]. These protocol searches for the best alignment between the rooms segmented and the ones in the GT and allows to report the *exact* and *partial* matches.

First of all we create a *match_score* table where rows represent the rooms segmented by the system and columns are the rooms in the GT. Each table position $(i\ j)$ specifies the overlapping between the segmented room i and the groundtruthed room j. It is calculated as:

$$match\_score(i\ j) = \frac{area(d[i] \bigcap g[j])}{\max(area(d[i])\ area(g[j]))} \tag{4.25}$$

In the *match_score* table, a *one2one*/exact match is given when the overlapping score in $(i\ j)$ overcomes an *acceptance threshold*, and the rest of the row and column are below a *rejection threshold*. This means that the room segment i matches with groundtruth room j and does not match with any other. Then, the partial matches are calculated as it is described in [83] and they are divided into the following categories:

- *g_one2many*: A room in the ground truth overlaps with more than one detected rooms.
- *g_many2one*: More than one room in the ground truth overlaps with a detected room.
- *d_one2many*: A detected room overlaps with more than one room in the ground truth.
- *d_many2one*: More then one detected rooms overlap with a room in the ground truth.

Finally, the detection rate (DR), the recognition accuracy (RA), and the *one2onerate* are calculated as follows:

$$DR = \frac{one2one}{N} + \frac{g\_one2many}{N} + \frac{g\_many2one}{N} \tag{4.26}$$

$$RA = \frac{one2one}{M} + \frac{d\_one2many}{M} + \frac{d\_many2one}{M} \tag{4.27}$$

$$one2one\ rate = \frac{one2one}{N} \tag{4.28}$$

where N and M are the total number of ground truth and detected rooms, respectively.

4.4.2 Results on room detection

In Table 4.2 we show the quantitative results of the three proposed methods for in the different datasets specified in Chapter 6. This table also compares this methods with some approaches proposed in the literature. In the following, we analyze separately the results obtained for each of the proposals.

Structural method

Quantitative examples for this system are shown in Figures 4.11b, 4.12b, 4.13b, and 4.14b. Moreover, we illustrate in Table 4.2 the quantitative results compared with the rest of the methods. At a glance, we can verify that the performance obtained in terms of the detection rate (DR) is practically perfect in all the datasets always over 90% . This fact indicates that the vast majority of the room instances are detected; regardless of the notations of the floor plan documents. Very differently, the recognition accuracy (RA) strongly varies depending on the dataset. For the Black dataset the RA is very high, substantially outperforming the rest of the methods. Yet, for the rest of the datasets, it considerably decreases as a result of a worse detection of the walls. In the Textured, Textured2, and Parallel datasets the notation is more challenging than in the Black dataset. This leads to the wall detection approach to produce several false positive instances on the outskirts of the building boundaries. Then, the intelligent search finds alignments between these walls producing false room hypothesis. Moreover, this fact not only lowers the RA but it also increments the oversegmentation of the rooms. This fact is clearly seen in Figures 4.13b and 4.14b and corroborated by the -1 score in these datasets.

Syntactic-0

The syntactical mode Syntactic-0 has been designed as a preliminary approach to study the feasibility of applying syntactic models for floor plan understanding. Furthermore, the notation-specific methodology for the wall extraction determines the applicability of the interpretation. It can only be applied on floor plans of the Black dataset. The learning of this model is carried out following a leave-one-out strategy, i.e. using one image for testing and the rest for learning the parameters at each time. The quantitative results are shown in Table 4.2 and qualitative example is shown in Figure 4.11c. It is interesting to analyze the results by closely comparing them with [72] as both methods are constructed from the same extraction of walls and doors. As we can observe, the results in DR practically the same. Yet, our method slightly outperforms the original approach in terms of RA. This small difference is produced by the top-down inference, where inconsistent rooms, such as non accessible spaces or unprovable, are ruled-out producing better interpretation hypothesis.

Syntactic

We show in Figures 4.11d, 4.12c, 4.13c, and 4.14c visual examples of the performance of Syntactic in the different datasets. Syntactic is a grammatical formalism that models probabilistically the representation of the floor plans using PGM s. The parameters of this model are learned using a leave-one-out strategy for the complete collection of images. This is that for every test image, the rest 121 are used for learning. It is worth to notice that independently of the testing image, we train using images of different resolution and notations. Our aim is to study the robustness of the method when no there is not prior knowledge on the type of image.

We analyze the results of the Syntactic in each dataset separately to draw a final conclusion of its overall performance. In the Black dataset the interpretation it is almost perfect. It outperforms the rest of the approaches in RA while maintaining a DR very close to the Structural method. The reason of this hight performance is a good graph initialization combined with an appropriate modeling of the floor plan structure. When we focus on the results for the Textured Dataset, the DR is significantly higher than the Structural. Contrarily, the RA drops for the same cause it also drops for the Structural method. The reason is that, since the detection of the walls it is more challenging in this dataset, the untrained method several false positives instances that adversely affect the initialization of G. This problem extrapolates also for the Textured2 and Parallel datasets. Even though the DR results are more than satisfactory, the multiple rooms generated out from the building boundaries decrease the RA score. Clear examples of this problem are shown in Figures 4.13c and 4.14c.

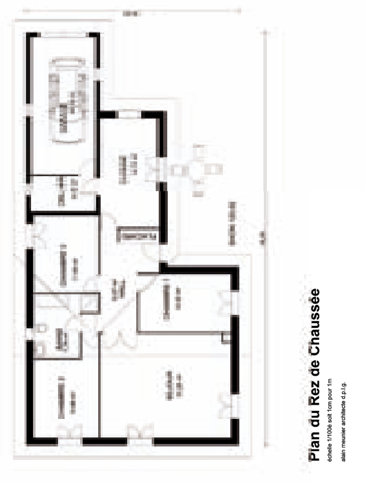
4.4.3 Discussion of the Results

There are several key reasons, beyond those strictly related to performance, that make the presented systems Structural and Syntactic-2 very attractive for floor plan interpretation. Firstly, these methodologies are able work on documents of multiple graphical vocabularies. The usage of statistic approximations to detect the primitive symbols in the floor plan permits to deal with completely different notations. Moreover, this technique also allows to relax some of the structural assumptions made by some works in literature, such as assume that walls are straight elements, and always horizontally and vertically aligned. In addition to that, both methods not only detect the rooms in a floor plan but they also output the complete structure of the documents. This representation is expressed in an attributed graph that makes explicit the attributes of the elements and their mutual contextual dependences. As we will explain in the next chapter, this explicit structural information will allow us to perform further semantic reasoning on the floor plan representations.

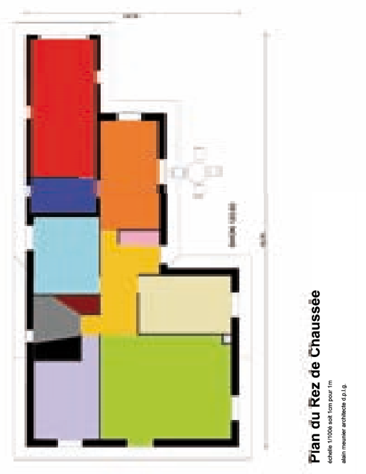
Finally, we also want to emphasize the generality of the presented approaches that allows them to easily adapt to other graphics recognition tasks. On one side, the Structural method has already been adopted to implement part of the interpretation of flowcharts drawings in patent documents [92]. The process of room hypothesis detection has been adapted to find the nodes in the flowcharts. Then, the proposed heuristic search algorithm has been applied to find correlations between symbols, and text boxes and symbols. On the other side, the Statistical method proposes a complete different manner to tackle the problem. A probabilistic model learns automatically the structure of the documents in order to guide the recognition procedure. Therefore, this syntactical learning combined with appropriated parsing strategies can be easily adapted to different applied frameworks.

4.5 Conclusion

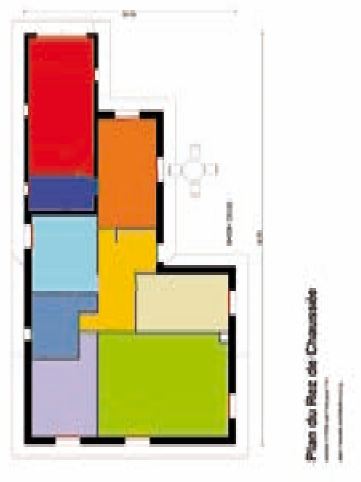
In this chapter we have explained how graphical documents express complex semantic concepts by a set of structurally related symbols. Therefore, in order to make the computers able to *understand* these documents, there is need of obtaining and managing this structural information.


Plan du Rez de Chaussée
échelle 1/100è soit 1cm pour 1m
alain meunier architecte d.p.l.g.

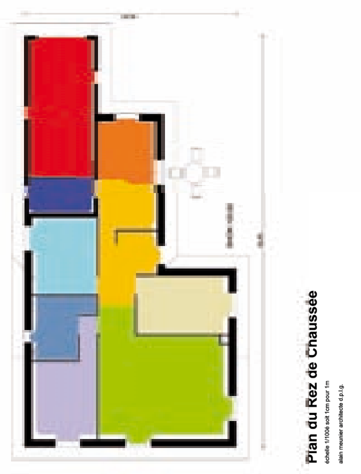
(a) Original image.


Plan du Rez de Chaussée
échelle 1/100è soit 1cm pour 1m
alain meunier architecte d.p.l.g.

(b) DetectedRooms: Structural.

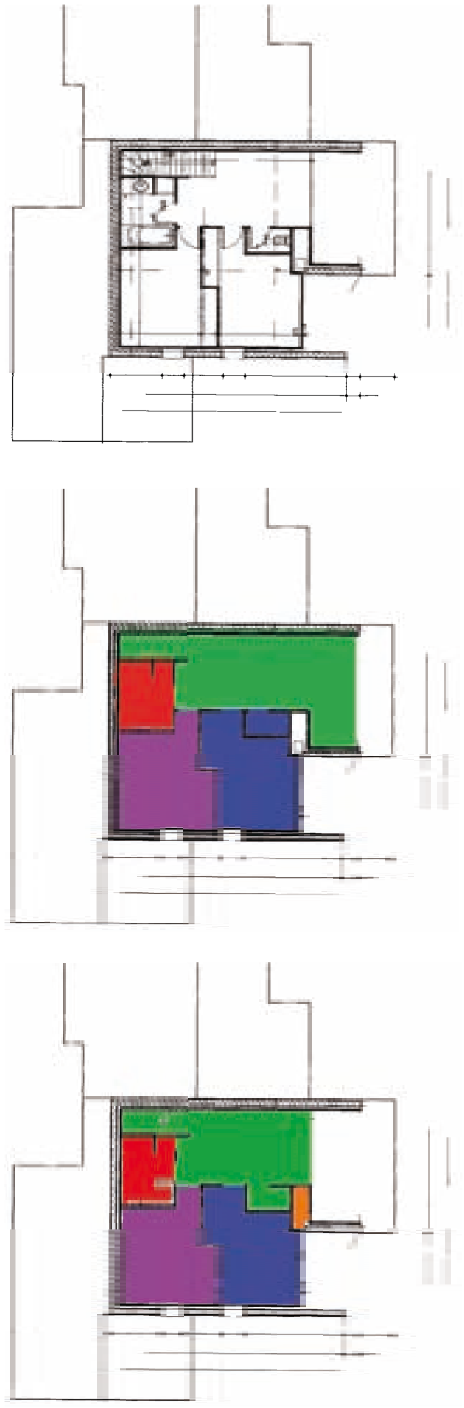


(c) DetectedRooms: Syntactic-0.


Plan du Rez de Chaussée
échelle 1/100è soit 1cm pour 1m
alain meunier architecte d.p.l.g.

(d) DetectedRooms: Syntactic.

Figure 4.11: Room segmentation results for the Black dataset.

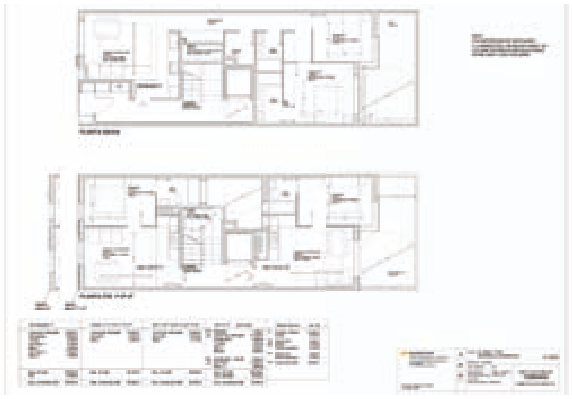


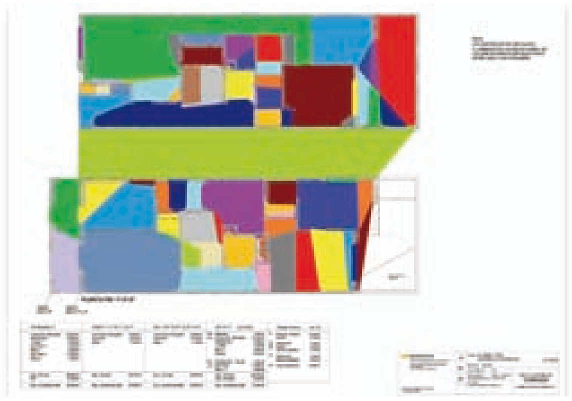
(a) Original image.

(b) DetectedRooms: Structural.

(c) DetectedRooms: Syntactic.

Figure 4.12: Room segmentation results for the Textured dataset.


(a) Original image.


(b) DetectedRooms: Structural.

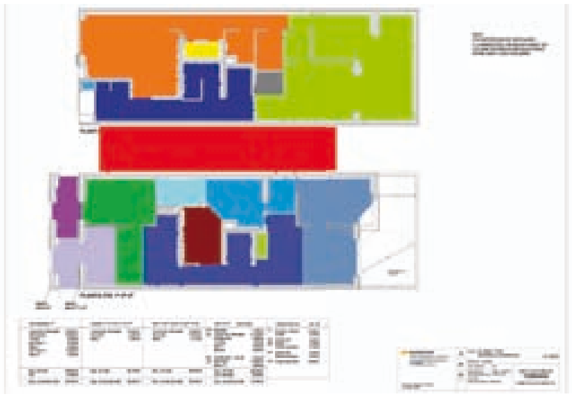
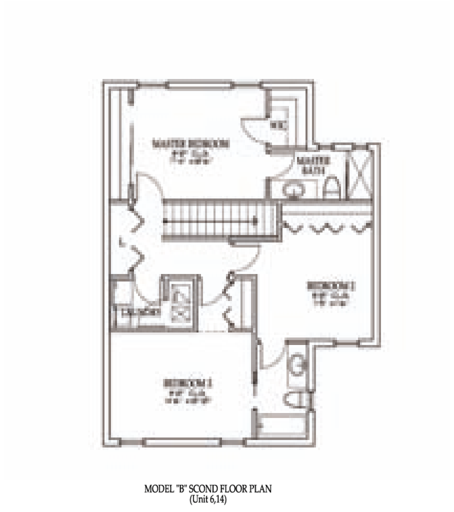
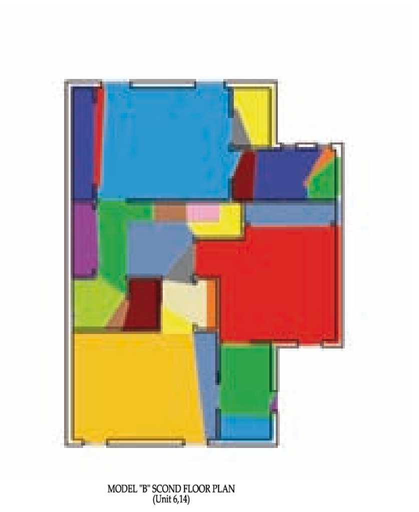

(c) DetectedRooms: Syntactic.

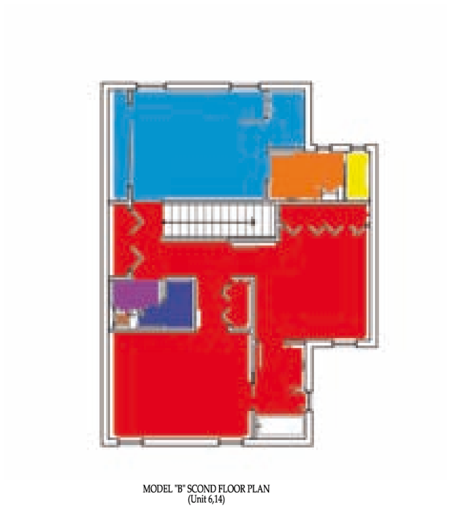
Figure 4.13: Room segmentation results for the Textured2 dataset.



(a) Original image.



(b) DetectedRooms: Structural.



(c) DetectedRooms: Syntactic.

Figure 4.14: Room segmentation results for the Parallel dataset.

Table 4.2: Results on room detection

Dataset		DR (%)	RA (%)	1 −	− 1
Black	[72]	85	69	2	0.76
	[13, 14]	89	79	1.50	1.65
	[15]	94.88	81.3	1.48	2.14
	Structural	94.76	94.29	1.34	2.24
	Syntactic-0	84.86	71.52	1.82	1.20
	Syntactic	93.51	95.02	0.84	4.92
Textured	Structural	90.74	85.65	1.4	3.4
	Syntactic	96.36	77.30	1.3	4.4
Textured2	Structural	99.44	40.38	10.44	0.16
	Syntactic	83.70	26.26	1.6	8.05
Parallel	Structural	100	65.31	4.75	0
	Syntactic	93.75	44.66	1	3.75

We have presented two methods for extracting the structure of graphical documents. They have been applied for room detection in floor plans. Both techniques employ graph-based formalisms since they provide the appropriate tools to express structural information. The first approach is a combination of a statistical detection with a structural method that recognizes the contextual information. It starts by detecting walls, doors, and windows using the symbol detection method presented in Section 3.4. Then, an adaptation of the A* algorithm looks for spatial dependences between the extracted primitives. In the end, this contextual relations are analyzed to detect the rooms in the floor plan. Contrarily, the second approach learns automatically the structure of the documents from annotated data, using this knowledge to recognize their structure. This is a syntactic approach based on an attribute graph grammar that models the contextual dependences of the architectural elements by PGM s. Then, a Gredy parsing strategy retrieves that graph representation that maximizes the posterior probability for the input floor plan. Both techniques output a graph representation expressing the structural information extracted from the documents.

After analyzing the presented results, we would like to stress the importance of three characteristics inherent to these methods. Firstly, the results obtained are very satisfactory. Both approaches highly outperform the rest of the methods in terms of recondition accuracy while obtaining practically the same detection rate. Secondly, the use of statistical-based techniques for symbol extraction allows them to deal with different graphical notations and document resolutions. Moreover, some constrains concerning the symbol variability assumed in some literature methods are completely relaxed here. We have reported the results on for collections consisting of totally different floor plan documents. Finally, both presented methods have been designed as general as possible to ease their adaptation to other graphical recognition frameworks.

Chapter 5

Semantic analysis

5.1 Introduction

As already mentioned in this thesis, the graphical documents convey complex semantic concepts understandable by humans. This information is structured agreeing to a visual language; consisting of a vocabulary graphical symbols and a syntax contextual relations . Therefore, the semantic content expressed in a document is defined by the contextualized meaning of its structurally related symbols. Let us exemplify this fact by making again an analogy with natural languages.

The following two sentences are correct in terms of vocabulary and syntax:

People drive cars
Cars drive people

> Even though both sentences consist of the same set of words, their structure syntactical positioning leads the them to express completely different meanings. Moreover, given our natural knowledge of the language domain the real world , we can assert that one of the sentences expresses an unlikely event.

Alike to natural language comprehension, graphical understanding requires the knowledge of the document domain. This knowledge defines the meaning of the compounding items in a determined context. For instance, in Figure 5.1, the visual syntax, e.g color matching and relative object location, combined with the domain knowledge of the graphical document allows us to answer complex questions such as *who won the Catalan elections of 2012?*. Thus, aiming at making computers able to understand graphical documents, we need to provide them with the appropriate tools to define, store, and employ this knowledge.

Ontologies are machine-interpretable specifications of conceptualizations [51]. They make explicit the description of concepts (classes), their attributes (properties) and mutual relationships that can exist in a domain. The domain definitions are written in formal languages with an expressive power close to the first-order logic; the language definition is independent to the data structure. Therefore, ontologies allow to describe

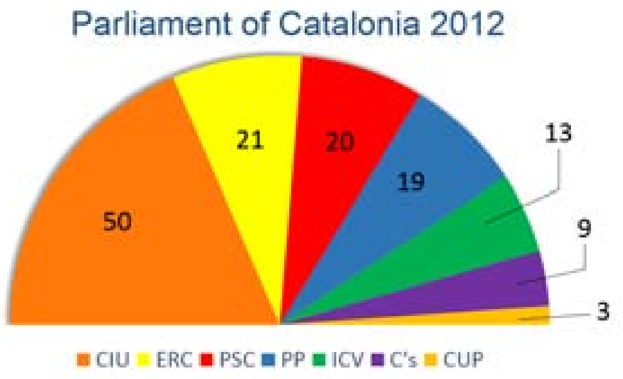


Figure 5.1: Catalonia Parliament elections of 2012

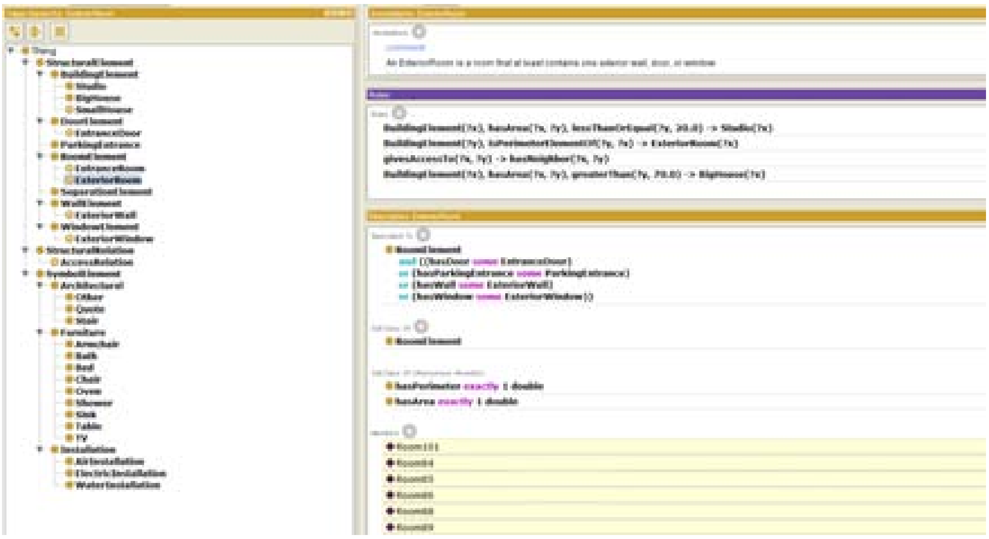
a domain knowledge in a manner that it can be reused, incremented, and shared by disparate agents. Additionally, an ontological definition together with individual instances of the classes conforms a knowledge base that can be analyzed, queried, and classified semantically. Ontological definitions have already demonstrated their suitability in multiple Computer Vision scenarios, e.g. object categorization [73], and recognition [98], medical imaging [76], and natural image description [78]. In consequence, and given its properties, ontologies are convenient tools to express the domain of graphical documents.

In this chapter we complete our pipeline for the graphical document understanding by tentatively exploring the ontological modeling of the graphical documents. We have created a domain ontology to describe floor plan documents. This definition allows us to perform semantic classification, retrieval, and validation of the knowledge base. In the following, we firstly introduce the Floor Plan knowledge base, we secondly overview the experiments performed, and we finally conclude the chapter.

5.2 Floor plan knowledge-base

We have created a knowledge base consisting of a formal definition of floor plan documents and a set real instances coming from both, automatic interpretation and manual annotation. This knowledge base has been created aiming at filling the following intentions:

- To define specifically the semantics of our domain. We have created a floor plan ontology that permits us to describe formally the taxonomy of the concepts conveyed in floor plans, their properties, and relations.
- To permit the reutilization and maintenance of the domain. Since this is a long term project, the formal definition of the domain eases its maintenance; there is an independence between the interacting implementations and the ontology. Moreover, it allows to other agents, either human or automatic, to reuse and upgrade our definition at their convenience.



```
Thing
  StructuralElement
    BuildingElement
      Studio
      BigHouse
      SmallHouse
    DoorElement
      EntranceDoor
    ParkingEntrance
    RoomElement
      EntranceRoom
      ExteriorRoom
    SeparationElement
    WallElement
      ExteriorWall
    WindowElement
      ExteriorWindow
  StructuralRelation
    AccessRelation
  SymbolElement
    Architectural
      Office
      Quote
      Stair
    Furniture
      Armchair
      Bath
      Bed
      Chair
      Oven
      Shower
      Sink
      Table
      TV
    Installation
      AirInstallation
      ElectricInstallation
      WaterInstallation
```

```
An ExteriorRoom is a room that at least contains one exterior wall, door, or window

BuildingElement(?x), hasArea(?x, ?y), lessThanOrEqual(?y, 20.0) -> Studio(?x)
BuildingElement(?y), isPerimeterElementOf(?y, ?x) -> ExteriorRoom(?x)
givesAccessTo(?x, ?y) -> hasNeighbor(?x, ?y)
BuildingElement(?x), hasArea(?x, ?y), greaterThan(?y, 70.0) -> BigHouse(?x)

RoomElement
  and ((hasDoor some EntranceDoor)
  or (hasParkingEntrance some ParkingEntrance)
  or (hasWall some ExteriorWall)
  or (hasWindow some ExteriorWindow))

RoomElement

hasPerimeter exactly 1 double
hasArea exactly 1 double

Room131
Room64
Room65
Room66
Room68
Room69
```

Figure 5.2: Floor plan ontology at the Protégé UI

- To allow semantic reasoning with real data. The inclusion of instances agreeing the ontological framework allow to classify and validate them regarding the definition of the concepts, attributes, and relations.

These aims have lead us to write The Floor Plan knowledge-base in the Web Ontology Language OWL2 [57] on the Protégé5 [9] ontology editor. In the following we summarize the reasons of these decisions.

- OWL2 is a logic-based description language for the semantic web that is able to explicitly represent complex knowledge about things and their relations. The expressiveness of OWL2 to represent machine-interpretable content overcomes other existing languages such as RDF [10], DAML [82], and DAML+OIL [31]. Several semantic reasoners exist for OWL, as Fact++ from the University of Manchester and Hermit from the University of Oxford, which allow inferring automatically semantic properties of ontology defined-classes. Furthermore, the Semantic web Rule Language SWRL [58] is an extension of the OWL model-theoretic semantics that provides a formal meaning for OWL ontologies including Horn-like rules written in RuleML. By this means, instance-based semantic assumptions in floor plan classes can be added to our ontology and automatically be reasoned. Finally, query languages as SPARQL [54] and OWL-SAIQL [66] allow to query the OWL ontology similarly as SQL in relational data-bases. OWL2, SWRL, and SPARQL are taken as W3C recommendation, which assures their promotion, maintenance, and upgrade.

- Protégé is a software developed by the University of Stanford to construct ontologies and knowledge-base applications in a friendly UI. It is currently used in several research and private projects[1] given its wide spectrum of functionalities for ontology design and application. It supports, among others, OWL, SWRL, and SPARQL. A snapshot of the floor plan ontology in the Protégé can be seen in Figure 5.2

Once the intentions and the technical issues of the knowledge-base are settled, we firstly explain the floor plan ontology and we subsequently describe how we constructed the knowledge-base construction from real data.

5.2.1 Floor plan ontology

The design of the floor plan ontology started by deciding the functionality it is intended to. In our case, we have constructed an ontology to represent the knowledge on floor plan documents within the scope of architectural understanding. Therefore, since it has to encapsulate the structural configuration of these documents, the classes (concepts), properties (attributes), and relations (contextual dependences) are pretty much alike to that ones defined in our syntactic representation in Section4.3.

Here we take a brief look into the ontological design, but we include its complete definition in the Annex X.

Class Taxonomy

The classes in the ontology define objects or concepts. In our case these are the structural symbols appearing as nodes in our floor plan structural representation: Building, Rooms, Domain, Wall, Door, Window, and Separator. Notice yet, that these classes are disjoint under a semantic point of view . This means that one instance can only belong to one of the defined classes, e.g a *wall* belonging to the class Wall cannot be at the same time an instance of the Room. Therefore, the hierarchical configuration of these classes totally differs from that one defined in Section 4.3.2. Semantically, all these classes are disjoint siblings from a common parent class StructuralElement, see Figure 5.3. For instance, a building is a individual of the class Building, which is a *kind of* StructuralElement.

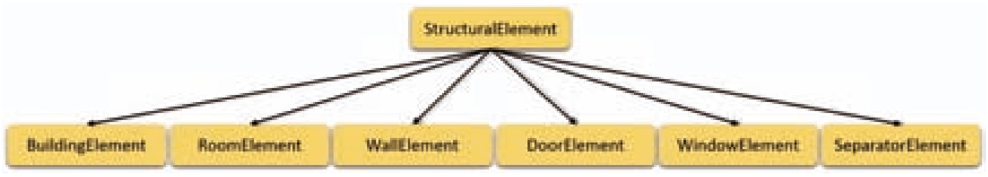


Figure 5.3: Class taxonomy.

[1]http://protege.cim3.net/cgi-bin/wiki.pl?ProjectsThatUseProtege

Object Properties

Object properties are binary relations between individuals. In the floor plan ontology they describe the structural dependences described in Section 4.3.2. These are the *neighborhood* and *accessibility* relation between rooms, and the *incidence* relation between walls, doors, windows, and separators. As it is seen in Figure 5.4, the object properties permit allow to reproduce the hierarchical definition at our syntactic model. Furthermore, we also define a taxonomy of object properties, e.g. the relations *hasRoom*, *hasWall*, *hasDoor*, *hasWindow*, and *hasSeparation* are subproberties of *hasStructuralElement*. This relation is transitive, which implies that, when a individual *A hasStructuralElement B* and, at the same time this *B hasStructuralElement C*, the *A hasStructuralElement C*.

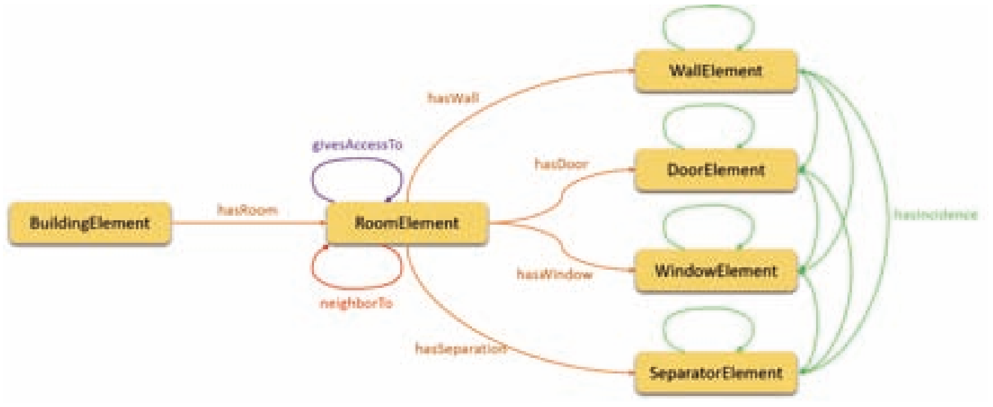


Figure 5.4: Object properties.

Data Properties

The object classes may have defined some properties or attributes that link their individuals to an XML Schema Datatype. For instance, we defined in our syntactic representation that buildings and rooms cover an area or space. Therefore, we can define a data property named *hasArea* that relates the individuals of these classes with a numerical value. In Figure 5.5 we show the data properties for the StucturalElements.

5.2.2 Introducing real instances into our knowledge base

Once our domain is described, we have created a knowledge-base by introducing real instances into our ontological definition. Our aim is to perform semantic reasoning on this data and thus, validate our ontological design together with our incoming floor plan representations. This input data comes from two different sources. On the one hand, it is acquired from the interpretation methods explicated in chapter 4. These recognition approaches output graph representations carrying the structure of the documents. On the other hand, it is collected from the structured GT explicated in

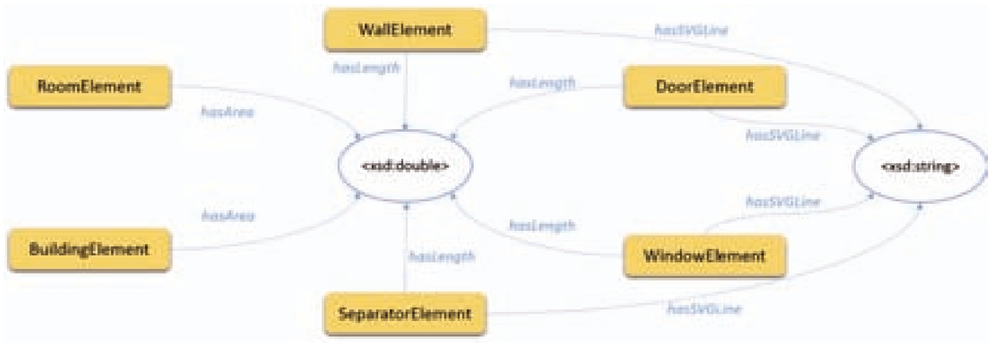


Figure 5.5: Data properties.

Chapter 6. This manually annotated documents not only incorporate the labellings of the objects, but they also make explicit the structural relations between objects.

Event though there are several frameworks and API's available to transform our definition into a practicable implementations, e.g. Jena [2] and Sesame [8] in JAVA$^{\text{TM}}$, we have addressed this task in the opposite way. We have introduced our instances into the OWL definition and thus, use Protégé to perform the reasoning. This has been done by implementing a simple wrapper in JAVA$^{\text{TM}}$ that is able to parse both, the interpreted representations and the SVG files of the GT.

5.3 Experimental validation

In this section we explain a set of simple experiments carried on our knowledge-base on floor plans to get an idea of the multiple application possibilities when semantic reasoning is available. These experiments are divided in two main tasks. Firstly, we show how semantic reasoning allows to automatically classify instances into object classes regarding their properties and relations. Secondly, we explain how the semantic reasoner has helped us to construct and validate our GT of floor plans.

5.3.1 Automatic instance classification

On the simple ontological specification presented in this chapter, we can create new object classes whose individuals comply certain characteristics. We will use the reasoner to automatically compute the new class hierarchy and classify the instances that satisfy that specifications.

We have created a new object property namely *isPerimeterOf* that relates an architectural physical primitive –wall, door, or window– with a building instance; it specifies that a certain primitive is part of the exterior perimeter of a particular building. Then, we can define three object classes ExteriorWallElement, ExteriorDoorElement, and ExteriorWindowElement consisting of exterior primitives:

ExteriorWall := WallElement **and** (isPerimeterOf **some** BuildingElement)

ExteriorDoor := DoorElement **and** (isPerimeterOf **some** BuildingElement)
ExteriorWindow := WindowElement **and** (isPerimeterOf **some** BuildingElement).

When we run the reasoner, it automatically infers that the classes ExteriorWall, ExteriorDoor, and ExteriorWindow are subclasses of WallElement, DoorElement, and WindowElement respectively. Furthermore, it automatically classifies that primitive individuals with a valid *isPerimeterOf* relation. Now we want to define what an exterior room is. We can do it as follows:

ExteriorRoom := RoomElement
and((hasWall **some** ExteriorWall)
or(hasDoor **some** ExteriorDoor)
or(hasWindow **some** ExteriorWindow))

Therefore, an exterior room is a room instance that has a wall, a door, or a window that belongs to the exterior perimeter of a building. Let s now define what an entrance room of a building is. We do it as:

EntranceRoom := RoomElement
and(hasDoor **some** ExteriorDoor)

The reader may notice that both, ExteriorRoom and EntranceRoom are defined as subclasses of RoomElement. Yet, the reasoner actually infers that the class EntranceRoom is a subclass of ExteriorRoom, i.e. all instances of EntranceRoom are instances of ExteriorRoom at the same time. Figure 5.6 shows a snapshot of the class hierarchy before and after applying the reasoner. This feature is really helpful when the size of the ontology (the number of classes) starts to significantly increase and keeping the multiclass hierarchy becomes a challenging task.

Now, we can imagine that this knowledge base belongs to real estate company that allows to search online their available flats for rent. It may be interesting to classify the dwellings according to their usable space. Therefore, we can predefine some classes to define different building types concerning their area:

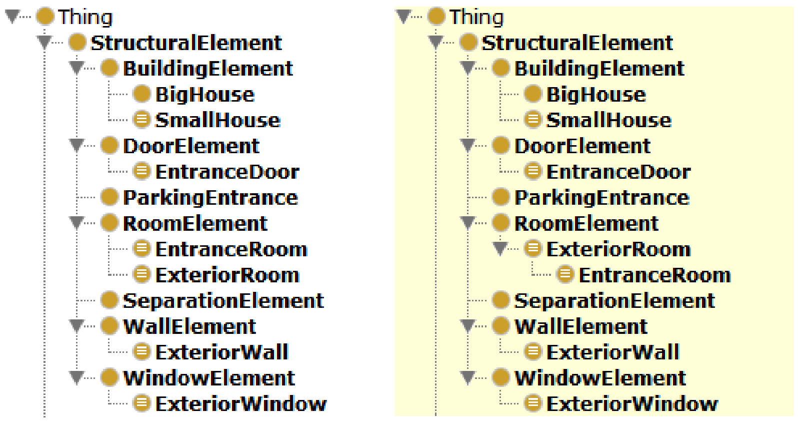


Figure 5.6: Class hierarchy before and after the automatic inference

Studio := BuildingElement
and hasArea **double**[<= 20]

SmallHouse := BuildingElement
(**and** hasArea **double**[> 20])
(**and** hasArea **double**[<= 70])

BigHouse := BuildingElement
and hasArea **double**[> 70]

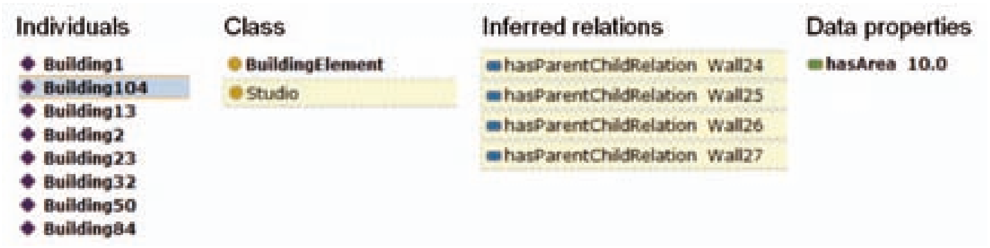
We can also declare this classes using SWRL. For instance in the case of the Studio:

BuildingElement(?x) hasArea(?x ?y) lessThanOrEqual(?y 20)   Studio(?x)

SWRL also allow us to define constrains between relationships. For instance, we can define that all the rooms that are accessible from each other are also neighbors:

givesAccessTo(?x ?y)   hasNeighbor(?x ?y)

Finally, imagine that we are very interested on finding buildings that are exterior. This means that at least 3 rooms are at the boundaries of the building. We therefore can define the ExteriorBuilding class as:



Individuals
- Building1
- Building104
- Building13
- Building2
- Building23
- Building32
- Building50
- Building84

Class
- BuildingElement
- Studio

Inferred relations
- hasParentChildRelation Wall24
- hasParentChildRelation Wall25
- hasParentChildRelation Wall26
- hasParentChildRelation Wall27

Data properties
- hasArea 10.0

Figure 5.7: Automatic instance classification. The reasoner categorizes the instance *Building104* as **Studio** according to its area. The reasoner infers the building *parentChildRelation* with those primitives that belong to its rooms.

$$\begin{aligned}&\mathrm{BuildingElement(?x), hasRoom(?x, ?y), makeBag(?b, ?y),}\\&\mathrm{greaterThan(?b, 2)} \rightarrow \mathrm{ExteriorBuilding(?x)}\end{aligned}$$

We have introduced some of the interpreted and GT instances into our knowledge-base to analyze their semantic room classification. The JAVA wrapper writes into the ontology the structured data, already specifying which instances belong to the exterior boundary of a building. In Figure 5.7, we show two visual examples to illustrate this automatic classification.

5.3.2 Automatic instance validation

The automatic verification of the instance description w.r.t the domain ontology has been a crucial process for the generation of a consistent floor plan GT. As we explain in the next chapter, we have create a labeling tool that allows to make specific the structural relations between the different architectural elements. Nevertheless, this tool does not control whether the relations are well defined in terms of the individual instances. Since the manual annotation is susceptible to errors, the consistency of labeled images can be strongly prejudiced. Therefore, we have incorporated every GT image into our knowledge-base and used the reasoner to spot transgressing instances w.r.t the domain definition. Most of the inconsistencies have been reported on the domain or scope of the object properties with the actual instances classes that take part. In that cases, the reasoner outputs the encountered inconsistency and facilitates the correction of mislabeled images.

5.4 Conclusions

In this chapter we have seen how we can specify the semantics in a domain of knowledge using ontologies. An ontology is an attributed directed graph that represents the concepts, properties, and context that exist in a formal and machine-interpretable manner. Thus, this domain specification can be adopted or upgraded by multiple agents for their own convenience. In addition to that, an ontological definition together with real instances compose a knowledge base on the domain. Given the formal structure of this knowledge, ontological reasoners can analyze this data and extract complex semantic concepts from the structured data.

We have created an ontology to define the semantic meaning expressed in floor plan documents. This ontology has allowed us to specifically define the architectural concepts appearing in this documents, their attributes, and relations. It has been written in the Ontology Web Language due to its convenient expressibility and the multiple tools available for this language. Into our definition we have introduced real instances from both, interpreted and GT floor plans, to create a knowledge base of floor plans. Then, we have experimentally illustrated the applied possibilities when we can perform semantic reasoning. For instance, we have created new concepts by associating the existing knowledge and used to classify the individuals accordingly. Finally, we have explained why the ontological definition of our domain has been of essential help at the GT generation.

Chapter 6

Floor plan database

6.1 Introduction

Current advances on structured learning methods in many pattern recognition tasks have driven to the development of new approaches encoding structural information. For instance, structured SVM [62] have been used for object segmentation [20], and conditional random fields [60] have largely applied in many object recognition tasks [87]. In the field of document image analysis, there is a long experience on structural methods for information extraction and analysis of multiple types of documents: Markov logic networks [90] have been applied for contextual word spotting on historical documents [46], and graph matching algorithms have been used for symbol recognition on technical drawings [68]. With all, these systems usually need conveniently annotated databases to extract and learn the structural interrelations among objects. The lack of such available databases may constrain the research advances in some domains, which is for instance, the case of automatic floor plan understanding.

In order to representatively evaluate all the contributions presented in this thesis, some of them based on structured learning, we have created a database of real floor plans that fulfill three different requirements:

1. The database should model the real variability of the problem. This means that it has to incorporate floor plans of different graphical notations, resolutions, purpose, and sort of information.
2. The dataset must be adequately annotated to guarantee a fair performance evaluation for the different contributions.
3. The collection must grant the extraction of structural interrelations among the architectural elements. This would let the structured learning systems to explicitly learn contextual object dependences and trigger better interpretation.

Yet, the creation of databases entails a main difficulty: the image labeling. Even though it is a straightforward procedure, the creation of ground truth (GT) is, for the most part, tedious and slow. Thus, tools allowing complex GT generation in an efficient way are highly required to speed up this procedure and make it as lighter as possible. This calls for tools that support the cooperative work, with user-friendly frameworks, fluent and consistent operational, security, and version control.

In order to give solution to the problems mentioned, we present in this section three different contributions:

1. We make publicly available a database *named CVC-FP* of real floor plans for research purposes. The collection consists of 122 scanned floor plans documents divided in 4 different subsets regarding their origin and style. It contains documents of different qualities, resolutions, and modeling styles, which is suitable to test the robustness of the analysis techniques.
2. The dataset is fully groundtruthed for the structural symbols: *rooms*, *walls*, *doors*, *windows*, *parking doors*, and *room separations*. The GT not only makes specific their locations in the images, but also includes structural relations between them.
3. We release freely for research purposes the tool used to create this GT. The large experience of our research group on the creation of groundtruthed collections has aided us in the conceiving of an efficient tool for structural labeling. This tool, named SGT tool, can easily be installed in any web server and an simple user administration system allows the collaborative ground truth task.

All these resources are available at the CVC-FP web page[1], including a benchmarking summary on *wall segmentation* and *room detection* tasks presented in this dissertation. Moreover, we publicly make available the evaluation scripts. Our intention is to ease and promote the researchers to test and compare their own interpretation methods.

We have organized this chapter as follows. In Section 6.2 we review existing related databases and groundtruthing tools. Then, we start by introducing the SGT tool in Section 6.3. This will allow us to explain in detail in Section 6.4 the structural content and format of the groundtruth generated. Section 6.5 is devoted to present the images of the 4 datasets that conform the CVC-FP Database. We finally conclude the Chapter in Section 6.6.

6.2 Related work

In order to put into context our work, we briefly explain the existing databases related to floor plans analysis tasks. We subsequently overview the characteristics of the available annotation tools to generate GT in documents.

6.2.1 Floor plan databases

Everyday, the amount of available datasets for research purposes is increasing thanks to the collaborative work of the community. Technical committees, research centers,

[1]`http://dag.cvc.uab.es/resources/floorplans`

and universities are highly contributing by updating, maintaining, and sharing their resources [11]. Yet, we are still far away from having a wide range of representative benchmark datasets for the different scenarios in document analysis. Testing and comparing different approaches in distinct domains is limited to few well known labeled collections. This fact sometimes can favor ad-hoc systems that fit very well in the existing datasets over those ones which better fill in the large variability of the real world. For these reason, new annotated datasets, well structured and detailed, that fill empty spaces in any research domain are always welcome.

In our area of interest, graphics recognition in documents, multiple available databases have been incorporated for the different sub-areas that it covers. These datasets can be created either by means of synthetic data generation or by real document annotation. On the one hand, synthetic databases consist of data generated by varying a predefined set of parameters to model different degrees of distortion, noise, and degradation than real documents may suffer. The generation of these sort of collections tends to be much faster than the annotated ones. In return, the model has to be closed enough to the reality to allow strong conclusions when using them. On the other hand, the annotated databases of real documents reflect the real variability of the world. However, collecting and manually groundtruthing the images can be very time demanding. This issue can be relaxed by semi-automatic annotation procedures [77].

One example of synthetic database is the GREC 2003 [100]. It was conceived in the IAPR International Workshop on Graphics Recognition in 2003 to settle up a common evaluation strategy for symbol recognition. This challenge dataset contains 50 cropped models from architectural and electrical documents. The primitives of these symbols are lines and arcs, which are subjected to different levels of noise, shape distortions, and linear transformations. Lately, the GREC 2011 dataset [99] was created not only as an extension of GREC 2003 in terms of recognition, but also included a symbol spotting contest in both architectural and electrical documents.

One of the most used databases for symbol recognition related tasks is the SESYD database [35]. It is a collection of labeled synthetic images. They include architectural and electrical documents for symbol spotting, recognition, and retrieval. Additional datasets for text-graphic separation and text segmentation are included. Regarding its floor plan collections, they are specifically generated for detection purposes; leaving aside the semantic assembly between symbols and the building structure.

The FPLAN-POLY database [91] is, to our best knowledge, the only available collection of annotated real floor plans. Nevertheless, it aims for symbol spotting tasks. It contains 38 symbol models in a collection consisting of 48 vectorized images.

Despite there is not any floor plan database for complete analysis purposes, on other structured drawings such as flowchart diagrams, several work has been pursued on structural and semantic understanding. Thereby, the CLEF-IP initiative investigates information retrieval techniques on patent documents. One of the goals of that challenge consists of extracting the structural information from patent flowcharts in order to be queried semantically a posteriori. This process entails not only the detection and recognition of the elements participating in the diagrams (nodes, text, arrows), but also the structural assembly between them and their semantic meaning [85,92].

6.2.2 Groundtruthing tools

In the document analysis domain we can find a large set of tools developed for the generation of GT. We analyze them by describing their functionality and limitations.

Most of the existing groundtruthing tools for document analysis related tasks are oriented to deal with textual documents. On the one hand, some of them address the evaluation of logical and physical layout methods, e.g. Aletheia [30], GEDI [37], TRUEVIZ [52], PinkPanther [105], and GiDoc[2]. Here, entities are represented by rectangular or polygonal regions by both physical and logical information. Physical information usually belongs to *textual* regions, *pictures*, *figures*, *tables*, etc. while logical information usually denote the semantic meaning of each physical entity in the document context, i.e. *headers*, *title*, *footnote*, etc. On the other hand, some tools focus on performance evaluation at pixel level. These tools aim at a very accurate pixel annotation and include semi-automatic labeling tools to improve the groundtruthing efficiency. Examples of these tools are the multi-platform based on Java^{TM} PixLabeler [94], and the very recent web-based tools WebGT [22] and APEP-te [63].

The specific focus of the previously cited tools hinder their usability on other document analysis tasks, i.e. graphics recognition. Some of them only allow to label rectangular segments [48, 52, 103]. Others delimit the definition of object categories into a small set of predefined classes [22, 94]. Moreover, the definition of object dependences usually rely on hierarchical information [63, 105] and limited structural concepts [30, 37], i.e. reading order and relative location. Furthermore, to our best knowledge, only [30] has a multilayer representation that permits the labeling of fully overlapped objects.

Finally, it is worth to mention that the current tendency is to design multiuser tools that foster real-time grountruthing cooperation either by versioning control [22, 63] or following crowdsourcing strategies [17, 48]. Moreover, the vast majority of the recent tools use slight variation based on XML for GT specification, i.e. the PAGE format [86]. This fact permits to easily adapt the existing platforms to parse GT files generated by other applications. Yet, none of the existing web-based tools uses the SVG format to naturally display the GT at the web browser interfaces.

6.3 The structural groundtruthing tool

The SGT tool is thought to perform general purpose groundtruthing; not restricted only to one specific domain as most of the existing tools are. It grants full flexibility since the proprietaries of the databases can create, modify, and erase their own object classes. Additionally, it is possible to define and declare n-ary properties for the objects, that allows to represent the ground truth as an attributed graph, where nodes are labeled objects and edges are relations between them. In Figure 6.1 we can see an scheme of the SGT tool architecture. The SGT tool is user-friendly, it allows 2 different labeling options, and the output is in the standard Scalable Vector Graphics (SVG). The tool is a cross-platform running on a web service, which enforces coworking without sacrificing security. It has been implemented in php5 and HTML5,

[2]`https://prhlt.iti.upv.es/page/projects/multimodal/idoc/gidoc`

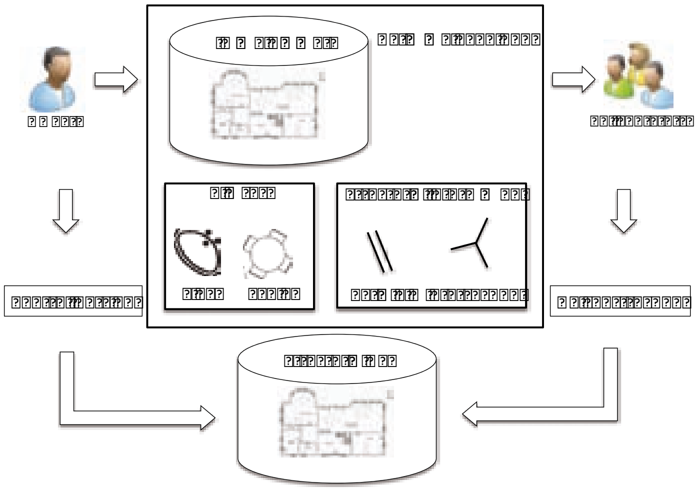


Figure 6.1: Overview of SGT tool architecture. Each database is stored into a different folder. For each database a particular set of classes and structural information is defined

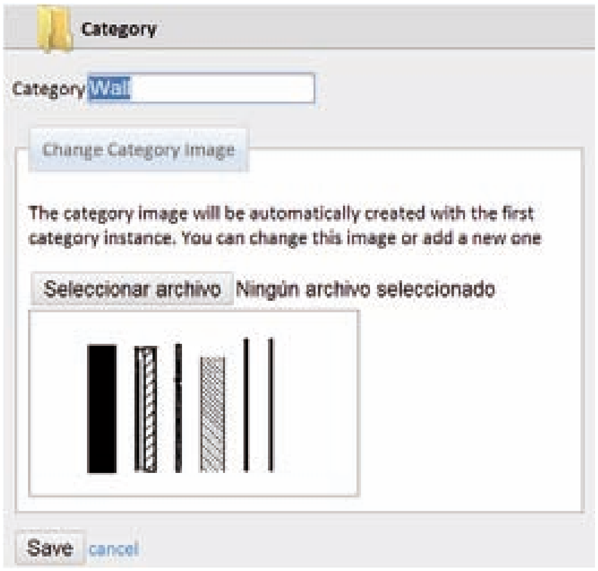
and the collections are stored in a relational database like MySQL [7].

In this section we overview the SGT tool. For a further detailed explanation we encourage the users to read the *user guide*, available in the project CVC-FP web page.

6.3.1 Classes and structural relations. Definitions and labeling

The SGT tool can be used in multiple domains since it allows the user to define their own object classes. For example, in the floor plan interpretation framework that we are interested in, we define object classes as *Wall*, *Room*, and *Door*. Contrarily, for symbol spotting we would rather define *Bed-type1*, *Bed-type2* and *Shower-bath*, and for textual document layout analysis *Title*, *Legend*, and *Graphic*. The classes are defined in a *Class Management* window, where the user can define, modify, and delete their own classes. When a new class is created an example image of the object can be added into its definition, see Figure 6.2. This image is shown at labeling time to help unexperienced users in cooperative groundtruthing task. The classes are defined at datasets level, which permits define classes without risk of mixing labels between different databases.

Object properties not only allow to define attributes for the different elements, but they also permit to declare structural and semantic dependences among multiple object instances. They are similarly defined and administered as classes at the *Relation Management* window. At definition time, the user can define the *arity* of the property: they can be specific for a single object or relating *n*. A brief description to help users can be also written in their definition, see Figure 6.3. Their labeling is



The category image will be automatically created with the first category instance. You can change this image or add a new one

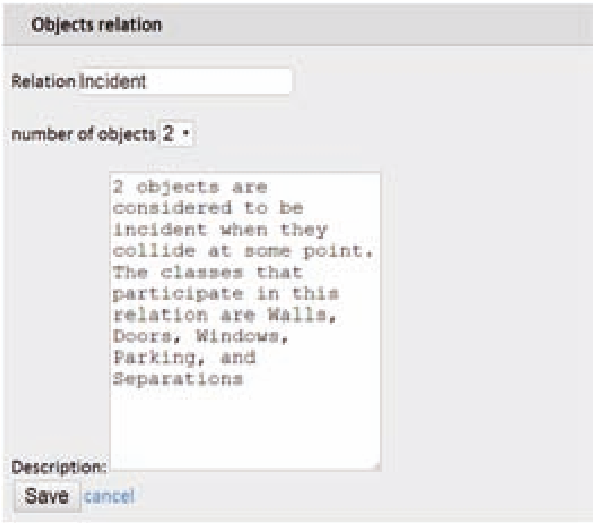
Figure 6.2: Window for new category creation.

done by selecting first the desired property, and then by picking those labeled objects that participate in it. The SGT tool ensures that the *arity* declared agrees with the property definition. Thus, object properties not only allow to define attributes for the different elements, but they can also make reference to structural and semantic dependences among multiple object instances.

SGT tool facilitates the user the labeling procedure with a clear interface, see Figure 6.4. Objects can be labeled either by drawing their bounding box through selecting just two corners; or by drawing their polygon through a sequence of clicks. Moreover, it allows to make local zooming to ease the labeling of tiny objects. In other GT tools, the visualization and selection of the desired objects can become a challenging task in crowded images with multiple objects that overlap among them. Since SGT tool uses a multilayer representation for each object categories, the users can display or hide object annotations at their convenience. This functionality extrapolates to object properties.

6.3.2 Creation and version control of a database

A registered user that uploads a collection of images to the SGT tool is its proprietary. Once uploaded, any registered user can participate in the groundtruthing task. They have only to select one image and start the annotation. Then, the tool will automatically avoid concurrent edition by controlling the access to the *in use* documents. For each of them, the new GT version associated with its author is stored by the versioning control system. Thereby, the database proprietary can track and control the whole groundtruthing procedure.



2 objects are considered to be incident when they collide at some point. The classes that participate in this relation are Walls, Doors, Windows, Parking, and Separations

Figure 6.3: Window for new relation creation.

6.3.3 Input images and Ground truth SVG files

Concerning the input documents, our application accepts the most common types of image formats: PNG, JPG, and TIFF. When an image is uploaded, it is stored by its file name and indexed locally in its database. The SGT tool has been implemented to support heavy files, it behaves smoothly with images around 20 mega pixels.

To make easier the exchange of classes and relations between databases, the SGT tool incorporates importing/exporting tools. For a given image called X.png, the tool generates one Scalable Vector Graphics (SVG) named X_gt_vY.svg, where Y is the number of the GT version. We have chosen SVG for formatting our GT mainly because of three reasons: it uses a well-structured format XML-based language, it is a recommendation of W3C[3], which ensures evolution and maintenance and, finally, allows to describe 2-dimensional vectorial graphics which are displayable in most of the Internet browsers. It is worth to notice that, since the SGT tool is web-based, the user interface is displayed at the Internet browser. Therefore, the use of SVG permits to adapt to different browser preferences while maintaing the same labeling visualization. Obviously, the tool also allows to import external SVG files to update the GT.

The format of a generated SVG file includes own metadata information and is defined as follows. It first has a header for defining the XML and SVG version. Then, the rest of information is divided in three main blocks. Firstly, the general information regarding the GT is specified. It includes the image dimensions (width and height) in pixels, the number of different instances labeled, the number of classes appearing in the document, and the name of all the classes that appear in the dataset. Secondly, it contains the list of the elements in the image. Each text-line describes one object

[3]`http://www.w3.org/Graphics/SVG/`

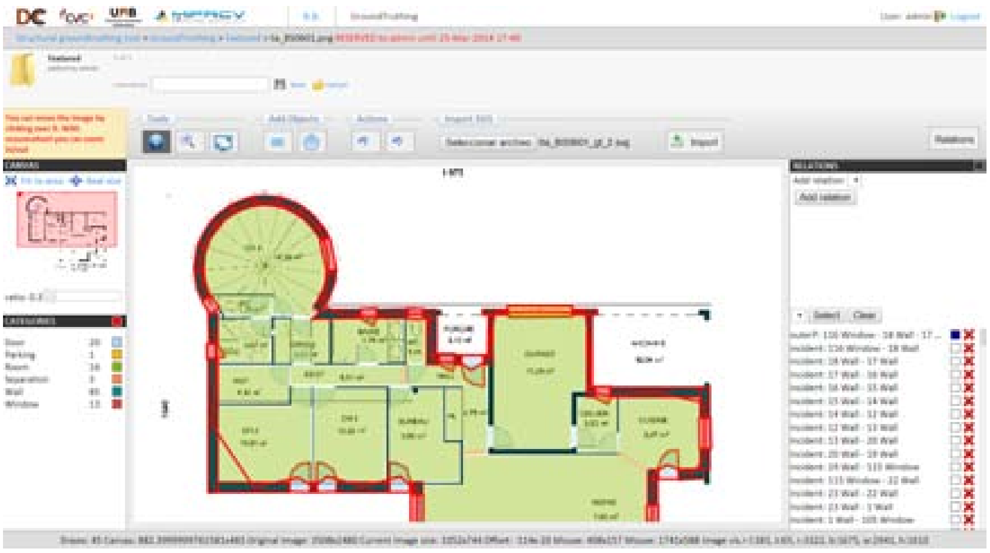


Figure 6.4: View of the *editing* page. Among other functionalities, the user can import an existing GT, choose the labeling procedure, label objects and structural information, and select those objects and relations want to show/hide.

by its label, its document-unique identity number, and its polygon composed by the extremity points selected by the user. Finally, the document describes the relations between the objects. Each relation is identified by its type and the identities of the elements involved.

6.4 Structural Floor Plan GT

In this section we review in detail the GT for the CVC-FP database constructed using the SGT tool. This GT not only contains the location of the architectural elements, but also those structural relations that we have considered to be of the interest for floor plan analysis systems. Despite it is worthy to remark that this is our own definition and it will vary for different applications, images, and experts, we have defined this database by taken into account several considerations. We have contacted a team of architects to address their needs in automatic interpretation applications. We experienced several cooperations with research and private companies aiming for different applications related to floor plan interpretation. We have considered other floor plan definitions in the literature that entail some sort of structural understanding, such is the case of [107] for evacuation building simulation, and [101] for structural floor plan retrieval. Additionally, we have also been inspired by the relevance of the structural information for high-level understanding in graphical documents, i.e. flowchart interpretation in patent documents [85]. Obviously, since the

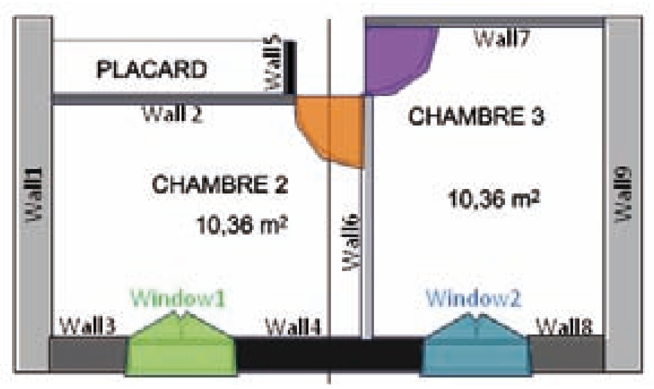


Figure 6.5: Wall, door and window labeling.

SGT tool is shared freely, the GT data can be modified or upgraded agreeing to every system requirements.

Nine people working on distinct areas of graphics recognition have participated in the generation of this GT. Thanks to the version and user control of the SGT tool, the creation of the GT has been parallelized for the complete collection of images. Once the annotation has been completed, one single person has checked the correctness and consistency of the data according to the definitions settled a priori. This task has been pursued to correct different subjective perceptions for the distinct users that have participated. Since the SGT tool is designed in a way that every category and relation can be displayed with independence to the rest, this process has been easily attended. We firstly review the convention followed in the object labeling and secondly the relations instantiation.

6.4.1 Element labels

Let us explain how we performed the labeling of the structural symbols. These are rooms, walls, doors, windows, parking doors, and separations. The labeling of each object has been pursued by selecting that polygon that maximizes the overlapping of its area; this is by selecting each of the extremities of the object.

- *Walls* work mainly to bear the structure of buildings, to isolate, and to delimit room space. Aiming for simplicity, they are usually rectangular-shaped generating corners at their intersections, and gaps to locate doors and windows. However, with the lack of additional architectural information, it is not clear how wall-instances should be separated. We have followed our own convention trying to stick to their structural purpose. We split walls when they have different thickness, and when they intersect at some point generating a L-shaped corner. In Figure 6.5 we show a detailed example to clarify our strategy.

- The labeling of *doors*, *windows*, and *parking doors* has been much easier since their boundaries are well defined. Yet, to label those objects with curved shapes

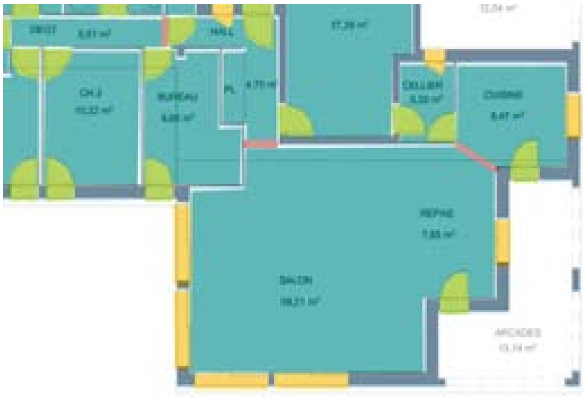


Figure 6.6: Rooms labeling. *Rooms* are drawn in turquoise and *separations* in red.

(doors and windows) we have followed a traded-off between an accurate adjustment to the boundaries and object representation simplicity. Few examples are shown in Figure 6.5.

- The labeling of *rooms* sometimes enclose ambiguity as their limits are not clearly defined. An example is shown in Figure 6.6, where thanks to the text and the structural shape of the building we can presume the separation between the dining room (*repas*) and the kitchen (*cuisine*); the separation between the *salon* and the *hall*; and the separation of this latter and the corridor (*degt.*); although none of them they are physically separated. On the contrary, the text also instantiates the *salon* and the *repas* to be separated habitations. This time yet, the lack of furniture and the building structure are not helping on presuming the hypothetical separation between these two rooms. Therefore, the labeling becomes a very subjective to the expert perception. Due to the difficulty on creating a clear convention on these situations and given the lack of additional information, each room annotation has been examined in detail a posteriori by a single person trying to keep an agreement in the whole collection of images.

- *Separations* are rectangular abstract elements that separate two neighbor rooms without physical frontiers, see Fig 6.6. These elements aim to make clear the accessibility area between these rooms.

6.4.2 Structural relations

Similarly to the object properties in ontologies, the SGT tool permits the definition of relations between object instances. In other words, the SGT tool allows to define attributed graphs to enclose the mutual dependences among the labeled elements. In these graphs the annotated elements are the nodes whereas the contextual relations among the different objects are defined by attributed edges. This fact enriches the expressiveness our GT and allows systems to learn complex features and affinities between elements. We have defined 5 following relations:

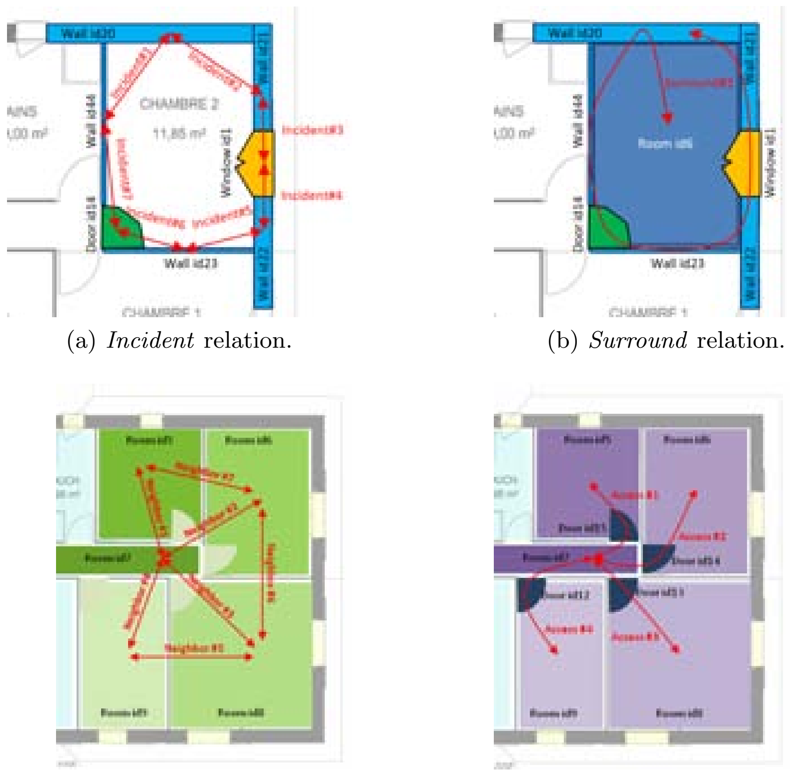
- **Incident**. Two elements are called to be *incident* when they intersect or collide at some point. An example on incident relation is shown in Figure 6.7a. The elements with incident relations are *walls*, *doors*, *windows*, and *separations*.
- **Surround**. Several *walls doors*, *windows*, *parking entrances*, and *separations* can delimit the space of a room. The surrounding relation, as it can be seen at Figure 6.7b, creates a graph of these elements connected with the room they encircle.
- **Neighborhood**. Two rooms are called to be *neighbors* when they share at least one *wall*, one *door*, one *window*, or one *separation* in their surrounding perimeters. Figure 6.7c shows the *neighbor* graph that generates a little part of a plan.
- **Access**. This relations put in correspondence two rooms which are accessible from each other through a door or a separation. It is also used for defining which rooms through which doors are possible entrances to the dwellings. Figure 6.7d shows the *access* graph that generates a little part of a plan.
- **Surrounding perimeter**. It defines the exterior boundary of a building. It is composed by *walls*, *doors*, *windows*, *parking entrances*, and *separations*. Each isolated building only contain one *surrounding perimeter* relation.

6.5 The CVC-FP Images

Let us now introduce the images in the CVC-FP database. This is a collection of real floor plan documents compiled and groundtruthed during the last recent years. It all started with the SCANPLAN[4] project in 2008, and still today the Document Analysis Group of the Computer Vision Centre is working on these graphical documents in multiple domains, such as structural analysis, semantic reasoning, and symbol spotting and recognition. The dataset is composed of 122 scanned documents and a partially groundtruthed version was presented in [34]. Nevertheless, these documents have been shared much before to foster the research in floor plan analysis [13, 14, 72].

The 4 sets have completely different drawing styles, image qualities and resolutions, and incorporate different sort of information. This is not an arbitrary fact; we have pursued the creation of a heterogeneous dataset to either foster the creation of robust techniques that can deal with different image scenarios, and also permit to assess the adaptability of the methods to the different graphical styles of the floor plans. It is important to take into account that different architects and architectural studios usually have their own graphical conventions. Therefore, there is a need of constructing systems that are able to learn each specific notation to be able to generalize for the existing architectural conventions. In addition to that, the different

[4]SCANPLAN PROJECT: `http://www.eurekanetwork.org/project/-/id/4462`



(a) *Incident* relation. (b) *Surround* relation.

(c) *Neighbor* relation. (d) *Access* relation.

Figure 6.7: Examples for the different structural relations between objects.

amount of images in each dataset permits to test the effectiveness of the proposed methodologies either when there is a large or a small set of documents available for learning purposes. We subsequently overview the characteristics of each subset separately, focusing on the structural information of the images, their symbolism, and the textual information.

6.5.1 Black Dataset

The name of this subset, as the rest does, references the graphical modeling of the walls; a thick black line as it can be seen in Fig 6.8. It consists of 90 floor plan binary images of good-quality with a resolution of 2480×3508 or 3508×2480 pixels, depending on the orientation of the building. These plans were conceived to sample the structural distributions of the buildings to possible customers, so they do not contain an excessive amount of technical information.

In this dataset, building drawings are centered and well oriented with respect to the document, most of the architectural lines are parallel to the horizontal and vertical axis. They model the ground floor of detached houses, usually including terraces,

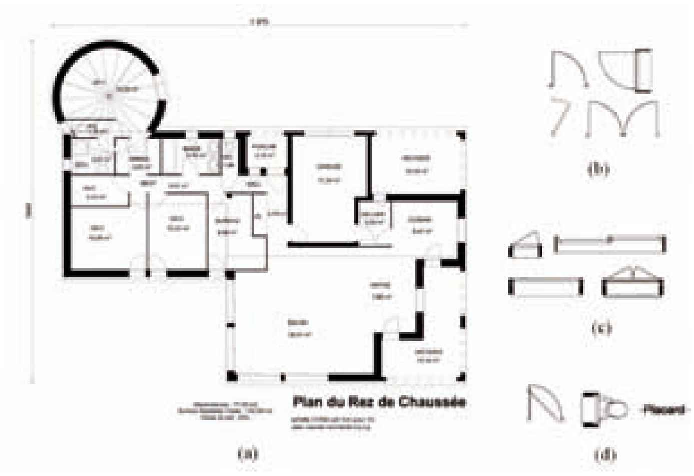


Figure 6.8: Black Dataset. In (a) we show a sample image from this dataset. In (b) we show the different types of doors, in (c) the window's models, and in (d) some of the difficulties of the dataset.

porches, and garages with cars. The drawing style is clear, with few elements crossing among them. Concerning the structural symbols, walls are mostly modeled by black lines of three different thicknesses whether they are main, interior, or exterior walls. Just in 3 plans, walls are modeled by parallel lines. Simple doors are drawn by a quarter of a circle arc whereas building's main doors have an additional rectangular base of the size of their incident walls. Moreover, toilet doors are represented by a quarter circle arc, and double doors by two consecutive arcs centered in each of the wall limits and tangent in the center of the accessible area (see Figure 6.8b). The window models can highly vary, see Figure 6.8c. We can find full opened windows, partly opened windows, and sliding windows; all of them with different breadths. The last of the structural symbols we can find are the stairs. They are modeled by consecutive parallel rectangles. In terms of non-structural symbols, the floor plans contain mostly symbols making reference to bath utilities. Different kind of sinks, toilets, shower baths, and bathtubs are the only ones repeated in all the images. In addition to that, occasionally we can find living room furniture and in buildings delighting of a terrace or a porch may include a garden-table with 4 chairs.

Text can be found in these documents. Each floor plan has a title with big bold letters that it can be read "Plan du Rez de Chauseé", in English "plan of the ground floor". As a subtitle we find the scale of the model (always 1/100, 1cm is 1m) and information about the architectural studio. In some plans next to the title we can find information about the surface area of the dependencies, the building utile area, and the slope of the roof. Less frequently, information of the surface and the orientation of the windows is included in the subtitle. Moreover, each room encloses the text describing its functionality and area –in squared meters–. Finally, each plan has two dimensions measuring in meters the rectangular surface of the building. They are located in the limits of the building perimeter.

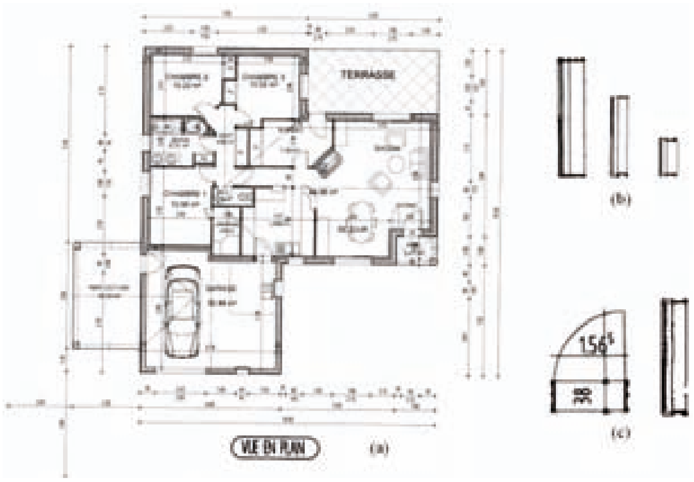


Figure 6.9: Textured Dataset. In (a) we show a sample image from this dataset. In (b) we show 3 different window symbols. In (c) we show some difficulties in the dataset: the multiple intersection of symbols and text to the left, and the side effects of binarizing in poor quality plans to the right.

6.5.2 Textured Dataset

This is the second floor plan dataset compiled in this thesis. It consists of 10 poor quality and grey-scale images whose resolutions can vary from 1098 × 905 pixels the smallest to 2218 × 2227 the largest, see Figure 6.9a. They are computer drawings of detached houses containing not only structural symbols but also furniture, several dimension quotes and textual information.

Here walls are modeled by two parallel lines with a diagonal line pattern in between for the exteriors, and a heterogeneous gray-dotted pattern for the interiors. The notation of doors and stairs is exactly the same of Black Dataset. Contrarily, all the windows follow a rectangular pattern of different breadths, which can be seen in Figure 6.9b. In this dataset, terraces are indicated by a repetitive pattern of squares. Regarding non-structural symbols, we mainly can find sofas, tables, and bath and kitchen utilities such as sinks, baths, and ovens. Furthermore, most of the buildings have a garage with the drawing of a car in it.

This dataset contains textural information, most of them belonging to numbers of dimension measurements. All the rooms are labeled with their name and their area –in squared meters–. Some plans have also a big bold text at the bottom of the image that says "vue en plan", in English "floor plan's view". Additionally, some extra structural information is written in barely readable text.

6.5.3 Textured2 Dataset

The Textured2 dataset is composed by 18 high-resolution images (7383 × 5671 pixels) collected from a local architectural project in Barcelona. The singularity of this

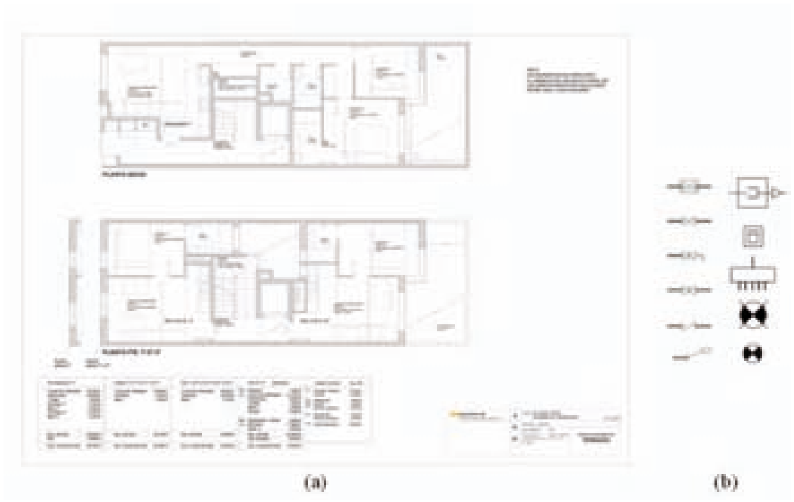


Figure 6.10: Textured2 Dataset. In (a) we show the structural distribution of all the floors in the flat. In (b) we show the different types symbols: from left to right: water and electrical symbols.

dataset is that the 18 floor plans belong to a single flat of 6 floors. The first image, which is shown in Figure 6.10a, contains the drawings of two different floors: the one corresponding to the ground floor, and just beside the overlapping of the 1st, the 2nd and the 3rd floors, which are identical. Similarly to the first, the second image contains the plans of the basement and the 4th floor. The rest of the images contain the same drawings but not exactly located and with different sort of information. The two first contain general structural information; this is for instance the utile area of each floor, the area of living rooms, and the area of the sleeping rooms. The second couple of images contain the architectural dimensions detailed. The third couple includes the information of the surface materials; whether the ground is made of parked or marble and the walls are covered by either plastering or natural stone. The fourth pair of images shows the distribution of the dropped or suspended ceiling. The fifth shows the plumbing distribution whereas the sixth displays the waste plumbing distribution. In the seventh the building's electrical installation is detailed. The eighth shows the gas installation and finally, the ninth is for heating installation.

The walls are modeled similarly to the Textured Dataset, this time with a higher frequency diagonal pattern between the two parallel lines. Doors are drawn by 90° arcs and windows follow the same model. Mostly all the utilities and furniture symbols are drawn in the first couple of images: sinks, toilets, bathtubs, ovens, beds, tables, and wardrobes. Meanwhile the rest of the images enclose the different type of symbols agreeing to their architectural purpose, see Fig 6.10b. The types of suspended ceilings are represented by different textural patterns, and water, electrical, gas, and heating symbolism is specified in their respective legends. Meanwhile, textual information is omnipresent in all the images. Firstly, a text-table situated at the bottom right corner of each image specifies the information regarding the architectural studio, the

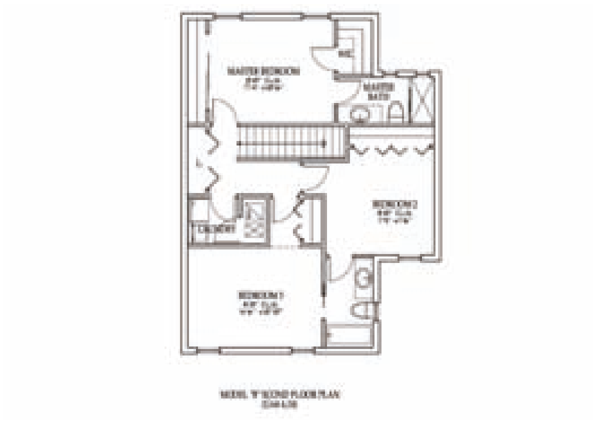


Figure 6.11: Parallel Dataset image.

project, and the document. Secondly, each document except for those enclosing the architectural dimensions contains a legend detailing the semantic meaning of the symbol encountered in the plan. Finally, every document has specific text in key positions to help its interpretation. This text includes for instance room's naming, dimensions, floor statements, walls' height, and facade orientation.

6.5.4 Parallel Dataset

This last collection is composed by only 4 images and was added to perform wall segmentation on walls drawn by simple parallel lines. They are extracted from Google Images[5] and are created by one single architectural studio to sample online the building distribution of 2 detached houses for sale. A instance of this dataset is shown in Figure 6.11.

The binary images are of good quality and high resolution (2550 $\times$ 3300 pixels). As mentioned, walls are modeled by simple parallel lines, doors by a 90° arcs, and windows following a rectangular model. Some house utilities are drawn, as the usual from bath and kitchen. Moreover, since the buildings delight of a laundry room, washing and drying machines symbols can be found. Text also appears in these images. Each room has written in its perimeter its functionality. In addition, in 2 of them, those belonging to the ground floor, have a text-table with the characteristics of the different surface areas –in squared feet–.

6.6 Conclusions

Recent results on structured learning methods have shown the impact of structural information in the performance of a wide range of pattern recognition tasks. Yet,

[5]`http://www.google.com/imghp`

these techniques usually need conveniently annotated databases to learn the interrelation among the objects of interest. In this chapter we have presented the CVC-FP database. It is composed of real floor plan documents that are fully annotated for architectural symbols and make specific their structural interrelations. This sort of information agrees the semantic definition of the domain set in Chapter 5, and will let floor plan analysis systems to learn directly from the knowledge based data how the elements are structurally arranged and thus, to trigger better interpretation.

The groundtruthing tool used to generate this database, the so-called SGT tool, is a general purpose groundtruthing tool. It is a web-based service that permits to create own objects classes and relations using a very intuitive user-interface. This tool fosters the collaboration by allowing standalone and multi-user handling, including user and version control. Thus, the SGT tool is suitable for the creation, the upgrade, and the maintenance of databases in domains where making specific additional structural information can be of great interest.

The CVC-FP database, the SGT tool, and the evaluation scripts are freely released to the research community to ease comparisons and boosting reproducible research.

Regarding future work, in a short-middle term we are planning to upgrade the SGT tool to allow the arrangement of object classes and relations in a taxonomic way. The aim is to organize and facilitate complex labeling procedures, and to foster the reutilization by defining formally the ground truth domain. This taxonomy will be defined by either creating a class and relation hierarchy in the SGT tool, or uploading a formal ontology definition. The SGT tool will be able to export the GT in an ontological framework for further semantic reasoning, i.e. to check the ground truth consistency according its definition. In a longer term, we plan to include this ontological functionality to the tool. Hence, the SGT tool will be able to help, correct and suggest the user at real time.

Chapter 7

Conclusions

Throughout this dissertation we have presented several methods at the different levels of the graphical document understanding pipeline. In this chapter we summarize the contributions presented in each chapter trying to focus on their strengths and weaknesses. Finally, we introduce several working lines to be considered in the future.

7.0.1 Summary and contributions

In Chapter 1 we have introduced the problem of graphical document understanding and the difficulties it currently entails. We have also presented our contributions in this topic, which consist on several relational models at each of the steps in the typical document interpretation architecture. Finally, we have introduced the application framework (architectural drawings), and why it is representative of the general problem.

Chapter 2 has introduced the main difficulties experienced in the literature when tackling the problem of graphical document understanding. Moreover, to contextualize our work, we have reviewed the cutting edge methodologies on floor plan interpretation. The review of this techniques has allowed us to conclude that most of the existing contributions focus on specific domains, data, and assumptions. This fact has constrained the progress on graphical document understanding w.r.t. other GR and DIA tasks.

We have presented two different contributions in Chapter 3 on symbol detection. The first addresses the problem from a structural perspective, which a predefined general knowledge of the domain guides the process. Contrarily, the second is an statistical approach that learns automatically the appearance of the symbols from annotated data. Furthermore, after analyzing the advantages and drawback of each, we have combined them into a new method that benefits from the strength of both approaches; it does not need to learn every predefined notation and it robust for different shapes and textures. Finally, the effective of this methods has been evaluated for wall detection on floor plans.

Chapter 4 has addressed the extraction of the visual context in graphical documents. We have presented two relational strategies that again tackle the problem from very different perspectives. The first is a full bottom-up method that heuristically searches in a graph the contextual relations among symbols. The second approach is a syntactic method that models probabilistically the structure of the images. This model is learned automatically and guides the parsing in both, bottom-up and top-down. The presented approaches have been evaluated for room detection of floor plans.

In Chapter 5 we have constructed a knowledge-based model from an ontological definition of our domain together with real data. This model has permitted the aforementioned syntactic method to learn the context of the documents, i.e. it has guided the interpretation of the documents. Additionally, the explicit definition of the domain has permitted to find semantic inconsistencies in the construction of the structural GT in Chapter 6.

Finally in Chapter 6, we have presented the data used to evaluate each of the contributions presented in this dissertation. This data consists of 122 real floor plans separated in four datasets regarding their graphical notations. These images have been manually labeled not only for the architectural symbols, but also making specific the relations among this symbols. In this chapter we have also introduced the SGT tool, that permits to create structural GT in an efficient manner by fostering cooperative work.

In general, in this thesis we have presented several relational models intended to solve the problem of graphical document understanding. This problem entails the detection of symbols, their structure and semantics in a visual context. The proposed models have demonstrated to be able to encapsulate this sort information and use it in the interpretation. Their suitability has been evaluated on floor plan understanding. We also want to highlight that, with the aim at fostering research in this topic, we share freely all the resources used in this dissertation.

7.0.2 Future work lines

We have several ideas regarding future work lines after this dissertation. We summarize them in the following:

- In Chapter 3 we have used statistical patch-based methods to segment walls, doors, and windows. Then, the syntactic approach in Chapter 4 uses this segmentation to recognize the real entities of these elements. What we propose is to introduce the patch level representation into the syntactic model. Thus, the contextual knowledge introduced by the probabilistic model would allow to infer contextual relations at patch level thus, enhance both, the initial segmentation and the final structural and semantic representation.

- As we have introduced in Chapter 1, we have chosen architectural drawings as a relevant framework to evaluate our methods on graphical document understanding for two main reasons. Firstly, there is a big variability on the vocabulary notation. Secondly, the amount of information (color, dimension lines, text, etc.) strongly depends on the floor plan intent. Therefore, we have studied how our relational models are able to adapt to different vocabularies and structures within this domain. Yet, in order to asses their generality without a considerable re-engineering, we would need to extrapolate our contributions to other domains, such as flowcharts, maps, etc.

- Finally, we have explained in Chapter 5 that the ontological definition of our domain has helped us in the semantic verification of our GT. In this process, every plan has been checked after its labeling whether it agreed to the domain definition. This process was slow because multiple inconsistencies were found in every plan. Therefore, we had planned and finally we did not implement it because of time constrains to integrate the ontological definition into the SGT tool. In this manner, the tool would help, suggest, and correct the user on real time at labeling time.

Bibliography

[1] IAPR International Workshop on Graphics Recognition. grec2013.loria.fr/GREC2013/, 2013.

[2] Apache Jena - A free and open source Java framework for building Semantic Web and Linked Data applications. https://jena.apache.org/index.html, 2014.

[3] Autodesk Homestyler - Home design and decorating ideas to get inspired and get expert tips. http://www.homestyler.com/, 2014.

[4] Floorplanner - Create floor plans, house plans and home plans on-line with Floorplanner. http://en.floorplanner.com/, 2014.

[5] Google maps indoor. http://www.google.com/maps/about/partners/indoormaps/, 2014.

[6] International Association for Pattern Recognition. http://iapr.org/, 2014.

[7] Mysql: The world s most popular open source database. www.mysql.com, 2014.

[8] OpenRDF Sesame - A de-facto standard framework for processing RDF data. http://www.openrdf.org/, 2014.

[9] Protégé 5: A free, open-source ontology editor and framework for building intelligent systems. http://protege.stanford.edu/, 2014.

[10] RDF: Resource Description Framework. http://www.w3.org/RDF/, 2014.

[11] Technical commitee on graphics recognition. http://iapr-tc10.univ-lr.fr/, 2014.

[12] C. Ah-soon and K. Tombre. Variations on the analysis of architectural drawings. In *In Proceedings of Fourth International Conference on Document Analysis and Recognition*, pages 347 351, 1997.

[13] S. Ahmed, M. Liwicki, M. Weber, and A. Dengel. Improved automatic analysis of architectural floor plans. In *In International Conference on Document Analysis and Recognition*, 2011.

[14] S. Ahmed, M. Liwicki, M. Weber, and A. Dengel. Automatic room detection and room labeling from architectural floor plans. In *10th . IAPR International Workshop on Document Analysis Systems (DAS-2012)*, pages 339 343. IEEE, 2012.

[15] S. Ahmed, M. Weber, M. Liwicki, C. Langenhan, A. Dengel, and F. Petzold. Automatic analysis and sketch-based retrieval of architectural floor plans. *Pattern Recognition Letters*, 35(0):91 100, 2014. Frontiers in Handwriting Processing.

[16] F. Álvaro, J.-A. Sánchez, and J.-M. Benedi. Recognition of printed mathematical expressions using two-dimensional stochastic context-free grammars. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 1225 1229, 2011.

[17] A. Amato, A. D. Sappa, A. Fornés, F. Lumbreras, and J. Lladós. Divide and conquer: Atomizing and parallelizing a task in a mobile crowdsourcing platform. In *Proceedings of the 2Nd ACM International Workshop on Crowdsourcing for Multimedia*, pages 21 22, 2013.

[18] Y. Aoki, A. Shio, H. Arai, and K. Odaka. A prototype system for interpreting hand-sketched floor plans. In *Proceedings of the 13th International Conference on Pattern Recognition*, volume 3, pages 747 751, 1996.

[19] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *Proceedings of the European Conference on Computer Vision*, pages 404 417. 2006.

[20] L. Bertelli, t. Yu, D. Vu, and B. Gokturk. Kernelized structural svm learning for supervised object segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition 2011*, 2011.

[21] M. Bicego, A. Lagorio, E. Grosso, and M. Tistarelli. On the use of sift features for face authentication. In *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW '06. Conference on*, pages 35 35, June 2006.

[22] O. Biller, A. Asi, K. Kedem, J. El-Sana, and I. Dinstein. Webgt: An interactive web-based system for historical document ground truth generation. *12th International Conference on Document Analysis and Recognition*, 0:305 308, 2013.

[23] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(11):1222 1239, 2001.

[24] H. Bunke. Attributed programmed graph grammars and their application to schematic diagram interpretation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 4(6):574 582, Nov. 1982.

[25] H. Bunke and K. Riesen. Recent advances in graph-based pattern recognition with applications in document analysis. *Pattern Recognition*, 44(5):1057 1067, 2011.

[26] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1 27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[27] J. Cherneff, R. Logcher, J. Connor, and N. Patrikalakis. Knowledge-based interpretation of architectural drawings. *Research in Engineering Design*, 3:195 210, 1992.

[28] A. Chhabra. Graphic symbol recognition: An overview. In K. Tombre and A. Chhabra, editors, *Graphics Recognition Algorithms and Systems*, volume 1389 of *Lecture Notes in Computer Science*, pages 68 79. Springer Berlin Heidelberg, 1998.

[29] M. J. Choi, A. Torralba, and A. S. Willsky. Context models and out-of-context objects. *Pattern Recognition Letters*, 33(7):853 862, 2012. Special Issue on Awards from ICPR 2010.

[30] C. Clausner, S. Pletschacher, and A. Antonacopoulos. Aletheia - an advanced document layout and text ground-truthing system for production environments. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 48 52, Sept 2011.

[31] D. Connolly, F. van Harmelen, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. DAML+OIL: Reference Description. http://www.w3.org/TR/daml+oil-reference, 2001.

[32] L. Cordella and M. Vento. Symbol recognition in documents: a collection of techniques? *International Journal on Document Analysis and Recognition*, 3(2):73 88, 2000.

[33] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886 893 vol. 1, 2005.

[34] L.-P. de las Heras, S. Ahmed, M. Liwicki, E. Valveny, and G. Sánchez. Statistical segmentation and structural recognition for floor plan interpretation. *International Journal on Document Analysis and Recognition (IJDAR)*, pages 1 17, 2013.

[35] M. Delalandre, T. Pridmore, E. Valveny, H. Locteau, and E. Trupin. Building synthetic graphical documents for performance evaluation. In *Graphics Recognition. Recent Advances and New Opportunities*, pages 288 298. Springer Berlin/Heidelberg, 2008.

[36] D. Doermann and K. Tombre, editors. *Handbook of Document Image Processing and Recognition*. Springer, 2014.

[37] D. Doermann, E. Zotkina, and H. Li. Gedi - a groundtruthing environment for document images. In *Ninth IAPR International Workshop on Document Analysis Systems*, 2010. Submitted.

[38] P. Dosch and G. Masini. Reconstruction of the 3d structure of a building from the 2d drawings of its floors. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 487 490, 1999.

[39] P. Dosch, K. Tombre, C. Ah-Soon, and G. Masini. A complete system for the analysis of architectural drawings. *International Journal on Document Analysis and Recognition*, 3:102 116, 2000.

[40] A. Dutta. *Inexact Subgraph Matching Applied to Symbol Spotting in Graphical Documents*. PhD thesis, Universitat Autònoma de Barcelona, 2014.

[41] I. El Rube, M. Ahmed, and M. Kamel. Wavelet approximation-based affine invariant shape representation functions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(2):323 327, 2006.

[42] C. Elkan. Using the triangle inequality to accelerate k-means. In *Proceedings of the 20th International Conference on Machine Learning*, pages 147 153, 2003.

[43] S. Escalera, A. Fornes, O. Pujol, A. Escudero, and P. Radeva. Circular blurred shape model for symbol spotting in documents. In *Proceedings of the 26th IEEE International Conference on Image Processing*, pages 2005 2008, 2009.

[44] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88:303 338, 2010.

[45] H. Fahmy and D. Blostein. A survey of graph grammars: theory and applications. pages 294 298, aug. 1992.

[46] D. Fernández, S. Marinai, J. Lladós, and A. Fornés. Contextual word spotting in historical manuscripts using markov logic networks. In *Proceedings of the 2Nd International Workshop on Historical Document Imaging and Processing*, HIP 13, pages 36 43, New York, NY, USA, 2013. ACM.

[47] G. Flitton, T. Breckon, and N. Megherbi Bouallagu. Object recognition using 3d sift in complex ct volumes. In *Proceedings of the British Machine Vision Conference*, pages 11.1 11.12. BMVA Press, 2010. doi:10.5244/C.24.11.

[48] A. Fornés, J. Lladós, J. Mas, J. M. Pujades, and A. Cabré. A bimodal crowdsourcing platform for demographic historical manuscripts. In *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage*, DATeCH 14, pages 103 108, 2014.

[49] B. Fulkerson, A. Vedaldi, and S. Soatto. Class segmentation and object localization with superpixel neighborhoods. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 670 677, 2009.

[50] J. Gibert. *Jaume GibertVector Space Embedding of Graphs via Statistics of Labelling Information*. PhD thesis, Universitat Autònoma de Barcelona, 2012.

[51] T. R. Gruber. A translation approach to portable ontology specification. *Knowledge Acquisition*, 5:199 220, 1993.

[52] C. Ha Lee and T. Kanungo. The architecture of TRUEVIZ: A groundTRUth/metadata Editing and VIsualizing toolkit. Technical report, LAMP, 2001.

[53] J. M. Hammersley and P. E. Clifford. Markov random fields on finite graphs and lattices. Unpublished manuscript, 1971.

[54] S. Harris and A. Seaborne. SPARQL 1.1 Query Language. http://www.w3.org/TR/sparql11-query/, 2013.

[55] T. Heskes. Convexity arguments for efficient minimization of the bethe and kikuchi free energies. *J. Artif. Int. Res.*, 26(1):153 190, 2006.

[56] M. Hilbert and P. López. The worlds technological capacity to store, communicate, and compute information. *Science*, 6025:60 65, 2011.

[57] P. Hitzler, M. Krotzsch, B. Parsia, P. F. Patel-Schneider, and S. Rudolph. OWL 2: Web Ontology Language. http://www.w3.org/TR/owl2-primer/, 2012.

[58] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. http://www.w3.org/Submission/SWRL/, 2004.

[59] M.-K. Hu. Visual pattern recognition by moment invariants. *Information Theory, IRE Transactions on*, 8(2):179 187, 1962.

[60] A. M. J. Lafferty and F. Pareira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.

[61] X. Y. Jiang and H. Bunke. An optimal algorithm for extracting the regions of a plane graph. *Pattern Recogn. Lett.*, 14(7):553 558, 1993.

[62] T. Joachims, T. Finley, and C. J. Yu. Cutting-plane training of structural svms. *Mach. Learn.*, 77(1):27 59, Oct. 2009.

[63] D. Karatzas, S. Robles, and L. Gomez. An online platform for ground truthing and performance evaluation of text extraction systems. In *Proceedings of the International Workshop on Document Analysis Systems*, 2014.

[64] D. Keysers, T. Deselaers, C. Gollan, and H. Ney. Deformation models for image recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29:1422 1435, 2007.

[65] J. Kittler, M. Hatef, R. Duin, and J. Matas. On combining classifiers. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(3):226 239, 1998.

[66] A. Kubias, S. Schenk, S. Staab, and J. Z. Pan. OWL SAIQL - An OWL DL query language for ontology extraction. In *Workshop on OWL: Experiences and directions OWLED-07*, 2007.

[67] J. Llados, J. Lopez-Krahe, and E. Marti. A system to understand hand-drawn floor plans using subgraph isomorphism and hough transform. *Machine Vision and Applications*, 10:150 158, 1997. 10.1007/s001380050068.

[68] J. Lladós, E. Martí, and J. Villanueva. Symbol recognition by error-tolerant subgraph matching between region adjacency graphs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(10):1137 1143, 2001.

[69] J. Lladós, E. Valveny, G. Sánchez, and E. Martí. Symbol recognition: Current advances and perspectives. In D. Blostein and Y.-B. Kwon, editors, *Graphics Recognition Algorithms and Applications*, volume 2390 of *Lecture Notes in Computer Science*, pages 104 128. Springer Berlin / Heidelberg, 2002.

[70] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91 110, Nov. 2004.

[71] T. Lu, H. Yang, R. Yang, and S. Cai. Automatic analysis and integration of architectural drawings. *International Journal on Document Analysis and Recognition*, 9:31 47, 2007.

[72] S. Macé, H. Locteau, E. Valveny, and S. Tabbone. A system to detect rooms in architectural floor plan images. In *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, DAS 10, pages 167 174, 2010.

[73] N. E. Maillot and M. Thonnat. Ontology based complex object recognition. *Image and Vision Computing*, 26(1):102 113, 2008.

[74] S. Marinai. Introduction to document analysis and recognition. in machine learning in document analysis and recognition (pp. 1-20). springer berlin heidelberg. *Machine learning in document analysis and recognition*, pages 1 20, 2008.

[75] J. Mas. *A Syntactic Pattern Recognition Approach based on a Distribution Tolerant Adjacency Grammar and a Spatial Indexed Parser. Application to Sketched Document Recognition.* PhD thesis, Universitat Autònoma de Barcelona, 2010.

[76] M. Moller. Fusion of spatial information models with formal ontologies in the medical domain. *DFKI thesis*, 2011.

[77] K. Nakagawa, A. Fujiyoshi, and M. Suzuki. Ground-truthed dataset of chemical structure images in japanese published patent applications. In *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, DAS 10, pages 455 462, New York, NY, USA, 2010. ACM.

[78] I. Nwogu, Y. Zhou, and C. Brown. An ontology for generating descriptions about natural outdoor scenes. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 656 663, 2011.

[79] J. Ogier, R. Mullot, J. Labiche, and Y. Lecourtier. Semantic coherency: the basis of an image interpretation device-application to the cadastral map interpretation. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 30(2):322 338, 2000.

[80] S.-H. Or, K.-H. Wong, Y.-K. Yu, and M. M.-Y. Chang. Highly automatic approach to architectural floorplan image understanding & model generation. *Proceedings of the Vision, Modeling, and Visualization*, page 2532, 2005.

[81] N. Ouwayed and A. Belaid. A general approach for multi-oriented text line extraction of handwritten document. *International Journal on Document Analysis and Recognition*, 14(4), Sept. 2011.

[82] M. Pagels. DAML - The DARPA Agent Markup Language. www.daml.org, 2006.

[83] I. Phillips and A. Chhabra. Empirical performance evaluation of graphics recognition systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(9):849 870, 1999.

[84] M. Pierrot Deseilligny, R. Mariani, J. Labiche, and R. Mullot. Topographic maps automatic interpretation : Some proposed strategies. In *Graphics Recognition Algorithms and Systems*, volume 1389 of *Lecture Notes in Computer Science*, pages 175 193. Springer Berlin Heidelberg, 1998.

[85] F. Piroi, M. Lupu, A. Hanbury, A. Sexton, W. Magdy, and I. Filippov. Clef-ip 2012: Retrieval experiments in the intellectual property domain. *CLEF 2012 evaluation labs and workshop*, (Online Working Notes), 2012.

[86] S. Pletschacher and A. Antonacopoulos. The page (page analysis and ground-truth elements) format framework. In *Proceedings of the 2010 20th International Conference on Pattern Recognition*, ICPR 10, pages 257 260, Washington, DC, USA, 2010. IEEE Computer Society.

[87] A. Quattoni, M. Collins, and T. Darrell. Conditional random fields for object recognition. In *In NIPS*, pages 1097 1104. MIT Press, 2004.

[88] R. Raveaux. *Graph mining and graph Classification: Application to catastral map analysis.* PhD thesis, University of La Rochelle, 2010.

[89] J. Rendek, G. Masini, P. Dosch, and K. Tombre. The search for genericity in graphics recognition applications: Design issues of the qgar software system. In *Document Analysis Systems VI*, volume 3163 of *Lecture Notes in Computer Science*, pages 366 377. 2004.

[90] M. Richardson and P. Domingos. Markov logic networks. *Mach. Learn.*, 62(1-2):107 136, Feb. 2006.

[91] M. Rusinol, A. Borràs, and J. Lladós. Relational indexing of vectorial primitives for symbol spotting in line-drawing images. *Pattern Recognition Letters*, 31(3):188 201, 2010.

[92] M. Rusinol, L.-P. de las Heras, and O. Terrades. Flowchart recognition for non-textual information retrieval in patent search. *Information Retrieval*, pages 1 18, 2013.

[93] K. Ryall, S. Shieber, J. Marks, and M. Mazer. Semi-automatic delineation of regions in floor plans. In *Proceedings of the Third International Conference on Document Analysis and Recognition*, pages 964 983, 1995.

[94] E. Saund, J. Lin, and P. Sarkar. Pixlabeler: User interface for pixel-level labeling of elements in document images. In *ICDAR*, pages 646 650. IEEE Computer Society, 2009.

[95] F. Schneider and O. Heckmann. *Grundrißatlas Wohnungsbau / Floor Plan Manual Housing.* Birkhauser, 2012.

[96] S. Se, D. Lowe, and J. Little. Vision-based global localization and mapping for mobile robots. *Robotics, IEEE Transactions on*, 21(3):364 375, June 2005.

[97] K. Tombre, S. Tabbone, L. Pélissier, B. Lamiroy, and P. Dosch. Text/graphics separation revisited. In *Document Analysis Systems V*, Lecture Notes in Computer Science, pages 615 620. 2002.

[98] S. Tongphu, B. Suntisrivaraporn, B. Uyyanonvara, and M. Dailey. Ontology-based object recognition of car sides. In *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2012 9th International Conference on*, pages 1 4, 2012.

[99] E. Valveny, M. Delalandre, R. Raveaux, and B. Lamiroy. Report on the symbol recognition and spotting contest. In *Graphics Recognition. New Trends and Challenges*, volume 7423 of *Lecture Notes in Computer Science*, pages 198 207. 2013.

[100] E. Valveny and P. Dosch. Symbol recognition contest: A synthesis. In *Graphics Recognition. Recent Advances and Perspectives*, volume 3088 of *Lecture Notes in Computer Science*, pages 368 385. 2004.

[101] M. Weber, C. Langenhan, T. Roth-Berghofer, M. Liwicki, A. Dengel, and F. Petzold. a.SCatch: Semantic Structure for Architectural Floor Plan Retrieval. In *18th International Conference on Case-Based Reasoning*, pages 510 524, 2010.

[102] R. Wessel, I. Blumel, and R. Klein. The room connectivity graph: Shape retrieval in the architectural domain. In *Proceedings of the 16th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, 2008.

[103] S. Yacoub, V. Saxena, and S. Sami. Perfectdoc: a ground truthing environment for complex documents. In *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*, pages 452 456 Vol. 1, Aug 2005.

[104] S. Yang. Symbol recognition via statistical integration of pixel-level constraint histograms: a new descriptor. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(2):278 281, 2005.

[105] B. Yanikoglu and L. Vincent. Pink panther: A complete environment for ground-truthing and benchmarking document page segmentation. *Pattern Recognition*, 31:1191 1204, 1998.

[106] D. Zhang and G. Lu. Review of shape representation and description techniques. *Pattern Recognition*, 37(1):1 19, 2004.

[107] G. Zhi, S. Lo, and Z. Fang. A graph-based algorithm for extracting units and loops from architectural floor plans for a building evacuation model. *Computer-Aided Design*, 35(1):1 14, 2003.

[108] S. Zhu and D. Mumford. Stochastic grammar of images. *Foundations and Trends*, 2006.