Inexact Subgraph Matching Applied to Symbol Spotting in Graphical Documents

A dissertation submitted by **Anjan Dutta** at Universitat Autònoma de Barcelona to fulfil the degree of **Doctor of Philosophy**.

Director: **Dr. Josep Lladós Canet**.
Departament: Ciències de la Computació, Escola d'Enginyeria, UAB.
PhD Program: Informàtica.

Bellaterra, March 24, 2014

Director	**Dr. Josep Lladós Canet** Dept. Ciències de la Computació & Centre de Visió per Computador Universitat Autònoma de Barcelona
Thesis committee	**Prof. Dr. Jean-Marc Ogier** Laboratoire Informatique, Image et Interaction Université de La Rochelle **Dr. Ernest Valveny** Dept. Ciències de la Computació & Centre de Visió per Computador Universitat Autònoma de Barcelona **Dr. Pierre Héroux** Laboratoire LITIS Université de Rouen **Dr. Francesc Serratosa** Enginyeria Informàtica i Matemàtiques Universitat Rovira i Virgili **Dr. Maria Vanrell** Dept. Ciències de la Computació & Centre de Visió per Computador Universitat Autònoma de Barcelona
European evaluators	**Prof. Dr. Luc Brun** École Nationale Supérieur d'Ingénieurs de Caen Université de Caen Basse-Normandie **Prof. Dr. Jean-Yves Ramel** Polytech'Tours - LI Tours Université de François - Rabelais, Tours

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:

Printed by Ediciones Gráficas Rey, S.L.

To my parents...

An idea that is not dangerous is unworthy of being called as an idea at all
Oscar Wilde (1854 - 1900)

Highly organized research is guaranteed to produce nothing new
Frank Herbert (1920 - 1986)

Acknowledgement

I would like to express my gratitude to some people, organizations and institutions who have supported me during these years of my thesis and without them this thesis would have not been completed.

First of all I would like to thank the Centre de Visiò per Computador (CVC) and the Universitat Autònoma de Barcelona (UAB) for allowing me to pursue my doctoral study and for providing me all the facilities. I would like to thank the Agéncia de Gestio d'Ajuts Universitats i Recerca (AGAUR) for providing me a three years PhD scholarship (2011_FIB 01022, 2012FI_B1 00174 and 2013FI_B2 00074) and a mobility scholarship (2011 BE1 00169) for doing a three months research stay abroad, without that for sure it would have not been possible.

I would like to gratefully and sincerely thank my supervisor Dr. Josep Lladós, first of all for giving me an opportunity to do PhD and also for his guidance, understanding, patience and most importantly, his friendship during this long journey. I am not sure whether many PhD students are given the opportunity to develop their own individuality and self-sufficiency by being allowed to work with such independence. For everything you have done for me, Josep, I thank you so much.

I would like to specially thank my co-supervisor Dr. Umapada Pal, from whom I got the inspirations for doing PhD. Thank you very much Sir for encouraging me and extending me to such opportunities.

My special thanks to Prof. Em. Dr. Horst Bunke for having lot of fruitful discussions, meetings, comments and ideas. Thank you very much Horst, without you my thesis would have not been completed.

My sincere thanks to Prof. Dr. Xiaoyi Jiang of University of Münster, Germany, who gave me the opportunity of spending my research stay in the Computer Vision and Pattern Recognition Group where I have learnt lot of important things.

I would like to thank Prof. Dr. Koichi Kise and Dr. Masakazu Iwamura of Osaka Prefecture University, Japan for giving me an opportunity of doing a research stay there, where I have learnt some nice things.

During my Ph.D. I was fortunate to have several collaborations. I would like to thank Klaus Broelemann of University of Münster, for collaborating with me. I would like to thank Lluis-Pere de las Heras, David Fernández, Dr. Joan Mas, Dr. Gemma Sanchez, Dr. Dimosthenis Karatzas, Dr. Oriol Ramos Terrades, Dr. Marçal Rusinyol, Dr. Alicia Fornés, Dr. Albert Gordo of CVC for conducting fruitful collaborations with me. I specially thank Alicia for collaborating with me from the beginning of my days in CVC. Even my first paper from CVC was also resulted from the collaboration

with her, thank you so much.

I express my gratitude to all my friends from CVC, especially, the ones who started doing PhD with me: Jon, David, Lluis, Toni. Also Fran, Camp, Ivet, Joan, Yainuvis and Alejandro with whom I have shared so many talks, lunches, coffees, parties and now I consider as very good friends. Thank you so much guys, with your presence, the feeling of the absence of my family in far India was minimized. My special thanks to Bhaskar, Partha, Naveen, Elena, Carlos, Martin, Arindam, Pragna and Suman for their company during different time in my whole stay in Barcelona.

I cordially thank to all the administrative and technical supporting personnel of CVC for being patience in front of my extensive requirements, specially, Montse, Gigi, Claire, Marc for helping me out of any office and technical work. For sure without you it would have been much much difficult.

Finally my thanks, love and respect to those persons, very close to my heart, my family in India. My thanks and regards to my parents for their unequivocal support throughout, as always, for which my mere expression of thanks would never be enough. My special thanks to Strutti for understanding me and extending her love, care and support everyday.

Abstract

There is a resurgence in the use of structural approaches in the usual object recognition and retrieval problem. Graph theory, in particular, graph matching plays a relevant role in that. Specifically, the detection of an object (or a part of that) in an image in terms of structural features can be formulated as a subgraph matching. Subgraph matching is a challenging task. Specially due to the presence of outliers most of the graph matching algorithms do not perform well in subgraph matching scenario. Also exact subgraph isomorphism has proven to be an NP-complete problem. So naturally, in graph matching community, there are lot of efforts addressing the problem of subgraph matching within suboptimal bound. Most of them work with approximate algorithms that try to get an inexact solution in approximate way. In addition, usual recognition must cope with distortion. Inexact graph matching consists in finding the best isomorphism under a similarity measure. Theoretically this thesis proposes algorithms for solving subgraph matching in an approximate and inexact way.

We consider the symbol spotting problem on graphical documents or line drawings from application point of view. This is a well known problem in the graphics recognition community. It can be further applied for indexing and classification of documents based on their contents. The structural nature of this kind of documents easily motivates one for giving a graph based representation. So the symbol spotting problem on graphical documents can be considered as a subgraph matching problem. The main challenges in this application domain is the noise and distortions that might come during the usage, digitalization and raster to vector conversion of those documents. Apart from that computer vision nowadays is not any more confined within a limited number of images. So dealing a huge number of images with graph based method is a further challenge.

In this thesis, on one hand, we have worked on efficient and robust graph representation to cope with the noise and distortions coming from documents. On the other hand, we have worked on different graph based methods and framework to solve the subgraph matching problem in a better approximated way, which can also deal with considerable number of images. Firstly, we propose a symbol spotting method by hashing serialized subgraphs. Graph serialization allows to create factorized substructures such as graph paths, which can be organized in hash tables depending on the structural similarities of the serialized subgraphs. The involvement of hashing techniques helps to reduce the search space substantially and speeds up the spotting procedure. Secondly, we introduce contextual similarities based on the walk based propagation on tensor product graph. These contextual similarities involve higher

order information and more reliable than pairwise similarities. We use these higher order similarities to formulate subgraph matching as a node and edge selection problem in the tensor product graph. Thirdly, we propose near convex grouping to form near convex region adjacency graph which eliminates the limitations of traditional region adjacency graph representation for graphic recognition. Fourthly, we propose a hierarchical graph representation by simplifying/correcting the structural errors to create a hierarchical graph of the base graph. Later these hierarchical graph structures are matched with some graph matching methods. Apart from that, in this thesis we have provided an overall experimental comparison of all the methods and some of the state-of-the-art methods. Furthermore, some dataset models have also been proposed.

Resumen

Existe un resurgimiento en el uso de métodos estructurales para el problema de reconocimiento y recuperación por contenido de objetos en imágenes. La teoría de grafos, en particular la puesta en correspondencia de grafos (graph matching) juega un papel relevante en ello. Así, la detección de un objeto (o una parte) en una imagen se puede formular como un emparejamiento de subgrafos en términos de características estructurals. El matching de subgrafos es una tarea difícil. Especialmente debido a la presencia de valores atípicos, muchos de los algoritmos existentes para el matching de grafos tienen dificultades en el escenario de matching de subgrafos. Además, el apareamiento de subgrafos de manera exacta ha demostrado ser una problema NP-completo . Así que hay una actividad intensa en la comunidad científica para proporcionar algoritmos eficaces para abordar el problema de manera suboptimal. La mayoría de ellos trabajan con algoritmos aproximados que tratan de obtener una solución inexacta en forma aproximada. Además, el reconocimiento habitualmente debe hacer frente a la distorsión. El emparejamiento de subgrafos de manera inexacta consiste en encontrar el mejor isomorfismo bajo una medida de similitud. Desde el punto de vista teórico, esta tesis propone algoritmos para la solución al problema del emparejamiento de subgrafos de manera aproximada e inexacta.

Desde un punto de vista aplicado, esta tesis trata el problema de la detección de símbolos en imágenes de documentos gráficos o dibujos lineales (symbol spotting). Este es un problema conocido en la comunidad de reconocimiento de gráficos. Se puede aplicar para la indexación y clasificación de documentos sobre la base de sus contenidos. El carácter estructural de este tipo de documentos motiva de forma natural la utilización de una representación de grafos. Así el problema de detectar símbolos en documentos gráficos puede ser considerado como un problema de apareamiento de subgrafos. Los principales desafíos en este dominio de aplicación son el ruido y las distorsiones que provienen del uso, la digitalización y la conversión de raster a vectores de estos documentos. Aparte de que la visión por computador en la actualidad no limita las aplicaciones a un número limitado de imágenes. Así que el paso a la escala y tratar un gran número de imágenes en el reconocimiento de documentos gráficos es otro desafío.

En esta tesis, por una parte, hemos trabajado en representaciones de grafos eficientes y robustas para solucionar el ruido y las distorsiones de los documentos. Por otra parte, hemos trabajado en diferentes métodos de matching de grafos para resolver el problema del emparejamiento inexacto de subgrafos, que también sea escalable ante un considerable número de imágenes. En primer lugar, se propone un método para de-

tectar símbolos mediante funciones de hash de subgrafos serializados. La organización del grafo una vez factorizado en subestructuras comunes, que se pueden organizar en tablas hash en función de las similitudes estructurales, y la serialización de las mismas en estructuras unidimensionales como caminos, son dos aportaciones de esta parte de la tesis. El uso de las técnicas de hashing ayuda a reducir sustancialmente el espacio de búsqueda y acelera el procedimiento de la detección. En segundo lugar, presentamos mecanismos de similitud contextual basadas en la propagación basada en caminos (walks) sobre el grafo producto (tensor product graph). Estas similitudes contextuales implican más información de orden superior y más fiable que las similitudes locales. Utilizamos estas similitudes de orden superior para formular el apareamiento de subgrafos como una problema de selección de nodos y aristas al grafo producto. En tercer lugar, proponemos agrupamiento perceptual basado en convexidades para formar regiones casi convexas que elimina las limitaciones de la representación tradicional de los grafos de regiones para el reconocimiento gráfico. En cuarto lugar, se propone una representación de grafo jerárquico mediante la simplificación/corrección de los errores estructurales para crear un grafo jerárquico del grafo de base. Estos estructuras de grafos jerárquicos integran en métodos de emparejamiento de grafos. Aparte de esto, en esta tesis hemos proporcionado una comparación experimental general de todos los métodos y algunos de los métodos del estado del arte. Además, también se han proporcionado bases de datos de experimentación.

Resum

Existeix un ressorgiment en l'ús de mètodes estructurals per al problema de reconeixement i recuperació per contingut d'objectes en imatges. La teoria de grafs, en particular, la posada en correspondència de grafs (graph matching) juga un paper rellevant en això. En particular, la detecció d'un objecte (o una part) en una imatge es pot formular com un aparellament de subgrafs en termes de característiques estructurals. El matching de subgrafs és una tasca difícil. Especialment a causa de la presència de valors atípics, molts dels algoritmes existents per al matching de grafs tenen dificultats en l'escenari de matching de subgrafs. A més, l'aparellament de subgrafs de manera exacta s'ha demostrat ser una problema NP-complet. Així que hi ha una activitat intensa a la comunitat científica per proporcionar algoritmes eficaços per abordar el problema de manera suboptimal. La majoria d'ells treballen amb algoritmes aproximats que tracten d'obtenir una solució inexacta en forma aproximada. A més, el reconeixement habitualment ha de fer front a la distorsió. L'aparellament de subgrafs de manera inexacta consisteix a trobar el millor isomorfisme sota una mesura de similitud. Des del punt de vista teòric, aquesta tesi proposa algoritmes per a la solució al problema de l'aparellament de subgrafs de manera aproximada i inexacta.

Des d'un punt de vista aplicat, aquesta tesi tracta el problema de la detecció de símbols en imatges de documents gràfics o dibuixos lineals (symbol spotting). Aquest és un problema ben conegut a la comunitat de reconeixement de gràfics. Es pot aplicar per a la indexació i classificació de documents sobre la base dels seus continguts. El caràcter estructural d'aquest tipus de documents motiva de forma natural la utilització d'una representació de grafs. Així el problema de detectar símbols en documents gràfics pot ser considerat com un problema d'aparellament de subgrafs. Els principals desafiaments en aquest domini d'aplicació són el soroll i les distorsions que provenen de l'ús, la digitalització i la conversió de ràster a vectors d'aquests documents. A part que la visió per computador en l'actualitat no limita les aplicacions a un nombre limitat d'imatges. Així que el pas a l'escala i tractar un gran nombre d'imatges en el reconeixement de documents gràfics és un altre desafiament.

En aquesta tesi, d'una banda, hem treballat en representacions de grafs eficients i robustes per solucionar el soroll i les distorsions dels documents. D'altra banda, hem treballat en diferents mètodes de matching de grafs per resoldre el problema de l'aparellament inexacte de subgrafs, que també sigui escalable davant d'un considerable nombre d'imatges. En primer lloc, es proposa un mètode per a detectar símbols mitjançant funcions de hash de subgrafs serialitzats. L'organització del graf

una vegada factoritzat en subestructures comunes, que es poden organitzar en taules hash en funció de les similituds estructurals, i la serialització de les mateixes en estructures unidimensionals com ara camins, són dues aportacions d'aquesta part de la tesi. L'ús de les tècniques de hashing ajuda a reduir substancialment l'espai de cerca i accelera el procediment de la detecció. En segon lloc, presentem mecanismes de similitud contextual basades en la propagació basada en camins (walks) sobre el graf producte (tensor product graph). Aquestes similituds contextuals impliquen més informació d'ordre superior i més fiable que les similituds locals. Utilitzem aquestes similituds d'ordre superior per formular l'aparellament de subgrafs com una problema de selecció de nodes i arestes al graf producte. En tercer lloc, proposem agrupament perceptual basat en convexitats per formar regions quasi convexes que elimina les limitacions de la representació tradicional dels grafs de regions per al reconeixement gràfic. En quart lloc, es proposa una representació de graf jeràrquic mitjançant la simplificació/correcció dels errors estructurals per crear un graf jeràrquic del graf de base. Aquests estructures de grafs jeràrquics s'integren en mètodes d'aparellament de grafs. A part d'això, en aquesta tesi hem proporcionat una comparació experimental general de tots els mètodes i alguns dels mètodes de l'estat de l'art. A més, també s'han proporcionat bases de dades d'experimentació.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This thesis work has two basic aspects: (1) theoretical and (2) applied. From theoretical point of view, it mostly addresses the approximate inexact subgraph matching algorithms: improvements in terms of performance, error tolerance, time complexity and large scale compatibility. And from application point of view, it addresses a typical problem of *document image analysis* (DIA): symbol spotting in graphical documents. By nature, the graphical documents can be represented by graphs in a robust way but there are issues concerning distortions, noise etc. So from applied perspective, this work considers symbol spotting problem as a subgraph matching problem and proposes different graph representation strategies for dealing with distortions, noise etc.

1.1 Graph matching in computer vision and pattern recognition

Let's start giving a tentative answer of the question: **What is pattern recognition (PR)?** Among all the existing answers, the one that fits best with the thesis and also to this chapter is: "Pattern recognition is the scientific discipline of machine learning (or artificial intelligence) that aims at classifying patterns (or data) into a number of categories or classes". **But what is a pattern?**

In 1985, Satoshi Watanabe [103] defined a pattern as "the opposite of chaos; it is an entity, vaguely defined, that could be given a name". In other words, a pattern can be any entity of interest which one needs to or has interest to recognise and/or identify: it is so worthy that one would like to know its name (its identity). Examples of patterns are: a pixel in an image, a 2D or 3D shape, a typewritten or handwritten character, the gait of an individual, a gesture etc.

In general, a pattern recognition system is an automatic system that aims at classifying the input pattern into a specific class. It proceeds into two successive

tasks: (1) the *analysis* (or description) that extracts the characteristics from the pattern being studied and (2) the *classification* (or recognition) that enables us to recognise an object (or a pattern) by using some characteristics derived from the first task.

Through several decades the pattern recognition community has been trying to use machines for solving various problem from various field. These include optical character recognition [3], mail sorting [102], text categorization [47], handwritten text recognition [75], writer identification [88], molecular structure analysis [36], fingerprint recognition [29], face detection and recognition [7], image classification [9], action recognition [76] and classification [71] and many more.

Methodologically, the field of pattern recognition is usually categorized into the statistical and the structural approach. Structural pattern recognition allows one to use powerful and flexible representation formalisms but offers only a limited repertoire of algorithmic tools needed to solve classification and clustering problems. By contrast, the statistical approach is mathematically well founded and offers many tools, but provides a representation formalism that is limited in its power and flexibility. Hence, both the subfields are complementary to each other.

Graphs, strings are typical representation paradigm from the structural pattern recognition community and widely used to represent complex structure in computer vision and pattern recognition applications. The description of a visual object can be further enhanced by an attributed relation between the nodes and edges. An *attributed relational graphs (or strings)* are the graphs (or strings) whose nodes and edges are associated by labels and parametric values which express some properties of the object to represent. So, for instance, the two dimensional information in any image plane can be well represented by the attributed graphs (see Figure 1.1). Moreover, graph matching is a suitable strategy for the comparison, detection and recognition of structural information. In general graphs can be matched by comparing the nodes and the edges between them.

In computer vision and pattern recognition, the research community has divided the graph matching methods into two broad categories: (1) *exact graph matching* and (2) *inexact graph matching*. The exact matching requires a strict correspondence among the two objects being matched or at least among their sub parts. Some examples of exact algorithms are [17, 20]. The inexact graph matching methods allow some tolerance and in this case a matching can occur where two graphs being matched can be structurally different to some extent. Some examples of inexact algorithms are [17, 18, 104].

On the other hand, graph matching is proved to be an NP-hard problem [62]. For that reason, in the research community there are substantial efforts for approximating the graph based algorithms so that a better solution can be obtained within a reasonable time complexity [16].

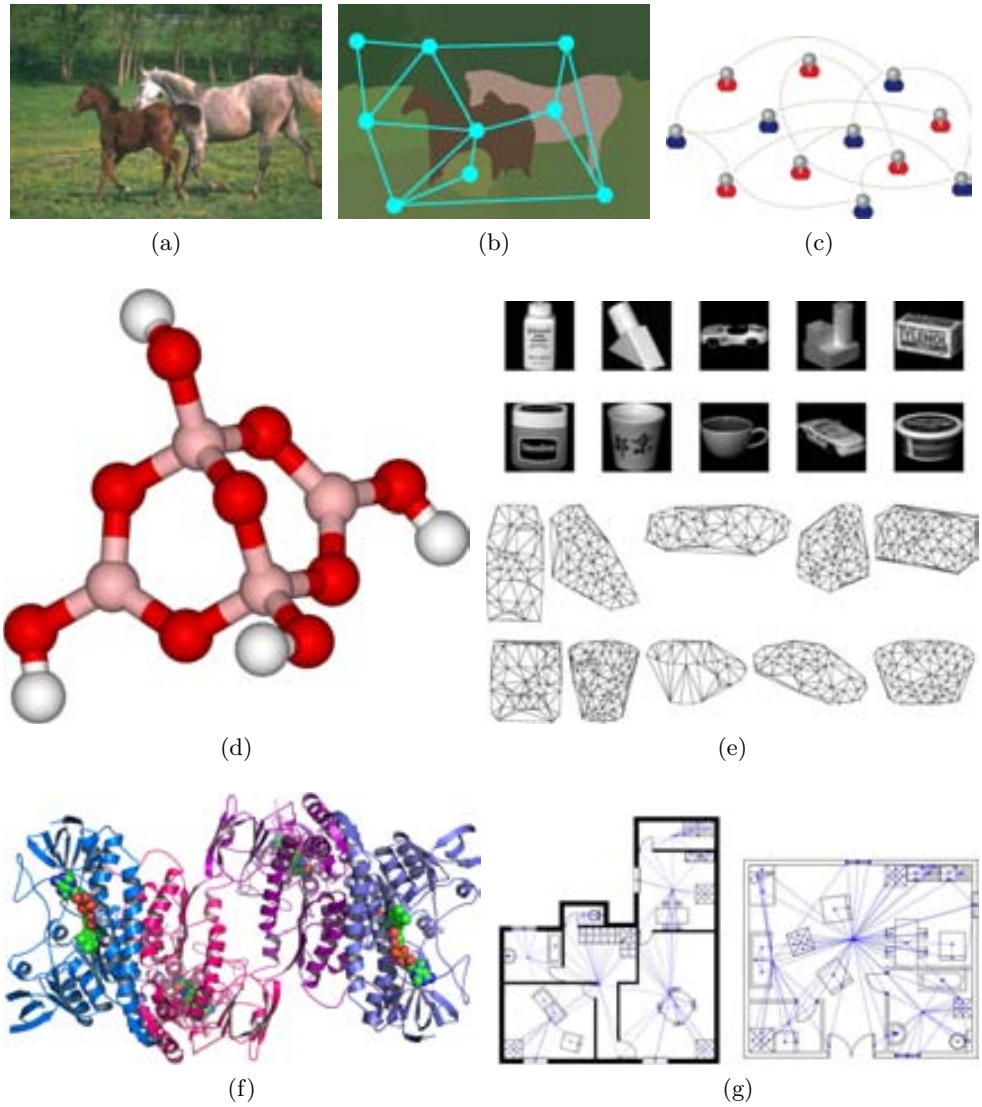


Figure 1.1: Graphs are everywhere: (a)-(b) natural scene, (c) social network, (d) protein structure, (e) visual objects, (f) protein structures and (g) graphical documents. Any visual objects can be represented by graphs. (Figure credit: (a)-(b) Harchaoui and Bach [39], (c)-(d), (f) Google images, (e) Han *et al.* [38], (g) Le Bodic *et al.* [8]).

1.2 Graphics recognition and Focused retrieval

Even after the extreme advancement of the present digital era, paper document still poses an important contribution in our regular workflow. Digitalization of those documents is justified for the portability and preservation issues. However, developing a system for browsing and querying collection of digital documents in an effective way still remains as a big challenge.

Graphics recognition is a subfield of DIA, where the research community mainly deals with graphics-rich documents, for example, maps, architectural drawings, engineering drawings, circuit diagrams and schematics etc. The main research activity concerns to propose methods and algorithms that can solve various analysis and recognition problems in graphical documents, viz., raster-to-vector conversion, text/graphics separation, symbol spotting and recognition etc. As the other stream of computer vision, neither DIA any more confined with limited number of images. It often encounters lot of digital or digitized information in poorly formatted way. So the main challenge lies in enriching the poorly structured information and add semantics.

Information spotting is a major branch of graphic recognition as well as of *information retrieval* (IR) methods. It can be defined by locating a given query information in a large collection of relevant data. In DIA, the research community is mainly focused on word spotting [57,78] for textual documents and symbol spotting for graphic rich documents [97]. Here it is to be mentioned that textual information can also be given a symbolic representation and approached by a symbol spotting technique. The main application of symbol spotting is indexing and retrieval into large database of graphical document images, e.g. finding a mechanical part into a database of engineering drawings or a logo into a database of invoices or administrative documents etc.

In general, *symbol spotting* can be defined as locating a given query symbol in a large graphical document or a database of graphical document without explicitly recognizing it. The desired output should be a ranked list of regions of interest likely to contain the similar shapes to the queried one. Symbol spotting can be considered as a specific application of *content based image retrieval* (CBIR), but differs in certain ways. The main difference is that any standard retrieval approaches retrieves the atomic documents leaving to the user the task of locating the real relevant information within the provided results, whereas symbol spotting methodologies give a direct access to the relevant information. The methods which give passages of interest within the document instead of the whole document are called *focused retrieval* (FR) [48]. Additionally, spotting systems are usually queried by example. That is, the user crops an object he wants to retrieve from the document database and this cropped image acts as the input query of the spotting system. This prevents the spotting methods to work on a specific set of model symbols and not to have an explicit learning stage where the relevant features describing a certain model symbol can be trained.

1.3 Motivation

Graphs are very effective tool to represent any visual objects and graph representation can efficiently handle various affine transformations viz. rotation, translation, scaling which is very useful for our purpose. Moreover, graphs have been widely adapted by the research community as a robust tool since a long back, as a result lots of efficient methods and algorithms are available to handle the graph based methods.

In spite of the progresses in the domain graph based methods in pattern recognition, a problem of the approximate and inexact methods still exists resulting from the consideration of outliers. This problem is often treated as a *subgraph matching* problem which recognizes one graph as part of the other. This is justified since in realistic computer vision applications due to the existence of background, occlusions and geometric transformation of objects, outliers do exist in lot of scenarios. Subgraph matching is a very difficult task; exact subgraph matching is proved to be an NP-complete problem. Moreover the issues concerning outliers and error tolerance in case of computer vision have made it more challenging. And for that reason it has drawn a huge attention of the research community and lot of research activity is going on addressing it. There are lot of work addressing the problem of graph matching with considerable effort for reducing the time complexity and making accurate approximation to obtain better results [16]. But not many of them work in subgraph matching scenarios due to the issues concerning outliers/background etc [107].

In this work we have concentrated on symbol spotting in graphical documents, mostly, in architectural line drawings and consider it as a subgraph matching problem. The main motivation behind it is the nature of the graphical documents, which usually have a relationship among the adjacent or nearby parts.

As being a part of real world entity, the graphical documents often suffer from noise. Modelling these distortions and noise in graph level is a difficult job and demands research activity. Also with the increase of improved imaging device the effort for digitalizing documents has been increased, as a result DIA is no more confined within a limited number of images. Graph based algorithms usually have a huge time complexity, so handling a lot of data is further difficult job and also requires investigation.

So the main motivation of this thesis is twofold. First comes from graph matching point of view, where we work on faster approximate graph matching algorithms. The second comes from the representation point of view where we address the difficulties representing graphical documents due to noise, distortions etc and work on efficient graph representation for having stable representation paradigm in presence of deformation.

1.4 Objectives and Contributions

The main aim of this thesis work is twofold. First, theoretically, the main objective is to propose an approximate subgraph matching algorithm that works in manageable time complexity and can deal with considerable number of visual data. Second, from application point of view, we consider the symbol spotting problem in graphical documents as a subgraph matching problem. As these documents are real instances, they contain lot of noise and distortions which demand efficient representation paradigm.

To reach the main aim we have defined the following atomic objectives:

1. Inexact subgraph matching: Since the graph representation of real world object employs the statistical description of object subparts, exact matching algorithms do not fit with realistic applications. So to propose a subgraph matching based symbol spotting method, we aim to work with inexact subgraph matching algorithms.
2. Error tolerance: Most of the realistic data to some extend contains noise and/or distortions, graphical documents are no exception of that. So the graph representations should aim to tolerate the noise, distortions in the data.
3. Scalability: With the increase of improved imaging device the effort for digitalizing documents has been increased, as a result DIA is no more confined within a limited number of images. So to solve the problem of symbol spotting with subgraph matching we aim to emphasise on the scalability issue.
4. Time complexity: As an application symbol spotting can be used for quick indexing, retrieval and classification of graphical documents. For that reason methodologically it demands a very fast processing of queries. So all the algorithms should aim time efficiency.

To meet the above objectives, we have made the following contributions in this thesis:

1. Symbol spotting by serialized subgraphs: Graph/Subgraph matching is considered as a computationally hard problem, so dealing a large collection of graphical documents directly with graph is not efficient. Factorizing a graph and then finding the common factorized substructures for the whole database is effective since it can considerably reduce the computational burden for the whole database. We factorize the graphs by finding the acyclic graph paths between each pair of connected nodes. Graph paths are serialized substructures, which are efficient in terms of computation. At the same time the factorized parts are helpful for recognizing partial/superimposed contents which is good for error toleration. This work is presented in Chapter 4.
2. Product graph based inexact subgraph matching: A particular type of state-of-the-art algorithms formulate the subgraph matching problem as an optimization problem where they mainly work with pairwise (dis)similarities of the node and edge attributes, which is not reliable in many cases. In this work we propose

a walk based way on tensor product graph to obtain higher order contextual similarities and formulate subgraph matching as a node and edge selection problem in the tensor product graph. Moreover, we propose a dual edge graph representation which aims at tolerating noise and distortions in line drawings. This work is discussed in Chapter 5.

3. Near convex region adjacency graph: Region based representation is a typical type of representation where the information is represented by connecting the adjacent regions. This kind of graph is called region adjacency graph (RAG). But often all the necessary information cannot be represented with bounded regions. It has been studied that mostly the interesting information can be represented with convex regions or a combination of them. In this work we have proposed a near convex grouping to create near convex regions and construct a near convex region adjacency graph with them which solve the problem of RAG. This work is explained in Chapter 6.
4. Hierarchical graph representation: Graph representation of graphical documents often suffer from noise viz. spurious nodes, spurious edges and their discontinuity etc. In general this kind of structural errors occur during the low-level image processing viz. binarization, skeletonization, vectorization etc while transforming documents to graphs. In this work we solve this problem by hierarchically merging/simplifying node-node and node-edge depending on the distance. We introduce plausibilities to different simplifications and later use these plausibility values while matching two hierarchical structures. This work is described in Chapter 7.

1.5 Outline

The rest of the thesis is organized as follows:

- Chapter 2 presents the definitions, notations of graph theory, particularly the ones that are necessary for our work to be described later. After that we present a review of the state of the art methods for graph and subgraph matching.
- Chapter 3 presents a detailed review of the state of the art methods for symbol spotting. This chapter classifies the symbol spotting methods into five broad classes and gives an overview, pros and cons and examples of different category of methods.
- Chapter 4 introduces a symbol spotting method by factorizing the graph representing the graphical information in documents. The idea of graph factorization to serialized subgraphs (graph paths) is introduced here. Locality sensitive hashing (LSH) is used for creating a indexation structure for later fast retrieval.
- Chapter 5 presents a subgraph matching method using product graph and a symbol spotting methodology is proposed using that. To cope with the structural errors a new dual graph based representation is proposed which is proved to be effective in graphical documents.

- Chapter 6 presents a region based graph representation to solve the limitations of the approach encountered by factorization based method described in Chapter 4. This chapter introduces near convex region adjacency graph (NCRAG) which solves the limitations of the basic region adjacency graph (RAG) in graphical documents.
- Chapter 7 addresses the structural errors encountered in the graphical documents. To solve the problem it proposes a hierarchical graph representation which can correct the errors/distortions in hierarchical steps.
- Chapter 8 presents a unified experimental evaluation of all the proposed methods and comparisons with some state of the art methods.
- Chapter 9 concludes the thesis and defines the future direction of the work.
- App. A provides a brief description on the datasets that we have used in this thesis work.

Chapter 2

Graph Matching

Graph is a powerful tool for any visual object representation. In this chapter, we will discuss some of the key definitions regarding graphs. Graph comparison is a crucial operation and it is needed for the sake of comparison of two objects represented with graphs. This process can roughly be defined as *graph matching*. In pattern recognition, the methods of graph matching are broadly divided into two categories viz. *exact* and *inexact*. In this chapter we will discuss the concepts of exact and inexact graph matching techniques. Also some of the key graph comparison approaches such as graph edit distance, graph indexing, graph kernel and some related concepts will be reviewed. As this thesis methodologically focuses on subgraph matching, at the end of this chapter we will give a brief overviews and state-of-the-art reviews on the other subgraph matching algorithms. Some of the definitions, concepts in this chapter are written inspired from the PhD thesis of Dr. Jaume Gibert [32].

2.1 Graphs and Subgraphs

Definition 2.1 (Graph)**.** Let L_V and L_E be any two sets of labels. An *attributed graph* is a 4-tuple $G = (V, E, \alpha, \beta)$, where V is a finite set of nodes or vertices, $E \subseteq V \times V$ is the set edges, $\alpha : V \to L_V$ is the node labelling function assigning a label from the set L_V to each of the nodes in V and $\beta : E \to L_E$ is the edge labelling function assigning a label from the set L_E to each of the edges in E.

The number of nodes of a graph G is denoted by $|V|$. Edges of a graph are usually identified by the pair of nodes they link. An edge $e \in E$ can thus be represented as $e = (u, v)$, where $u, v \in V$ are the nodes joined by the edge e. The number of edges in a graph G is denoted by $|E|$. Based on the definitions of the labelling sets L_V, L_E and the labelling functions α, β, there also exist different types of graphs. For instance, the graphs whose labelling sets are sets of discrete values is called *discretely attributed graphs*. On the other hand the graphs whose labelling sets are any subset

of $\mathbb{R}^d$ for $d > 1$, are called *continuously attributed graphs*. Moreover, L_V and/or L_E can be empty sets, in that case the graphs are named as *unattributed graphs*.

Definition 2.2 (Directed Graph)**.** A *directed graph* is a graph where all the edges have a specific direction. For example, by the edge $e = (u, v)$, we mean the edge e is originated at the node u and terminated at the node v. The existence of an edge $e = (u, v)$ does not assure the existence of the edge $e' = (v, u)$.

Definition 2.3 (Undirected Graph)**.** An *undirected graph* is a graph where for any edge $e = (u, v)$, there always exist an edge $e' = (v, u)$ such that $\beta(e) = \beta(e')$.

Definition 2.4 (Subgraph)**.** Let $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ and $G_2 = (V_2, E_2, \alpha_2, \beta_2)$ be two graphs. Then the graph G_1 is said to be a *subgraph* of G_2 and is denoted by $G_1 \subseteq G_2$ if the following conditions hold.

- $V_1 \subseteq V_2$
- $E_1 = E_2 \cap V_1 \times V_2$
- $\alpha_1(u) = \alpha_2(u), \forall u \in V_1$
- $\beta_1(e) = \beta(e), \forall e \in E_1$

From this definition, a subgraph of a graph can be obtained by removing some nodes and all their incident edges. In this case the subgraph is called *induced*. If, however, the second condition of the definition is substituted by $E_1 \subseteq E_2$, not only the incident edges to the deleted nodes are removed but also some other edges have to be removed. In this case the subgraph is called *non-induced*.

2.2 Graph matching

Given two attributed graphs, *graph matching* can roughly be defined as a process for finding correspondence between the node and edge sets of two graphs that satisfies some constraints. In pattern recognition, the research community has divided the graph matching methods into two broad categories: (1) exact and (2) inexact graph matching. The *exact matching* requires a strict correspondence among the two objects being matched or at least among their sub parts. On the other hand the *inexact graph matching* methods allow some tolerance for matching and in this case a matching can occur where two graphs being matched can be structurally different to some extent. This kind of graph matching also called as *error tolerant subgraph matching*.

As in all subfields of mathematics, also in graph theory the relation between two objects or entities can be established in terms of a mapping. Depending on the nature of mapping graph matching can also be of different types: *graph isomorphism*, *graph monomorphism* and *graph homomorphism*, where graph isomorphism can be defined as follows:

Definition 2.5 (Graph Isomorphism)**.** Let $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ and $G_2 = (V_2, E_2, \alpha_2, \beta_2)$ be two graphs. G_1 is said to be isomorphic to G_2, denoted by $G_1 \cong G_2$, if there exist a bijection $f : V_1 \to V_2$ such that for any $u, v \in V_1$ and $(u, v) \in E_1$ if and only if $\exists f(u), f(v) \in V_2$ and $(f(u), f(v)) \in E_2$. In this case the two graphs G_1 and G_2 are called *isomorphic*.

Similarly the graph homomorphism and monomorphism can also be defined depending on the type of functions f.

A particular class of graph matching algorithms resulting from the consideration of outliers is called *subgraph matching*. It can roughly be defined as recognizing one graph as part of the other. Like graph matching, subgraph matching also includes altogether subgraph homomorphism, isomorphism and monomorphism. Subgraph isomorphism can be formally defined as follows:

Definition 2.6 (Subgraph Isomorphism)**.** Let $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ and $G_2 = (V_2, E_2, \alpha_2, \beta_2)$ be two graphs. G_1 is said to be isomorphic to a subgraph S_2 of G_2, denoted by $G_1 \cong S_2 \subseteq G_2$, if there exist an injection $\phi : V_1 \to V_2$ such that for any $u, v \in V_1$, $(u, v) \in E_1$ if and only if $\exists \phi(u), \phi(v) \in V_2$ and $(\phi(u), \phi(v)) \in V_2$.

Later in this chapter we provide a detailed review and references of subgraph matching algorithms.

2.3 Graph edit distance

Graph edit distance comes from the necessity of having a dissimilarity measure between two graphs. In pattern recognition when two objects are represented with graphs, it is often needed to have a distance or similarity measure between them. This is a very difficult task as graph can be multidimensional. The basic idea behind *graph edit distance* is to define a dissimilarity measure between two graphs by the minimum amount of edition needed to transform one graph to another [98]. To this end, a number of edit operations e, consisting of the *insertion*, *deletion* and *substitution* of pair of nodes and edges together with *merging* of a series of nodes and edges must be defined. Then for a pair of graphs G_1 and G_2, there exist a sequence of edit operations or edit path $p(G_1, G_2) = (e_1, \ldots, e_k)$, (where each e_i denotes an edit operation) that transforms G_1 to G_2 or vice versa. The total distance measure of editing a graph to another can be obtained by attaching an edit cost to each of the edit operations and summing them up. In general, there might exist more than one edit path that transform the graph G_1 to G_2, let $\mathbb{P}(G_1, G_2)$ be the set of all such paths. One example of such paths $p(G_1, G_2) \in \mathbb{P}(G_1, G_2)$ is shown in Figure 2.1. The edit distance between two graphs G_1 and G_2 is defined as the minimum total edit cost of all such paths that transform G_1 to G_2. It is formally defined as follows:

Definition 2.7 (Graph edit distance)**.** Given two graphs $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ and $G_2 = (V_2, E_2, \alpha_2, \beta_2)$, the graph edit distance between G_1 and G_2 is defined by:

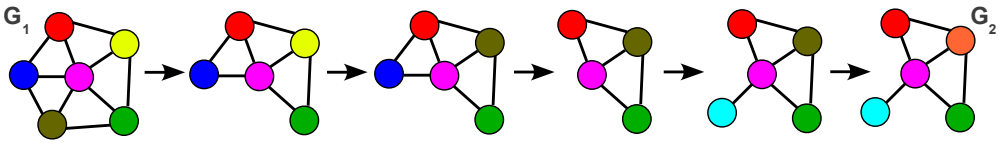


Figure 2.1: An edit path between two graphs. Node labels are represented by different colours.

$$d(G_1, G_2) = \min_{(e_1,\ldots,e_k)\in\mathbb{P}(G_1,G_2)} \sum_{i=1}^{k} c(e_i)$$

where $c(e)$ denotes the cost of an edit operation e.

There are many kind of algorithms to compute graph edit distance. Optimal algorithms are based on combinatorial search procedures that explores all the possible mappings of nodes and edges of one graph to the nodes and edges of the second graph [13]. The major drawback of such approach is the time complexity which is exponential to the size of the graphs. Then a number of suboptimal methods have been proposed to make the graph edit distance less computationally demanding [69]. A linear programming method to compute the graph edit distance with unlabelled edges is proposed in [49]. An efficient suboptimal algorithm for graph edit distance computation is proposed based on bipartite optimization procedure in [80]. There are also some effort going on towards quadratic time approximation of graph edit distance [30].

2.4 Graph indexing

In the core of many graph related applications, lies a common and critical problem: how to efficiently process graph queries and retrieve related graphs. In some cases, the success of an application directly relies on the efficiency of the query processing system. The classical graph query problem can be defined as: given a graph database $D = \{g_1, g_2, \ldots, g_n\}$ and a graph query q, find all the graphs in D, in which q is a subgraph. It is inefficient to perform a sequential search on D and check whether q is a subgraph of any graph $g_i \in D$. Sequential search in such a manner is inefficient because in this way one not only has to access the whole graph database but also has to check subgraph isomorphism which is an NP-complete problem. Since generally the size of D is huge, sequential searching in this manner is nearly impossible. This creates the necessity to build graph indices in order to help the processing of graph queries. XML query is a simple kind of graph query, which is usually built around path expressions. There are various indexing methods available [35,50,89,91,105,108]. The methods can be categorized in terms of the basic indexing unit such as *graph paths* [35], *frequent graph structure* [105], *subtree* [89] and they are developed depending on the type of applications.

2.5 Graph embedding

The necessity of graph embedding comes from some of the major drawbacks of graph based methods, that is the significantly increased complexity of many algorithms. For example, the comparison between two vectors for identity can be done in linear time complexity with respect to the length of the vectors. On the other hand, for testing two graphs for isomorphism only the exponential algorithms are known today. Apart from that some of the basic operations like weighted summation or multiplication etc. of pair of entities in graph domain are not defined but these are elementary operations in many classification and clustering algorithms. *Graph embedding* can roughly be defined as the procedure of mapping graphs either explicitly or implicitly into high dimensional spaces for the sake of performing basic mathematical operation required by various statistical pattern recognition techniques. The graph embedding methods are formally categorized as *explicit graph embedding* and *implicit graph embedding*.

2.5.1 Explicit graph embedding

Definition 2.8. *Explicit graph embedding* can be defined as the procedure of mapping graphs from arbitrary graph domains $\mathcal{G}$ to a real vector space $\mathbb{R}^n$ by means of functions such as $\varphi : \mathcal{G} \to \mathbb{R}^n$.

The main aim of this kind of graph embedding is to provide an n-dimensional vector for each graph $G \in \mathcal{G}$ such that the distance between the embedded vectors x_i and x_j respectively of the graphs G_i and G_j is as close as possible to the distance between G_i and G_j. There are many explicit graph embedding algorithms based on different paradigms such as spectral properties [58], dissimilarity measures to selected prototypes [14], node attribute statistics [33], fuzzy assignment [61] etc.

2.5.2 Implicit graph embedding: graph kernels

Definition 2.9. *Implicit graph embedding* methods are based on *graph kernels*. A graph kernel is a function $\kappa : \mathcal{G} \times \mathcal{G} \to \mathbb{R}$ for that a mapping $\Phi : \mathcal{G} \to \mathcal{H}$ to Hilbert space $\mathcal{H}$ exists, such that $\kappa(G_1, G_2) = \langle \Phi(G_1), \Phi(G_2) \rangle$ for all $G_1, G_2 \in \mathcal{G}$.

A graph kernel is a positive definite kernel on the set of graphs $\mathcal{G}$. Graph kernels can be defined on different substructures such as random walks [31], shortest paths [11], cyclic patterns [41], backtrackless walks [4] etc. The idea behind random walk kernel is to measure the similarity between graphs by counting the weighted common random walks between the operand graphs. This can be done by computing the *tensor product graph* and using the property of the adjacency matrix of the product graph.

2.6 Product graph

A *product graph* is a graph generated from the *graph product* operation on two graphs. In graph theory *graph product* is a binary operation that takes two graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ and produces a graph $G_X(V_X, E_X)$ with a vertex set V_X as the Cartesian product $V_1 \times V_2$ and an edge set E_X depending on the criterion imposed during the graph product operation. In graph theory we talk about different product graphs depending on the definition of the edge set. In this section we will have a quick look on different type of product graphs, before that let us consider the following example which we will use to explain different definitions.

Let $G_1(V_1, E_1, L_{V_1}, L_{E_1})$ and $G_2(V_2, E_2, L_{V_2}, L_{E_2})$ be two graphs with sets of vertices $V_1 = \{1, 2, 3, 4, 5\}$, $V_2 = \{1', 2', 3'\}$ and sets of edges $E_1 = \{(1,2), (2,1), (1,3), (3,1), (2,3), (3,2), (2,4), (4,2), (3,5), (5,3), (4,5), (4,5)\}$, $E_2 = \{(1',2'), (2',1'), (1',3'), (3',1'), (2',3'), (3',2')\}$ (see Figure 2.2).

Definition 2.10. The simplest product graph is the *Cartesian product graph*. It is defined as a product graph G_X of two graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ such that the vertex set V_X of G_X is the Cartesian product $V_1 \times V_2$ *i.e.*

$$V_X = \{(u_1, u_2) : u_1 \in V_1, u_2 \in V_2\}$$

and the edge set is defined as:

$$E_X = \{((u_1, u_2), (v_1, v_2)) : u_1 = v_1 \text{ and } (u_2, v_2) \in E_2 \text{ or } u_2 = v_2 \text{ and } (u_1, v_1) \in E_1, u_1, v_1 \in V_1, u_2, v_2 \in V_2\}$$

An example of Cartesian product graph is shown in Figure 2.2.

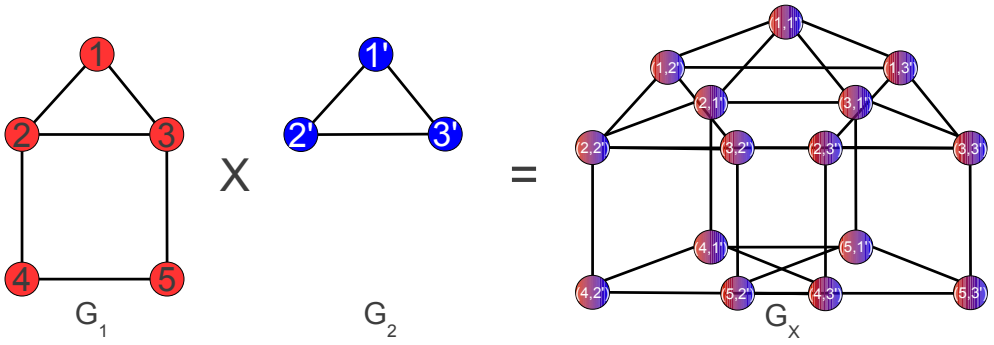


Figure 2.2: An example of Cartesian product graph.

Definition 2.11. The *strong product graph* G_X of two graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ is a graph such that the vertex set V_X of G_X is the Cartesian product $V_1 \times V_2$ *i.e.*

$$V_X = \{(u_1, u_2) : u_1 \in V_1, u_2 \in V_2\}$$

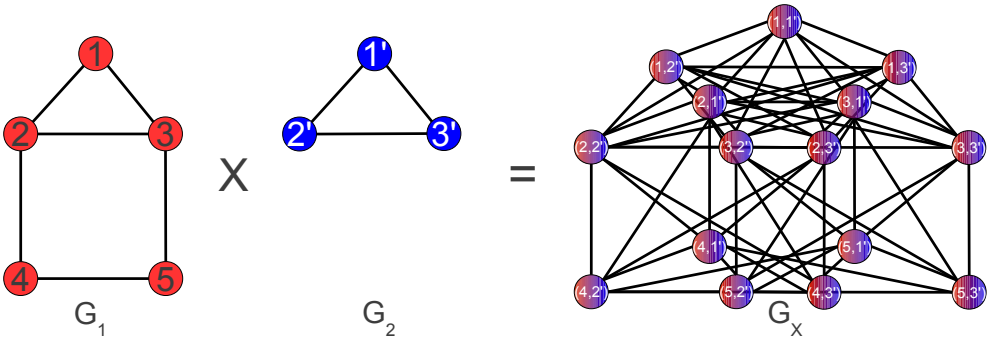


Figure 2.3: An example of a strong product graph.

and the edge set is defined as:

$$E_X = \{((u_1, u_2), (v_1, v_2)) : u_1 = v_1 \text{ or } (u_1, v_1) \in E_1 \text{ and } u_2 = v_2 \text{ or } (u_2, v_2) \in E_2, u_1, v_1 \in V_1, u_2, v_2 \in V_2\}$$

An example of strong product graph is shown in Figure 2.3.

Definition 2.12. The *tensor product graph* G_X of two graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ is a graph such that the vertex set V_X of G_X is the Cartesian product $V_1 \times V_2$ *i.e.*

$$V_X = \{(u_1, u_2) : u_1 \in V_1, u_2 \in V_2\}$$

and the edge set is defined as:

$$E_X = \{((u_1, u_2), (v_1, v_2)) : (u_1, v_1) \in E_1 \text{ and } (u_2, v_2) \in E_2, u_1, v_1 \in V_1, u_2, v_2 \in V_2\}$$

An example of tensor product graph is shown in Figure 2.4.

Apart from the above instances, there are some other product graphs also, viz., *lexicographical product graph*, *modular product graph*, *co-normal product graph* etc. In graph theory, mainly in artificial intelligence or machine learning the tensor product graph has got popularity for computing walk based graph kernel. Tensor product graph can also be called as *direct product graph*, *Kronecker product graph*, *categorical product graph*, *cardinal product graph* etc.

2.7 State-of-the-art in graph matching

As this thesis mainly deals with subgraph matching algorithm, in this section we will exclusively review the subgraph matching algorithms. Interested readers can find

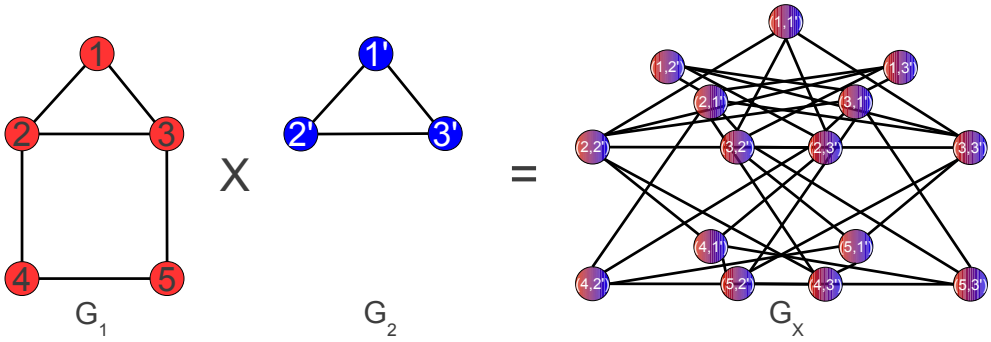


Figure 2.4: An example of a tensor product graph.

a very robust and detailed review of various graph matching methods for pattern recognition problem in [16]. In pattern recognition, the research community has divided the graph matching methods into two broad categories: (1) exact and (2) inexact graph matching. The exact matching requires a strict correspondence among the two objects being matched or at least among their sub parts [21,100]. On the other hand the inexact graph matching methods allow some tolerance for matching and in this case a matching can occur where two graphs being matched can be structurally different to some extent [54, 73].

Most of the exact techniques rely on some kind of *tree search with backtracking*. The basic idea is initially an empty matched list with pair of nodes is expanded with new pair of matched nodes. Each pair of nodes is chosen if they satisfy some kind of condition. For this kind of algorithm usually a set of heuristic is set which can eliminate some kind of absurd matching. The algorithm backtracks in case it reaches a partial condition where further expansion is not possible. The first algorithm in this category was proposed by Ullman [100]. Cordella *et al.*proposed two algorithms viz. VF [19] and VF2 [21] which also belong to this category. As it is mentioned before this kind of algorithm is not efficient when one has to find different solutions. Also determining the heuristic is also bit tricky, then also the algorithm can reach to a unfruitful solution.

Exact matching problem can also be formulated as *constraint satisfaction problem* [53, 90]. This is a problem that has been studied deeply in discrete optimization framework and operational research. As most of the optimization based techniques, they aim to find local optima in approximated way and for that reason it is always possible for these algorithms to stuck in a solution near to the local optima. Moreover, optimization based techniques have high time complexity.

There are some subgraph matching algorithms aimed at reducing the matching time of one input graph against a large library of graphs. The method proposed by Messmer and Bunke in [65] can be classified in this category. Here a recursive decomposition of each graph of the library is performed. This decompose the graphs

into smaller subgraphs, until trivial, one node graphs are reached. The matching process, then, exploits the fact that some of the parts are common to several graphs in the library. This fact speeds up the matching procedure. Later a more impressive algorithm was proposed by the same author where they builds a decision tree from the graph library [64]. The main problem that has been noticed of these algorithms is the space to store the built in libraries, which is exponential with respect to the number of nodes.

There are lot of arguments supporting the exact and inexact matching methods. Researchers often mention the characteristic of exact matching algorithms do not fit with realistic applications. This is because the description of object (or part of object) represented by graph is a vector resulted from some statistical analysis. So researchers proposed inexact graph matching.

Tree search with backtracking can also be used for inexact subgraph matching problem. Usually this kind of algorithms are directed by the cost of partial matching obtained so far and a heuristic estimate of the matching cost for the remaining nodes. One of the first algorithms of this branch was proposed by Tsai and Hu [98, 99]. In these papers they introduced the graph edit costs *i.e.* the costs for substituting, deleting and inserting nodes and edges. The main idea of such family of methods is to search for the minimum cost graph edit sequence that transform one graph to another.

Continuous optimization methods have also been used for inexact graph matching. It consists in finding a matching matrix X, between a subset of the nodes of the first graph and a subset of the nodes of the second graph. The desired matching must optimize some function depending on the weights of the edges preserved by the match. The elements of X are constrained to take the discrete values 0 and 1. One of the first algorithms in this branch was proposed by Almohamad and Duffuaa [2]. Then based on different optimization methods different algorithms were proposed, for example, graduated assignment graph matching [37], maximal clique finding [10, 73]. None of the above algorithms addressed the subgraph matching problem. Very recently Le Bodic *et al.*proposed an integer linear programming based optimization method exclusively for subgraph isomorphism [8]. The main problem of this kind of optimization methods for graph matching is that they could stuck in a local optima. There are some algorithms that can take care of this situation but this technique is not feasible for all scenario. Also optimization methods are expensive and inefficient for bigger graph.

Spectral methods for graph matching are based on the observation that the eigenvalues and the eigenvectors of the adjacency matrix of a graph are invariant with respect to node permutations. Hence, if two graphs are isomorphic, their adjacency matrix will have the same eigenvalues and eigenvectors. The method which is slightly related to subgraph matching with spectral methods is presented by Shokoufandeh and Dickinson [89]. The method is about indexing a hierarchical structure represented by a directed acyclic graph (DAG). They proposed to index each of the nodes of DAG in terms of the topological properties encoded by the spectral method under that node. To the best of knowledge we do not find many works in this category.

Table 2.1: Summary table of different graph matching techniques.

Method	**Heuristic**	**Risk of local optima**	**Size constraints**	**Inexact matching**	**Time efficient**
Ullman [100] and Cordella [19, 21]	✓	✓	✗	✗	✓
Larrosa and Valiente [53] and Solnon [90]	✗	✓	-	✗	✗
Messmer and Bunke in [64, 65]	✗	✗	✓	✗	✓
Tsai and Hu [98, 99]	✓	✗	✗	✓	✓
Almohamad and Duffuaa [2]	✗	✗	✗	✓	✓
Bomze *et al.* [10] and Pelillo *et al.* [73]	✗	✓	✗	✓	-
Le Bodic *et al.* [8]	✗	✗	✓	✓	✗
Shokoufandeh and Dickinson [89]	✗	✗	✓	✓	✓

2.8 Conclusions

In the literature we encounter different type of subgraph matching algorithms. But there are lack of methods that can deal with a big dataset of graphs. Nowadays, due to a huge development of imagery device the number of images in any of the field of computer vision is not limited. So developing algorithms that can deal with a substantial number of images with manageable time complexity is necessary. Also, different application field evolve different problem from distortion, noise point of view. Dealing these noisy data in the graph level is a challenging task. So this demands some work on having stable graph representation tolerating the distortions. These are the main motivations for working on the graph representations and algorithms in this thesis.

Chapter 3

State-of-the-art in Symbol Spotting

Symbol spotting has experienced a lot of interests among the graphics recognition community. Here the main task starts by querying a given graphic symbol or model object, usually cropped from a bigger document. The main aim is to search the model object in a target document or set of target documents. In this dissertation we will alternatively name the model object to be queried as *model symbol* or *query symbol* or *pattern* and the corresponding graph representing them as *query graph* or *model graph* or *pattern graph*. And the document or the set of documents where the user intends to find the model symbol as *input document* or *target document* and the corresponding graph as *target graph* or *input graph*. Most of the time the user is interested in getting a ranked list of retrieved zones supposed to contain the queried symbol depending on the similarity or dissimilarity measure. This Chapter contains a review of the state of the art of symbol spotting methods. The major existing research can be classified into five broad families as in [83], which are listed in Table 3.1. We review those families as follows:

3.1 Hidden Markov Models (HMMs)

HMMs are powerful tools to represent dynamic models which vary in terms of time or space. Their major advantage in space series classification results from their ability to align a pattern along their states using a probability density function for each state. This estimates the probability of a certain part of the pattern belonging to the state. HMMs have been successfully applied for off-line handwriting recognition [27, 28], where the characters represent pattern changes in space whilst moving from left to right. Also, HMMs have been applied to the problems of image classification and shape recognition [40]. Muller and Rigoll [66] proposed pseudo 2-D HMMs to model the two-dimensional arrangements of symbolic objects. This is one of the first few approaches we can find for symbol spotting, where the document is first partitioned by a fixed sized grid. Then each small cell acts as an input to a trained 2-dimensional

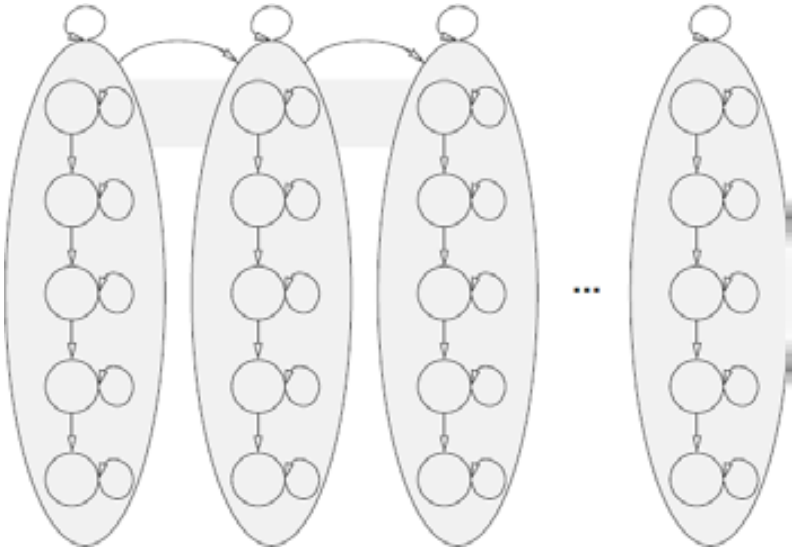


Figure 3.1: Pseudo 2-D Hidden Markov Model (Figure credit: Muller and Rigoll [66])

HMM to identify the locations where the symbols from the model database is likely to be found. Previously, HMMs were also applied to word spotting, and this work is an adaptation of HMMs for 2D shapes. The method does not need pre-segmentation, and also it could be used in noisy or occluded conditions, but since it depends on the training of a HMM, it loses one of the main assumptions of symbol spotting methodologies.

3.2 Graph-based approaches

The methods based on graphs rely on the structural representation of graphical objects and propose some kind of (sub)graph matching techniques to spot symbols in the documents. Graph matching can be solved with a structural matching approach in the graph domain or solved by a statistical classifier in the embedded vector space of the graphs. In both cases these techniques include an error model which allows inexact graph matching to tolerate structural noise in documents. Because of the structural nature of graphical documents, graph based representation has proven to be a robust paradigm. For that reason graph matching based symbol spotting techniques has drawn a huge attention of the researchers. There are an adequate number of methods based on graphs [5, 8, 54, 55, 59, 63, 68, 77, 87]. In general the structural properties of the graphical entities are encoded in terms of attributed graphs and then a subgraph matching algorithm is proposed to localize or recognize the symbol in the document in a single step. The (sub)graph matching algorithms conceive some noise models

Table 3.1: Different families of symbol spotting research with their advantages and disadvantages.

Family	Method	Advantages	Disadvantages
HMM	[66]	segmentation-free; Robust in noise	Needs training
Graph based	[5, 8, 26, 54,55,59, 63,68,77, 87]	Simultaneous symbol segmentation and recognition	Computationally expensive
Raster features	[70, 95]	Robust symbol representation; Computationally fast	Ad-hoc selection of regions; Inefficient for binary images
Symbol signatures	[23, 109]	Simple symbol description; Computationally fast	Prone to noise
Hierarchial symbol representation	[110]	Linear matching is avoided by using an indexing technique	Dendogram structure is strongly dependent on the merging criterion.

to incorporate image distortion, which is defined as inexact (sub)graph matching. Since (sub)graph matching is an NP-hard problem [62], these algorithms often suffer from a huge computational burden. Among the methods available, Messmer and Bunke in [63] represented graphic symbols and line drawings by Attributed Relational Graphs (ARG). Then the recognition process of the drawings was undertaken in terms of error-tolerant subgraph isomorphisms from the query symbol graph to the drawing graph. Lladós *et al.*in [54] proposed Region Adjacency Graphs (RAG) to recognize symbols in hand drawn diagrams. They represented the regions in the diagrams by polylines where a set of edit operations is defined to measure the similarity between the cyclic attributed strings corresponding to the polylines.

In [5], Barbu *et al.*presented a method based on frequent subgraph discovery with some rules among the discovered subgraphs. Their main application is the indexing of different graphical documents based on the occurrence of symbols. Qureshi *et al.* [77] proposed a two-stage method for symbol recognition in graphical documents. In the first stage the method only creates an attributed graph from the line drawing images and in the second stage the graph is used to spot interesting parts of the image that

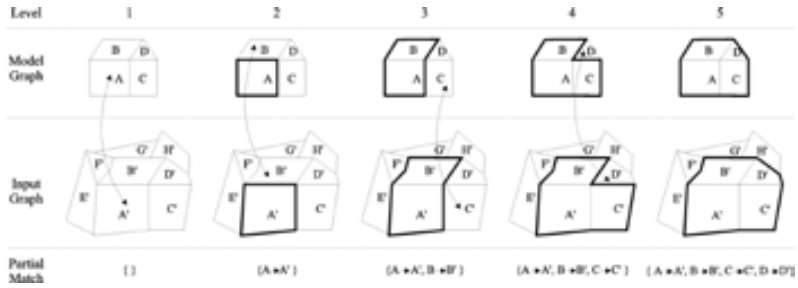


Figure 3.2: Example of string growing in terms of the neighbourhood string similarity (Figure credit: Lladós *et al.* [54]).

potentially correspond to symbols. Then in the recognition phase each of the cropped portions from the images are passed to an error tolerant graph matching algorithm to find the queried symbols. Here the procedure of finding the probable regions restricts the method only to work for some specific symbols, which violates the assumption of symbol spotting. Locteau *et al.* [55] present a symbol spotting methodology based on a visibility graph. There they apply a clique detection method, which corresponds to a perceptual grouping of primitives to detect regions of particular interest.

In [87] Rusinol *et al.*proposed a symbol spotting method based on the decomposition of line drawings into primitives of closed regions. An efficient indexing methodology was used to organize the attributed strings of primitives. Nayef and Breuel [68] proposed a branch and bound algorithm for spotting symbols in documents, where they used geometric primitives as features. Recently Luqman *et al.* [59] also proposed a method based on fuzzy graph embedding for symbol spotting, a priori they also used one pre-segmentation technique as in [77] to get the probable regions of interest which may contain the graphic symbols. Subsequently, these ROIs are then converted to fuzzy structural signatures to find out the regions that contain a symbol similar to the queried one. Recently, Le Bodic *et al.* [8] proposed substitution-tolerant subgraph isomorphism to solve symbol spotting in technical drawings. They represent the graphical documents with RAG and model the subgraph isomorphism as an optimization problem. The whole procedure is performed for each pair of query and document. The subgraph matching is done based on integer linear programming based optimization technique. Moreover, since the method works with RAG, it does not work well for the symbols having open regions or regions with discontinuous boundary.

3.3 Raster features

Some of the methods work with low-level pixel features for spotting symbols. To reduce the computational burden they extract the feature descriptors on some regions of the documents. These regions may come from a sliding window or spatial interest

Table 3.2: Comparison of the key existing works of symbol spotting.

Method	**Segmentation free**	**Robust in noise**	**Training free**	**Time efficient**	**Large database**
Muller and Rigoll [66]	✓	✓	✗	✓	-
Messmer and Bunke [63]	✓	-	-	✗	✗
Lladós *et al.* [54]	✓	-	✓	✗	✗
Barbu *et al.* [5]	✓	-	✓	✗	✗
Qureshi *et al.* [77]	✗	-	✓	✗	✗
Locteau *et al.* [55]	✓	✗	✓	✓	✗
Rusinol *et al.* [87]	✓	-	✓	✗	✓
Rusinol *et al.* [84]	✓	-	✓	✓	✓
Tabbone *et al.* [95]	✗	✗	✓	✓	-
LeBodic *et al.* [8]	✓	✗	✓	✗	✗

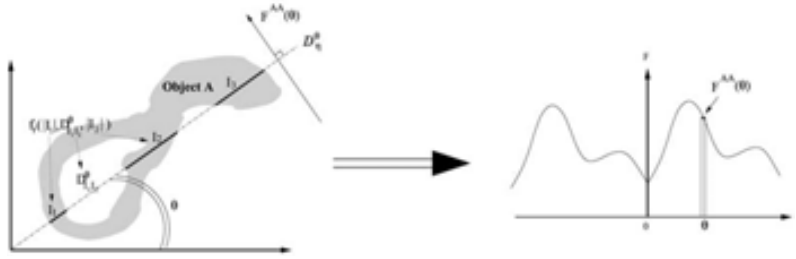


Figure 3.3: Definition of an $\mathcal{F}$-signature (Figure credit: Tabbone *et al.* [95]).

point detectors. These kinds of pixel features robustly represent the region of interest. Apart from those methods mentioned, other methods find some probable regions for symbols by examining the loop structures [77] or just use a text/graphic separation to estimate the occurrence of the symbols [95]. After ad-hoc segmentation, global pixel-based statistical descriptors [70, 95] are computed at each of the locations in sequential order and compared with the model symbols. A distance metric is also used to decide the retrieval ranks and to check whether the retrievals are relevant or not. The one-to-one feature matching is a clear limitation of this kind of methods and also the ad-hoc segmentation step only allows it to work for a limited set of symbols.

3.4 Symbol signatures

Like the previous category, this group of methods [23, 85, 109] also works with ad-hoc segmentation, but instead of pixel features they compute the vectorial signatures, which better represent the structural properties of the symbolic objects. Here vectorial signatures are the combination of simple features viz. number of graph nodes, relative lengths of graph edges etc. These methods are built on the assumptions that the symbols always fall into a region of interest and compute the vectorial signatures inside those regions. Since symbol signatures are highly affected by image noise, these methods do not work well in real-world applications.

3.5 Hierarchial symbol representation

Some of the methods work with the hierarchical definition of symbols, in which they hierarchically decompose the symbols and organize the symbols' parts in a network or dendogram structure [110]. Mainly, the symbols are split at the junction points and each of the subparts are described by a proprietary shape descriptor. These subparts are again merged by a measure of density, building the dendogram structure. Then the network structures are traversed in order to find the regions of interests of the

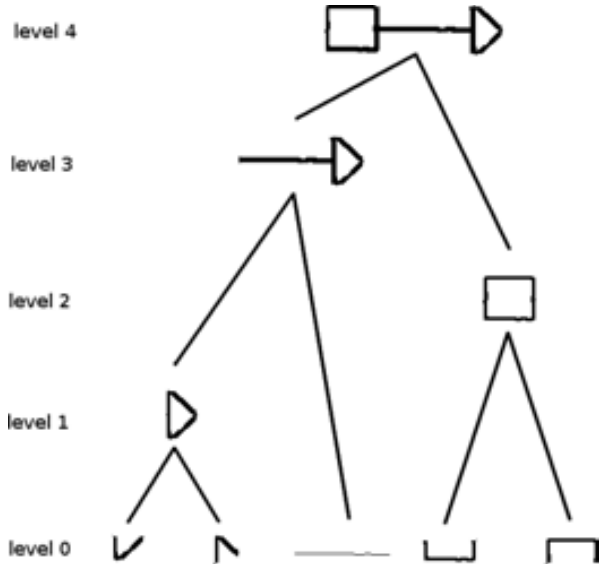


Figure 3.4: A dendogram showing the hierarchical decomposition of graphical object (Figure credit: Zuwala and Tabbone [110]).

polylines where the query symbol is likely to appear.

3.6 Conclusions

To conclude the literature review, some of the challenges of symbol spotting can be highlighted from the above state-of-the-art reviews. First, symbol spotting is concerned with various graphical documents viz. electronic documents, architectural floorplans etc., which in reality suffer from *noise* that may come from various sources such as low-level image processing, intervention of text, etc. So efficiently handling structural noise is crucial for symbol spotting in documents. Second, an example application of symbol spotting is to find any symbolic object from a large amount of documents. Hence, the method should be efficient enough to handle a *large database*. Third, symbol spotting is usually invoked by querying a cropped symbol from some document, which acts as a query to the system. So it implies infinite possibilities of the query symbols, and indirectly *restricts the possibility of training* in the system. Finally, since symbol spotting is related to real-time applications, the method should have a *low computational complexity*. We chose these five important aspects (segmentation, robustness in noise, training free, computational expenses, robustness with a large database) of symbol spotting to specify the advantages and disadvantages of the key research, which is listed in Table 3.2.

Chapter 4

Symbol Spotting by Hashing Serialized Subgraphs

In this chapter we propose a symbol spotting technique in graphical documents. Graphs are used to represent the documents and an error tolerant (sub)graph matching technique is used to detect the symbols in them. We propose a graph serialization to reduce the usual computational complexity of graph matching. Serialization of graphs is performed by computing acyclic graph paths between each pair of connected nodes. Graph paths are one dimensional structures of graphs, handling which is less expensive in terms of computation. At the same time they enable robust localization even in the presence of noise and distortion. Indexing in large graph databases involves a computational burden as well. We utilize a graph factorization approach to tackle this problem. Factorization is intended to create a unified indexed structure over the database of graphical documents. Once graph paths are extracted, the entire database of graphical documents is indexed in hash tables by locality sensitive hashing (LSH) of shape descriptors of the paths. The hashing data structure aims to execute an approximate k-NN search in a sub-linear time. We have performed detailed experiments with various datasets of line drawings and compared our method with the state-of-the-art works. The results demonstrate the effectiveness and efficiency of our technique.

4.1 Introduction

In this chapter we propose a symbol spotting technique based on a graph representation of graphical documents, especially various kinds of line drawings. When graphs are attributed by geometric information, this also supports various affine transformations viz. translation, rotation, scaling etc. On the other hand, subgraph isomorphism is proved to be a NP-hard problem [62], so handling a large collection of graphical documents using graphs is difficult. To avoid computational burden, we propose a method

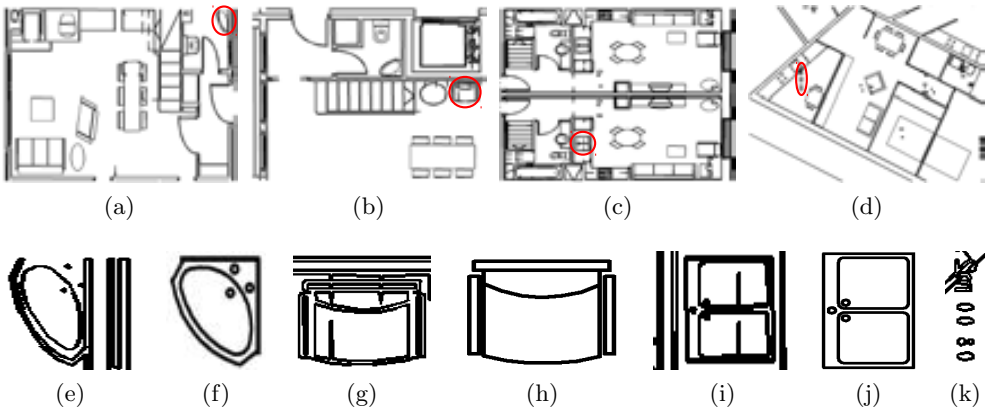


Figure 4.1: (a)-(d) Examples of floorplans from a real floorplan (FPLAN-POLY) database, (e),(g),(i),(k) Zoomed portions of the selected parts respectively shown in Figures 4.1a-4.1d show the difficulty of recognition due to noise and superimposition of textual and graphical information, (f),(h),(j) Actual instances of the symbols shown in (e),(g),(i) respectively.

based on the *factorization of graphs*. Informally, *graph factorization* can be defined as the method to extract graph sub-structures from larger graphs. This is helpful to find common subgraph structures from larger collections of graphs to define indexing keys in terms of such common subgraphs. This indexing structure is supposed to reduce the search space by clustering similar subgraphs. In our case, factorization is performed by splitting the graphs into a set of all acyclic paths between each pair of connected nodes. The paths carry the geometrical information of a structure which are considered as attributes. The decomposition of a graph into graph paths can be seen as a *serialization* process where the complex two-dimensional graph structure is converted to a one-dimensional string to reduce computational complexity, usually present in subgraph matching algorithms. In this work we follow both factorization and serialization to create an inexpensive and unified structure. Graph factorization creates a unified representation of the whole database and at the same time it allows for robust detection with a certain tolerance to noise and distortions (see Figure 4.1). This also eases the segmentation-free recognition which is important for our purpose.

In this chapter, the shape descriptors of paths are compiled into hash tables by the Locality-Sensitive Hashing (LSH) algorithm [34,44]. The hashing data structure aims to organize similar paths in the same neighborhood into hash tables. The spotting of the query symbol is then undertaken by a spatial voting scheme, which is formulated in terms of the selected paths from the database.

However, the graph paths are indexed independently, ignoring any spatial relationship between them. Actually keeping the spatial relationship is not important for the method since we consider all the acyclic paths between each pair of connected nodes. Actually this fact better helps to incorporate the structural noise keeping the spatial relationship among paths. This way the spatial relationship is maintained as

the smaller paths are always subpaths of some longer paths and longer paths contain more global structural information.

Since the method represents a database of graphical documents in terms of unified representation of factorized substructures, it can handle a larger database of documents which is important for real-world applications. Moreover, the factorized substructures allow the method to handle structural noise up to a certain limit of tolerance. The proposed method does not work with any kind of pre-segmentation and training, which makes it capable of handling any possible combination of query symbols.

The rest of the chapter is outlined as follows: We present our proposed methodology in Section 4.2, followed by a series of experiments in Section 4.3. Section 4.4 concludes the chapter with discussions on future works.

4.2 Methodology

Our graph representation considers the critical points detected by the vectorization method as the nodes and the lines joining them as the edges. For our purpose we use the vectorization algorithm proposed by Rosin and West [82]. To avoid the computational burden we propose a method based on the factorization of graphs. The factorization is performed by splitting the graphs into a set of all acyclic paths between each pair of connected nodes; the paths carry the geometrical information of a structure as attributes. The factorization helps to create an unified representation of the whole database and at the same time it allows robust detection with certain tolerance to noise and distortion. This also eases the segmentation-free recognition which is important for our purpose. We have already mentioned that factorization of graphs is used in kernel based methods and it s principle motive was to cope with distortions. But the kernel based method can not utilize the power of indexation which is important for our case as we concentrate in spotting symbols in bigger datasets efficiently. So indexing the serialized subgraphical structures is a crucial part for our application. Our method takes the advantage of the error tolerance as proposed by the kernel based methods and at the same time the advantage of the indexation strategy to make the searching efficient. The shape descriptors of paths are compiled in hash tables by the Locality-Sensitive Hashing (LSH) algorithm [34, 44]. The hashing data structure aims to organize similar paths in the same neighborhood in hash tables and LSH is also proved to perform an approximate k-NN search in sub-linear time. The spotting of the query symbol is then performed by a spatial voting scheme, which is formulated in terms of the selected paths from the database. This path selection is performed by the approximate search mechanism during the hash table lookup procedure for the paths that compose the query symbol. The method is dependent on the overall structure of the paths. This technique is able to handle the existence of spurious nodes. And since we consider all the acyclic paths between each pair of connected nodes, the detection or recognition of a symbol is totally dependent on the overall structure of the majority of paths. This way the method is able to handle

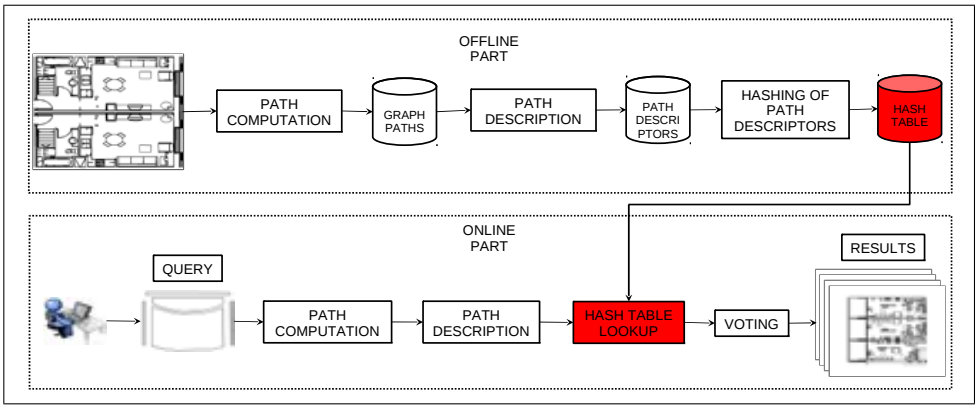


Figure 4.2: Symbol spotting framework for our method.

the problem of spurious nodes and edges. So the introduction of spurious edges and nodes only increases the computational time in the offline part without hampering the performance.

4.2.1 Framework

Our entire framework can be broadly divided into two parts viz. offline and online (see Figure 4.2). The algorithms are respectively shown in Algorithm 4.2.1 and Algorithm 4.2.2. The offline part (Algorithm 4.2.1) includes the computation of all the acyclic graph paths in the database, description of those paths with some proprietary descriptors and hashing of those descriptors using the LSH algorithm (see Figure 4.3). Each time a new document is included in the database, the offline steps for this document are repeated to update the hash table. To reduce the time complexity of the offline part the path and description information of the previously added documents are stored. On the other hand, the online part (Algorithm 4.2.2) includes the querying of the graphic symbol by an end user, the computation of all the acyclic paths for that symbol and description of them by the same method. Then a hash table lookup for each of the paths in the symbol and a voting procedure, which is based on the similarity measure of the paths, are also performed on the fly to undertake the spotting in the documents. The framework is designed to produce a ranked list of retrievals in which the true positive should appear first. The ranking is performed based on the total vote values (see Section 4.2.4) obtained by each retrieval.

Let us now describe the key steps of our framework in the following subsections.

Algorithm 4.2.1 Hash table creation

Require: A set $\mathcal{D} = \{D_1, \ldots, D_n\}$.
Ensure: A set $\mathcal{T}$ of hash tables.
//Let f_{all} be the set of all path descriptors.

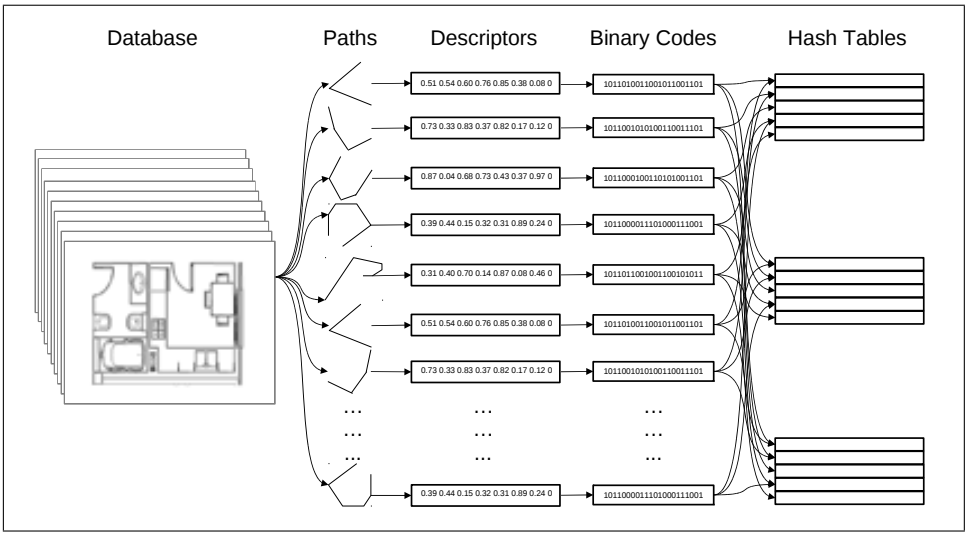


Figure 4.3: Hashing of paths provokes collisions in hash tables.

//Initialize f_{all}
f_{all} $\oslash$
for all D_i of $\mathcal{D}$ **do**
  P_i acyclic paths (D_i)
  for all p of P_i **do**
    f descriptors of (p) // Zernike moments or Hu moment invariants
    f_{all} f_{all} f
  end for
end for
//Create the set of hash tables
$\mathcal{T}$ LSH(f_{all})

4.2.2 Path description

Let $\mathcal{D} = D_1, D_2, ..., D_n$ be the set of all documents in a database, and $G_i(V_i, E_i, \alpha_i)$ be the node attributed graph for the document D_i. Here $\alpha_i : V_i$ L_v is a function, in this case $L_v = \mathbb{N}^2$, where the labels for each of the nodes is its position in terms of a two-dimensional coordinate system.

Definition 4.1. Given an attributed graph $G_i(V_i, E_i, \alpha_i)$, a *graph path* p_k between two connected nodes v_r and v_s in G_i is defined as the ordered sequence of vertices $(v_r, ..., v_s)$ starting from v_r to v_s.

Definition 4.2. An *embedding* function f of a graph path is defined as a function $f : P$ $\mathbb{R}^n$, defined in the space of a graph path P and maps a path to an n-dimensional feature space.

Let $P_i = \ p_1, p_2, ..., p_{n_i}\ $ be the set of all graph paths in the document D_i, where n_i is the total number of paths in the document D_i. Therefore $P = \ _iP_i$ is the set of all paths from all the documents in $\mathcal{D}$. From the definition of a graph path, a path p_k can be represented as an ordered sequence of nodes *i.e.* $p_k = [(x_1, y_1), (x_2, y_2), ...] = p_k(x, y)$. So formally speaking, given a path $p_k(x, y)$ and a shape descriptor $f : P \ \mathbb{R}^n$ defined over the space of all graph paths, applying f to each of the graph paths in P will generate a feature vector of dimension n. Below is the brief description of the shape descriptors used in this work. We define the embedding function f by means of Zernike moments and Hu moment invariants.

Embedding function based on Zernike moments

Zernike moments are robust shape descriptors which were first introduced in [96] using a set of complex polynomials. They are expressed as A_{mn} as follows:

$$A_{mn} = \frac{m+1}{\pi} \int_x \int_y p_k(x, y)[V_{mn}(x, y)]^* dxdy, \text{ where } x^2 + y^2 \leq 1 \tag{4.1}$$

where $m = 0, 1, 2, ...,$ and defines the order, $p_k(x, y)$ is the path being described and $*$ denotes the complex conjugate. Here n is an integer (that can be positive or negative) depicting the angular dependence, or rotation, subject to the conditions $m - n = even$, $n \leq m$ and $A^*_{mn} = A_{m,-n}$ is true. The Zernike polynomials $V_{mn}(x, y)$ can be expressed in polar coordinates as follows:

$$V_{mn}(x, y) = V_{mn}(r, \theta) = \sum_{s=0}^{\frac{m-|n|}{2}} (-1)^s \frac{(m-s)!}{s!(\frac{m+|n|}{2} - s)!(\frac{m-|n|}{2} - s)!} r^{m-2s} exp(in\theta) \tag{4.2}$$

The final descriptor function $f_{Zernike}(p_k)$ for p_k is then constructed by concatenating several Zernike coefficients of the polynomials. Zernike moments have been widely utilized in pattern or object recognition, image reconstruction, content-based image retrieval etc. but its direct computation takes a large amount of time. Realizing this disadvantage, several algorithms [42] have been proposed to speed up the accurate computation process. For line drawings, Lambert *et al.* [51, 52] also formulated Zernike moments as computationally efficient line moments. But in our case the computation is performed based on the interpolated points of the vectorized data using fast accurate calculations.

Embedding function based on Hu moment invariants

The set of seven Hu invariants of moments proposed in [43] involving moments up to order three, are widely used as shape descriptors. In general the central $(r + s)$th order moment for a function $p_k(x, y)$ is calculated as follows:

$$\mu_{rs} = \sum_{x} \sum_{y} (x - \bar{x})^r (y - \bar{y})^s \tag{4.3}$$

The function $f_{Hu}(p_k)$ describing p_k is then constructed by concatenating the seven Hu invariants of the above central moments. The use of centroid $c = (\bar{x}, \bar{y})$ allow the descriptor to be translation invariant. A normalization by the object area is used to achieve invariance to scale. The geometric moments can also be computed on the contour of the objects by only considering the pixels of the boundary of the object. As in the case of Zernike moments, these moments can also be calculated in terms of line moments [51, 52] for the objects represented by vectorized contours, which are obviously efficient in terms of computation.

4.2.3 Locality Sensitive Hashing (LSH)

In order to avoid one-to-one path matching [25], we use the LSH algorithm which performs an approximate k-NN search that efficiently results in a set of candidates that mostly lie in the neighborhood of the query point (path). LSH is used to perform contraction of the search space and quick indexation of the data. LSH was introduced by Indyk and Motwani [44] and later modified by Gionis *et al.* [34]. It has been proved to perform an approximate k-NN search in sub-linear time and used for many real-time computer vision applications.

Let $f(p_k) = (f_1, ..., f_d)$ $\mathbb{R}^d$ be the descriptors of a graph path p_k in the d-dimensional space. This point in the d-dimensional space is transformed in a binary vector space by the following function:

$$b(f(p_k)) = (Unary_C(f_1), ..., Unary_C(f_d)) \tag{4.4}$$

Here if C is the highest coordinate value in the path descriptor space then $Unary_C(f_p)$ is a C bit representation function where f_p bits of 1 s are followed by $C - f_p$ bits of 0 s. Thus, the distance between two path vectors $f(p_1)$, $f(p_2)$ can be computed by the Hamming distance between their respective binary representations $b(f(p_1))$, $b(f(p_2))$. Actually, Eqn. 4.4 allows the embedding of the descriptors fs into the Hamming cube $H^{d'}$ of dimension $d' = Cd$. The construction of the function in Eqn. 4.4 assumes the positive integer coordinates of f, but clearly any coordinates can be made positive by proper translation in $\mathbb{R}^d$. Also the coordinates can be converted to an integer by multiplying them with a suitably large number and rounding to the nearest integers.

Now let $h: \ 0,1^{\ d'} \quad 0,1$ be a function which projects a point $b \quad 0,1^{\ d'}$ to any of its d' coordinate axes, and $\mathcal{F}$ be a set of such hash functions $h(b)$, which can be formally defined as:

$$\mathcal{F} = \ h(b) \ h(b) = b_i, i = 1, ..., d'$$

where b_i is the ith coordinate of b. The final hash functions $\mathcal{H}$s can be created by randomly selecting at most K such bitwise hash functions $h(h)$ and concatenating them sequentially. This actually results in bucket indices in the hash tables. The LSH algorithm then creates a set $\mathcal{T}$ of L hash tables, each of which is constructed based on different $\mathcal{H}$s. L and K are considered as the parameters to construct the hashing data structures. Then given a descriptor f_q of a query path (point), the algorithm iterates over all the hash tables in $\mathcal{T}$ retrieving the data points that are hashed into the same bucket. The final list of retrievals is the union of all such matched buckets from different hash tables.

The entire procedure can be better understood with the following example: let $f(p_1) = (1, 6, 5)$, $f(p_2) = (3, 5, 2)$ and $f_(p_3) = (2, 4, 3)$ be three different descriptors in a three-dimensional ($d = 3$) space with $C = 6$. Their binary representation after applying the function in Eqn. 4.4 is:

$$
\begin{aligned}
b(f(p_1)) &= 100000\ 111111\ 111110 \\
b(f(p_2)) &= 111000\ 111110\ 110000 \\
b(f(p_3)) &= 110000\ 111100\ 111000
\end{aligned}
$$

Now let us create an LSH data structure with $L = 3$ and $K = 5$. So, we can randomly create 3 hash functions with at most 5 bits in each of them as follows:

$$
\begin{aligned}
\mathcal{H}_1 &= h_5, h_{10}, h_{16} \\
\mathcal{H}_2 &= h_1, h_9, h_{14}, h_{15}, h_{17} \\
\mathcal{H}_3 &= h_4, h_8, h_{13}, h_{18}
\end{aligned}
$$

This defines which components of the binary vector will be considered to create the hash bucket index. For example, applying G_2 to a binary vector results in a binary index concatenating the first, ninth, fourteenth, fifteenth and seventeenth bit values respectively. After applying the above functions to our data we obtain the following bucket indices:

$$
\begin{aligned}
&\mathcal{H}_1(f(p_1)) = 011,\ \mathcal{H}_2(f(p_1)) = 11111,\ \mathcal{H}_3(f(p_1)) = 0110 \\
&\mathcal{H}_1(f(p_2)) = 010,\ \mathcal{H}_2(f(p_2)) = 11100,\ \mathcal{H}_3(f(p_2)) = 0110 \\
&\mathcal{H}_1(f(p_3)) = 010,\ \mathcal{H}_2(f(p_3)) = 11110,\ \mathcal{H}_3(f(p_3)) = 0110
\end{aligned}
$$

Then for a query $f_{p_q} = (3, 4, 5)$ we have

$$
\begin{aligned}
b(f(p_q)) &= 111000\ 111100\ 111110 \\
\mathcal{H}_1(f(p_q)) &= 011,\ \mathcal{H}_2(f(p_q)) = 11111,\ \mathcal{H}_3(f(p_q)) = 0110
\end{aligned}
$$

Thus, we obtain $f(p_1)$ as the nearest descriptor to the query since it collides in each of the hash tables.

Similarly, for each of the graph path descriptors in the query symbol, we get a set of paths that belong to the database. Consequently, we get the similarity distances of the paths in the vectorial space. This similarity distance is useful during the voting procedure to spot the symbol and is used to calculate the vote values.

4.2.4 Voting scheme

A voting space is defined over each of the images in the database dividing them into grids of three different sizes (10×10, 20×20 and 30×30). Multiresolution grids are used to detect the symbols accurately within the image and the sizes of them are experimentally determined to have the best performance. This kind of voting scheme helps to detect the regions of occurrence even though there is a lack of overlapping between a certain pair of graph paths. It was mentioned earlier that the voting is performed in the online step of the system when the user query is accepted with a model symbol S_m. We factorize the graph representing S_m in the same way as the documents and let us say $P_{S_m} = p_1^{S_m}, ..., p_t^{S_m}$ be the set of all paths of S_m and $F_{S_m} = f_{p_1^{S_m}}, ..., f_{p_t^{S_m}}$ be the set of descriptors for the paths in P_{S_m}. The searching in the hash table is then performed in a path by path wise manner and consecutively the voting is performed in the image space. For a particular model path, $p_l^{S_m} \quad P_{S_m}$, the LSH lookup procedure returns a union of several buckets (this is how the LSH is constructed). Let us say B_l be the union of all buckets returned when queried with a path $p_l^{S_m}$. In the next step, for each path $f_{p_{B_i}} \quad B_l$ we accumulate the votes to the nine neighboring grids of each of the two terminals of $f_{p_{B_i}}$ (see Figure 4.4). The vote to a particular grid is inversely proportional to the path distance metric (in this case the Euclidean distance between the Zernike moments descriptors) and is weighted by the Euclidean distance to the centers of the respective grids (in Figure 4.4 the centers of the grids are shown in red) from the terminal of the selected path. The grids constituting the higher peaks are filtered by the k-means algorithm applied in the voting space with k=2. Here we only keep the cluster having the higher votes, all the higher voted points from all the three grids are then considered for spatial clustering. Here we compute the distances among all these points and use this distance matrix to cluster the points hierarchically. Here we use a threshold th_1 to cut the dendogram and have the clusters. The selection of th_1 is performed experimentally to give the best performance. Each of the clusters of points is considered as a retrieval; the total vote values of the grids in each cluster are considered for ranking the retrievals.

Algorithm 4.2.2 Spotting of query symbols in documents

Require: A model symbol (S_m) with the set of path descriptors $f_{p_1^{S_m}}, \ldots, f_{p_t^{S_m}}$ and a set $\mathcal{T}$ of hash tables.

Ensure: A ranked list $ROI = \quad R_1, R_2, \ldots$ of regions of interest.

//Search for the nearest buckets

for all $f_{p_i^{S_m}}$ of $f_{p_1^{S_m}}, \ldots, f_{p_t^{S_m}}$ **do**

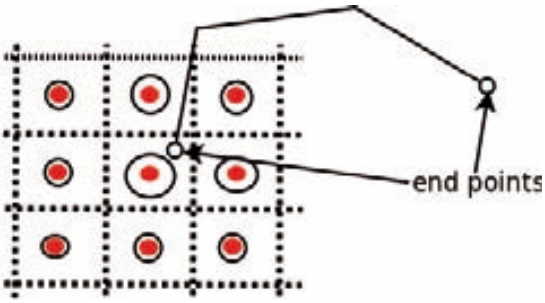


Figure 4.4: Illustration of voting: For each of the selected paths from the hash table, we accumulate the votes to the nine nearest grids of each of the 2 terminal vertices of that path.

$B_i \Leftarrow$ nearest bucket of $f_{p_i^{S_m}} \in \mathcal{T}$
//Calculate the matching scores
for all $f_{p_{B_j}}$ of B_i **do**
$MS_{i\ j} \Leftarrow$ matching score of $(f_{p_i^{S_m}}, f_{p_{B_j}})$
end for
end for
//Define and initialize the voting space
for all $D_k \in \mathcal{D}$ **do**
// Grids of three different sizes
for all $gsize$ of $\{[10 \times 10]\ [20 \times 20]\ [30 \times 30]\}$ **do**
$G^{D_k}_{gsize} \Leftarrow \oslash$ //Grids on documents
$GV^{D_k}_{gsize} \Leftarrow \oslash$ //Vote values for the grids
end for
end for
//Voting
for all B_i of $B_1 \quad B_t$ **do**
for all $f_{p_{B_j}}$ of B_i **do**
$D \Leftarrow$ document of $f_{p_{B_j}}$
$[pt_1\ pt_2] \Leftarrow$ two end points of $f_{p_{B_j}}$
for all $gsize$ of $\{[10 \times 10]\ [20 \times 20]\ [30 \times 30]\}$ **do**
for all pt_i of $[pt_1\ pt_2]$ **do**
$G_D(1:9) \Leftarrow$ Nine neighbouring grids of pt_i
$CG_{gsize}(1:9) \Leftarrow$ Centres of $G^D(1:9)$
$GDist(1:9) \Leftarrow$ distance between $(CG_{gsize}(1:9), pt_i)$
$GV^D_{gsize}(G^D_{gsize}(1:9)) \Leftarrow GV^D_{gsize}(G^D_{gsize}(1:9)) + GDist(1:9) \times \frac{1}{MS_{i\ j}}$
end for
end for
end for
end for

//Spotting
$S \Leftarrow \oslash$
for all $D_k \in \mathcal{D}$ **do**
  for all $gsize \in \{[10 \times 10], [20 \times 20], [30 \times 30]\}$ **do**
    $[Class^{D_k}_{gsize}(h), Class^{D_k}_{gsize}(l)]$=kmeans$(GV^{D_k}_{gsize}, 2)$
    //mean$(GV^{D_k}_{gsize}(Class^{D_k}_{gsize}(l)))\leq$mean$(GV^{D_k}_{gsize}(Class^{D_k}_{gsize}(h)))$,
    //where $GV^{D_k}_{gsize}(Class^{D_k}_{gsize}(h))$ are the higher voted grids
  end for
  $G^{D_k}_{all} \Leftarrow G^{D_k}_{[10\times10]}(Class^{D_k}_{[10\times10]}(h)) \cup G^{D_k}_{[20\times20]}(Class^{D_k}_{[20\times20]}(h)) \cup G^{D_k}_{[30\times30]}(Class^{D_k}_{[30\times30]}(h))$
  $\{(s_1, total\_votes(s_1)), (s_2, total\_votes(s_2)), \ldots\} \Leftarrow$ spatial clustering$(G^{D_k}_{all})$
  $S \Leftarrow S \cup \{(s_1, total\_votes(s_1)), (s_2, total\_votes(s_2)), \ldots\}$
end for
$ROI = sort(S, key = total\_votes)$

4.3 Experimental results

In this section we present the results of several experiments. The first experiment is designed to see the efficiency between the Zernike moments and the Hu moment invariants in a comparative way to represent the graph paths. The second experiment is undertaken to show the variation of symbol spotting results by varying the L and K parameters of the hash table creation. Then a set of experiments is performed to test efficiency of the proposed method to spot the symbols on documents. For that we use four different sets of images with varying difficulties. The last experiment is performed to see the possibility of applying the proposed method to any other information spotting methodologies; for that we test the method with handwritten word spotting in some real historical handwritten documents. Next we present a comparative study with state-of-the-art methods. For all these experiments we mainly use two publicly available databases of architectural floorplans: (1) FPLAN-POLY and (2) SESYD (floorplans), a description on both of them is available in App. A. Apart from them we have used two more datasets: (1) SESYD-GN and (2) SESYD-VN, specifications on them are also available in App. A.

4.3.1 Zernike moments versus Hu moment invariants

This test aims to compare the two description methods used to describe graph paths. Finally, based on this experiment, the best method is used in the remaining experiments. We compare the performance of the presented algorithm by using both description methods. To undertake this experiment, we consider the FPLAN-POLY database and perform the path description with Hu moment invariants and Zernike moments with different orders (6 to 10). In Figure 4.5 we show a precision recall curve showing the performance with different descriptions. This shows that the Zernike moments with any order outperforms the Hu moment invariants, on average there is a gain of 6.3% precision for a given recall value. Finally, in this experiment, Zernike

moments with order 7 give the best trade-off in terms of performance. This gives the imperative to perform the rest of the experiments with Zernike moments descriptors with order 7.

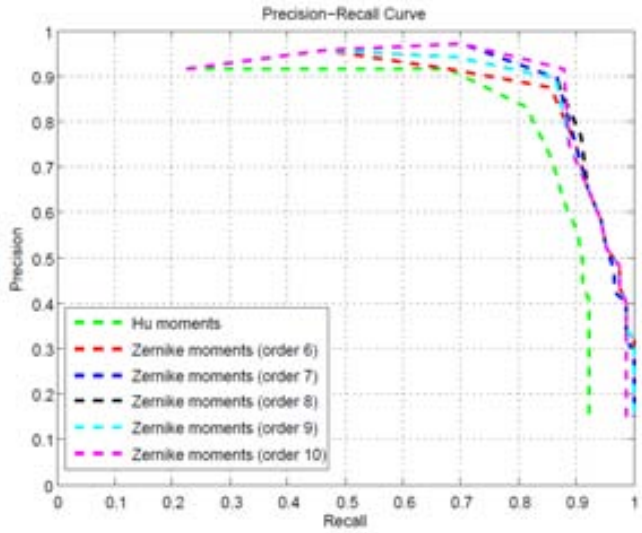
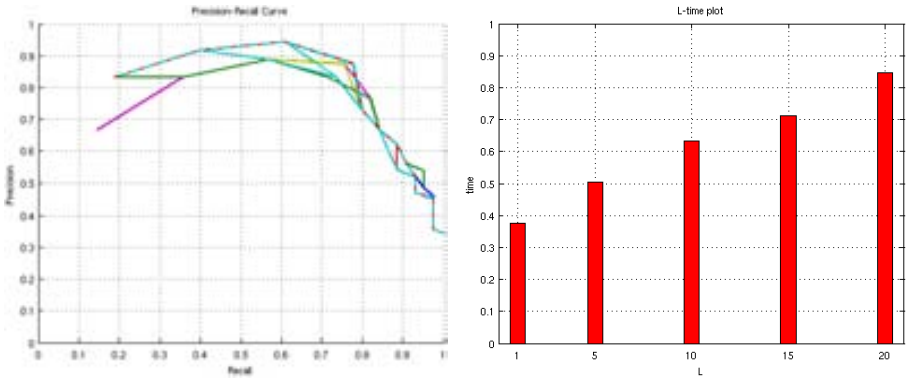


Figure 4.5: Precision-Recall plot showing the performance of the spotting method with the Hu moment invariants and Zernike moments of order 6 to 10.

4.3.2 Experiments on the influence of parameters L and K

Literally K is the maximum number of bits of the binary indices of different buckets in a table. So increasing K will increase the number of random combinations of the bit positions which ultimately increases the number of buckets in each of the hash tables. This creates tables in which many buckets with only a few instances appear, which separates the search space poorly. On the other hand, decreasing K will merge different instances incorrectly. The number of hash tables (L) is another parameter to play with, which indicates the number of tables to create for a database. Increasing L will increase the search space, since LSH considers the union of all the tables, so after a certain limit, increasing the number of tables will not improve the performance but only increase the retrieval time. So choosing the proper combination of L and K for a particular experiment is very important for efficient results.

In this experiment we chose a set of 10 floorplans from the FPLAN-POLY dataset and created the hashing data structures by varying L from 1 to 20 and K from 40 to 80. The performance of the spotting method is shown in terms of the precision-recall curves in Figure 4.6a, which shows similar performance for all the settings. But the time taken by the spotting method increases proportionally with the increment of L



(a) (b)

Figure 4.6: (a) The precision-recall plot of the spotting method by varying L 1 to 20 and K 40 to 80. (b) The plot of the time taken by the method to retrieve symbols for different values of L.

(Figure 4.6b).

4.3.3 Symbol spotting experiments

In order to evaluate the proposed spotting methodology, we present four different experiments. The first experiment is designed to test the method on the images of real world floorplans. The second experiment is performed to check the algorithm on a moderately large dataset which is a synthetically created benchmark. Then the experiments are performed to test the efficiency of the method on the images of handwritten sketch-like floorplans. Lastly we conducted some experiments to test the method on some noisy images, where the kind of noise is very similar to the noise introduced by scanning or any other low-level pre-processing.

The set of available query symbols for each dataset are used as queries to evaluate the ground truths. A particular retrieved symbol is regarded as true positive the bounding region of the symbol will have at least 50% overlapping with the corresponding ground truth. For each of the symbols, the performance of the algorithm is evaluated in terms of precision (**P**), recall (**R**) and average precision (**AveP**). In general, the precision (**P**) and recall (**R**) are computed as:

$$P = \frac{|ret \cap rel|}{|ret|}; R = \frac{|ret \cap rel|}{|rel|} \tag{4.5}$$

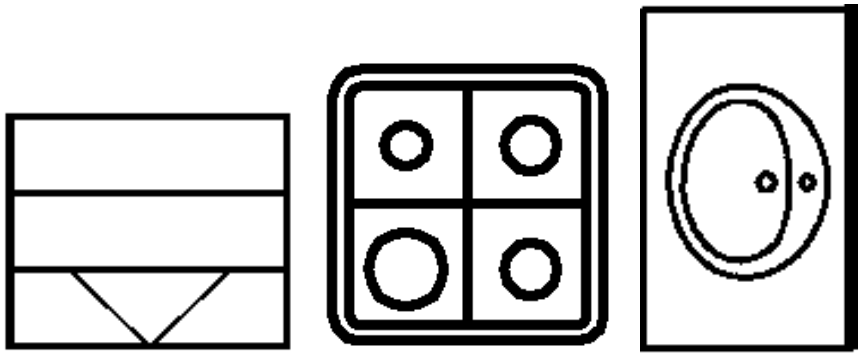
Here in Eqn. 4.5, the precision and recall measures are computed on the whole set of retrievals returned by the system. That is, they give information about the

final performance of the system after processing a query and do not take into account the quality of ranking in the resulting list. But IR systems return results ranked by a confidence value. The first retrieved items are the ones the system believes that are more likely to match the query. As the system provides more and more results, the probability to find non-relevant items increases. So in this experimental evaluation the precision value is computed as the $P(r_{max})$ *i.e.* the precision attained at r_{max}, where r_{max} is the maximum recall attained by the system and average precision is computed as:

$$AveP = \frac{\sum_{n=1}^{n=|ret|} P(n) \times r(n)}{|rel|} \tag{4.6}$$

where $r(n)$ is an indicator function equal to one, if the item at rank n is a relevant instance or zero otherwise. The interested reader is referred to [86] for the definition of the previously mentioned metrics for the symbol spotting problem. To examine the computation time we calculate the per document retrieval time (**T**) for each of the symbols. For each of the datasets the mean of the above mentioned metrics are shown to judge the overall performance of the algorithm.

All the experiments described below are performed with the Zernike moments descriptors with order 7 (dimension d=36). For LSH, the hashing data structures are created with L=10 and K=60. These parameters are experimentally decided to give the best performance. LSH reduces the search space significantly, for example SESYD (floorplans16-01) consists of approximately 1,465,000 paths and after lookup table construction, these paths are stored in 16,000 buckets, so compared to a one-to-one path comparison, the search space is reduced by a factor of 90.



(a) (b) (c)

Figure 4.7: Examples of model symbols from the FPLAN-POLY dataset used for our experiment.

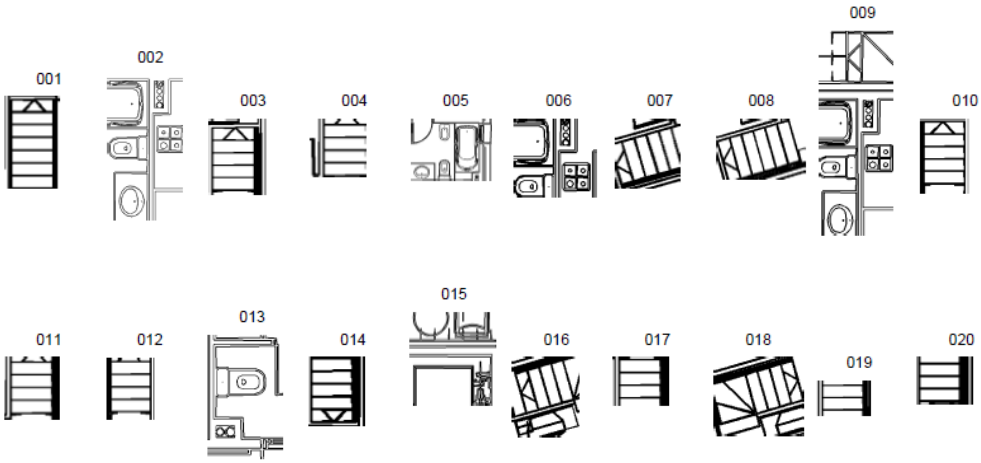


Figure 4.8: Qualitative results of the method: first 20 retrieved regions obtained by querying the symbol in Figure 4.7a in the FPLAN-POLY dataset.

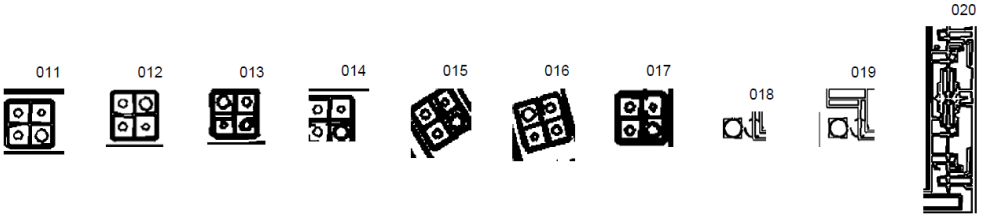


Figure 4.9: Qualitative results of the method: first 20 retrieved regions obtained by querying the symbol in Figure 4.7b in the FPLAN-POLY dataset.

Experiment on FPLAN-POLY with real-world images

We have tested our method with the FPLAN-POLY dataset. This experiment is undertaken to show the efficiency of the algorithm on real images, which could suffer from the noise introduced in the scanning process, vectorization etc.

The recall rate achieved by the method is 93% which shows the efficiency of the algorithm in retrieving the true symbols. The average precision obtained by the method is 79.52% which ensures the occupancy of the true positives at the beginning of the

ranked retrieval list. The precision value of the method is 77.87% which is more than 50% better than the precision reported by the latest state-of-the-art method [87] on this dataset. This signifies that the false positives are ranked worse than the correct results. This fact is also clear from Figure 4.8, 4.9, where we show the qualitative results obtained by the method. Also the method is efficient in terms of time complexity since the average time taken to spot a symbol per document is 0.18 sec.

Scalability experiment on SESYD

We have also tested our method on the SESYD (floorplans) dataset. This experiment is designed to test the scalability of the algorithm *i.e.* to check the performance of the method on a dataset which is sufficiently large.

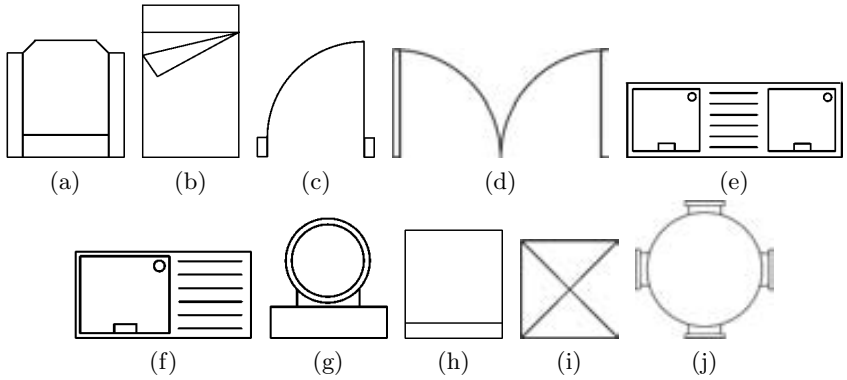


Figure 4.10: Example of different isolated symbols: (a) *armchair*, (b) *door2*, (c) *sink4*, (d) *table3*.

The mean measurements for each of the sub-datasets are shown in Table 4.1. The recall values for all the sub-datasets are quite good, although the average precisions are less than in the previous experiments. This is due to the existence of the similar substructures (graph paths) among different symbols (for example, between the symbols in Figure 4.10c and Figure 4.10d between the symbols in Figure 4.10e and Figure 4.10f, among the symbols in Figures 4.10a, 4.10b, 4.10h and 4.10i and etc). These similarities negatively affect the vote values considered for ranking the retrievals. There is an interesting observation regarding the average time taken for the retrieval procedure, which is 0.07 sec. to retrieve a symbol per document image, which is much less than the previous experiment. This is due to the hashing technique, which allows for the collision of the same structural elements and inserts them into the same buckets. So even though the search space increases due to hashing of the graph paths, it remains nearly constant for each of the model symbols. This ultimately reduces the per document retrieval time. To get an idea about the performance of the method, in Figures 4.11, 4.12, 4.13 and 4.14, we present some qualitative results on the SESYD dataset.

Table 4.1: Results with SESYD dataset

Database	P	R	AveP	T
floorplans16-01	41.33	82.66	52.46	0.07
floorplans16-02	45.27	82.00	56.17	0.09
floorplans16-03	48.75	85.52	71.19	0.07
floorplans16-04	54.51	74.92	65.89	0.05
floorplans16-05	53.25	91.67	67.79	0.08
floorplans16-06	52.70	78.91	60.67	0.07
floorplans16-07	52.78	83.95	65.34	0.07
floorplans16-08	49.74	90.19	58.15	0.08
floorplans16-09	51.92	77.77	47.68	0.07
floorplans16-10	50.96	83.01	63.39	0.08
mean	**50.32**	**83.06**	**60.87**	**0.07**

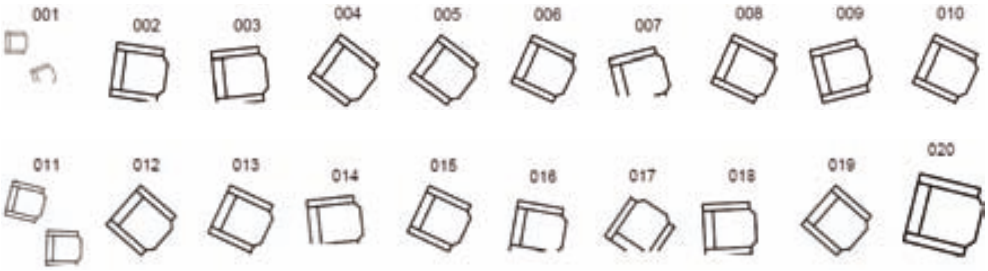


Figure 4.11: Qualitative results of the method: first 20 retrieved regions obtained by querying the symbol shown in Figure 4.10a in the SESYD (floorplans16-01) dataset.

Figure 4.12: Qualitative results of the method: first 20 retrieved regions obtained by querying the symbol shown in Figure 4.10d in the SESYD (floorplans16-05) dataset.

Experiment on SESYD-VN to test vectorial distortion

This experiment is undertaken to test the effectiveness of the algorithm on the hand-written sketch-like floorplans. For this, we have considered SESYD-VN dataset (see

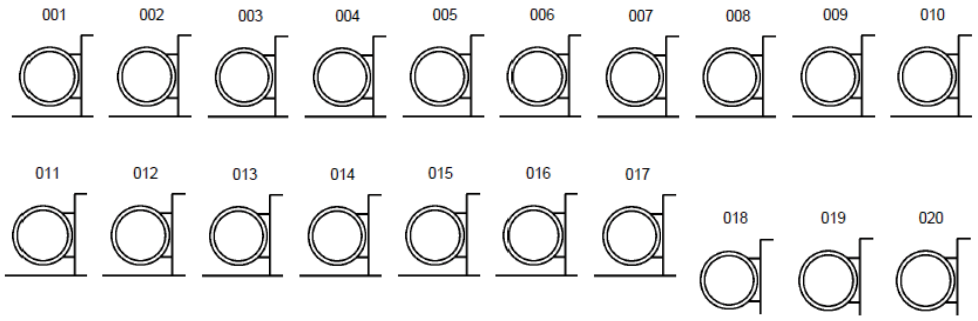


Figure 4.13: Qualitative results of the method: first 20 retrieved regions obtained by querying the symbol shown in Figure 4.10g in the SESYD (floorplans16-05) dataset.

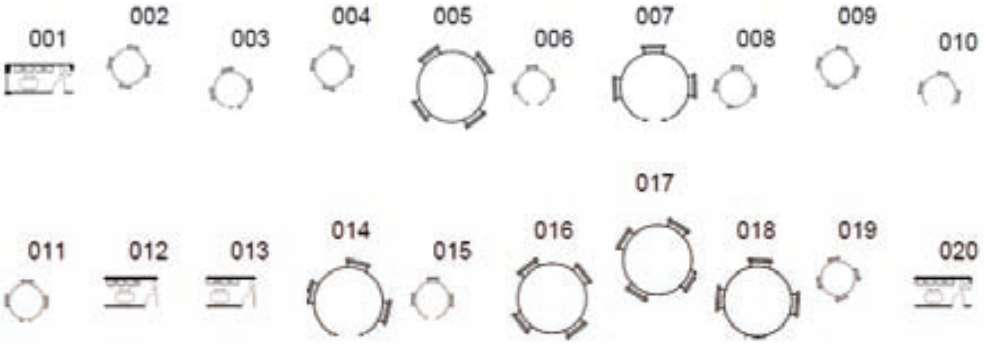


Figure 4.14: Qualitative results of the method: first 20 retrieved regions obtained by querying the symbol shown in Figure 4.10j in the SESYD (floorplans16-01) dataset.

Section A.5 for details). For this experiment we have created 3 levels of difficulty (for $r = 5, 10, 15$). For all the different distortions the same model symbols are used as queries.

Table 4.2: Results with SESYD-VN dataset

Radius (r)	**P**	**R**	**AveP**	**T**
r=5	63.64	92.19	65.27	0.25
r=10	47.49	87.01	56.82	0.26
r=15	34.37	82.16	47.80	0.25

The measurements of the method are shown in Table 4.2. The recall value for the dataset with minimum distortion ($r = 5$) is quite good, but it decreases with

the increment of distortion. The same incident is observed for average precision also. The distortion also introduces many false positives which harms the precision. In this experiment, the per document retrieval time of model symbols increases when compared to the previous experiment. This is due to the increment of randomness in the factorized graph paths which decreases the similarity among them. This compels the hashing technique to create a large number of buckets and hence ultimately increases the per document retrieval time.

Table 4.3: Results with SESYD-GN dataset

mean (m)	**variance** (σ)	**P**	**R**	**AveP**	**T**
0.1	0.01	24.36	94.86	74.07	0.25
	0.05	21.79	89.46	60.07	0.35
	0.09	15.38	67.77	42.85	1.47
0.2	0.01	24.36	94.87	73.43	0.26
	0.05	20.00	82.19	48.93	1.16
	0.09	15.38	65.44	30.97	1.58
0.3	0.01	24.10	93.34	65.79	2.12
	0.05	14.62	69.11	40.81	2.30
	0.09	12.05	54.12	25.62	3.15
0.4	0.01	15.89	72.45	36.32	1.95
	0.05	11.79	50.64	17.97	2.11
	0.09	11.54	43.78	15.29	2.49
0.5	0.01	9.74	34.56	10.00	0.52
	0.05	8.20	29.94	6.69	0.74
	0.09	9.23	36.07	11.14	0.84

Experiment on SESYD-GN with noisy images

The last symbol spotting experiment is performed to test the efficiency of the algorithm on noisy images, which might be generated in the scanning process. For this, we have considered the SESYD-GN dataset with the mean (m) of 0.1 to 0.5 with step 0.1 and with variance (σ) 0.01 to 0.09 with step 0.04, which generates a total 15 sets of images with different levels of noise (see Section A.4 for details). Practically, the increment of variance introduced more pepper noise into the images, whereas the increment of the mean introduced more and more white noise, which will detach the object pixel connection. Here we do not apply any kind of noise removal technique other than pruning, which eliminates isolated sets of pixels.

The mean measures of metrics are shown in Table 4.3 and the performance of the method is shown in Figure 4.15 in terms of the precision recall curves. Clearly, from the precision-recall curves, the impact of variance is more than that of the mean. This implies that with the introduction of more and more random black pixels, there

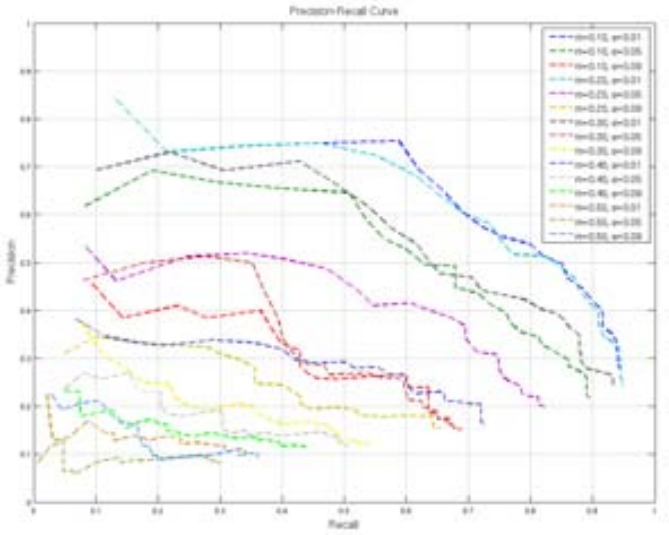


Figure 4.15: Precision-Recall plot generated by the spotting experiments with different levels of Gaussian noise.

is a decrease in the performance, which is due to the distortion in the object pixels that substantially affects the vectorization methods and changes the local structural features of the graph paths. On the other hand, the increment of the mean introduces white pixel noise which ultimately separates an object into different parts and which facilitates the loss of the local structural information. Increase in Gaussian noise introduces local distortions (both with black and white pixels) which introduces extra points, as well as discontinuity during the vectorization process. These random points increase the time for computing the paths and also the number of buckets due to the random structure of them. Since the increment of the mean after a certain stage breaks a component into several pieces, the vectorization results in simple structures of isolated components. These structures are quite similar, since in most of the cases they are the straight lines or simple combination of straight lines which further decrease the retrieval time as they reduce the number of buckets. This explains the increase of retrieval time up to a certain stage and then again the decrease. The increment of both mean and standard deviation of the Gaussian noise creates a lot of discontinuity within the structure of objects; this creates lot of spurious parts after vectorization. These parts are not distinctive among different symbolic objects, which explains the irregular shape of the precision recall curves with the increase of noise.

4.3.4 Experiment on handwritten word spotting

This experiment is performed to demonstrate the possibility of applying our method to any other kind of information spotting system. For that we have chosen a handwritten word spotting application which also has received some popularity amongst the

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

Figure 4.16: An image from the marriage register from the fifth century from the Barcelona cathedral, (a) The original image, (b) The binarized image of 4.16a, (c) The image in 4.16b after preprocessing (eliminating black border created due to scanning), (d) Graph constructed from the image in 4.16c: the inset also shows the zoomed part of a word Ramon .

research community. The experiment is performed on a set of 10 unsegmented handwritten images taken from a collection of historical manuscripts from the marriage register of the Barcelona cathedral (see Figure 4.16). Each page of the manuscripts contains approximately 300 words. The original larger dataset is intended for retrieval, indexing and to store in a digital archive for future access. We use skeletonization based vectorization to obtain the vectorized documents. Before skeletonization, the images undergo preprocessing such as binarization by Otsu s method [72] and removal of the black borders generated in the scanning process. Then we construct the graph from the vectorial information and proceed by considering this as a symbol spotting problem. The retrieval results of the method on the handwritten images are promising, which is also clear from the qualitative results shown in Figure 4.17. This shows a very good retrieval of the word de with almost perfect segmentation. We also observe some limitations of the method in spotting handwritten words, among them, when a particular query word is split into several characters or components, the method is more prone to retrieve the character, which is more discriminative with respect to the other characters in the word. This is due to the non-connectivity of the word symbol, which reduces the overall structural information. Another important observation is that the computation of paths takes a substantial amount of time for the handwritten documents, since handwritten characters contain many curves. This generate more and more spurious critical points in the images, which ultimately affects the path computation time.

Figure 4.17: The first 120 retrievals of the handwritten word de in the Marriage documents of the Barcelona Cathedral.

4.3.5 Discussions

We compare our results with three state-of-the-art methods respectively proposed by Luqman *et al.* [59], Rusinol *et al.* [84] and Qureshi *et al.* [77]. The method put forward by Luqman et al is based on graph embedding, the method due to Rusinol *et al.*is based on the relational indexing of primitive regions contained in the symbol and that proposed by Qureshi *et al.*is based on graph matching, where the methods due to Luqman *et al.*and Qureshi *et al.* [59, 77] use a pre-segmentation technique to find the regions of interest, which probably contain the graphic symbols. Generally this kind of localization method works to find some region containing loops and circular structures etc. Then a graph matching technique is applied either directly in the graph domain or in the embedded space to each of the regions in order to match the queried symbol. The method proposed by Rusinol *et al.* [84] works without any pre-segmentation. For experimentation, we considered the images from a sub-dataset of SESYD, The sub-dataset contains 200 images of floorplans. The mean measurements at the recall value of 90.00% are shown in Table 4.4 and the performance of the algorithm is shown in terms of the precision-recall plot in Figure 4.18. Clearly, the proposed method dominates over the existing methods. For any given recall, the precision given by our method is approximately 12% more than that reported by Qureshi *et al.* [77], 10% more than that indicated by Rusinol *et al.* [84] and 6% more than that resulted by Luqman *et al.* [59], which is a substantial improvement.

Finally, we use our algorithm as a distance measuring function between a pair of isolated architectural symbols, let us say, S_1 and S_2. In this case we do not perform

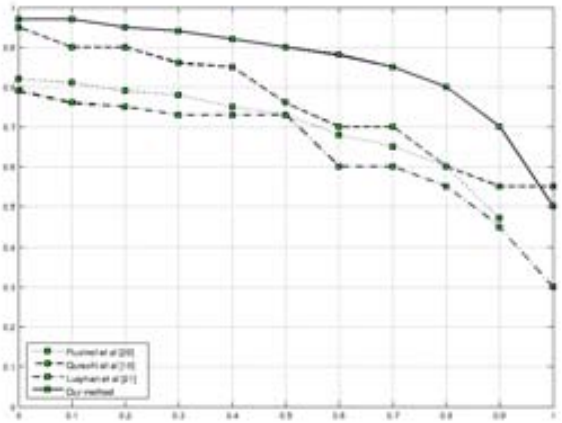


Figure 4.18: Precision-Recall plot generated by the spotting methods proposed by Luqman *et al.* [59], Qureshi *et al.* [77], Rusinol *et al.* [84] and our proposed method.

Table 4.4: Comparison with the state-of-the-art methods

Methods	P	R	AveP	T
Qureshi *et al.* [77]	45.10	90.00	64.45	1.21
Rusinol *et al.* [84]	47.89	90.00	64.51	-
Luqman *et al.* [59]	56.00	90.00	75.70	-
Our method	70.00	90.00	86.45	0.07

any hashing, instead we simply factorize the symbols into graph paths and describe them with some shape descriptors as explained in Section 4.2.2. Then we use these descriptors to match a path of, say, symbol S_1, to the most identical path of S_2. So the total distance between the symbols S_1 and S_2 is the sum of such distances, which can be regarded as a modified version of Hausdorff distance [30]:

$$\sum_{p_i \in S_1} \min_{p_j \in S_2} dist(p_i, p_j) + \sum_{p_j \in S_2} \min_{p_i \in S_1} dist(p_i, p_j)$$

We use this total distance to select the nearest neighbours of the query symbol. It is expected that for a pair of identical symbols, the algorithm will give a lower distance than for a non-identical symbol. This experiment is undertaken to compare our method with various symbol recognition methods available in the literature. When using the GREC2005 [24] dataset for our experiments, we only considered the set with 150 model symbols. The results are summarized in Table 4.5. We have achieved a 100% recognition rate for clear symbols (rotated and scaled) which shows that our

method can efficiently handle the variation in scale and rotation. Our method outperforms the GREC participants (results obtained from [24]) for degradation models 1, 2, 3 and 5. The recognition rate decreases drastically for models 4 and 6, this is because the models of degradation lose connectivity among the foreground pixels. So after the vectorization, the constructed graph can not represent the complete symbol, which explains the poorer results.

Table 4.5: Results of symbol recognition experiments

Database	Recognition rate
Clear symbols (rotated & scaled)	100.00
Rotated & degraded (model-1)	96.73
Rotated & degraded (model-2)	98.67
Rotated & degraded (model-3)	97.54
Rotated & degraded (model-4)	31.76
Rotated & degraded (model-5)	95.00
Rotated & degraded (model-6)	28.00

In general the symbol spotting results of the system on the SESYD database are worse than the FPLAN-POLY (see Table 4.6). This is due to the existence of more similar symbols in the collection, which often create confusion amongst the query samples. But the average time for retrieving the symbols per document is much lower than the FPLAN-POLY database. This is because of the hashing technique that allows collision of the same structural elements and inserts them into the same buckets. So even though the search space increases, due to hashing of the graph paths, it remains nearly constant for each of the model symbols, which ultimately reduces the per document retrieval time.

Table 4.6: Comparative results on two databases FPLAN-POLY & SESYD

Database	P	R	AveP	T
FPLAN-POLY	77.87	93.43	79.52	0.18
SESYD	50.32	83.06	60.87	0.07

Our system also produces some erroneous results (see Figures 4.8(002, 005, 006, 013, 015) and Figures 4.19(001, 002, 003, 004, 014, 019)) due to the appearance of similar substructures in nearby locations. For example the symbol in Figures 4.7a contains some rectangular box like subparts. The paths from these derived substructures of the symbol resemble some commonly occurring substructures (walls, mounting boxes etc.) in a floorplan. This creates a lot of false votes, which explains the retrieval of the false instances in Figure 4.8. Similarly, the subparts of the symbol in Figure 4.7c resemble the subparts of some architectural symbols. This explains the occurrence of the false retrievals in Figure 4.19.

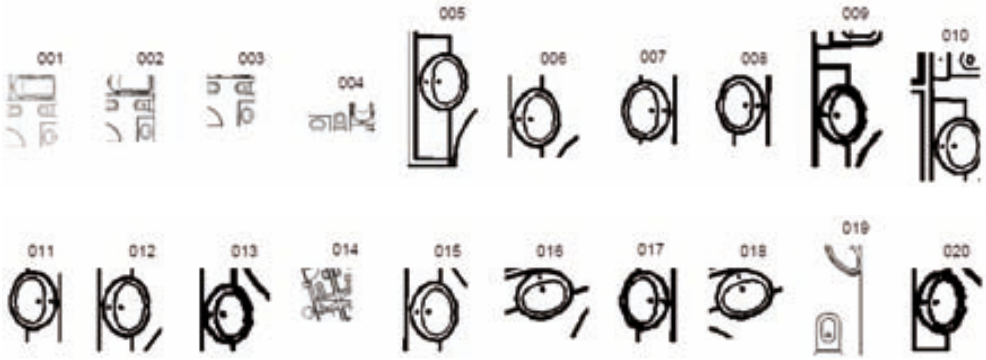


Figure 4.19: Qualitative results of the method: first 20 retrieved regions obtained by querying the symbol 4.7c in the FPLAN-POLY dataset.

4.4 Conclusions

In this chapter we have proposed a graph based approach for symbol spotting in graphical documents. We represent the documents with graphs where the critical points detected in the vectorized graphical documents are considered as the nodes and the lines joining them are considered as the edges. The document database is represented by the unification of the factorized substructures of graphs. Here the graph substructures are the acyclic graph paths between each pair of connected nodes. The factorized substructures are the one-dimensional (sub)graphs which give efficiency in terms of computation and since they provide a unified representation over the database, the computation is substantially reduced. Moreover, the paths give adaptation to some structural errors in documents with a certain degree of tolerance. We organize the graph paths in hash tables using the LSH technique, this helps to retrieve symbols in real-time. We have tested the method on different datasets of various kinds of document images.

Chapter 5

Product Graph based Inexact Subgraph Matching

In literature, there are methods that formulate (sub)graph matching as an optimization problem (OP) where the objective function is constructed from the pairwise (dis)similarities of the node, edge attributes. These methods usually emphasise on time efficient approximation of the OP. In this work we use walk based propagation of pairwise similarities on the tensor product graph (TPG) of two operand graphs to obtain higher order contextual information. We do it by counting the total number of weighted walks initiated from a certain node in TPG. We call them contextual similarities (CS) of the pair of nodes that constitute a node of TPG. After that we formulate the maximal common subgraph matching problem as a node and edge selection problem in TPG. To do that we use those CS to construct an objective function and optimize it with a linear programming (LP) formulation. With experiment we prove that the higher order CS add discriminations and allow one to efficiently approximate the optimization problem with LP. Since TPG takes into account higher order information, it is not surprise that we obtain more reliable similarities and better discrimination between the nodes/edges. Moreover, in this chapter, we propose a dual edge graph representation for line drawing images which solve the problem of distortion, noise. We apply our subgraph matching method for spotting symbols in line drawings represented by our dual graph representation.

5.1 Introduction

In this chapter we propose an inexact subgraph matching methodology based on *tensor product graph* (TPG). Roughly, TPG can be seen as a different form of association/affinity graph, where each node of the graph is an ordered pair of nodes from the two operand graphs. And two nodes are adjacent (or joined by an edge) if and only if the member nodes are adjacent in the corresponding operand graphs. Given

two attributed graphs it is quite straight forward to formulate them as the similarities (pairwise) and assign them as weights on the edges of TPG (step one in Figure 5.1). Now one can think of having a random walk from node to node considering those weights on the edges as the plausibility to proceed to the next node and add those plausibilities of the walks of different length between different pair of nodes, and perform this traversal through the whole graph. A similar phenomenon is termed as *diffusion on graph* and well known to capture higher order contextual information between objects [15, 106]. Finally, we accumulate the plausibilities of having a walk from each of the vertices, which we refer as *contextual similarities* (CS). This information can be obtained by simple algebraic operation of the adjacency (or weight) matrix of the product graph (step two in Figure 5.1). We formulate maximal common subgraph (MCS) matching as a node and edge selection problem in TPG. To do that we use those CS and formulate a constrained optimization problem (COP) to optimize a function constructed from the higher order similarity values (step three in Figure 5.1). We solve the COP with a linear programming (LP) which is solvable in polynomial time. In Section 5.3, with experiments we have shown that the higher order contextual similarities allow us to relax the constrained optimization problem in real world scenarios.

The main contributions of this chapter are threefold:

- First, we propose a random walk based way to obtain higher order contextual similarities between nodes/edges that capture higher order information. These similarities can be used instead of the pairwise similarities and give better results in real scenario. We have proved the effectiveness of the proposal with experimental study.
- Second, we formulate subgraph matching procedure as a node and edge selection procedure in the product graph which further can be formulated as a constrained optimization problem. For that we model a LP problem which can be solved quite efficiently with minimum number of outliers. Here we are motivated by the ILP formulation in [8]. We proved with experiments that the higher order CS allow one to relax the constrained optimization problem.
- The third contribution is application dependent, where we have considered symbol spotting in graphical documents as an inexact subgraph matching problem which is a widely used approach for spotting symbols in graphical documents. The problem of robust graph representation for graphical documents is very common and also difficult. For example, famous graph representation technique such as region adjacency graph (RAG) can not handle the objects that are not confined in well defined regions. On the other hand representing a graphical document with a graph involves several low level image processing steps, approximation etc. which introduce lot of structural noise. In this chapter, motivated by dual graph of a plan graph representation, we introduce dual graph of an edge graph representation to handle the structural distortions and some limitations that can not be handled by popular graph representations.

The rest of the chapter is organized in four sections. In Section 5.2, we present a

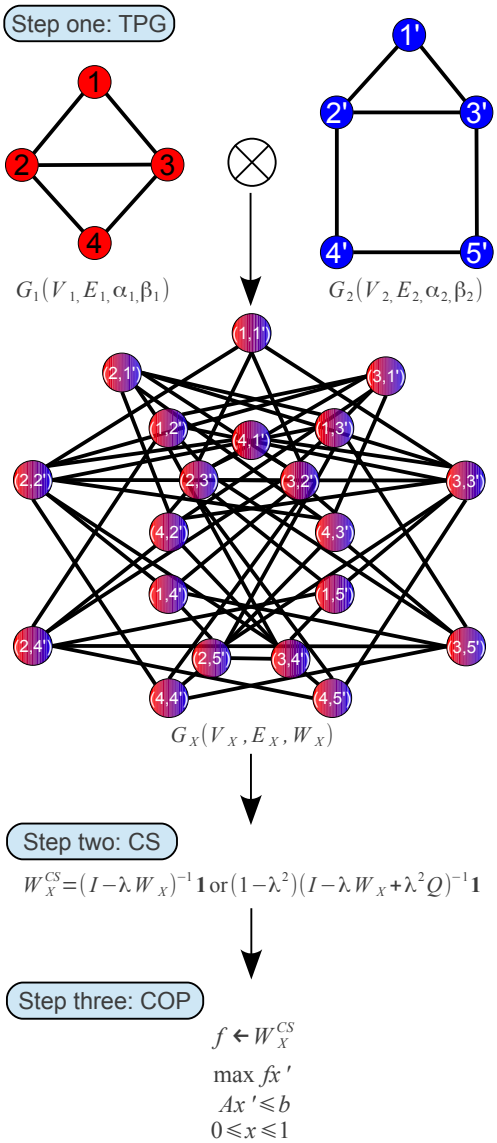


Figure 5.1: Outline of the proposed method: Step one: computation of the tensor product graph (TPG), Step two: algebraic procedure to obtain contextual similarities (CS), Step three: constrained optimization problem (COP) for matching subgraph.

product graph based subgraph matching methodology which includes the description of having the CS and also the formulation of MCS as a COP. In Section 5.3 we present the experimental results carried out to show the robustness of the method, apart from that here we discuss the dual graph representation for graphical documents. At last in Section 5.4 we conclude the chapter and possible future directions of this work are presented.

5.2 Methodology

A very nice property of the adjacency matrix A of any graph G is that the $(i\ j)$th entry of A^n denotes the number of walks of length n from the node i to the node j. One can bring the same analogy to an edge weighted graphs where each edge is associated with weights in [0 1] and can be considered as the plausibilities of moving from one node to another. Then following the same idea the $(i\ j)$th entry of W^n denotes the plausibility of having a walk of length n from the node i to the node j. To avoid the dependency on n one can consider all the walks upto the length infinity and add them up. Let SW be the sum of all such powers of W upto infinity. In that case the value of $SW(i\ j)$ signifies the combined plausibility of reaching the node j from the node i. Let SW_c be the row wise summation of SW, then SW_c is a column vector and $SW_c(i)$ indicates the likelihood of starting a walk from the node i or in other words it signifies the plausibility of visiting the node i, which we refer as CS. Here a higher entry reveals that the corresponding node is better connected with the rest of the graph in terms of the weights on the edges.

The same procedure can be simulated on a TPG of two operand graphs and can be used to capture higher order contextual information between pair of objects represented as nodes. The process can be started by assigning the pairwise similarities between nodes and edges as the weights on the corresponding edges of TPG, let W_X be such a weight matrix. Then simultaneous walking can be performed from node to node taking the weights on the edges as the plausibilities to move from one node to the next one. Let SW_X be the sum of all such powers of W_X upto infinity and let SW_{X_c} be the row wise summation of SW_X. Similar to the previous explanation, here $SW_{X_c}(i)$ indicates the likelihood of starting a walk from the node i and here a higher entry reveals that the node is better connected with the rest of the TPG in terms of the weights on the edges. As the weights on the edges of TPG come from the similarities of nodes and edges of the operand graphs, here a better connected node of TPG supposed to constitute a promising pair of matched nodes. Since the walking is performed through the edges of TPG, the accumulated weights take into account contextual information. This procedure of accumulating weights by considering connections between objects is proved to be more discriminative for objects where contextual informations are important such as graph. This is also the inner idea of *graph diffusion*, which is well known to capture higher order context similarities and intrinsic relations when done with pair of objects [15, 94, 106]. In this work we model the procedure of walking in two different ways, which we describe below.

5.2.1 Random walks

The easiest way to get the contextual similarities between pair of objects through graph is by propagating the pairwise similarity information with random walks on TPG. Below we use $\mathbf{I}$ to denote the identity matrix and $\mathbf{1}$ to denote a column vector whose all elements are set to 1. When it is clear from context we will not mention the dimensions of these vectors and matrices. Let W_X be the weight matrix of G_X, then the process of obtaining contextual similarities with random walks can be defined as:

$$SW_X = \lim_{n} \sum_{k=0}^{n} \lambda^k W_{X_k} \tag{5.1}$$

where

$$W_{X_k} = W_X^k \tag{5.2}$$

Here λ is a weighting factor to discount the longer walks, as they often contain redundant or repeated information. In this chapter we always choose $\lambda = \frac{1}{a}$, where $a = \min(\Delta^+(W_X)\ \Delta^-(W_X))$. Here $\Delta^+(W_X)$, $\Delta^-(W_X)$ are respectively the maximum outward and inward degree of W_X [31].

The above summation converges for sufficiently small value of λ. In that case, to get rid of the iterative matrix multiplication procedures, one can consider the infinite sum as follows:

$$SW_X = \lim_{n} \sum_{k=0}^{n} \lambda^k W_{X_k} = (\mathbf{I} - \lambda W_X)^{-1} \tag{5.3}$$

Then a vector containing the contextual similarities for all the nodes can be obtained by a matrix-vector multiplication as follows:

$$W_X^{CS} = (\mathbf{I} - \lambda W_X)^{-1}\mathbf{1} \tag{5.4}$$

Eqn. 5.4 can be efficiently computed by solving $(\mathbf{I} - \lambda W_X)x = \mathbf{1}$ by conjugate gradient methods which allows to avoid the expensive matrix inversion. An entry $W_X^{CS}(\ _1\ \ _2)$ indicates the plausibility of having a random walk to any node from the node $(\ _1\ \ _2)$ of TPG. Here since the weights on the edges of TPG are derived from the pairwise similarities of node, edge attributes, a higher value in $W_X^{CS}(\ _1\ \ _2)$ reveals a better candidate for pairwise matching. Here it is to be noted that W_{CS}^X is a column vector, the comma separated double subscript is used just to ease the understanding.

For the sake of understandability, let us illustrate the method using a simple

example. Let us take a very simple weight matrix W as follows:

$$W = \begin{bmatrix} 0 & w_{12} & w_{13} \\ w_{21} & 0 & w_{23} \\ w_{31} & w_{32} & 0 \end{bmatrix}$$

where w_{ij} denotes the similarity between the nodes i and j. Now the summation upto iteration 2 is:

$$W_1 + W_2 = \begin{bmatrix} w_{12}w_{21} + w_{13}w_{31} & w_{12} + w_{13}w_{32} & w_{13} + w_{12}w_{23} \\ w_{21} + w_{23}w_{31} & w_{21}w_{12} + w_{23}w_{32} & w_{23} + w_{21}w_{13} \\ w_{31} + w_{32}w_{21} & w_{32} + w_{31}w_{12} & w_{31}w_{13} + w_{32}w_{23} \end{bmatrix}$$

Here it is clear that the exponentiation plus the summation procedure takes into account information from the context to determine the strength of a particular edge. For example, to determine the strength of the edge (1 2), it considers the weights on the edges (1 3) and (3 2). An actual occurrence of a pattern graph in the target graph creates higher similarities in the neighbourhood, this also effects the connected nodes. This formulation enhances the pairwise similarities with more information from context. On the other hand, in this formulation the effect of occurrence of an outlier gets minimized. Now the ith entry of the row wise summation of $W_1 + W_2$ gives the plausibility of initiating a walk of length two from the node i. As explained before, here a higher entry reveals how well a node is connected with rest of the TPG in terms of similarities. In other words, it gives a similarity measures of the pair of nodes in the operand graphs.

The main problem of the random walk based procedure is that it backtracks an edge in case of a undirected graph and reduce the discriminations. To solve this limitation, *backtrackless walks*, a variation of random walks have been recently proposed [4]. In the next section we describe how to adapt this to our approach.

5.2.2 Backtrackless walks

A similar formulation as random walks can also be done with backtrackless walks [4]. The backtrackless walks are also random walks but do not backtrack an edge and for that a variation of exponentiation is available [92]. Let W_X be the weight matrix of G_X, then the process of obtaining contextual similarities with backtrackless walks can be defined as:

$$SW_X = \lim_{n} \sum_{k=1}^{n} \lambda^k W_{X_k} \tag{5.5}$$

where

$$W_{X_k} = \begin{cases} W_X & \text{if } k = 1 \\ W_X^2 - (Q_X + I) & \text{if } k = 2 \\ W_{X_{k-1}} W_X - W_{X_{k-2}} Q_X & \text{if } k \geq 3 \end{cases} \tag{5.6}$$

Here Q_X is a diagonal matrix where the ith or (i i)th element is equal to the ith or (i i)th element of W_X^2 minus one. Here also λ serves the same purpose. The above summation in Eqn. 5.5 converges for sufficiently small value of λ. In that case, to get rid of the iterative matrix multiplication procedures, one can consider the infinite sum as follows (for derivation see appendix):

$$SW_X = \lim_{n} \sum_{k=1}^{n} \lambda^k W_{X_k} = (1-\lambda^2)(\mathbf{I} - \lambda W_X + \lambda^2 Q_X)^{-1} \tag{5.7}$$

Then the weight vector for each node can be obtained by a matrix-vector multiplication as follows:

$$W_X^{CS} = (1-\lambda^2)(\mathbf{I} - \lambda W_X + \lambda^2 Q_X)^{-1}\mathbf{1} \tag{5.8}$$

Similar to Eqn. 5.4, Eqn. 5.8 can also be computed by solving $(\mathbf{I} - \lambda W_X + \lambda^2 Q_X)x = \mathbf{1}$ and then multiplying the solution by $(1-\lambda^2)$.

Here also the phenomena regarding context can be explained with the same example as follows:

$$W = \begin{bmatrix} 0 & w_{12} & w_{13} \\ w_{21} & 0 & w_{23} \\ w_{31} & w_{32} & 0 \end{bmatrix}$$

where w_{ij} denotes the similarity between the nodes i and j. Then Q_X can be written as follows:

$$Q_X = \begin{bmatrix} w_{12}w_{21} + w_{13}w_{31} - 1 & 0 & 0 \\ 0 & w_{12}w_{21} + w_{23}w_{32} - 1 & 0 \\ 0 & 0 & w_{13}w_{31} + w_{23}w_{32} - 1 \end{bmatrix}$$

Now the summation of the series upto the iteration 2 of W is:

$$W_1 + W_2 = \begin{bmatrix} 0 & w_{12} + w_{13}w_{32} & w_{13} + w_{12}w_{23} \\ w_{21} + w_{23}w_{31} & 0 & w_{23} + w_{21}w_{13} \\ w_{31} + w_{32}w_{21} & w_{32} + w_{31}w_{12} & 0 \end{bmatrix}$$

Here also it is clear that the exponentiation plus the summation procedure also takes into account contextual information to determine the strength of a particular edge and in each iteration it eliminates the tottering effect (by eliminating the loops) with the special algebraic formulation in Eqn. 5.6. An actual occurrence of a pattern graph in the target graph creates higher pairwise similarities in the neighbourhood, this also effects the connected nodes. This formulation enhances the pairwise similarities with more information from context/connection. On the other hand, in this formulation an occurrence of an outlier gets minimized. Now the ith entry of the row wise summation of $W_1 + W_2$ gives the plausibility of having a walk of length two from the node i. As

before, here a higher entry reveals how well a node is connected with rest of the TPG in terms of similarities. It gives a similarity measures of the pair of nodes in the operand graphs.

We use the contextual similarities obtained in the steps explained above to formulate maximal common subgraph matching algorithm as a constrained optimization problem.

5.2.3 Subgraph matching as a constrained optimization problem

We formulate a maximal common subgraph matching as a node, edge selection problem in TPG. To do that we use the CS obtained in the previous step to construct a maximization problem and solve it with a LP formulation. We construct two vectors S_V and S_E as follows:

$$S_V(u_1\ u_2) = W_X^{CS}(u_1\ u_2)\ \ (u_1\ u_2) \quad V_X$$

$$S_E((u_1\ u_2)\ (v_1\ v_2)) = \frac{W_X^{CS}(u_1\ u_2)}{\Delta^+(u_1\ u_2)} + \frac{W_X^{CS}(v_1\ v_2)}{\Delta^+(v_1\ v_2)}\ \ (u_1\ u_2) \quad V_X$$

Here $\Delta^+(u_1\ u_2)$ denotes the maximum outward degree of the node $(u_1\ u_2) \quad V_X$, S_V contains the higher order affinities of all the nodes $(u_1\ u_2) \quad V_X$ and S_E contains that for all the edges $((u_1\ u_2)\ (v_1\ v_2)) \quad E_X$. Here it is to be clarified that both of S_V and S_E are row vectors, the comma separated double subscripts are just to ease the understanding. Now clearly the dimension S_V is V_X and that of S_E is E_X . We formulate the maximal common subgraph (MCS) matching problem as a node, edge selection problem in the product graph. This can be formulated as a constrained optimization problem which maximize a function of higher order similarities of the nodes and edges. We construct the objective function as follows:

$$f(x\ y) = S_V x\ + S_E y \tag{5.9}$$

where x and y are row vectors containing variables denoting the probabilities of matching (selecting) the nodes and edges in the product graph respectively. For example, x_{u_1,u_2} denote the probability of matching the node $u_1 \quad V_1$ with the node $u_2 \quad V_2$. Similarly, $y_{u_1u_2,v_1v_2}$ denote the probability of matching the edge $(u_1\ v_1)$ E_1 with the edge $(u_2\ v_2) \quad E_2$. x and y contain probabilities as the optimization problem in Eqn. 5.9 is solved with linear or continuous programming with the domain $[0\ 1]$.

Now let us introduce a set of constraints on the variables to satisfy the maximal common subgraph matching problem between the operand graphs G_1 and G_2 in TPG G_X.

- **Pattern node constrain** Each node $u_1 \quad V_1$ can be matched with at most L number of nodes $u_2 \quad V_2$ *i.e.*there can be at most L number of nodes $(u_1\ u_2)$

V_X for each u_1 V_1.

$$\sum_{u_2 \ V_2} x_{u_1,u_2} <= L \quad u_1 \quad V_1$$

Here L is the number of instances of the pattern graph to be searched in the target graph.

- **Pattern edge constrain** Each edge $(u_1\ v_1)$ E_1 can be matched with at most L number of edges $(u_2\ v_2)$ E_2 *i.e.*there can be at most L number of edges $((u_1\ u_2)\ (v_1\ v_2))$ E_X for each $(u_1\ v_1)$ E_1

$$\sum_{(u_2,v_2) \ E_2} y_{u_1u_2,v_1v_2} <= L \quad (u_1\ v_1) \quad E_1$$

Here L is the number of instances of the pattern graph to be searched in the target graph.

- **Target node constrain** Each node u_2 V_2 can be matched with at most one node in u_1 V_1 *i.e.*there can be at most one node $(u_1\ u_2)$ V_X for each u_2 V_2.

$$\sum_{u_1 \ V_1} x_{u_1,u_2} <= 1 \quad u_2 \quad V_2$$

- **Target edge constrain** Each edge $(u_2\ v_2)$ E_2 can be matched with at most one of edge $(u_1\ v_1)$ E_1 *i.e.*there can be at most one edge $((u_1\ u_2)\ (v_1\ v_2))$ E_X for each $(u_2\ v_2)$ E_2

$$\sum_{(u_1,v_1) \ E_1} y_{u_1u_2,v_1v_2} <= 1 \quad (u_2\ v_2) \quad E_2$$

- **Outward degree constrain** The outward degree of a node $(u_1\ u_2)$ V_X is bounded above by the minimum of the outward degree of the node u_1 V_1 and u_2 V_2 *i.e.*the number of outgoing edges from the node $(u_1\ u_2)$ V_X is less than or equal to the minimum of the number of outward degree of the nodes u_1 V_1 and u_2 V_2.

$$\sum_{(v_1,v_2) \ V_X} y_{u_1u_2,v_1v_2} <= x_{u_1,u_2} \ \min\left(\Delta^+(u_1)\ \Delta^+(u_2)\right) \quad (u_1\ u_2) \quad V_X$$

- **Inward degree constrain** The inward degree of a node $(v_1\ v_2)$ V_X is bounded above by the minimum of the inward degree of the node v_1 V_1 and v_2 V_2 *i.e.*the number of incoming edges to the node $(v_1\ v_2)$ V_X is less than or equal to the minimum of number of inward degree of the nodes v_1 V_1 and v_2 V_2.

$$\sum_{(u_1,u_2) \ V_X} y_{u_1u_2,v_1v_2} <= x_{v_1,v_2} \ \min\left(\Delta^-(v_1)\ \Delta^-(v_2)\right) \quad (v_1\ v_2) \quad V_X$$

Table 5.1: Execution time of the exact graph matching experiment.

tvn	50			100			250			500		
pvn	rw	btlw	pw	rw	btlw	pw	rw	btlw	pw	rw	btlw	pw
10	0.03	0.04	0.03	0.12	0.13	0.11	0.52	0.49	0.46	1.73	1.77	2.17
20	0.13	0.012	0.14	0.43	0.41	0.46	1.83	1.80	2.10	13.24	13.21	15.34
50	0.77	0.75	0.78	2.99	2.91	3.20	21.72	21.67	33.23	292.60	291.98	475.24

- **Domain constrain** Finally, we restrict all the variables to be in between [0 1].

$$x_{u_1,u_2} \quad [0\ 1] \quad (u_1\ u_2) \quad V_X$$

$$y_{u_1u_2,v_1v_2} \quad [0\ 1] \quad ((u_1\ u_2)\ (v_1\ v_2)) \quad E_X$$

For having a node-node and edge-edge correspondence between the pattern and target graph, we consider all the non-zero values of the variables in x and y and consider it as matchings.

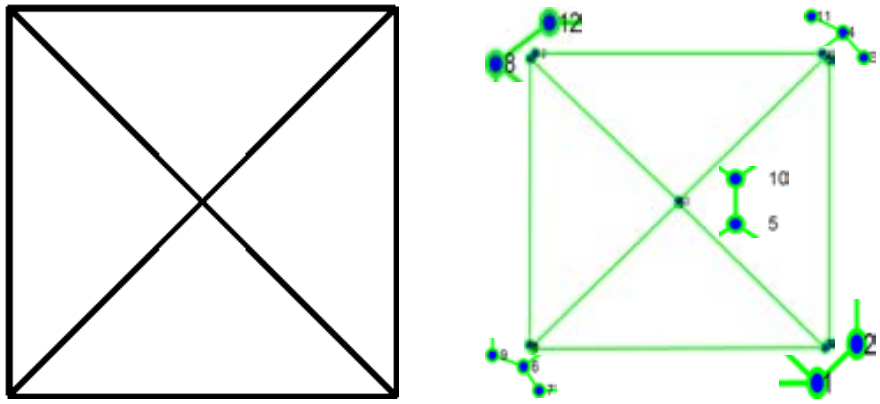
5.3 Experimental framework

We have performed two different experiments. The first one is designed to show the functionality of our proposed subgraph matching algorithm in an exact subgraph matching scenario. The second experiment is more focused on an application of inexact or error tolerant subgraph matching perspective. Here we convert the symbol spotting problem in graphical documents as a subgraph matching problem and apply the proposed product graph based subgraph matching algorithm for spotting symbols. All the experiments are done in a workstation with Intel Xeon 2.67GHz processor and 12GB of RAM. Our unoptimized Matlab code that we have been used for the experimentations is available in `http://www.cvc.uab.es/~adutta/ProductGraph`.

5.3.1 Exact subgraph matching

For this experiment we have considered the two synthetic subset of the ILPIso dataset (see Section A.7 for details), these two subsets contain graphs that are connected *i.e.*they do not have any isolated nodes. We run our subgraph matching algorithm with all the pattern-target pairs with $L = 1$.

We use the Euclidean distance to compute the node and edge distance while computing the product graph and given the distance d between two nodes (or edges), the similarity between them is computed as $s = e^{-d}$. For each pair we perform the subgraph matching in three different node, edge similarity settings: (1) higher order similarities with random walks, (2) higher order similarities with backtrackless walks and (3) pairwise similarities. As expected our proposed subgraph matching algorithm solved all the pair of instances with all the three different settings. Table 5.1 contains



(a) (b)

Figure 5.2: (a) An example symbol, (b) Graph representation of the symbol in (a) considering the critical points (detected by the vectorization algorithm) as the nodes and lines joining the critical points as the edges. Note the spurious nodes and edges generated near the junctions and corners. It is to be mentioned that in this case the vectorization is done by QGAR.

a comparison of average time (average over ten consecutive runs) taken to solve each pair of instances with the edge probability of 0 1 for three different settings.

In this experiment it is observed that with the increase in the number of nodes of the operand graphs the required time to solve a problem increases which is well expected. And also it is observed that for bigger operand graphs considering higher order contextual similarities gives benefits in terms of time. We explain this phenomena as an advantage of contextual similarities which adds more discrimination to the node and edge labels.

5.3.2 Symbol spotting as an inexact subgraph matching problem

For this experiment we have considered the two subsets (*floorplans16-05* and *floorplans16-06*) of SESYD (floorplans) dataset (see Section A.1 for details).

Efficient graph representation of graphical documents is a popular but difficult problem in the graphics recognition field. The steps converting a document to a graph involve several low level image processing such as binarization, skeletonization, polygonization etc., which introduce structural noise such as spurious nodes, edges (see Figure 5.2).

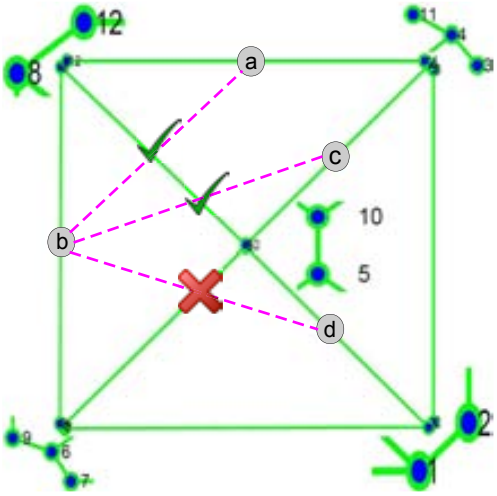


Figure 5.3: An illustration showing the details of dual graph representation. Here a, b, c and d are dual nodes and we consider $l = 1$. For that reason (b, a) and (b, c) are dual edges (shown in magenta and discontinuous line) since the corresponding edges (shown in green and continuous line) in the original graph are reachable with a shortest walk of length 1. There is no dual edge (b, d) since the shortest walk between the corresponding edges of b and d is 2. Consider the details near the junctions and corners.

Dual graph representation

To resolve this problem we consider a variation of *dual graph* representation. Originally dual graph of a plane graph G_F is a graph that has vertex corresponding to each face of G_F and an edge joining two neighbouring faces sharing a common edge in G_F. We bring the same analogy to our problem. Initially we have edge graphs G_E where each critical point (obtained by the vectorization algorithm) is represented as node and the line joining each pair of nodes as edge (as shown in Figure 5.2b). In our dual graph representation we assign a node to each edge of G_E. Any pair of dual nodes (nodes of dual graph) are joined by a dual edge (edge of dual graph) if the corresponding edges in the edge graph are reachable from each other with a shortest walk of length l. Lets call this graph representation as *dual edge graph* representation.

Let $G_1 = (V_1, E_1, \alpha_1, \beta_1)$ and $G_2 = (V_2, E_2, \alpha_2, \beta_2)$ be respectively the attributed dual edge graphs for the pattern and the target. Here $\alpha_1 : V_1(u) \rightarrow \mathbb{R}^{m \times 7}$ is a node labelling function and is defined as the set of seven Hu moments invariants [43] of the acyclic graph paths joining the extremities of an edge in the edge graph (m is total number of paths). α_2 is also defined in same way in V_2. $\beta_1 : E_1(u_1, v_1) \rightarrow \mathbb{R}^3$ is an edge labelling function and is constructed of three components:

- normalized angle (angle divided by 180°) between two edges (of the edge graph)

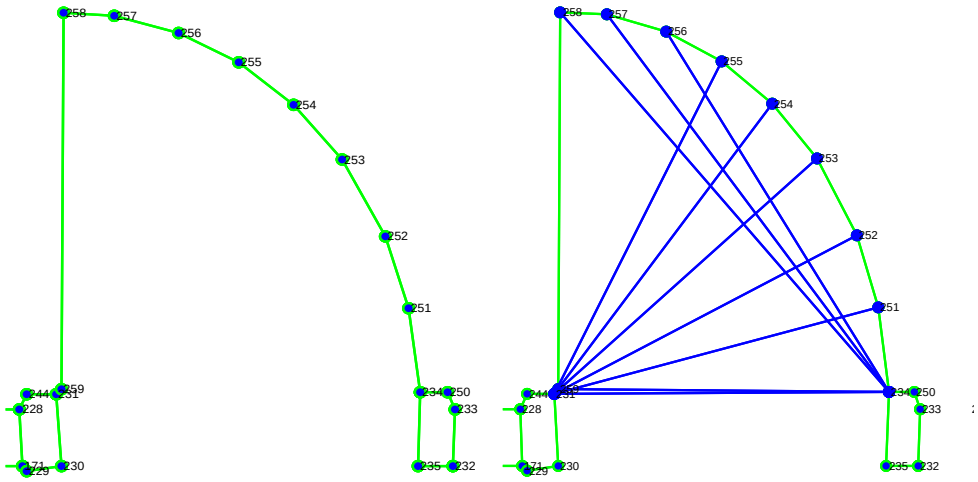


Figure 5.4: Transitive closure.

joined by the dual edge.

- the ratio of the two edges (of the edge graph) joined by the dual edge.
- the ratio of the distance between the mid points of the two edges (of the edge graph) and the total length of the edges joined by the dual edge.

Similarly, β_2 is defined in E_2.

The above representation fails when the extremities of a certain edge (of the edge graph) are not connected with other graph paths. In that case the corresponding dual node loses local discrimination as shown in Figure 5.4a. We resolve this difficulty by connecting those nodes in the edge graph by a relation inspired by the *transitive closure* of a graph. In general, *transitive closure* of a graph $G = (V\ E)$ is also a graph $G^\star = (V\ E^\star)$ such that $E^\star$ contains an edge $(u\ v)$ if and only if G contains a path (of at least one edge) from u to v. In our case, we only join those u, w so that $w = \arg\max_v dsp(u\ v)$. Here $dsp(u\ v)$ denotes the minimum number of nodes to be traversed to reach v from u.

Given the dual edge graph representation of the floorplan (target) and the symbol (pattern) we compute the product graph between them. For computing the distance between the dual nodes we use a modified version of Hausdorff distance as follows [30]:

$$d(A\ B) = \sum_{a\ A} \min_{b\ B} dm(a\ b) + \sum_{b\ B} \min_{a\ A} dm(a\ b)$$

Here $dm(a\ b)$ denotes the distance of $a\ \ A$ from $b\ \ B$ and $d(A\ B)$ denotes the distance between the sets A and B. Since the node label in our dual edge graph

representation is a set of Hu moments, we use this particular distance computation. For having the distance between the edges we use the Euclidean distance. Given the distance d between two nodes (or edges) we obtain the similarity between them as e^{-d}. Given the contextual similarities we perform the optimization based maximal subgraph matching to obtain the correspondences in target graph. Here also all the experiments were done with $L = 1$.

Initially we show the node-node matching for different pattern graphs from Figure 5.5 to 5.16 in pattern graph wise manner. In the above figures the left sub figures show the correspondences obtained by the context similarities due to random walks, the ones in the middle show that for backtrackless walks and those in the right show the same for pairwise similarities. Here it is clear that the context similarities really enhance the truly matched nodes, as in the case of random, backtrackless walks the most of the obtained correspondences focus to the actual instance. After obtaining the correspondences, we perform a simple clustering to group the neighbouring dual nodes to obtain a bounding box for comparing with the ground truths.

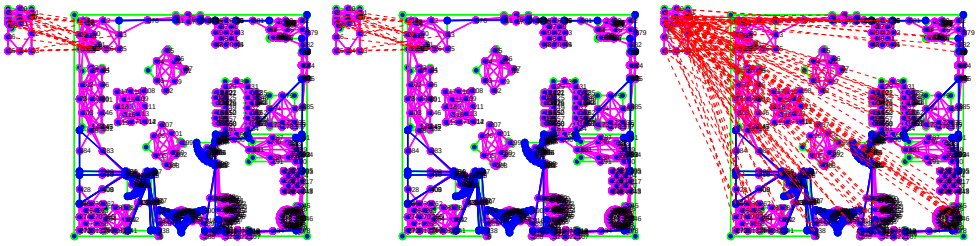


Figure 5.5: Matchings: *bed.*

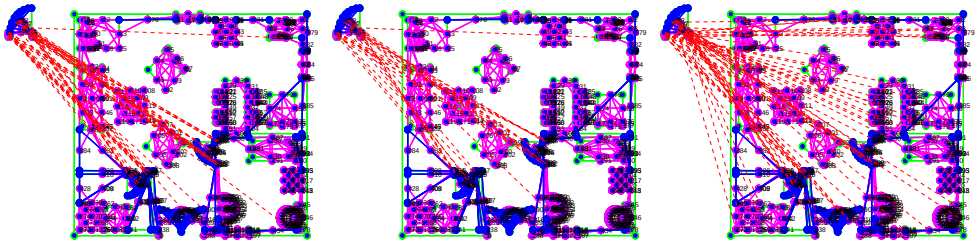


Figure 5.6: Matchings: *door1.*

Performance evaluation and results

The decision whether a retrieved subgraph is true or false is done by overlapping the bounding boxes, that is, if B be the rectangular bounding box for a retrieval and T be the bounding box of its corresponding ground truth, B is considered true if

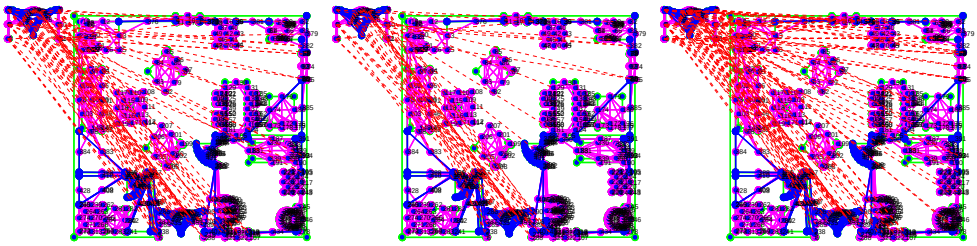


Figure 5.7: Matchings: *door2.*

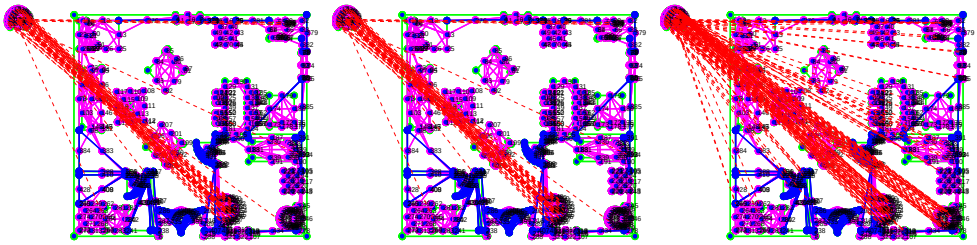


Figure 5.8: Matchings: *sink1.*

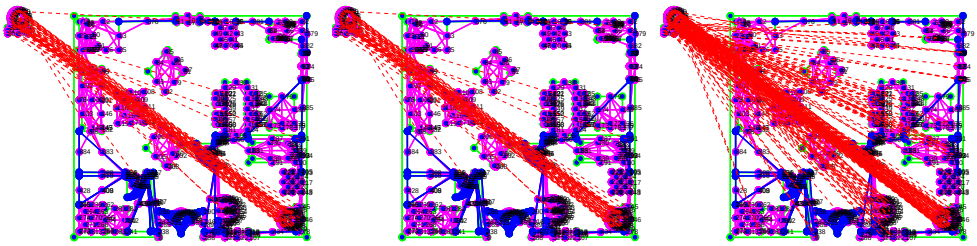


Figure 5.9: Matchings: *sink4.*

$\frac{B\ T}{B\ T} \geq 0\,5$. For quantitative evaluation, we have computed the usual metrics for evaluating a retrieval system such as precision (**P**), recall (**R**), F-measure (**F**) and average precision (**AP**). For detailed definitions of all such metrics, one can have a look to the paper [86]. To get an idea about the time complexity we have also shown the meantime (**T**) of matching a pattern graph in a target graph.

The results obtained are listed in Table 5.2. From the results it is clear that the method really performs well with contextual similarities than the pairwise similarities and there is not much difference between the higher order similarities learned from random walk based and backtrackless walks. As the first experiment it is also observed here that the average time taken by the method is significantly lower than the pairwise

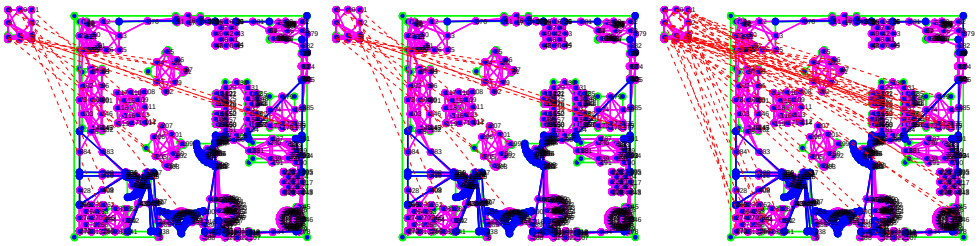


Figure 5.10: Matchings: *sofa1.*

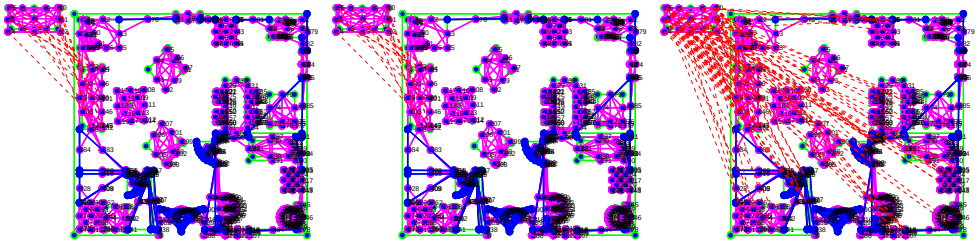


Figure 5.11: Matchings: *sofa2.*

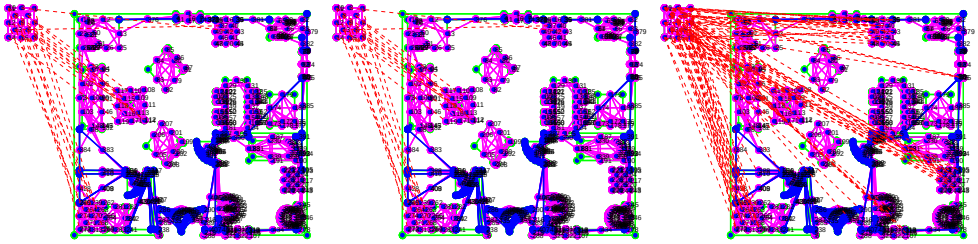


Figure 5.12: Matchings: *table1.*

similarities. We explain this phenomena as a benefit of using higher order similarities, which add more discrimination to the nodes and edges. Our method works quite well except for *sink3*, *sofa1*, *table1*, *window1*. The failures occur when the pattern graph has more than one component (for example *sink3*) and when there are many instances of the same symbol (*table1*, *sofa1* and *window1*).

Some of the qualitative results are shown in 5.18 to 5.25 in symbol wise manner. In each of the figures the left one is obtained by using the context similarities due to random walks, the middle one is obtained due to backtrackless walks and right one is obtained with pairwise similarities. In the figures, the bounding boxes with green border indicate the true positives and the ones with red border signify false positive.

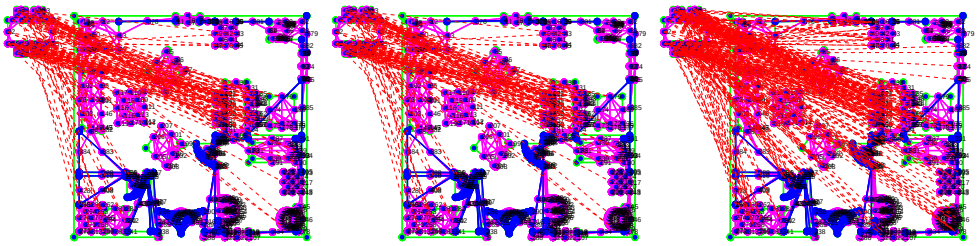


Figure 5.13: Matchings: *table2.*

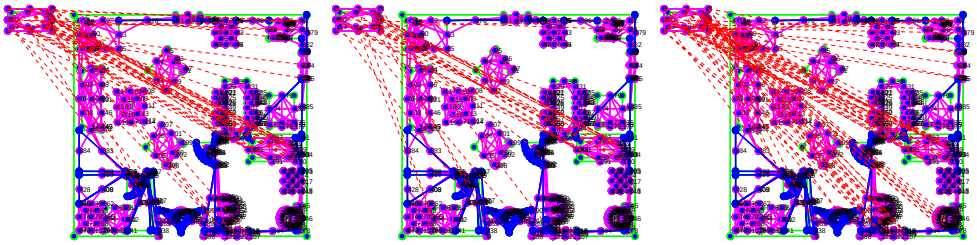


Figure 5.14: Matchings: *tub.*

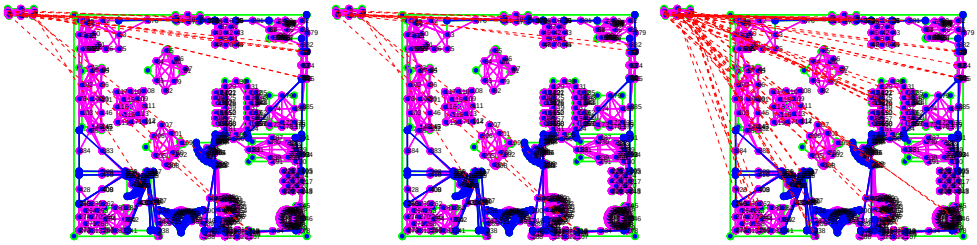


Figure 5.15: Matchings: *window1.*

Each of the retrieved bounding boxes are also accompanied with a similarity value. The similarity value for each of the bounding boxes is the summation of all the similarity of matched nodes. This similarity value is used to rank the retrievals. Rest of the qualitative results are available in `http://www.cvc.uab.es/~adutta/ProductGraph`.

To have a comparison between a state-of-the-art method, we have a considered the method by Le Bodic*et al.* [8]. For the comparisons the same floorplans are represented with RAG as done by the authors in the paper and each of the nodes is attributed with 24 Zernike moments and each of the edges is attributed by the ratio of the areas of the regions joined by the edges and distance between the cen-

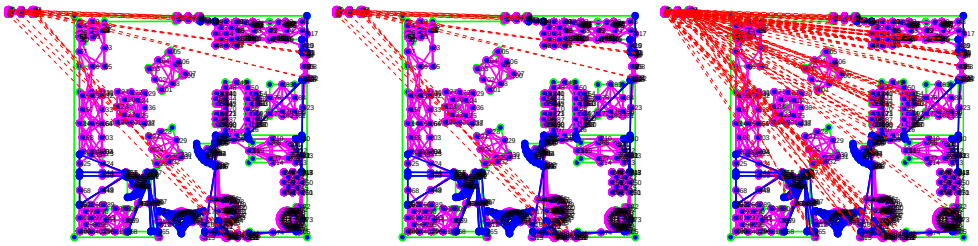


Figure 5.16: Matchings: *window2*.

Table 5.2: Overall results with three different settings.

	P	R	F	AP	T
random	69.90	84.95	78.78	81.10	33.43
backtrackless	70.56	86.29	80.10	82.98	33.37
pairwise	61.60	72.81	64.94	60.21	53.60

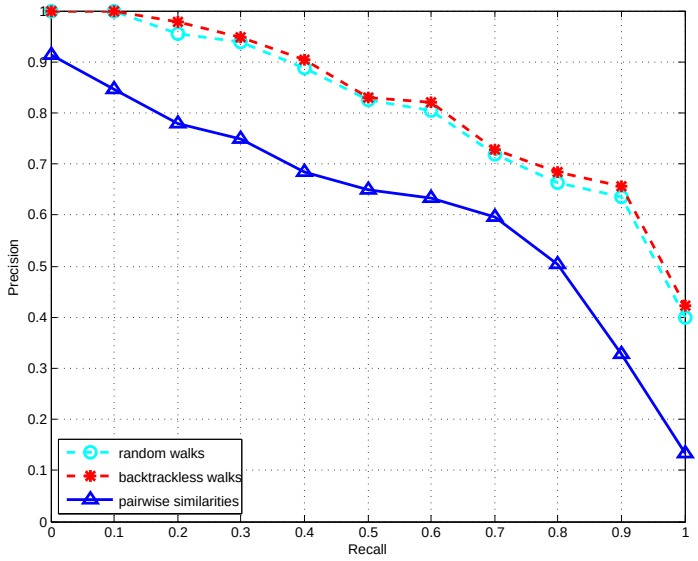


Figure 5.17: *Precision Recall curve.*

tre of the regions. The implementation of the method was taken from the website `http://litis-ilpiso.univ-rouen.fr/ILPIso`. In each of the target documents a single pattern symbol was queried for five instances. A retrieved instance is classified as true/false depending on the same overlapping criteria as the proposed method. The results obtained are listed in the first row of Table 5.3. The

Figure 5.18: Symbol spotting: *bed*. Green boxes are the true positives, the red ones are false positives.

Figure 5.19: Symbol spotting: *door1*. Green boxes are the true positives, the red ones are false positives.

Figure 5.20: Symbol spotting: *door2*. Green boxes are the true positives, the red ones are false positives.

method can perfectly retrieve all the instances of the symbol named *sink1*, *sink2*, *sink3*, *sofa2*, *table1*, *tub*, *window1*, *window2*. The method fails completely for *door1*, *door2* as the region adjacency graph can not represent them in stable way. For *sink4*, we observe there are some instability among the regions in the pattern and target

Figure 5.21: Symbol spotting: *sink1*. Green boxes are the true positives, the red ones are false positives.

Figure 5.22: Symbol spotting: *sink2*. Green boxes are the true positives, the red ones are false positives.

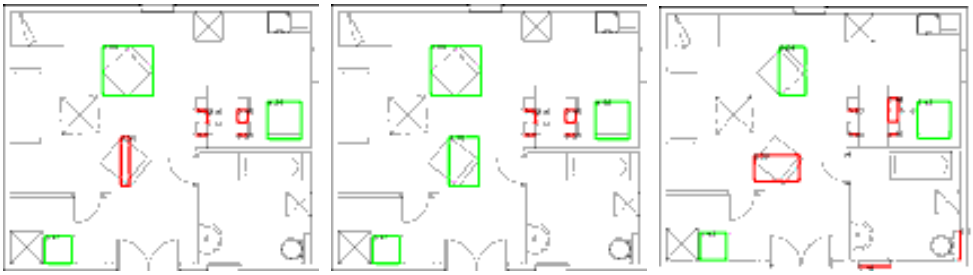


Figure 5.23: Symbol spotting: *sofa1*. Green boxes are the true positives, the red ones are false positives.

graphs. For *table2* and *table3* the optimization did not finish for many cases and the cases when the optimization finished they either have partial or wrong detection.

Figure 5.24: Symbol spotting: *sofa2*. Green boxes are the true positives, the red ones are false positives.

Figure 5.25: Symbol spotting: *window2*. Green boxes are the true positives, the red ones are false positives.

5.4 Conclusions

In this work we have proposed a product graph based subgraph matching approach. We have used the walk based similarity propagation through edges to have higher order similarities between the nodes and edges. With experiments we have proved that the higher order similarities are more robust than the pairwise similarities. Since higher order similarities add more discrimination to nodes and edges the optimization performs faster than the pairwise one. In this chapter we have also formulated a maximal common subgraph matching problem as an optimization problem and solved it with a linear programming relaxation.

Table 5.3: Comparative results with Le Bodic *et al.* [8].

	P	**R**	**F**	**AP**	**T**
Le Bodic*et al.*	65.44	58.23	59.11	57.75	27.29
Our approach	70.56	86.29	80.10	82.98	33.37

Since the optimization is solved using a LP, mostly in inexact cases (eg. symbol spotting experiment) we do not get a one-one mapping between the nodes-nodes and edges-edges. Instead we have performed a simple grouping to cluster the nodes and rank them depending on the total similarity. In future it will be interesting to add a density maximization based module that would discard the outliers and automatically clusters the inlier [101].

Chapter 6

Near Convex Region Adjacency Graph

This chapter deals with a subgraph matching problem in Region Adjacency Graph (RAG) applied to symbol spotting in graphical documents. RAG is a very important, efficient and natural way of representing graphical information with a graph but this is limited to cases where the information is well defined with perfectly delineated regions. What if the information we are interested in is not confined within well defined regions? This chapter addresses this particular problem and solves it by defining near convex grouping of oriented line segments which results in near convex regions. Pure convexity imposes hard constraints and can not handle all the cases efficiently. Hence to solve this problem we have defined a new type of convexity of regions, which allows convex regions to have concavity to some extend. We call this kind of regions as Near Convex Regions (NCRs). These NCRs are then used to create the Near Convex Region Adjacency Graph (NCRAG) and with this representation we have formulated the problem of symbol spotting in graphical documents as a subgraph matching problem. For subgraph matching we have used the Approximate Edit Distance Algorithm (AEDA) on the neighborhood string, which starts working after finding a key node in the input or target graph and iteratively identifies similar nodes of the pattern graph in the neighborhood of the key node. The experiments are performed on artificial, real and distorted datasets.

6.1 Introduction

Many of the symbol spotting methods have proposed some sort of subgraph matching as the solution, where pattern graphs represent the query symbols and the target graphs represent the graphical documents. Often this kind of methods use the Region Adjacency Graph (RAG) as the way of representing graphical information [6, 8, 54], where a region is roughly defined as a white connected component. This is well

justified since RAG allows to capture regionwise contextual information. RAG has also been widely used for classification [39] and object detection and recognition [45, 93] in other fields of computer vision. The main advantage of RAG is that it is natural and robust, and allows one to capture region wise contextual information, which encode higher order representation. But it is not always representative when the region boundaries are not clearly defined or they have some discontinuities (as in the symbol *door1* and *door2* respectively in Figure 6.1a and Figure 6.1b and in the synthetically distorted example of symbol *table1* in Figure 6.1d). So to solve these problems, in this chapter we define Near Convex Region Adjacency Graph (NCRAG) where the regions must not be clearly and continuously bounded, but can be nearly convex. This is done by near convex grouping of the oriented line segments and defining the convexity of the regions. Then we use this NCRAG representation to solve the problem of subgraph matching and apply it for symbol spotting in graphical documents. The first step of the method is to create two NCRAGs, one from a graphical document and the other from a symbol and then in the second step apply the Approximate Edit Distance Algorithm (AEDA) for subgraph matching.

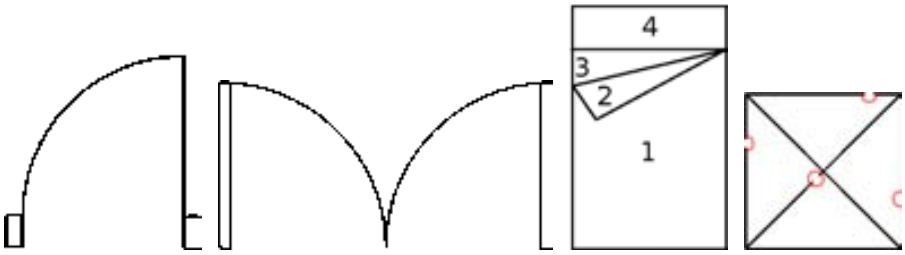


Figure 6.1: Limitations of RAG and convex region based representation: (a) the symbol *door1* contains open region, (b) the symbol *door2* also contains open regions, (c) the symbol *bed* contains a region (region 1) which is not convex, (d) the symbol *table1* contains discontinuous boundaries.

Convexity of objects is a very important property and it has been shown that most of the objects, even though they are not fully convex, can be decomposed into multiple convex parts [46]. Also it is important to note that often the object of interest is almost convex. So the property of convexity has already been studied in the field of computer vision and pattern recognition for object detection and recognition [56] and recently it has also been studied in document analysis for symbol spotting [6, 8, 67]. But, as it has been mentioned before, the object of interest might not always be perfectly convex but include some concavity in some parts (as the region 1 of the symbol *bed* in Figure 6.1c). Of course, such regions can be split into multiple strictly convex parts as it is studied in [45,46] but it is inefficient dealing with a large number of smaller purely convex parts rather than few near convex parts. Also small concavity provides discrimination in the representation of objects, so it is an important property to be considered for description. Convexity or near convex decomposition has also been studied in [79] very recently. Hence representing the graphical documents with

NCRAG seems worthwhile and useful.

The main contributions of the work in this chapter are: (1) Formulation of NCRs using the near convex grouping of a set of oriented line segments which not necessarily have to be closed and the use of these NCRs to construct the NCRAGs. This NCRAG is able to handle concavity within the convex regions and at the same time it is as expressive as RAG. (2) Application of the Approximate Edit Distance Algorithm (AEDA) [69] to solve the problem of subgraph matching for faster symbol spotting in graphical documents. The method does not need any learning or offline step and can be computed in reasonable time as shown in Table 6.1 and Table 6.2.

The rest of the chapter is organized into four sections. In Section 6.2 we explain the detailed methodology. Section 6.3 shows the experimental results. In Section 6.4 we provide a detailed discussion about limitations of this kind of representation and compare the results with a previously proposed method and note the improvements. At last in Section 6.5 we conclude the chapter and discuss future directions of work.

6.2 Methodology

The first step of the method is to create two NCRAGs, one from the target graphical document and the other from the query symbol. Formally we define an NCRAG as a graph $G(V\ E\ \alpha\ \beta)$, where V is the set of nodes and $E \subseteq V \times V$ is the set of edges of the graph G and are defined as follows:

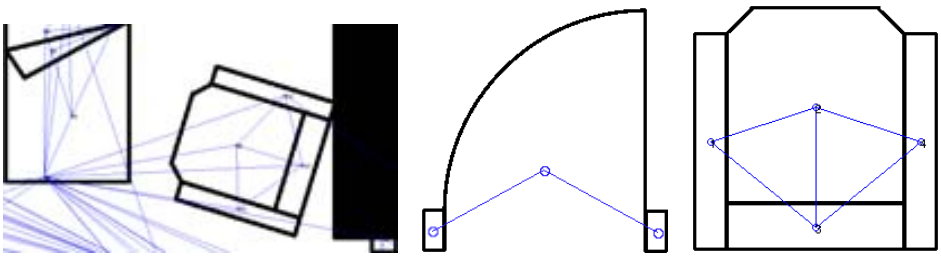
$$V = \ v_i : v_i \text{ is a convex region (nearly) in the document}$$

$$E = \ (v_i\ v_j) : v_i\ v_j \quad V \text{ and } v_i\ v_j \text{ are adjacent regions}$$

$\alpha : V \quad \mathbb{R}^n$ is the node labeling function, in this case, the Hu moments invariants concatenated with the Euler number and solidity of each of the regions. Therefore, the node label has the dimension nine and all values are normalized between 0 and 1. $\beta : E \quad \mathbb{R}$ is the edge labeling function, in this case, the ratio of the length of the common boundary to the length of the larger boundary between the two regions connecting the edge. Given two NCRAGs, the symbol spotting problem can be formulated as a subgraph matching problem, where the task is to find an instance of the smaller query symbol graph in the larger document graph. Let us denote the NCRAG of the query symbol as the pattern graph G_1 and that of the document as the input or target graph G_2. As the second step, for matching the subgraph we have used the efficient AEDA proposed by Neuhaus and Bunke in [69]. These two steps are explained in the subsequent subsections.

6.2.1 Near Convex Region Adjacency Graph (NCRAG)

This step starts working on the vectorized images which contain the approximated line segments. Here each line segment is considered as two oriented line segments where an oriented line segment is defined as a line segment where one endpoint is considered



(a) (b) (c)

Figure 6.2: NCRAG representing (a) a part of a floorplan, (b) a symbol with open region (*door1*), (c) a symbol with all closed regions (*armchair*).

as its first endpoint [46]. If l_i is an oriented line segment, then we consider $l_{i,1}$ as it s first endpoint and $l_{i,2}$ is its second. Let us consider their coordinates as $(x_{i1}\ \ y_{i1})$ and $(x_{i2}\ \ y_{i2})$ respectively. Just to clarify, if l_i and l_j are two consecutive oriented line segments coinciding end-to-end then the coordinate $(x_{i2}\ \ y_{i2})$ and $(x_{j1}\ \ y_j)$ denote the coordinates of the same point. Now let $S_n = \ l_1\ l_2 \qquad l_n\ $ be a sequence of oriented line segments and L_i be the length of the segment l_i and γ_i be the gap between $l_{i,2}$ and $l_{i+1,1}$. Then according to the original algorithm [46] we have:

$$L_{i,n} = \sum_{i=1}^{n} L_i \qquad \gamma_{i,n} = \sum_{i=1}^{n} \gamma_i \tag{6.1}$$

The saliency measurement of the convex group S_n can be defined as:

$$Saliency(S_n) = \frac{L_{i,n}}{L_{i,n} + \gamma_{i,n}} \tag{6.2}$$

The saliency parameter helps to incorporate the erroneous gaps that might be generated during binarization or vectorization as we have shown in one of our experiments in Section 6.3. Before adding any oriented line segment to a sequence, the saliency measurement of the sequence is checked. In case the saliency of the sequence is less than t_{sal} the current line segment is added to the sequence.

The convexity of the group S_n is defined as:

$$Convexity(S_n) = \frac{area(S_n)}{area(CHS_n)} \tag{6.3}$$

where CHS_n is the convex hull of S_n. Since any group S_n is not guaranteed to be closed, its area is computed as:

$$area(S_n) = \frac{\sum_{i=1}^{n}(x_{i1}y_{i2} - x_{i2}y_{i1}) + (x_{i2}y_{((i+1)\%n)2} - x_{((i+1)\%n)1}y_{i2})}{2}. \tag{6.4}$$

Like the saliency measurement, before adding any oriented line segment to a sequence S_n, its convexity together with S_n is checked and if it is less than t_{conv}, it is added to the sequence.

To make the idea clear it is to be mentioned that for efficient computation, for each oriented line segment l_i, the original algorithm precomputes the list of all other oriented line segments $List(l_i)$ with which it is mutually convex and sorts them according to the distance. Secondly, it also precomputes the angle that is turned when going from one oriented line segment to another. Since we take into account the convexity of a sequence S_n we only sort the list according to the distance and check the saliency and convexity of the current sequence together with the line segment to be added before adding it to S_n. These NCRs are then used to create NCRAG. Figure 6.2 shows some results of the NCRAG construction. Construction of the NCRAG can be done in time complexity of $\mathcal{O}(m^2 logm + m^2)$, where m is the number of oriented line segments.

6.2.2 Approximate Edit Distance Algorithm (AEDA)

The AEDA starts by finding a similar node in $G_2(V_2\ E_2\ \alpha_2\ \beta_2)$ to a node in $G_1(V_1\ E_1\ \alpha_1\ \beta_1)$. These nodes are called the key nodes [69]. The similarity of the nodes is inversely proportional to the Euclidean distance of the node labels, and the edge labels of the graph are not taken into account here. Then the algorithm looks at the neighborhood nodes considering the key nodes as the center nodes. The neighborhood nodes are then arranged in clockwise order to create a string. Here the connectivity information between the neighborhood nodes is taken into account. If any two nodes are connected the corresponding edge label is concatenated with the incident node label and form the attributed string. After having constructed the attributed string, cyclic string edit distance is applied to get the node-to-node correspondences. Then each of the nodes in each correspondence is considered as a key node and the previous steps are repeated. This algorithm continues working until it gets new correspondences. In the cyclic string the edge label is augmented with the originating node and the cost function is defined as:

$$\lambda\ \alpha_1 - \alpha_2\ + (1-\lambda)\ \beta_1 - \beta_2 \quad \text{where } 0 \leq \lambda \leq 1$$

where α and β are, respectively, the node and edge labels. For the original algorithm the readers are referred to [69].

For each node in G_1 we consider n key nodes in G_2 and perform the AEDA. Therefore for a single pattern graph G_1, we perform $n \times m$ iterations of the AEDA in G_2, where m is the number of nodes in G_1. In this case, n should be greater than the actual number of instances of the query symbol in a graphical document to get all the relevant instances. Here it is to be mentioned that greater values of n might produce more false positives but the system produces a ranked list of the relevant retrievals. So it does not hamper the performance, since the true positives suppose to appear at the beginning of the ranked list of retrieved symbols. The edge label is only used when we perform the cyclic string matching on the strings obtained from the neighborhood subgraphs by considering the nodes in clockwise order. At the end, we obtain a distance measure between a retrieved subgraph and the pattern graph by calculating the distance of the node labels. Later we use this distance to rank the retrieved subgraphs.

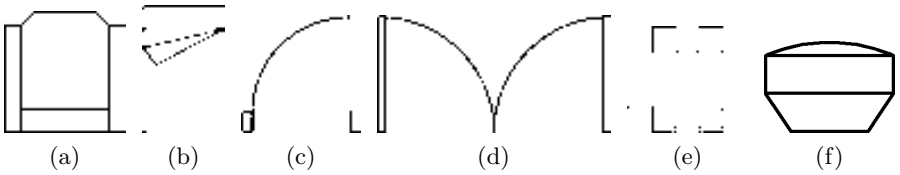


Figure 6.3: Model symbols: (a)-(e) SESYD, (f) FPLAN-POLY: (a) *armchair*, (b) *bed*, (c) *door1*, (d) *door2*, (e) *table2*, (f) *television*.

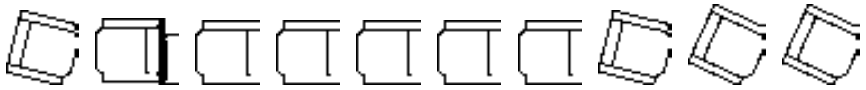


Figure 6.4: First ten retrievals of *armchair*.

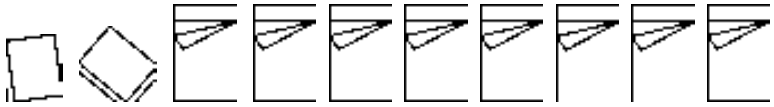


Figure 6.5: First ten retrievals of *bed*.

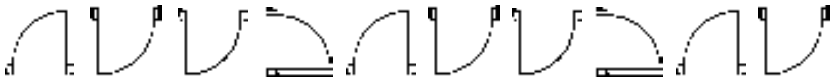


Figure 6.6: First ten retrievals of *door1*.

Figure 6.7: First ten retrievals of *television*.

6.3 Experimental results

Experiments are carried out to show (1) the robustness of the algorithm for constructing NCRAG and (2) the efficiency of the AEDA for subgraph matching in NCRAG. We have considered two different datasets: (1) SESYD (floorplans) and (2) FPLAN-POLY for experiments. For the details on these datasets one can have a look to App. A. To get the line segments, the vectorization algorithm of Qgar[1] is applied. For each of the symbols the performance of the algorithm is evaluated in terms of precision (**P**), recall (**R**) and average precision (**AveP**). To have an idea about the computation time we calculate the per document retrieval time (**T**) required for each of the symbols with each document. For each of the datasets the mean of the above mentioned metrics is shown (Table 6.1) to judge the overall performance of the algorithm. Throughout our experiments we have chosen $t_{sal} = 0\,95$, $t_{conv} = 0\,8$ and $\lambda = 0\,6$;

[1] www.qgar.com

we run a set experiments varying these parameters, then the best values for these parameters are chosen to give the best performance.

Figure 6.8: First 10 retrievals of *table2* on the database of floorplans having discontinuous line noise.

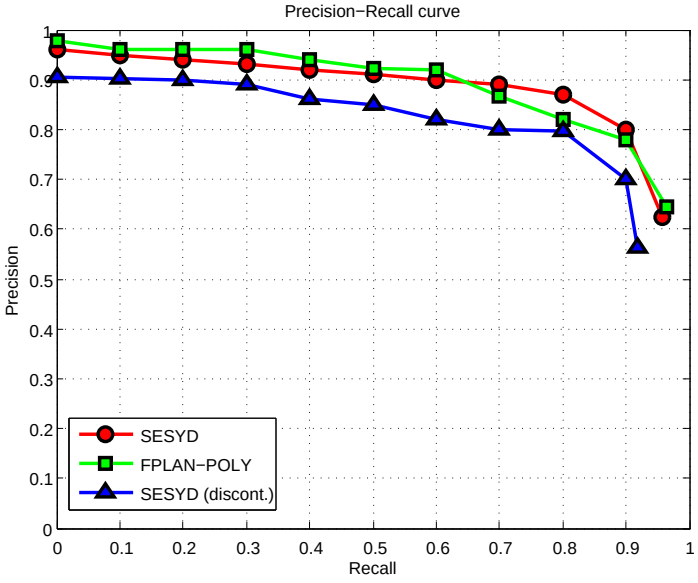


Figure 6.9: Precision recall curve for different dataset.

6.3.1 Experiments on SESYD

In this experiment we have only considered the subset called floorplans16-01. Each of the NCRAGs of each of the floorplans approximately contains 150 NCRs, whereas each of the query symbols contains 3-6 NCRs. The quantitative results are shown in the first row of Table 6.1. The high recall values for this dataset show that the algorithm works pretty well for most of the symbols. There are some cases of failure or partial detection, the reason of which will be discussed in Section 6.4. Qualitative results are shown in Figs. 6.4 to 6.6, which, respectively, include symbols with closed, near convex and open regions.

6.3.2 Experiments on FPLAN-POLY

Here we have used all the floorplans and 10 randomly chosen model symbols from this dataset. In this dataset each floorplan image contains approximately 110 NCRs, whereas a query symbol contains 4-8 NCRs. The recall value obtained in this dataset is also very good which is shown in Table 6.1. Qualitative results of querying *television* are shown in Fig. 6.7 (note the disappearance of some boundaries). The results obtained in this dataset is slightly better than SESYD. The reason is mentioned in the discussions part (Section 6.4). The parameter t_{sal} has less influence on SESYD and FPLAN-POLY dataset, since the line segments are end-to-end coinciding.

6.3.3 Experiments on SESYD-DN

This experiment is performed to prove the robustness of the algorithm constructing the NCRAG. To do that we have taken SESYD-DN dataste, the details of which are also described in App. A. After vectorizing the images from the dataset, we apply the algorithm to spot the symbol on it. The quantitative and qualitative results are respectively shown in the Table 6.1 (3^{rd} row) and Fig. 6.8 (note the white gaps on the black edges). Here the parameter t_{sal} poses an important role and to be tuned according to the existing gap in edges. The method fails when the drawn white lines remove substantial portion of a symbol. The precision recall curve (Fig. 6.9) shows the performance of the method for these three datasets, from that it is clear that the method performs worse in case of the discontinuous edges than the other two.

Table 6.1: Dataset wise mean results with NCRAG representation.

Symbol	P	R	F	AveP	T (secs)
SESYD (floorplans16-01)	62.33	95.67	74.76	70.66	0.57
FPLAN-POLY	64.56	96.32	76.21	73.68	0.65
SESYD (discont.)	56.33	91.75	70.81	64.65	0.59

6.4 Discussions

Although NCRAG is capable of capturing contextual information well in terms of regions, there are some serious limitations of NCRAG or, more generally, of the RAG based representation. One problem is shown in Fig. 6.10, where there are two symbols called *sink3* and *sink4* and the difference between them when they appear in a model symbol (Fig. 6.10(a),(c)) and in a document (Fig. 6.10(b),(d)). This is due to the difference in stroke width in images. Particularly, in the example of *sink3* when it appears in the document it looses the thin peripheral portion in the left of the region and also small circular part detaches the upper right corner part of the square. These create some difference in the regions but apparently they appear the same with our

high level vision. The dissimilarity in regions also changes the NCRAG representation. As a result it partially finds some nodes of a graph and results in partial detection or complete loss. Hence it lowers the similarity score and precision. Since in FPLAN-POLY the query symbol is generated by cropping the floorplan image, there is no discrepancy like that. This explains the slight better results in FPLAN-POLY.

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

Figure 6.10: Limitations of region based representation: (a) model symbol of *sink3*, (b) *sink3* as it appears in the document, (c) model symbol of *sink4*, (d) *sink4* as it appears in the document.

Table 6.2: Comparison between a state-of-the-art method and the current method.

Symbol	P	R	F	AveP	T (secs)
Dutta et al. [26]	41.33	82.66	51.24	52.46	0.07
Current method	62.33	95.67	74.76	70.66	0.57

As we have used the same dataset (floorplans16-01 of SESYD) as the method proposed in [26], we can do a direct comparison between the results. Table 6.2 shows the results obtained by these two methods. Clearly the NCRAG based representation improves the performance remarkably. This was expected since this kind of representation takes into account contextual information. But at the same time it has some limitations as explained above. Also the time complexity of the proposed method is quite high compared to the other one. Since [26] uses an indexation technique of the serialized subgraphical structure, the online part of the method is quite fast. But it is to be noted that the method needs an offline steps to create the indexation or hash table. The interested readers are referred to [26] for detailed comparisons of different symbol spotting methods.

6.5 Conclusions

In this chapter we have proposed a near convex grouping approach to create NCRAG on graphical documents. Then this representation has been used for spotting symbols on graphical documents with the application of the efficient AEDA. We have

shown the results of the proposed method on three different sets of images and the results obtained are quite satisfactory. We have also compared results with a previously proposed method and noticed the methodological differences and performance improvement. At the end we have shown some limitations of this kind of region based representation.

Chapter 7

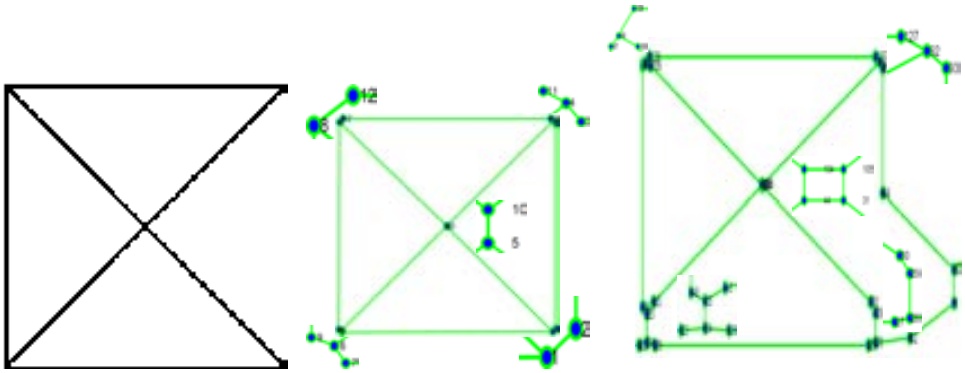
Hierarchical Graph Representation

Graph representation of graphical documents often suffer from noise viz. spurious nodes, spurious edges and their discontinuity etc. In general these errors occur during the low-level image processing viz. binarization, skeletonization, vectorization etc while transforming documents to graphs. Hierarchical graph representation is a nice and efficient way to solve this kind of problem by hierarchically merging node-node and node-edge depending on the distance. The current version of the work is an extended version of the work presented in [12]. In [12], the hierarchical graph was created by hierarchically merging different entities depending on some thresholds on the distance. This chapter introduces plausibilities to the nodes, edges of the hierarchical graphs as a function of distance between the entities and proposes a modified algorithm for matching subgraphs of the hierarchical graphs. The plausibility-annotated nodes help to improve the performance of the matching algorithm on two hierarchical structures. To show the potential of this approach, we conduct an experiment with the SESYD dataset.

7.1 Introduction

Documents often suffer from noise, as a results the representation with graphs also results in distorted graphs, for example with spurious nodes, spurious edges and discontinuity in them etc (see Figure 7.1). Hierarchical graph representation can be a way of solving this kind of structural errors hierarchically where the node-node and node-edge are merged hierarchically depending on the node-node or node-edge distance [12].

The main motivation of the work comes from [1], where the authors used the hierarchical representation of the segmented image regions and later used an approximated maximal clique finding algorithm on the association graph of the two hierarchical graphs to match the two hierarchical representations [10, 74]. In particular, the aforementioned work was applied to match two different natural images for



(a) (b) (c)

Figure 7.1: Examples of the structural distortions (spurious nodes, edges, discontinuous edges) for a graphical symbol: (a) A graphical symbol called *table1*, (b), (c) Graph representations of two different instances of the symbol *table1* when appeared in floorplans, these instances are cropped from bigger graphs representing floorplans. Graph representation of documents involves low level image processing viz. binarization, skeletonization, vectorization etc. which further add structural noise such as spurious nodes, edges etc. The example shows how even a undistorted symbol can become distorted after represented with graph (note the spurious nodes and edges near the junction and corners.).

classification.

The graph representation of documents is followed by some inter-dependent pre-processing steps viz. binarization, skeletonization, polygonal approximation etc. These low level pre-processing steps result in the vectorized documents which often contain some structural errors. In this work our graph representation considers the critical points as the nodes and the lines joining them as the edges. So often the graph representation contains spurious nodes, edges, disconnection between nodes etc (see Figure 7.1). The present version of the work is an extension of the work in [12] where we dealt with this kind of distortions in the graph level, to do that we proposed hierarchical representation of graphs because the hierarchical representation of graphs allows to incorporate the various segmentation errors hierarchically. But the node-node or node-edge merging was performed depending on a hard threshold which resulted in some loss of information. In this work we assign plausibilities to the nodes as a function of the distance and use them as a a measurement for matching.

The rest of the chapter is organized into four sections. In Section 7.2 we present the methodology of sub graph matching. Section 7.3 contains the detailed experimental results. After that, in Section 7.4, we conclude the chapter and discuss future work.

7.2 Methodology

An essential part for graph-based symbol spotting methods is the representation of the graphical objects. This representation often contains low-level vectorization errors that will affect later graph matching methods. In this section we present a hierarchical representation that overcomes these problems by covering different possible vectorizations and estimating their plausibilities.

First we will give a brief overview of the initial vectorization and some errors that can occur due to it. Afterwards we will describe our hierarchical representation and how this representation overcomes the vectorization errors.

7.2.1 Vectorization

Graph representation of documents follows some pre-processing steps, vectorization is one of them. Here vectorization can be defined as approximating the binary images to a polygonal representation. In our method we have used the Rosin-West algorithm [82] which is implemented in the Qgar package[1]. This algorithm works without any parameter except one to prune the isolated components. The algorithm produces a set of critical points and the information whether any two points are connected. Our graph representation considers the critical points as the nodes and the lines joining them as the edges.

Vectorization errors

The resulting graph can contain vectorization errors. Reasons for that can be inaccurate drawings, artefacts in the binarization or errors in the vectorization algorithm. There are different kinds of vectorization errors that can occur. Among these, we concentrated on the following ones:

Gaps In the drawing there can be small gaps between lines that ought to be connected. Reasons for that can be inaccurate drawings as well as mistakes in the binarization. The result can either be two unconnected nodes at the border of the gap or a node on one and an edge on the other side of the gap. Beside caused by errors, gaps can also be drawn intentionally to separate nearby symbols.

Split nodes On the other hand, one original node can be split into two or more nodes. This can happen, if lines in the drawing do not intersect exactly at one point. Another reason are artefacts from the skeletonization step. Nearby nodes that seem to be a split node can be the result of fine details instead of vectorization errors.

Dispensable nodes The vectorization can create nodes of order two that divide a straight edge into two or more parts. One reason for these nodes are small inaccuracies in the drawing that cause a local change in direction. For a later symbol spotting,

[1] `http://www.qgar.org/`

Figure 7.2: Three cases for simplification. Displayed are the original nodes and edges (black) and the simplified nodes and their edges (gray): (a) Merge nodes (b) Remove dispensable node (c) Merge node and edge.

these nodes are often undesired and should be removed. Nevertheless, in some cases such structures reflect details of the symbol. Examples of these errors can be seen in Figure 7.1.

Though all these errors can be corrected in a post-processing step, a simple post-processing causes other problems: often it is not clear for the system whether a situation is an error or intentional. To deal with this uncertainty, we introduce a hierarchical representation that will be described in the next section.

7.2.2 Hierarchical graph construction

This section describes the construction of hierarchical graph that is able to cover different possible vectorizations. This enables a later graph matching algorithm to deal with the uncertainties whether a part of the graph is intentionally made or caused by a vectorization error.

The basic idea of our approach is to extend a given graph G so that it contains the different possibilities of interpretation. These possibilities are connected hierarchically and assigned with a plausibility measure. The hierarchy allows us to embed the constraint just to match one interpretation into the graph matching. In Section 7.2.3 we will give further details for the graph matching, the hierarchical constraint and how to use plausibilities for the matching.

In order to create different possible vectorizations, we take the initial vectorization represented in G and simplify it step by step. For this purpose, we identify three cases that allow a simplification. These three cases will be motivated in the following. Afterwards the plausibilities are introduced and based on this a formal definition of our simplification steps is given.

Nearby nodes. Both gaps in drawing as well as split nodes result in nodes near to each other and can be solved by merging these nodes. Since nearby nodes can also be the result of correct vectorization, e.g. due to two nearby symbols, we store both versions and hierarchically connect the merged node with the basic nodes. The merged

nodes inherit all connection of its basic nodes. Figure 7.2 (a) shows an example for such a merging step.

Dispensable nodes. In case of dispensable nodes, the vectorization error can be solved be removing the node. Again, a hierarchical structure can store both versions. As described before we only consider dispensable nodes that have two neighbours. The simplified versions of these neighbours are directly connected. This is shown in Figure 7.2 (b). Applying this rule multiple times allows us to remove chains of dispensable nodes.

Nodes near to edges. The third simplification is the merging of nodes with nearby edges. In this way the second kind of gaps can be corrected. To merge a node with an edge, the edge has to be divided into two edges by a copy of the node. This can be seen for an example in Figure 7.2 (c).

Plausibility

The aim of these plausibilities is to measure the likelihood of a certain simplification to be correct. By doing so, we can prioritize matching of likely structures and still keep the ability of matching unlikely ones. To compute the plausibility for a certain simplification we identify basic features and describe the plausibility as function of these features. The features are described in the following:

Merging nodes. The plausibility for merging very near nodes is high and it decreases with increasing distance between the nodes. Thus, the distance between the nodes is taken as feature to measure the plausibility.

Removing nodes. Removing a node means to merge two edges. We consider the removal as plausible if the resulting edge is similar to the two original edges. There different features that can be used to measure this similarity. One possibility is the angle between bot edges. If the angle is near to 180^o, the resulting edge will be near to the original edges. Another measurement is the distance of the node to the resulting edge, either absolute or relative to the length of the edge. For our experiments we used the angle feature.

Merging nodes with edges. Similar to merging nodes, we take the distance of the edge to the node as feature for the plausibility.

To measure the plausibility for the three previously mentioned cases, we define three functions

1. function $\delta_1 : V \times V \quad \mathbb{R}$ to measure the plausibility for merging two nodes.
2. function $\delta_2 : V \quad \mathbb{R}$ to measure the plausibility for removing a node.
3. function $\delta_3 : V \times E \quad \mathbb{R}$ to measure the plausibility for merging a node with an edge.

For the concrete implementation we used exponential functions applied to the

features, e.g.

$$\delta_1(u\ v) = \alpha_1 \exp(-\beta_1 \cdot d(u\ v))$$

The functions δ_2 and δ_3 are defined analogous, replacing $d(u\ v)$ by the respective features.

Our approach also allows other plausibility measurements. Note that our previous work [12] without plausibilities can be seen as a special case of this work by choosing binary measurements, *i.e.*

$$\delta_1(u\ v) = \begin{array}{ll} 1 & \text{if } d(u\ v) < T_1 \\ 0 & \text{otherwise} \end{array}$$

The plausibilities are used to identify possible simplifications. For this purpose we define a threshold T_0 and only perform hierarchical simplifications for constellations that have a plausibility greater than a T_0.

Recursive definition

Based on the previous motivation now we will give a recursive definition of our hierarchical graphs that reflects the construction algorithm based on the vectorization outcome.

The result of the vectorization is an undirected graph $G(V_G\ E_G\ \alpha_G)$ where V_G is the set of nodes, $E_G \subseteq V_G \times V_G$ is the set of edges and $\alpha_G : V_G \quad \mathbb{R}^2$ is a labelling function that maps the nodes to their coordinates in the plane.

A hierarchical graph has two kinds of edges: undirected neighbourhood edges and directed hierarchical edges. Hierarchical edges represent simplification operations, *i.e.* they link nodes from the original graph arising from the vectorization to successor nodes representing simplified vectorizations. In addition, each node is assigned with a plausibility value. Formally, we define a hierarchical graph H as a five tuple $H(V\ E_N\ E_H\ \alpha\ p)$ with the neighbourhood edges $E_N \subseteq V \times V$, the hierarchical edges $E_H \subseteq V \times V$ and plausibility values $p : V \quad \mathbb{R}$.

Note that there is a difference between the plausibility of a node (given by the function p) and the plausibility of a simplification (given by δ_i, $i = 1\ 2\ 3$). The reason for this difference is, that a plausible simplification with implausible nodes results in implausible nodes.

Furthermore, given two nodes $u\ v \quad V$ let $u \quad v$ denote that v is a hierarchical successor of u and $L(u)$ denote the set of all predecessors of u that belong to G: $L(u) = \quad v \quad V_G\ v \quad u$. Based on these functions and formulations we can define the hierarchical simplification $\mathcal{H}(G) = H(V\ E_N\ E_H\ \alpha\ p)$ of G by the following rules:

Initial. As initialization for the recursion, G is a subgraph of H. We define a base plausibility $p(v) = 1$ for all initial nodes $v \quad V_G$.

Merging. For $u\ v \quad V$ with $\delta_1(u\ v) > T_0$ there is a merged node $w \quad V$ with

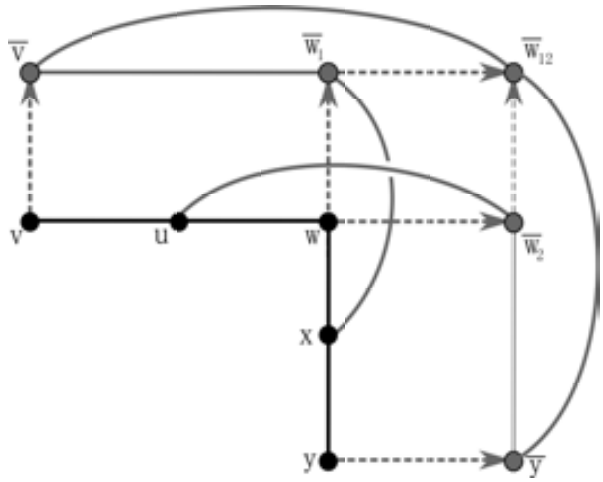


Figure 7.3: An example for removing nodes. Note that the possibility of removing two adjacent nodes of w creates four different possible interpretations of w, e.g. $\bar{w}_1$ stands for removing u but keeping x

- w is a hierarchical successor of u and v:
  $\forall s \in V : s \rightsquigarrow w \Leftrightarrow s \rightsquigarrow u \vee s \rightsquigarrow v \vee s \in \{u, v\}$
- w has all neighbours of u and v except u and v:
  $\forall s \in V : (s, w) \in E_N \Leftrightarrow ((s, u) \in E_N \vee (s, v) \in E_N) \wedge s \not\in \{u, v\}$
- w lies in the center of its leaf nodes:
  $\alpha(w) = \frac{1}{|L(w)|} \sum_{s \in L(w)} \alpha(s)$
- The plausibility of w is defined by δ_1 and the plausibilities of u and v:
  $p(w) = \delta_1(u, v)p(u)p(v)$
  If there are different ways to create w, we assign the maximal plausibility to w.

Removing. For a dispensable node $u \in V$ with $\delta_2(u) > T_0$ there exist two neighbour nodes $v, w \in V_G$, *i.e.* $(u, v), (u, w) \in E_N$. Since v and w can have hierarchical successors from other simplifications, these have to be included in the definition: for all $v_i : (v_i, u) \in E_N \wedge v \in L(v_i)$ there exists a $\bar{v}_i$. In the same way a set of $\bar{w}_j$ is defined.

- $\bar{v}_i$ hierarchical successor of v_i: $(v_i, \bar{v}_i), (w_j, \bar{w}_j) \in E_H$
- to cover all possibilities, there is neighbourhood connection between all of $\bar{v}_i$ and all $\bar{w}_j$. Furthermore, the $\bar{v}_i$ has the same connections as v_i with exception of the removed node u:
  $(s, \bar{v}_i) \in E_N \Leftrightarrow ((s, v_i) \in E_N \wedge s \neq u) \vee \exists j : s = w_j$. (analogous for w_j)
- The coordinates do not change: $\alpha(v_i) = \alpha(\bar{v}_i)$, $\alpha(w_j) = \alpha(\bar{w}_j)$
- $p(\bar{v}_i) = \delta_2(u)p(v_i)$

In this definition the successors of u and w have to be included. The reason for this can be seen in the example in Figure 7.3: if the removing is done iteratively, removing u will lead to $\bar{v}$ and $\bar{w_1}$. A subsequent removal of x has to create $\bar{w_2}$ and $\bar{w_{12}}$ in order to cover both possibilities: just remove x and remove u and x. This will give a plausibility of:

$$p(\bar{w_{12}}) = \delta_2(x)p(\bar{w_1}) = \delta_2(x)\delta_2(u)p(w)$$

Node/Edge merging. For $u \quad V\ e = (v\ w) \quad E$ with $\delta_3(u\ e) > T_0$ there exist simplifications $\bar{u}\ \bar{v}\ \bar{w}$ with

- $\bar{u}\ \bar{v}\ \bar{w}$ are hierarchically above $u\ v\ w$:
  $s \quad V : s \quad \bar{u} \quad s \quad u \quad s = u$ (analogue for v,w)
- $\bar{u}$ intersects the edge between $\bar{v}$ and $\bar{w}$:
  $s \quad V : (s\ \bar{u}) \quad E_N \quad ((s\ u) \quad E_N \quad s \quad \bar{v}\ \bar{w}$
- The coordinates do not change: $\alpha(u) = \alpha(\bar{u})$, $\alpha(v) = \alpha(\bar{v})$ and $\alpha(w) = \alpha(\bar{w})$
- $p(\bar{u}) = \delta_3(u\ e)p(u)$ (analog for v,w)

Based on these recursive rules, we construct the smallest hierarchical graph that satisfies these rules, *i.e.* no additional nodes are added. Here it is to be noted that the hierarchical simplification $\mathcal{H}(G)$ of the graph G always contains the graph G.

7.2.3 Graph matching

In this section we will describe how to make use of the hierarchical graph representation described in the previous section for subgraph matching in order to spot symbols on technical drawings. Graph matching has a long history in pattern recognition and there exist several algorithms for this problem [16]. Our approach is based on solving maximal weighted clique problem in association graphs [10]. In this section we will first give a brief overview over the graph matching algorithm. This method relies on similarities between nodes. Hence, we will present a geometric node similarity for hierarchical graphs afterwards.

Given two hierarchical graphs $H^i(V^i\ E_N^i\ E_H^i\ \alpha^i\ p^i)\ \ i = 1\ 2$, we construct the association A. Each node of A consists of a pair of nodes of H^1 and H^2, representing the matching between these nodes. Two nodes $(u_1\ u_2)$, $(v_1\ v_2) \quad H^1 \times H^2$ are connected in A, if the matching is consistent with each other. For hierarchical graphs we define the constraints for edges in A: u_i and v_i are different, not hierarchically connected and if u_1 and v_1 are neighbour, this also holds for u_2 and v_2. By forbidding the matching of hierarchically connected nodes, we force the matching algorithm to select one version of the vectorization. The first and the third constraint keep the structure between both subgraphs same.

We use replicator dynamics [10] to find the maximal weighted clique of the association graph and, hence, the best matching subgraphs of H^1 and H^2. Based on

the results of this, we perform the following steps to spot symbols. Let us consider H^1 be the pattern or model graph and H^2 be the target graph where we want to spot the instances of H^1. First of all we perform n iterations and in each iteration we perform the replicator dynamics to find the correspondences of the H^1 to H^2. Since the replicator dynamics only provide a one-to-one matching, in each iteration we obtain the correspondences from the nodes of H^1 to the nodes of H^2. So for m nodes in H^1 we get m nodes in H^2. But it is not constrained that these m nodes in H^2 will belong to the same instance of H^1. So to obtain the different instances of the H^1 we consider each of the m nodes in the H^2 and all the neighbourhood nodes of a node which can be reached within a k graph path distance. The graph path distance between two nodes is calculated as the minimum total number of nodes between the two nodes. Let us denote this set of nodes as V_s^2 and consider all the hierarchical and neighbourhood edges connecting the nodes in V_s^2 as in H^2, this forms a subgraph which we can denote as $H_s^2(V_s^2\ E_{s_N}^2\ E_{s_H}^2\ \alpha_s^2\ p_s^2)$. We again apply the replicator dynamics to get the best matching subgraph and compute the bounding box around the nodes of best correspondences. The bounding box gives the best matching region of interest expected to contain instance of a query symbol.

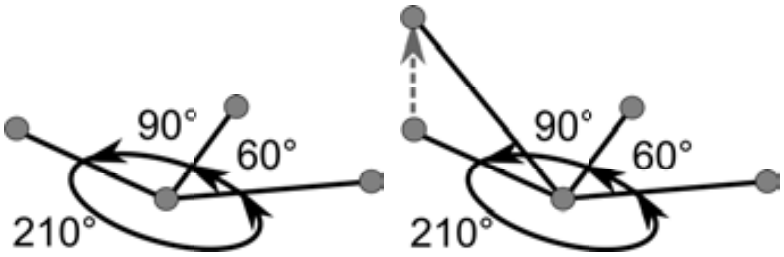
The complexity of replicator dynamics is $\mathcal{O}(\ A\ ^2)$ (see [1]). Since we perform n iterations, we get a complexity of $\mathcal{O}(n \cdot\ A\ ^2)$. In order to reduce the computation time we use the fact that the symbols are much smaller than floorplans. We create overlapping parts of the floorplan and perform the matching between the symbol and the parts. These parts have to be big enough to ensure that the symbols are completely included in at least one part. Then the construction of the hierarchical graph takes about 2 or 3 seconds for an average floorplan, the matching takes several minutes.

Node attributes

The graph matching algorithm operates on the association graph with similarity labels for the nodes. To use this algorithm, we have to define the similarity between two nodes of the hierarchical graph. Since the matching reflects geometric structures, we use geometric attributes for the similarity.

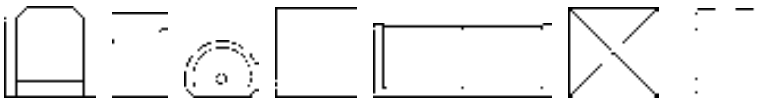
In a non-hierarchical planar graph, a single node can be labelled by the sequence of adjacent angles which sum up to 360°. Figure 7.4 (a) gives an example for such a labelling. This naive approach will cause some problems for hierarchical graph since nodes can have several hierarchically connected neighbours. Thus, the number of possible vectorizations has a strong influence on the node description. Because the number of possibilities is also affected by the level of distortion of the original image, such an approach is not robust to distortion.

To reduce the influence of the hierarchical structure and the distortion on the node labelling, we use only edges to nodes that have no predecessor connected with the central node. An example for that can be seen in Figure 7.4 (b): though the central node is connected to four nodes, only three edges are used to compute the node label.



(a) (b)

Figure 7.4: Example for node labels for graphs based on angles between edges: (a) for planar graphs and (b) for hierarchical graphs. Both will be labeled with $(90, 210, 60)$.



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

Figure 7.5: Model symbols in the SESYD dataset: (a) *armchair*, (b) *bed*, (c) *sink1*, (d) *sofa1*, (e) *sofa2*, (f) *table1*, (g) *table2*.

To compute the similarity between two node labels, we define an editing distance on these labels. The editing operations are rotating one edge, *i.e.* lowering one angle and rising another one, removing one edge, *i.e.* merging two angles, and rotating the whole description. The last operation is cost-free and makes the similarity rotation-invariant. The cost for rotating an edge is set to the angle of rotation. The cost for removing an edge is set to a fixed value.

Using this editing distance, we can define the similarity between nodes. This similarity is based on the nodes and their direct neighbourhood, but do not take into account the plausibilities of the nodes. In order to prefer matchings between plausible nodes, we multiply the similarity between two nodes with their plausibilities to get the weight for the corresponding node in the association graph.

(a) (b) (c)

Figure 7.6: Results of spotting *bed*, here the single instance of *bed* is correctly detected, note that in this case the instance is also attached with thin black pixel, (b) Results of spotting *bed* by the previous version of the method [12], (c) Results of spotting *bed* by Dutta *et al.* [26].

(a) (b) (c)

Figure 7.7: (a) Results of spotting *sofa2*, here both the instances are correctly detected among which one of them was partially attached with thick wall, (b) Results of spotting *sofa2* by the previous version of the method [12], (c) Results of spotting *sofa2* by Dutta *et al.* [26].

(a) (b) (c)

Figure 7.8: Results of spotting *table1*, note that all the instances of the symbol *table1* are correctly detected even the ones attached with the walls. In reality these walls are thin and hence less distorted during the vectorization, (b) Results of spotting *table1* by the previous version of the method [12], (c) Results of spotting *table1* by Dutta *et al.* [26].

(a) (b) (c)

Figure 7.9: Results of spotting *table1*, except one all the instances of the symbol *table1* are correctly detected. The one which is not detected is attached with the thick black pixels, (b) Results of spotting *table1* by the previous version of the method [12], (c) Results of spotting *table1* by Dutta *et al.* [26].

(a) (b) (c)

Figure 7.10: Results of spotting *table1*, note that all the instances of the symbol *table1* are correctly detected even the one which is connected with thick black pixels, (b) Results of spotting *table1* by the previous version of the method [12], (c) Results of spotting *table1* by Dutta *et al.* [26].

7.3 Experimental results

Since the work still is in progress, we have conducted a short experiment to check the performance of the algorithm. Our experiments were conducted on the images taken

(a) (b) (c)

Figure 7.11: Results of spotting *table1*, here two of the symbols are not detected and one of them are isolated but heavily distorted by the vectorization algorithm, (b) Results of spotting *table1* by the previous version of the method [12], (c) Results of spotting *table1* by Dutta *et al.* [26].

from the SESYD (floorplans) dataset (see App. A). For this short experiment we have taken the first twenty images from the subset *floorplan16-01* and three model symbols: *bed*, *sofa2* and *table1*. Some qualitative results of spotting the symbols *bed*, *sofa2* and *table1* are shown in Figure 7.8 to Figure 7.7. The results of the present method is also compared with the previous version [12] and also with a previously proposed symbol spotting method by Dutta *et al.* [26]. In Figure 7.8 to Figure 7.7, the sub figures with label (a) show the results obtained by the current method, the sub figures with label (b) show the results obtained by the previous version of the method [12] and those with label (c) show the results obtained by the Dutta *et al.* [26]. For all the cases we have only considered the same smaller subset of images and query symbols. The quantitative results are listed in the Table 7.1 where the first row shows that for current version of the method, the second row shows that for previous version of the method [12], the third row shows graph-matching with a planar graph (with the preprocessing of this paper) and the last row shows the quantitative results by the method proposed by Dutta *et al.* [26]. The present version of the method has obtained a precision of 100% and recall of 88.75%. High precision proves that all the spotted instances of the symbol is correct. The obtained recall value indicates that the system loses some of the instances and from our investigation we can say that this kind of mis-detections often occur when the symbol is attached to a thick portion of black pixel. Vectorization methods specially perform worse with black thick pixels and creates lot of distortion in the vectorized information. The previous version of the method had obtained the precision and recall respectively as 32.99% and 77.55%, clearly the present version has gained an improvement.

Table 7.1: Results obtained by the proposed method and the comparison with the previous version [12] and a previously proposed symbol spotting method [26].

Method	**P**	**R**	**F**
Current version	88.75	100.00	94.04
Previous version [12]	32.99	77.55	46.29
Without Hierarchy	54.55	90.00	67.92
Dutta *et al.* [26]	69.19	67.28	68.22

7.4 Conclusions

In this chapter we have proposed an extension of the previously proposed hierarchical graph representation. We have introduced plausibilities to the nodes of the hierarchical graph, these plausibilities help to better match the hierarchical substructures through the computation of the association graph. The present method still has some scope of improvement, as we have shown in the experimental results that all kind distortions particularly heavy distortions viz. connected to thick black walls and etc still can not be solved. So the future work will address the further improvement of the method regarding noise. With the improvement, the construction of the hierarchical graph for this kind of graph representation is becoming complex and time taking. So another direction of future work will also concern about constructing hierarchical graph for different kind of graph representation. We have investigated that the hierarchical matching algorithm used by us sometime fails to find local optima and hence the solution is erroneous. We further investigated that a little modification of the matching algorithm provides much better results. Therefore improvement of the hierarchical matching will also be considered as a future work.

Chapter 8

Experimental Evaluation

The main purpose of this chapter is to provide an overall experimental evaluation and comparison of all the proposed methodologies and some of the state-of-the-art methods proposed in the literature. Among the state-of-the-art methods we have considered those which work with graph based techniques and apply their algorithms for symbol spotting in graphical documents. To have a unified experimental framework we configure a symbol spotting system on graphical documents especially on architectural floorplans. The evaluation of a particular method is basically done depending on the capability of spotting symbols on the graphical document.

8.1 Introduction

In the previous four chapters we have provided individual experiments corresponding to the different contributions of the thesis. In this chapter we provide an integrated evaluation scheme. Experimental evaluation is an important step to judge the behaviour of algorithms. An experimental study should reveal the strengths and weaknesses of the methods under test. The analysis of these strong points and drawbacks should determine which method is the most suitable for a certain use case and predict its behaviour when using it in applications with real data. In this chapter, we experimentally evaluate the proposed and some of the state-of-the-art algorithms and provide an overall comparisons among them.

Before going to the results part, in the next section we have provided a brief description of the state-of-the-art algorithms on symbol spotting considered for the comparison. For referencing the algorithms we use some abbreviations of the names which are listed in Table 8.1.

The rest of the chapter is divided into four sections, where Section 8.2 is dedicated for describing in brief the state-of-the-art methods that we have considered for experimental evaluation and comparison. The experimentation is described in Section 8.3.

Table 8.1: Summary, abbreviations of the methods

Abbreviation	Method
SG	Symbol spotting by hashing of serialized subgraphs (Chapter 4).
PG	Product graph based subgraph matching (Chapter 5).
NCRAG	Near convex region adjacency graph (Chapter 6).
HGR	Hierarchical graph representation (Chapter 7).
SSGR	Symbol spotting using graph representations [77]
FGE	Subgraph spotting through fuzzy graph embedding [60].
ILPIso	Integer linear program for substitution tolerant subgraph isomorphism [8].

Section 8.4 contains discussions on the obtained results by various methods. After that in Section 8.5 the chapter is concluded.

8.2 Description of state-of-the-art methods

This section contains brief descriptions of the three state-of-the-art methods:

Symbol spotting using graph representations [77]: The algorithm was proposed by Qureshi *et al.* [77]. The proposed strategy has two main steps. In the first step, a graph based representation of a document image is generated that includes selection of description primitives (nodes of the graph) and relation of these features (edges) (see Figure 8.1). In the second step the graph is used to spot interesting parts of the image that potentially correspond to symbols. The sub-graphs associated to selected zones are then submitted to a graph matching algorithm in order to take the final decision and to recognize the class of the symbol. The experimental results obtained on different types of documents demonstrates that the system can handle different types of images without any modification.

Subgraph spotting through fuzzy graph embedding [60]: Here the author present a method for spotting subgraph in a graph repository. Their proposed method accomplishes subgraph spotting through graph embedding. They have done an automatic indexation of a graph repository during an off-line learning phase; where they (i) split the graphs into 2-node subgraphs, which are primitive building-blocks of a graph, (ii) embed the 2-node subgraphs into feature vectors by employing proposed explicit graph embedding technique, (iii) cluster the feature vectors in classes by employing a classic agglomerative clustering technique, (iv) build an index for the graph repository and (v) learn a Bayesian network classifier. The subgraph spotting is achieved during the on-line querying phase; where they (i) further split the query graph into 2-node subgraphs, (ii) embed them into feature vectors, (iii) employ the Bayesian network classifier for classifying the query 2-node subgraphs and (iv) re-

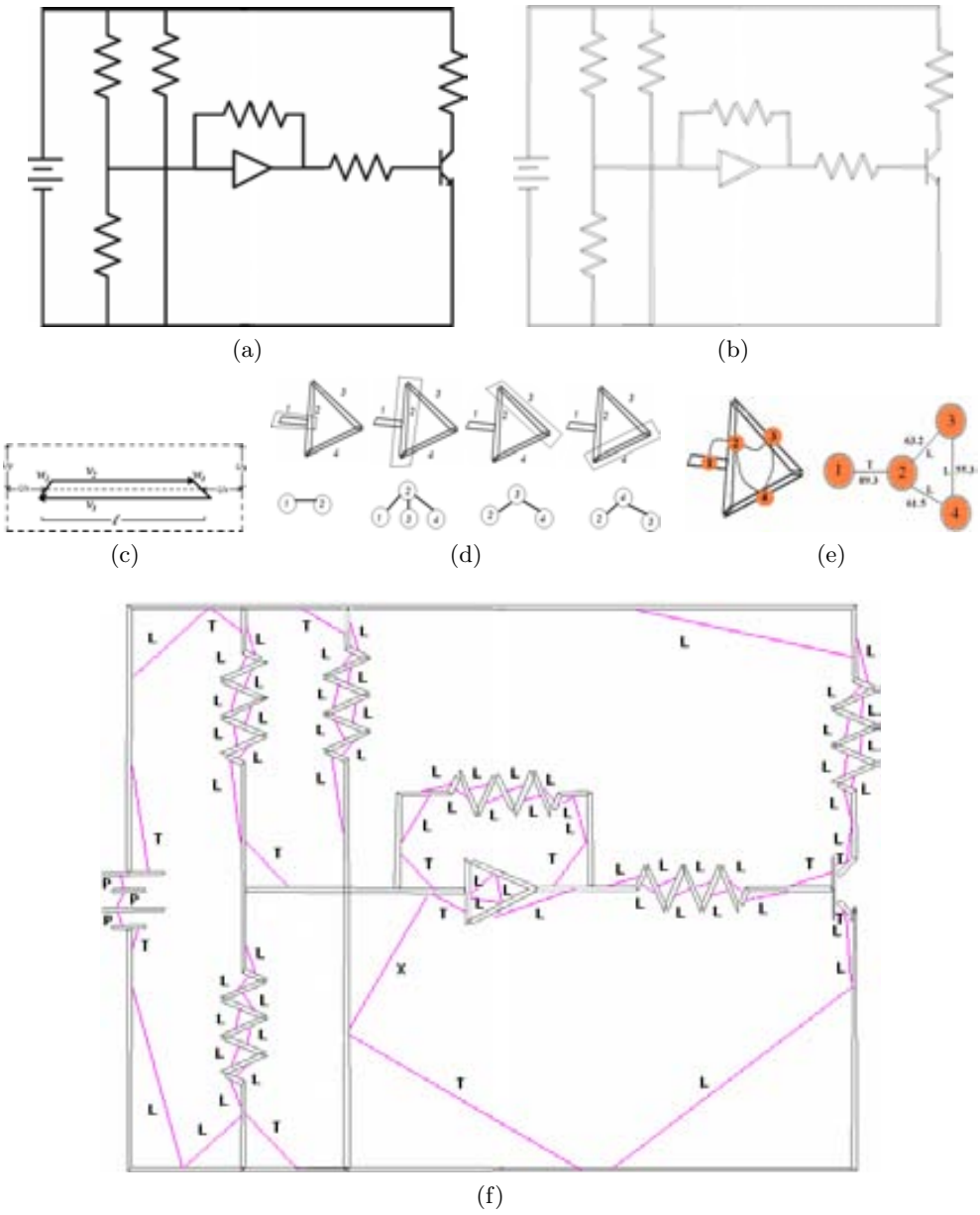


Figure 8.1: (a) Initial image, (b) Vectorization results, (c) Zone of influence of a quadrilateral, (d) Influence zone of the quadrilaterals and their corresponding sub-graphs respectively, (e) and (f) Graph representation. (Figure credit: Qureshi *et al.* [77]).

trieve the respective graphs by looking-up in the index of the graph repository. The graphs containing all query 2-node subgraphs form the set of result graphs for the query. Finally, they employ the adjacency matrix of each resultant graph along with a score function, for spotting the query graph in it. The proposed subgraph spotting method is equally applicable to a wide range of domains; offering ease of query by example (QBE) and granularity of focused retrieval.

Integer linear program for substitution tolerant subgraph isomorphism [8]: This paper tackles the problem of substitution-tolerant subgraph isomorphism which is a specific class of error-tolerant isomorphism. This problem aims at finding a subgraph isomorphism of a pattern graph S in a target graph G. This isomorphism only considers label substitutions and forbids vertex and edge insertion in G. This kind of subgraph isomorphism is often needed in pattern recognition problems when graphs are attributed with real values and no exact matching can be found between attributes due to noise.

The proposal to solve the problem of substitution-tolerant subgraph isomorphism relies on its formulation in the Integer Linear Program (ILP) formalism. Using a general ILP solver, the approach is able to find, if one exists, a mapping of a pattern graph in to a target graph such that the topology of the searched graph is kept and the editing operations between the label shave a minimal cost. The proposed subgraph matching has been applied for spotting symbols in graphical documents where document and symbol images are represented by vector-attributed Region Adjacency Graphs built from a segmentation process.

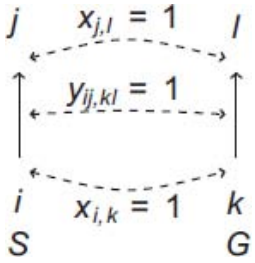


Figure 8.2: An example of matching. $\mathcal{S}$ and $\mathcal{G}$ both contain a single edge, respectively ij and kl. The following solution is represented on this figure: $x_{i,k} = 1$ (resp. $x_{j,l} = 1$, $y_{ij,kl} = 1$), *i.e.* i (resp. j, ij) is matched with k (resp. l, kl). Conversely, since i (resp. j) is not matched with l (resp. k), $x_{i,l} = 0$ (resp. $x_{j,k} = 0$). (Figure credit: Le Bodice *et al.* [8]).

8.3 Experimental results

For the experiments we choose two different subsets from the SESYD dataset, viz. *floorplans16-05* and *floorplans16-06*. One can find a details on SESYD in Section

A.1. These two particular subsets have been chosen keeping in mind the execution ability of different methods. This is because some of the graph based methods use special kind of vectorization algorithms which can not handle all kind of graphical structures such as thick walls etc.

Since in each of the individual chapters a detailed experimentations have already been documented for each of the proposed methods with different parameter settings, in this chapter we only mention the best results obtained by a particular method (by a particular parameter settings). This implies the best results from each of the method is considered for the experimental comparison. The results obtained by the methods SSGR [77] and FGE [60] are directly taken from the paper [60], where the authors had performed a comparison. And for the ILPIso method proposed by Le Bodic *et al.*, the implementation was downloaded from the project web page[1].

For all the methods proposed in this thesis and the ILPIso [8], a particular retrieved symbol is considered as true positive if it has a 50% overlapping with its corresponding ground truth. This signifies that if R be the bounding box of a retrieved symbol and T be the bounding box of the corresponding ground truth then the retrieved symbol is considered true positive if and only if $\frac{R \cap T}{R \cup T} >= 0.5$.

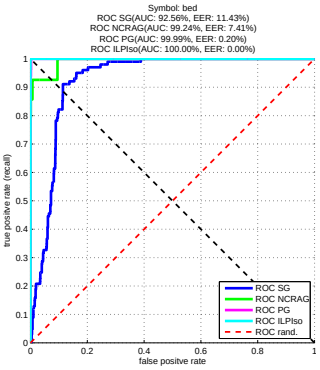
Three of the proposed methods viz. SG, PG, NCRAG and ILPIso are designed to provide a ranked list of retrievals. The HGR method does not provide a ranked list in this way. This is because this method works with a node attributes which take into account the angles and does not reflect similarity of shapes. To have an idea how a particular method can rank the true positives with respect to the false positives, we draw the receiver operating characteristic (ROC) curves obtained from the ranked list of retrievals of different symbols (see Figure 8.3). The ROC curves usually reveal how well a method can rank the true positives before the false positives, these curves are basically a test of the (dis)similarity functions designed inside each of the algorithms.

The quantitative results obtained by different methods are listed in the Table 8.2. We have mainly used the traditional measurements for a retrieval system such as precision (**P**), recall (**R**), F-measure (**F**) and average precision (**AveP**). The details of these measurements for symbol spotting application can be found in [86]. For having an idea on the time complexity of each of the methods we have also provided a mean time measurement for spotting instances of a query symbol in a single target document. Here it is to be mentioned that the mentioned time duration only includes the time taken in the online phase. The time taken for the necessary feature extraction, preprocessing, construction of graphs etc is not considered in this study as they are done in offline phase. One can consider the F-measure value for a very high level overview of the performance of the methods. But we believe that deciding the winner or looser is not the only aim of an experimental study. There are separate advantages and disadvantages of each methods.

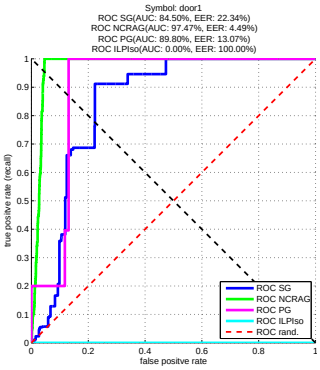
[1] `http://litis-ilpiso.univ-rouen.fr/ILPIso`

8.4 Discussions

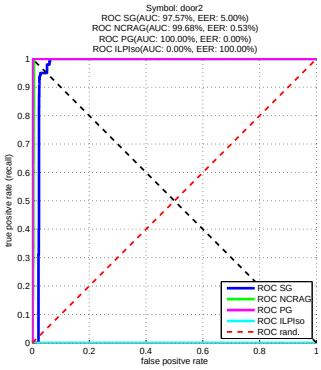
The SG method performs quite well for symbols with complex structures such as *bed*, *door2*, *sink1*, *sink2*, *sink4*, *table2*, *table3* etc. This is quite justified since the complex



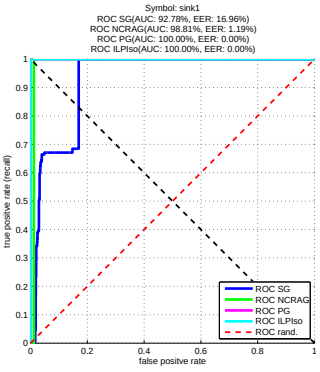
(a) *bed*



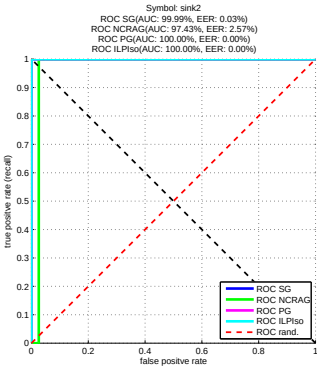
(b) *door1*



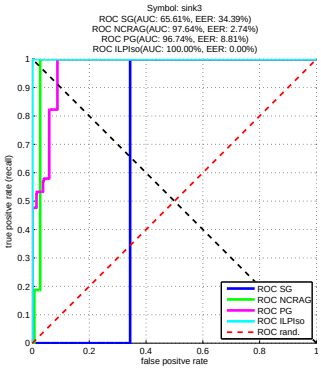
(c) *door2*



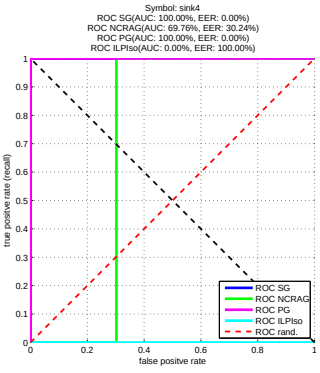
(d) *sink1*



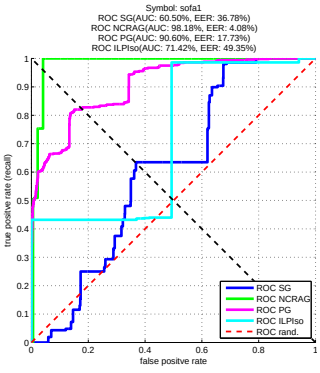
(e) *sink2*



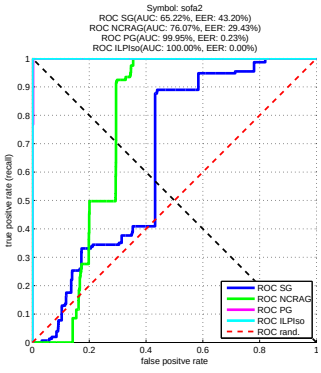
(f) *sink3*



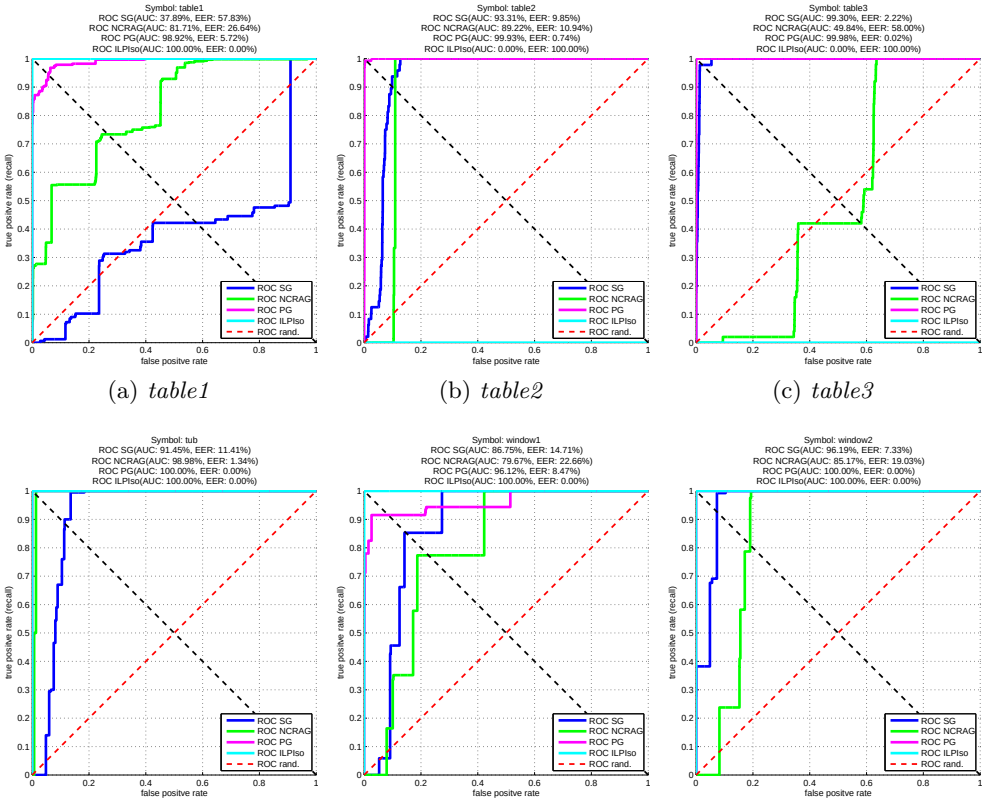
(g) *sink4*



(h) *sofa1*



(i) *sofa2*



(a) *table1* (b) *table2* (c) *table3*

(d) *tub* (e) *window1* (f) *window2*

Figure 8.3: Receiver operating characteristic (ROC) curves for different pattern graphs obtained by the method based on hashing of serialized graphs.

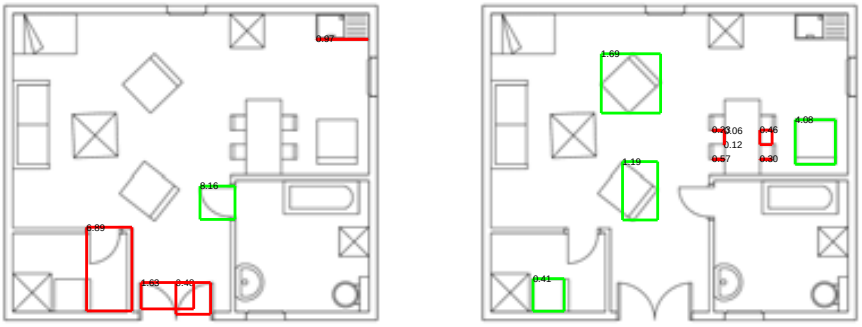
Table 8.2: Results

Methods	P	R	F	AveP	T
SG (Chapter 4)	54.19	83.98	65.87	65.43	**0.07s**
PG (Chapter 5)	**70.56**	86.29	**80.10**	**82.98**	33.37s
NCRAG (Chapter 6)	61.89	82.87	70.85	70.65	0.72s
HGR (Chapter 7)	30.11	33.76	31.83	-	48m24s
SSGR [77]	41.00	80.00	54.21	64.45	-
FGE [60]	56.00	**100.00**	71.79	75.50	-
ILPIso [8]	65.44	58.23	59.11	57.75	27.29s

parts provide enough discrimination to a particular class. As we will observe in the next part, that this phenomenon is quite common for the other methods too. The

SG method returns false positives while the query symbol contains a subpart of the other symbol. For example, it retrieves false *door1* because *door1* also occurs inside *door2* and it detects part of *door2*. For the same reason it retrieves false *sink3* as it belongs as a subpart in *sink2*. This problem is also mentioned in Chapter 4 and this is because the paths are indexed independently and there is no higher level organization of the serialized structures. This method also performs worse for very simple and frequently occurring structure such as *sofa1*, *sofa2*, *table1*, *tub* and *window1*. One of the advantages of this method is the execution time in the online phase which is quite less and can be considered as a benefit of the indexation technique. However, the indexation phase in the offline stage usually takes nearly two hours to create the hash table for this particular dataset.

The overall results obtained by the PG method is quite good. It has achieved the highest precision, F-measure and average precision values. A problem of this method results in from the computation of the bounding box to decide the position of the occurrence of an instance which is presently done by grouping the adjacent dual nodes. This way occurrence of a false dual node often creates bigger bounding box, as shown in Figure 8.4a (the red bounding box) while spotting *door1*. In our performance evaluation bigger bounding boxes are classified as false positives, this explains the bad results for *door1*. This method also performs worse in case of *sofa1* but this is due to the occurrence of similar structure in different parts other than the actual occurrences as shown in Figure 8.4b.



(a) *door1*: erroneous results obtained by PG method

(b) *sofa1*: erroneous results obtained by PG method

Figure 8.4: Erroneous results.

The results obtained by the NCRAG method is also good. Even the method worked very well for the difficult symbol *sofa1*. It is observed that this method is bit sensitive to the selection of the key regions. A region, which is adjacent to the most of the other regions in a symbol, can be a good candidate for a key region. Otherwise, a wrong expansion often results in with a lower cost. This problem is observed for the symbol *sink4* and *table3*. A similar problem also occurs for symmetric symbol such as *table1* where a finding a discriminating region is difficult. The issues concerning

the variation of the regions of the pattern and target graphs have been reported in Chapter 6, have not been observed in this dataset.

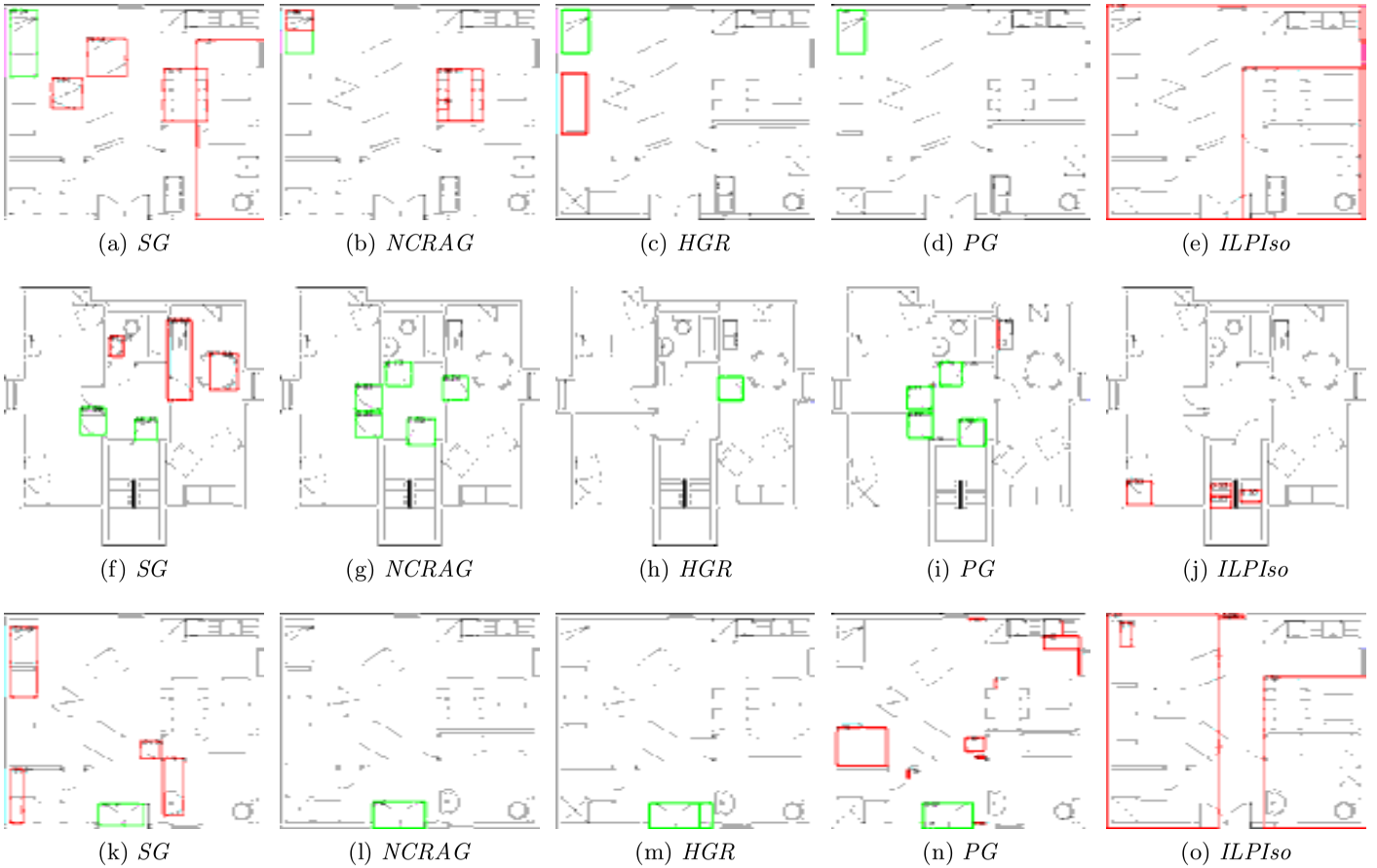
The HGR method worked only with six query symbols and they are *bed*, *door1*, *door2*, *sofa1*, *sofa2* and *table1*. The reason of failure for the other symbols might be due to the node attributes which are not stable in many scenario and also getting robust node attributes for this kind of unlabelled graphs is not easy. But for the successful symbols the method works quite good, the false retrievals are substantially less for this method. We can not provide a ranked list of retrieved symbols because in this case obtaining a similarity value for each of the retrievals is not straight forward for the nature of the node attributes.

We are not able to comment on the detailed results obtained by the method SSGR and FGE as the results are taken from the paper for quantitative comparisons. The ILPIso method proposed by Le Bodic *et al.*performed quite well with most of the scenario, as it obtained 100% F-measure for seven pattern graphs. As the other methods there is a usual problem with the occurrence of the same symbol as part of the other such as *sofa1* occurs in *table2*. Apart from that, the method can not provide any true retrievals for *door1*, *door2* etc. This is because of the kind graph representation used by the method, as region adjacency graph can not provide robust representation for these two symbols due to the existence of an open region. The method does not find any true instances for *sink4*, *table3*, may be this is because of the discrepancy of the regions in pattern and target graphs as mentioned for NCRAG method. The method does not finish the search procedure with *table2* and the search procedure has to be aborted manually after 60 minutes.

We have also provided some of the qualitative results for all the five methods, which are shown from Figure 8.5 to Figure 8.9.

8.5 Conclusions

In this chapter we have provided an overall experimental evaluation of all the proposed methods and some of the state-of-the-art methods. We have tried to figure out the advantages and disadvantages of different methods. The discussions on different methods can reveal in which scenario which kind of methodology fits better. There is not any single method which can resolve all the problems. This fact indicates the need of certain future work for different methodologies, at the same time, it points out a direction to investigate on combining more than one symbol spotting systems.



(a) *SG* (b) *NCRAG* (c) *HGR* (d) *PG* (e) *ILPIso*

(f) *SG* (g) *NCRAG* (h) *HGR* (i) *PG* (j) *ILPIso*

(k) *SG* (l) *NCRAG* (m) *HGR* (n) *PG* (o) *ILPIso*

Figure 8.5: Qualitative results: (a)-(e) *bed*, (f)-(j) *door1* and (k)-(o) *door2*.

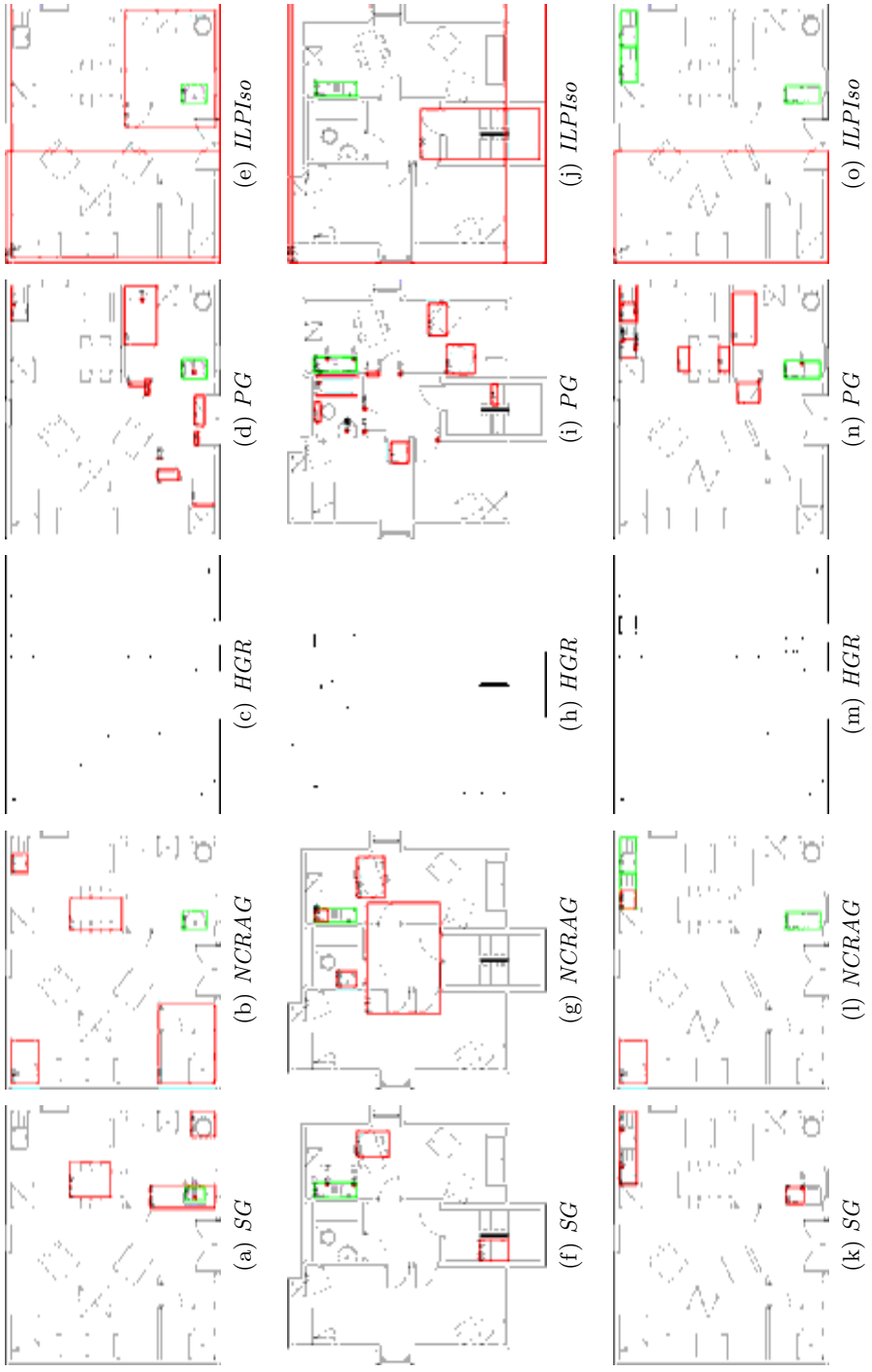
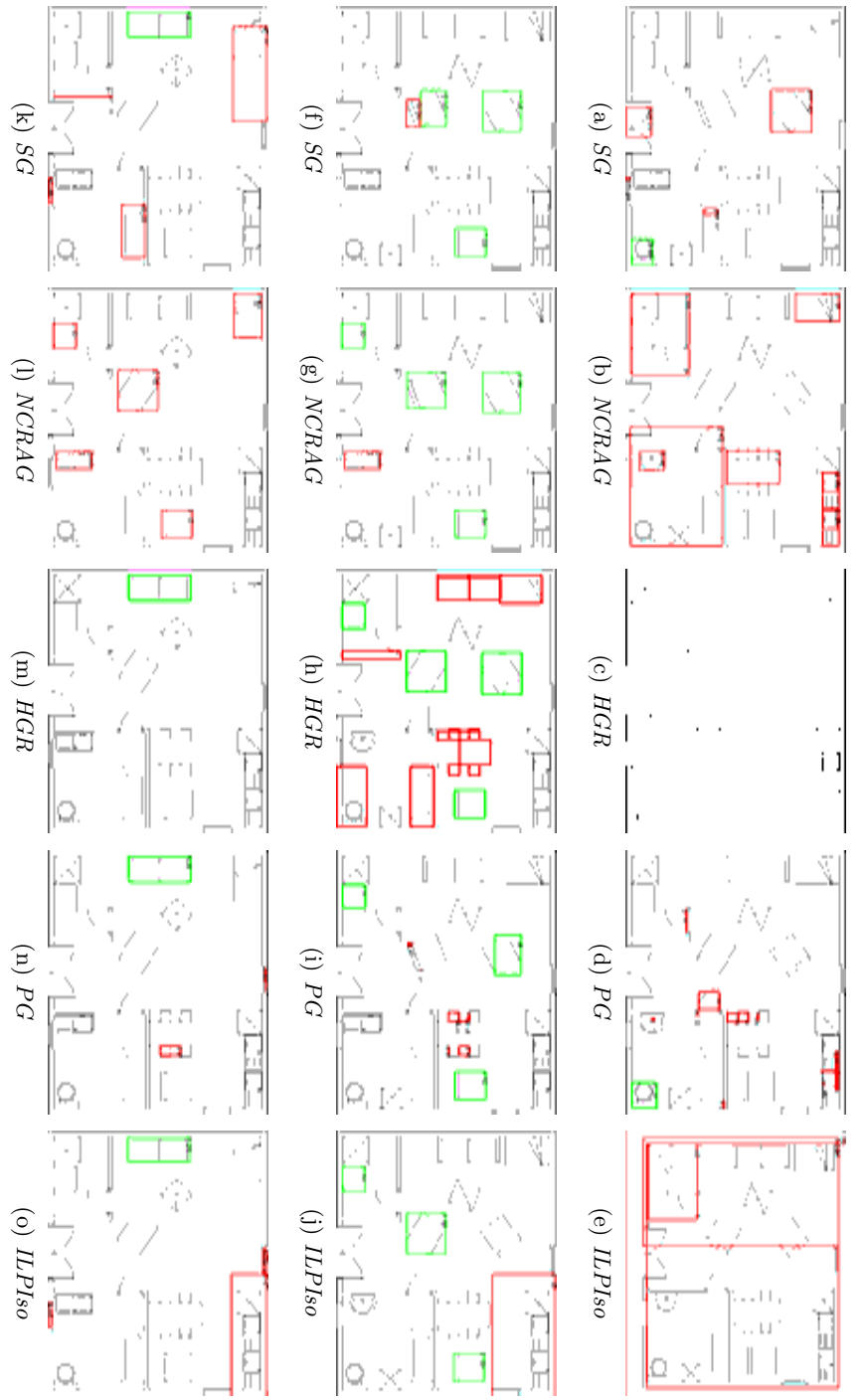


Figure 8.6: Qualitative results: (a)-(e) *sink1*, (f)-(j) *sink2* and (k)-(o) *sink3*.



(a) *SG* (b) *NCRAG* (c) *HGR* (d) *PG* (e) *ILPIso*

(f) *SG* (g) *NCRAG* (h) *HGR* (i) *PG* (j) *ILPIso*

(k) *SG* (l) *NCRAG* (m) *HGR* (n) *PG* (o) *ILPIso*

Figure 8.7: Qualitative results: (a)-(e) *sink4*, (f)-(j) *sofa1* and (k)-(o) *sofa2*.

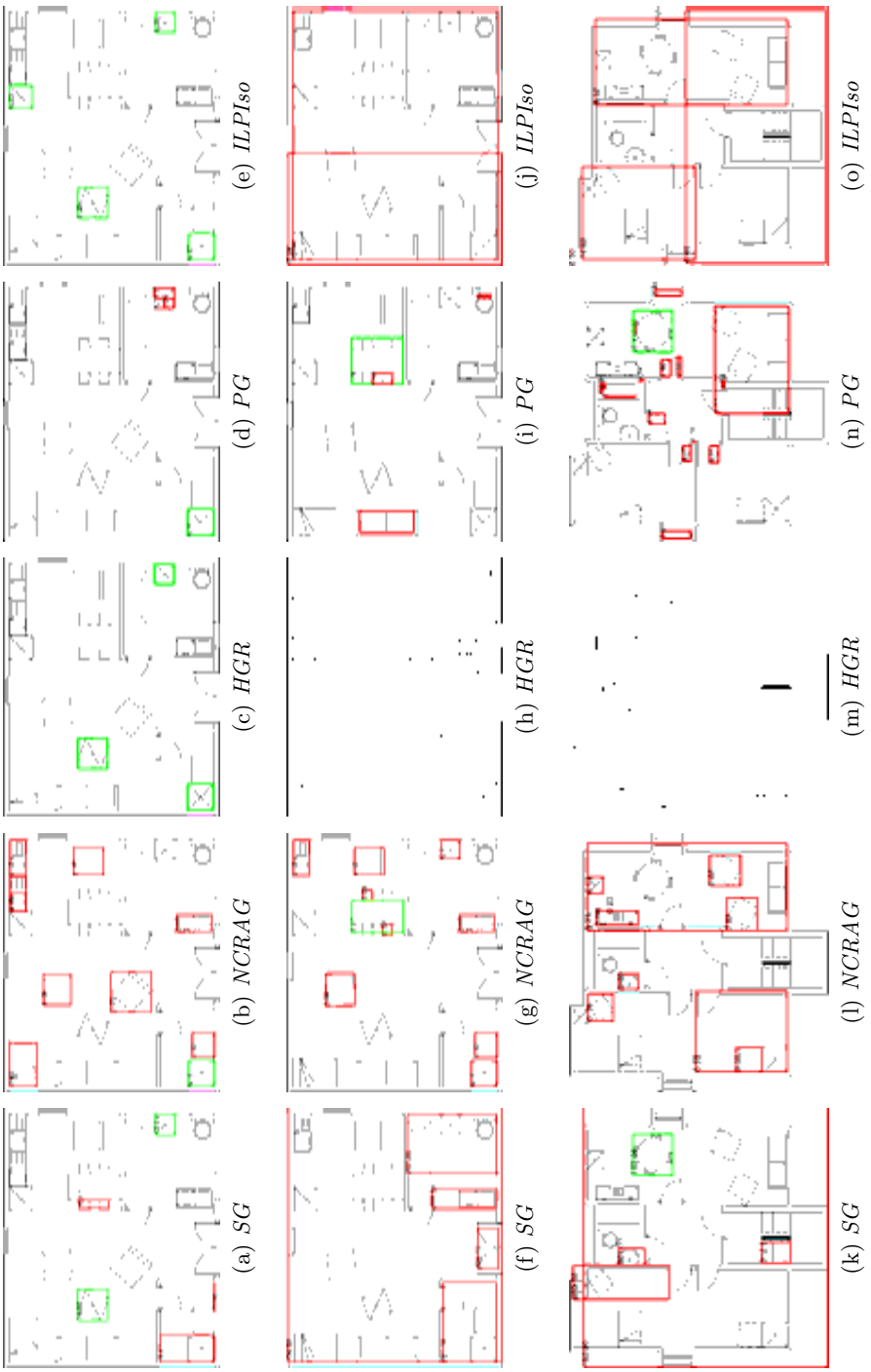
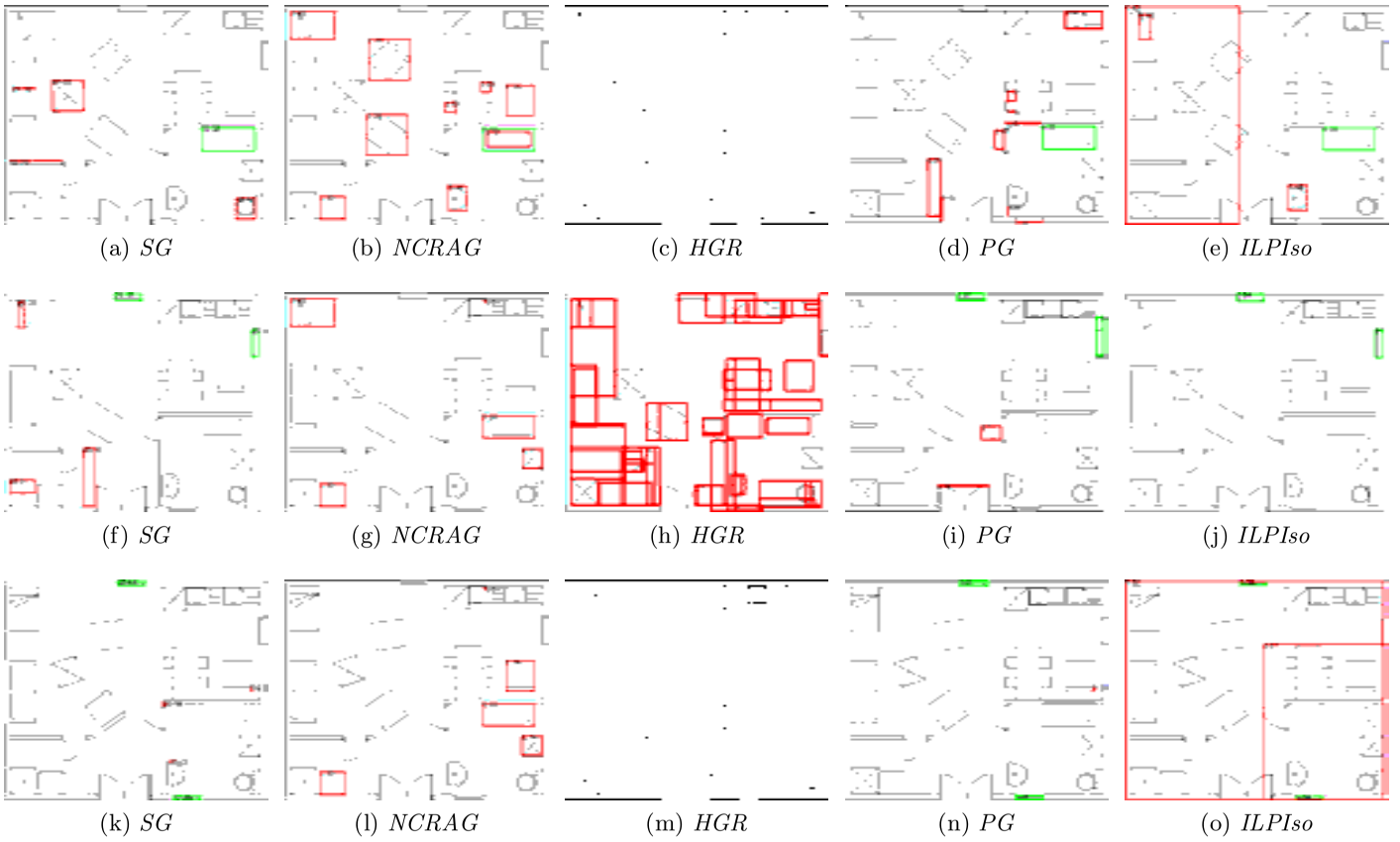


Figure 8.8: Qualitative results: (a)-(e) *table1*, (f)-(j) *table2* and (k)-(o) *table3*.



(a) *SG* (b) *NCRAG* (c) *HGR* (d) *PG* (e) *ILPIso*

(f) *SG* (g) *NCRAG* (h) *HGR* (i) *PG* (j) *ILPIso*

(k) *SG* (l) *NCRAG* (m) *HGR* (n) *PG* (o) *ILPIso*

Figure 8.9: Qualitative results: (a)-(e) *tub*, (f)-(j) *window1* and (k)-(o) *window2*.

Chapter 9

Conclusions

Throughout the dissertation several methods for subgraph matching applied to symbol spotting have been presented. This chapter summarizes each main chapter by revisiting their contributions, strengths and weaknesses. Finally, a brief overview of the future research possibilities in the area of subgraph matching and symbol spotting is discussed.

9.1 Summary and contributions

In this thesis work we have presented four different methods for symbol spotting on graphical documents represented as graphs. Chapter 1 has introduced the main idea of structural pattern recognition, graphs as a tool of structural pattern recognition and the problem of symbol spotting.

Chapter 2 has introduced some useful definitions, concepts of graph matching and a brief state-of-the-art methods are discussed. These were necessary since all our symbol spotting methods are based on graph representation and use graph based methodologies to solve the problem.

In Chapter 3, an overview of the state-of-the-art methods have been presented. Here we have divided the main methods of symbol spotting into five different categories. Literature review in each of these categories have been presented along with the advantages, disadvantages in different scenarios.

We have introduced the first approach to symbol spotting by hashing serialized graphs in Chapter 4. The major contribution in this work is to serialize the planar graphs and form one dimensional graph paths. Graph paths are used to index a given database. The main motivation of this work came from the idea of graph indexing, which is a popular approach for applications dealing with large number of graphs. We model the structure of a path by off-the-shelf shape descriptor and we have used locality sensitive hashing for indexing those individual paths.

Chapter 5 has presented a subgraph matching technique based on product graph and it has been used for spotting symbols on graphical documents. The main contribution was the introduction of higher order contextual similarities which are obtained by propagating the pairwise similarities. The next contribution of this work was to formulate subgraph graph matching as a node, edge selection problem of the product graph. For that we constructed a constrained optimization problem whose objective function was created from the higher order contextual similarities.

In the Chapter 6, we have introduced near convex region adjacency graph. The main contribution was the introduction of near convex regions and forming a graph representation based on that. There are certain drawbacks of region adjacency graph (RAG), for example, the information that are not bounded in regions can not be represented by RAG. This contribution solves the limitation.

Chapter 7 has presented a hierarchical graph representation of line drawing graphical documents. Certain line drawings often suffer from structural errors or distortions. Hierarchical graph representation is a way to solve those structural errors in hierarchical simplification.

Finally in Chapter 8, we have provided an experimental evaluation of all the proposed methods and some state-of-the-art methods and advantages and disadvantages have been pointed out.

In general, in this thesis we have proposed several inexact subgraph matching algorithms which intend to solve the problem of inexact subgraph matching in a better approximated way. Different graph representations have been used and the purpose of them was to tolerate the structural noise and distortions and bring stability in the representation. Also a hierarchical graph representation is proposed to simplify/-correct the structural errors in step-by-step manner. Detailed experimentations were performed to evaluate each of the methods and for comparison with some state-of-the-art algorithms and for that some model of datasets also have been proposed.

9.2 Future Perspective

There are ideas that have come out from different contributions of this thesis but could not be evaluated due to the time constraints. The future perspective of this thesis can be listed as follows:

- Since indexation of the serialized substructures was successful as shown in this thesis (Chapter 4), a very good continuation of this work can be done by factorizing the graphs into two dimensional structures and making hash/indexation structure of them. Here also we can use some kind of graph embedding to transfer those factorized subgraphs into a vector space and do the same procedure as the Chapter 4.

- In Chapter 5, we have proposed higher order contextual similarities by propagating the pairwise similarities through random walk. This is an idea that

crept in from random walk graph kernel. There are other kernel methods such as graphlet kernel, shortest path kernel etc. that measure the graph similarities by counting different common substructures such as graphlet, short path etc. It would be interesting to bring these ideas for the purpose of having contextual similarities and then apply the optimization technique as used in Chapter 5.

- Chapter 7 described a hierarchical graph representation which corrects the structural errors appeared in the base graph. In this particular work we consider a graph representation where we consider the critical points as the nodes and the lines joining them as the edges. As a consequence, the hierarchical graph representation resulted in become huge with lot of nodes/edges, these make problem while matching these hierarchical structures. The hierarchical graph representation is a very good idea for correcting the errors/distortions. It would be interesting to consider a different graph representation which considers more higher order entities as nodes/edges and then take into account the hierarchical graph representation of that. This could make the matching module faster and easier.

- In this thesis we have proposed different graph representations aimed to tolerate structural noise and distortions etc. At the same time there are different graph matching methods have been proposed. It would be interesting to evaluate each of the graph representations for each of the graph matching techniques. This needs the adaptation of the data structures to our matching methodologies, which demands some work.

- As it has been seen in the experimental evaluation chapter (Chapter 8) that some of the methods work very well on some specific class of pattern graph. This phenomenon evokes the idea of combining different methods to produce a unified system that can provide the best results depending on the majority voting. This of course needs some more research and also some engineering work which would need some more time.

Appendix A

Datasets

Through out this thesis work we have used several datasets, sometime they consist of floorplans with different variation, sometime isolated graphical objects or historical handwritten documents. Some of the datasets are generated by us to perform some specific experimentations. In this chapter we give a description on all of them. And also when they are created by us we explain a brief methodology, motivation for the creation. The rest of this chapter is divided into nine sections, each of which is dedicated for each of the datasets.

A.1 SESYD (floorplans)

This dataset is a collection of synthetically generated floorplans and isolated graphic symbols [22][1]. It contains 16 isolated symbols which can be seen in Figure A.1. Apart from that it has 10 different subsets of floorplans, each of which contains 100 floorplan images. All the floorplans in a subset are created upon a same floorplan template by putting different isolated symbols in different feasible places with random orientations and scales. From Figure A.2 to Figure A.5, we have shown some example images from this dataset. This dataset also contains ground truths, where it indicates the position of the bounding boxes of each of the isolated symbols in the corresponding floorplans.

A.2 FPLAN-POLY

This dataset is a collection of real floorplans and isolated symbols [86][2]. It contains 42 floorplan images which are basically parts of bigger real floorplans. Being the parts of real floorplans, the images contain distortions, text-graphic interference, etc.

[1] http://mathieu.delalandre.free.fr/projects/sesyd

[2] http://www.cvc.uab.es/~marcal/FPLAN-POLY

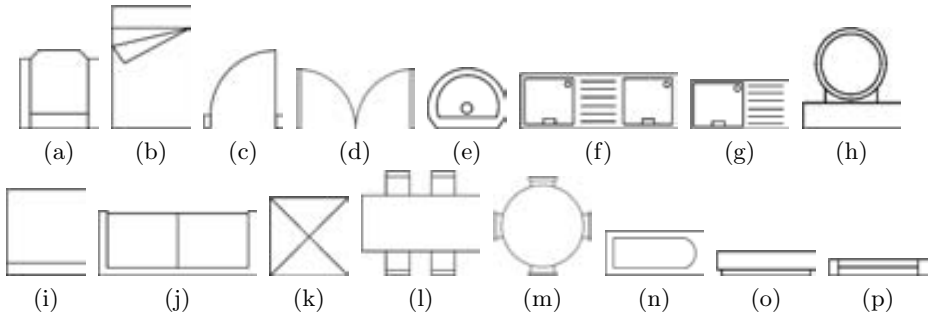


Figure A.1: Example of different isolated symbols: (a) *armchair*, (b) *bed*, (c) *door1*, (d) *door2*, (e) *sink1*, (f) *sink2*, (g) *sink3*, (h) *sink4*, (i) *sofa1*, (j) *sofa2*, (k) *table1*, (l) *table2*, (m) *table3*, (n) *tub*, (o) *window1*, (p) *window2*.

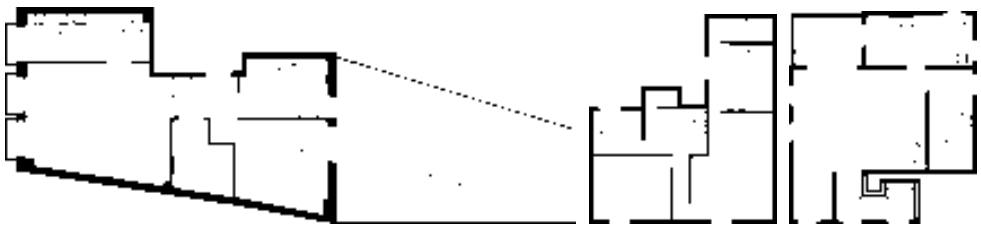


Figure A.2: Example of floorplans from SESYD (a) floorplans16-01 (b) floorplans16-02 and (c) floorplans16-03 subset.

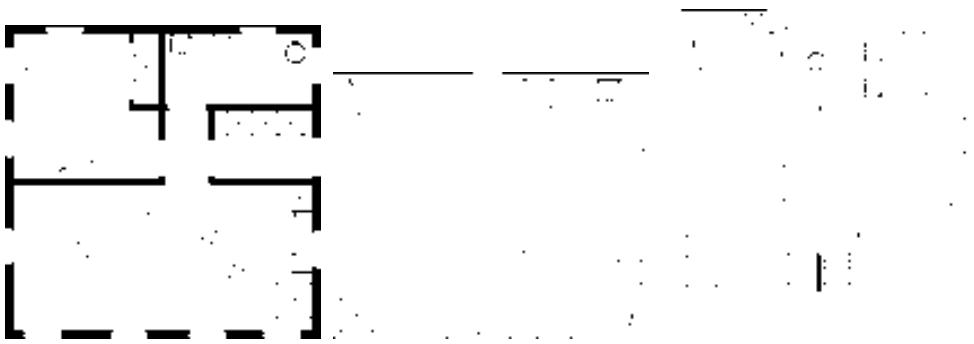


Figure A.3: Example of floorplans from SESYD (a) floorplans16-04 (b) floorplans16-05 and (c) floorplans16-06 subset.

This dataset basically contains the vectorized images, where all the images of this database have been converted by using a raster-to-vector algorithm implemented in

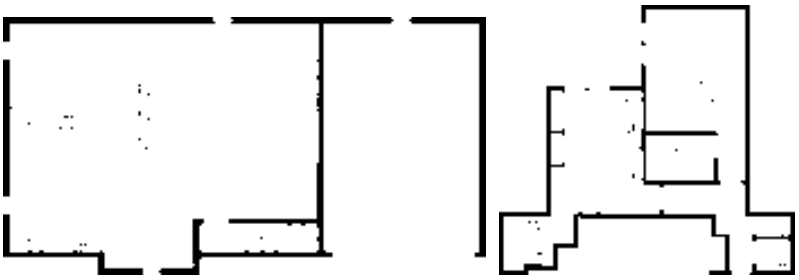


Figure A.4: Example of floorplans from SESYD (a) floorplans16-07 (b) floorplans16-08 subset.

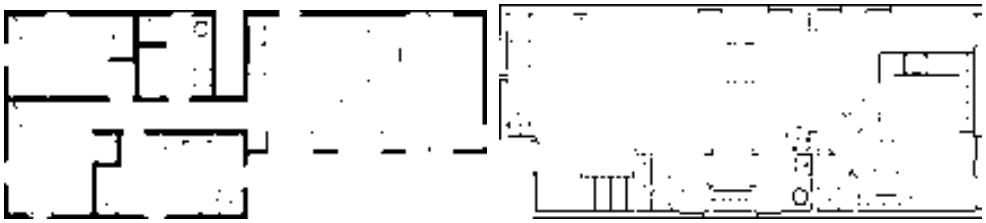


Figure A.5: Example of floorplans from SESYD (a) floorplans16-09 (b) floorplans16-10 subset.

the QGar[3] library. This dataset provides 38 isolated symbols for querying purpose, some of them are shown in the Figure A.6. Ground truths are also available which relate query symbol with their location in the respective floorplans.

A.3 SESYD-DN

This dataset has been generated by us on the original SESYD (floorplans) dataset. It has been done by randomly drawing white horizontal lines of 2-3 pixels width. This generates random discontinuity of black pixels. The main motivation of generating this dataset was to show the advantage of the proposed NCRAG representation over the traditional RAG representation. Two sample images from this dataset are shown in Figure A.8. Since the introduction of this kind of noise does not hamper the position of pixels we use the original ground truth provided in the SESYD dataset for the correspondences of isolated symbols to the target documents. The noise is only generated on the floorplans and not on the isolated symbols. This is to test the stability of the representation.

[3] http://www.qgar.org

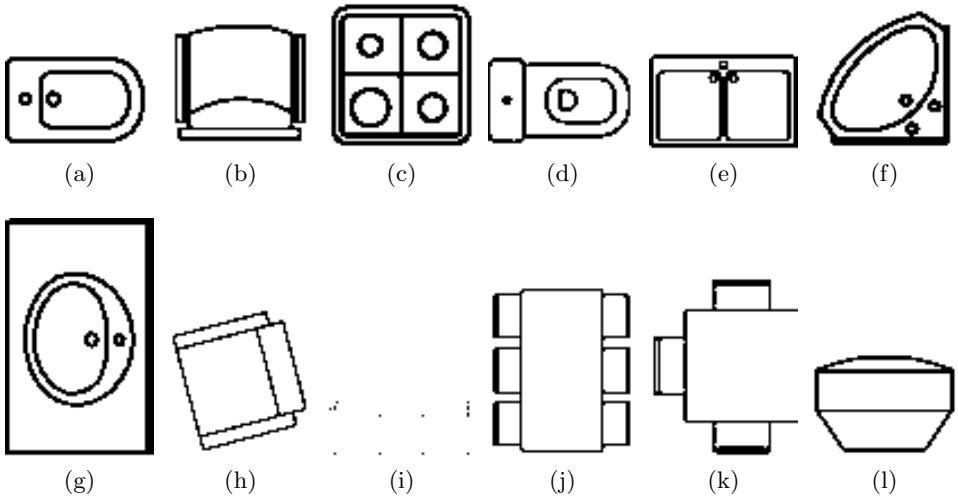


Figure A.6: Example of different query symbols from the FPLAN-POLY dataset.

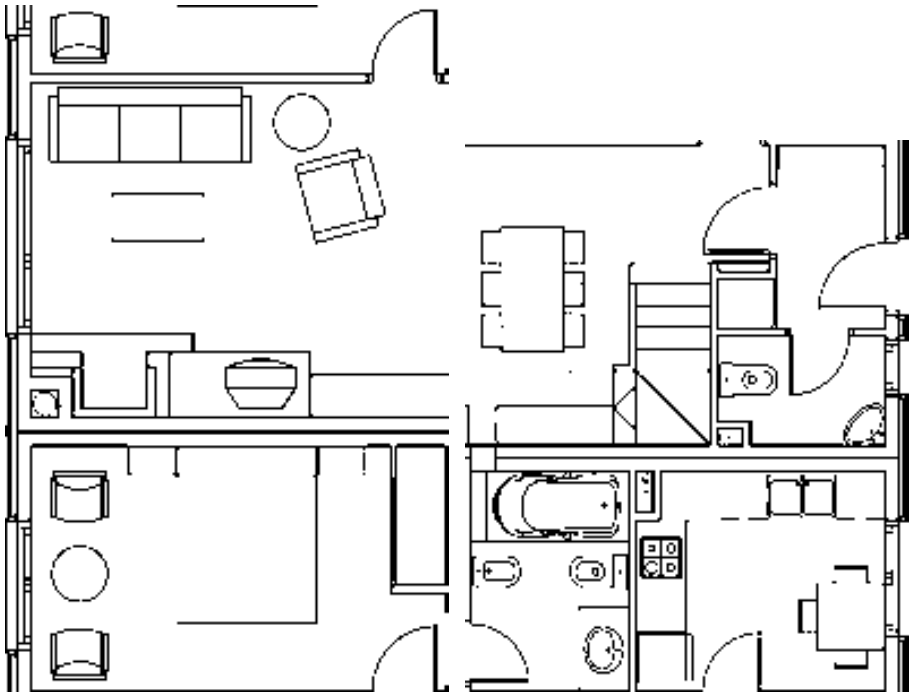
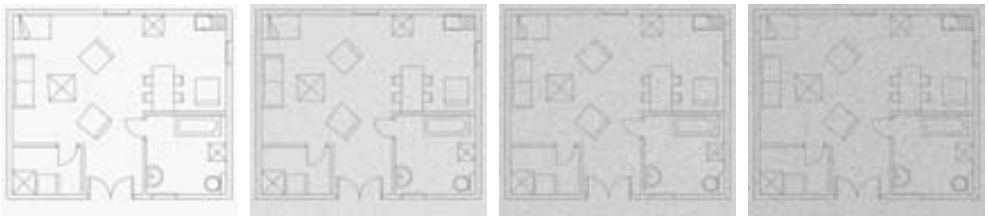


Figure A.7: Example of floorplans from the FPLAN-POLY dataset.

Figure A.8: Example of floorplans from the SESYD-DN dataset.

A.4 SESYD-GN

This dataset has also been generated on the SESYD (floorplans) dataset. For this we used the Gaussian noise at different levels by varying the mean (m) and variance (σ). Practically, the increment of variance introduced more pepper noise into the images, whereas the increment of the mean introduced more and more white noise, which will detach the object pixel connection. This kind of noise nearly simulates documents suffered due to scanning or some other low level image processing. Example of images can be seen in Figure A.9. This noise model is only applied only on the floorplans. Here also we consider the original ground truths from the dataset for the correspondence.

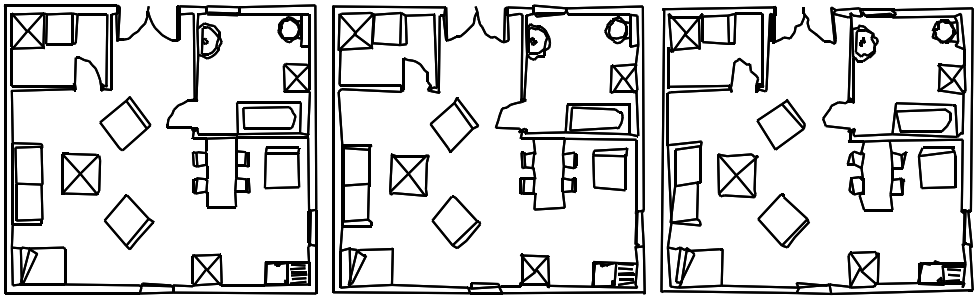


(a) $\sigma = 0.10$ (b) $\sigma = 0.40$ (c) $\sigma = 0.70$ (d) $\sigma = 1.00$

Figure A.9: Example of floorplans from the SESYD-GN dataset with $m = 0.30$.

A.5 SESYD-VN

As the previous two, this dataset has also been created by us and it has been done on the SESYD (floorplans) dataset. Here we introduced vectorial noise and it has been done by randomly shifting the primitive points (critical points detected by the vectorization process) within a circle of radius r. We vary r to get different level of vectorial distortions. This kind of noise simulate the documents containing handwritten sketch. Figure A.10 shows some example of images from this dataset.



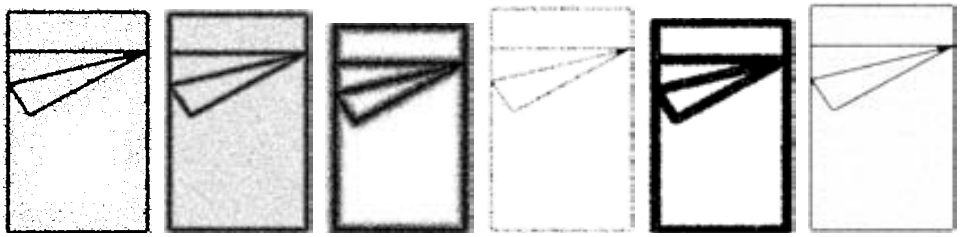
(a) $r = 5$ (b) $r = 10$ (c) $r = 15$

Figure A.10: Example of floorplans from the SESYD-VN dataset.

The Matlab codes and a subset of all the three datasets created by us are available in the link: `http://www.cvc.uab.es/~adutta/datasets`.

A.6 GREC 2005 dataset

This dataset is a collection of degraded isolated symbols and created with the motivation for conducting a symbol recognition contest in the IAPR Workshop on Graphics Recognition in the year 2005 [24][4]. The dataset comes with upto 150 different model symbols with six degradation models. The tests are available in four different configurations from rotation and scaling point of view: (1) not rotated and not scaled, (1) not rotated but scaled (3) rotated but not scaled and (4) rotated and scaled.



(a) degradation model1 (b) degradation model2 (c) degradation model3 (d) degradation model4 (e) degradation model5 (f) degradation model6

Figure A.11: Example of isolated images from the GREC-2005 dataset.

[4] `http://symbcontestgrec05.loria.fr/finaltest.php`

A.7 ILPIso dataset

This dataset is composed of synthetic and real sets of graphs and is published by Le Bodic *et al.* [8][5]. There are four synthetic datasets, all of which are node and edge attributed and contain set of pattern graphs, target graphs and ground truths that relate the pattern graphs with the target graphs. Two of these sets contain simple graphs that may contain isolated nodes. One of these two contains pairs of pattern, target graphs containing exact mapping between the corresponding vertices/edges *i.e.*there are perfect equality of labels. The other one contains pairs of pattern, target graphs where the mappings between the nodes/edges are disturbed by Gaussian noise ($m = 0$, $\sigma = 5$). The other two datasets of the synthetic category have the same characteristic except there is no isolated nodes *i.e.*all the nodes at least have a neighbour. The real dataset contains 16 pattern graphs and 200 target graphs. Pattern graphs are the RAG representation of the isolated symbols and target graphs are the same for the floorplans. The ground truth files are also available which relate the pattern graphs with the target graphs with region to region correspondence.

A.8 L'Esposallas dataset

This dataset is a collection of pages compiled from marriage licence books conserved at the archives of Barcelona cathedral [81][6]. The original marriage register is composed of 291 books/volumes with information of 600,000 marriages celebrated in 250 parishes in between 1451 and 1905. The dataset has three partitions: (1) the indices, (2) the marriage record and (3) the segmented lines. At present the indices part of this dataset contains only the indices of two volumes and the marriage record part is taken from a single volume. Some examples of indices and register pages are visible in Figure A.12.

[5] litis-ilpiso.univ-rouen.fr/ILPIso

[6] http://dag.cvc.uab.es/the-esposalles-database

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

Figure A.12: Example of pages from the marriage registers from the L Esposalles dataset: (a)-(b) indices, (c)-(d) register pages.

List of Publications

This dissertation has led to the following communications:

Journal Papers

- Anjan Dutta, Josep Lladós and Umapada Pal. A symbol spotting approach in graphical documents by hashing serialized graphs . In Pattern Recognition (PR), 46(3), pp. 752-768, March 2013.

- Josep Lladós, Marçal Rusinol, Alicia Fornés, David Fernández and Anjan Dutta. On the Influence of Word Representations for Handwritten Word Spotting in Historical Documents . In International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI), 26(5), August 2012.

Book Chapters

- Anjan Dutta, Josep Lladós and Umapada Pal. Bag-of-GraphPaths for symbol Recognition and Spotting in Line Drawings . In Graphic Recognition. New Trends and Challenges. Lecture Notes in Computer Science, Vol. 7423, pp. 208-217, 2013.

- Klaus Broelemann, Anjan Dutta, Xiaoyi Jiang, and Josep Lladós. Hierarchical graph representation for symbol spotting in graphical document images . In the Proceedings of the 14th International Workshop on Statistical, Structural and Syntactic Pattern Recognition (S+SSPR, 2012), pp. 529-538, Miyajima-Itsukushima, Hiroshima, November, 2012.

Conference Contributions

- Anjan Dutta, Josep Lladós, Horst Bunke and Umapada Pal. Near Convex Region Adjacency Graph and Approximate Neighborhood String Matching for Symbol Spotting in Graphical Documents . In the Proceedings of the 12th International Conference on Document Analysis and Recognition (ICDAR, 2013), pp. 1078-1082, Washington DC, USA, August, 2013.

- Anjan Dutta, Josep Lladós, Horst Bunke and Umapada Pal. A Product graph based method for dual subgraph matching applied to symbol spotting . In the Proceedings of the 10th IAPR Workshop on Graphics Recognition (GREC, 2013), pp. 7-11, Bethlehem, PA, USA, August, 2013.

- Klaus Broelemann, Anjan Dutta, Xiaoyi Jiang and Josep Lladós. Hierarchical Plausibility-Graphs for Symbol Spotting in Graphical Documents . In the Proceedings of the 10th IAPR Workshop on Graphics Recognition (GREC, 2013), pp. 13-18, Bethlehem, PA, USA, August, 2013.

- Anjan Dutta, Jaume Gibert, Josep Lladós, Horst Bunke and Umapada Pal. Combination of Product Graph and Random Walk Kernel for Symbol Spotting in Graphical Documents . In the Proceedings of the 21st International Conference on Pattern Recognition (ICPR, 2012), pp. 1663-1666, Tsukuba, Japan, November, 2012.

- Anjan Dutta, Josep Lladós and Umapada Pal. Symbol Spotting in Line Drawings Through Graph Paths Hashing . In the Proceedings of the 11th International Conference on Document Analysis and Recognition (ICDAR, 2011), pp. 982-986, Beijing, China, September, 2011.

- Anjan Dutta, Josep Lladós and Umapada Pal. Bag-of-GraphPaths for symbol Recognition and Spotting in Line Drawings . In the Proceedings of the 9th IAPR Workshop on Graphics Recognition (GREC, 2011), pp. 49-52, Seoul, South Korea, September, 2011.

- Anjan Dutta, Josep Lladós and Umapada Pal. A Bag-of-Paths based Serialized Subgraph Matching for Symbol Spotting in Line Drawings . In the Proceedings of the 5th Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA, 2011), Lecture Notes in Computer Science (LNCS 6669), Jordi Vitrià, Joao M. Sanches, Mario Hernández (Eds.), pp. 620-627, Las Palmas de Gran Canaria, Spain, June, 2011.

Bibliography

[1] Narendra Ahuja and Sinisa Todorovic. From region based image representation to object discovery and recognition. In Edwin R. Hancock, Richard C. Wilson, Terry Windeatt, Ilkay Ulusoy, and Francisco Escolano, editors, *Proceedings of the International Workshop on Structural, Syntactic and Statistical Pattern Recognition (SPR + SSPR)*, volume 6218 of *Lecture Notes in Computer Science*, pages 1 19. Springer Berlin / Heidelberg, 2010.

[2] H. A. Almohamad and S. O. Duffuaa. A linear programming approach for the weighted graph matching problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 15(5):522 525, 1993.

[3] N. Arica and F.T. Yarman-Vural. Optical character recognition for cursive handwriting. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 24(6):801 813, 2002.

[4] F. Aziz, R.C. Wilson, and E.R. Hancock. Backtrackless walks on a graph. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, 24(6):977 989, 2013.

[5] E. Barbu, P. Heroux, S. Adam, and E. Trupin. Frequent graph discovery: Application to line drawing document images. *Electronic Letters on Computer Vision and Image Analysis*, 5(2):47 54, 2005.

[6] A. Barducci and S. Marinai. Object recognition in floor plans by graphs of white connected components. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, pages 298 301, 2012.

[7] G. M. Beumer, Q. Tao, A. M. Bazen, and R. N J Veldhuis. A landmark paper in face recognition. In *Proceedings of the International Conference on Automatic Face and Gesture Recognition (ICAFGR)*, pages 78 83, 2006.

[8] Pierre Le Bodic, Pierre Hèroux, Sébastien Adam, and Yves Lecourtier. An integer linear program for substitution-tolerant subgraph isomorphism and its use for symbol spotting in technical drawings. *Pattern Recognition (PR)*, 45(12):4214 4224, 2012.

[9] O. Boiman, E. Shechtman, and M. Irani. In defense of nearest-neighbor based image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1 8, 2008.

[10] I.R. Bomze, M. Pelillo, and V. Stix. Approximating the maximum weight clique using replicator dynamics. *IEEE Transactions on Neural Network (TNN)*, 11(6):1228 1241, 2000.

[11] K.M. Borgwardt and H.-P. Kriegel. Shortest-path kernels on graphs. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 8 pp. , 2005.

[12] Klaus Broelemann, Anjan Dutta, Xiaoyi Jiang, and Josep Lladós. Hierarchical graph representation for symbol spotting in graphical document images. In Georgy Gimel farb, Edwin Hancock, Atsushi Imiya, Arjan Kuijper, Mineichi Kudo, Shinichiro Omachi, Terry Windeatt, and Keiji Yamada, editors, *Proceedings of the International Workshop on Structural, Syntactic, and Statistical Pattern Recognition (SPR + SSPR)*, volume 7626 of *Lecture Notes in Computer Science*, pages 529 538. Springer Berlin / Heidelberg, 2012.

[13] H Bunke and G Allermann. Inexact graph matching for structural pattern recognition. *Pattern Recognition Letters (PRL)*, 1(4):245 253, 1983.

[14] Horst Bunke and Kaspar Riesen. Improving vector space embedding of graphs through feature selection algorithms. *Pattern Recognition (PR)*, 44(9):1928 1940, 2010.

[15] Ronald R. Coifman and Stéphane Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5 30, 2006.

[16] D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI)*, 18(3):265 298, 2004.

[17] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. An improved algorithm for matching large graphs. In *Proceedings of the International Workshop on Graph-Based Representations in Pattern Recognition (GbRPR)*, pages 149 159, 2001.

[18] L.P. Cordella, P. Foggia, C. Sansone, and M. Vento. An efficient algorithm for the inexact matching of arg graphs using a contextual transformational model. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 3, pages 180 184, 1996.

[19] L.P. Cordella, P. Foggia, C. Sansone, and M. Vento. Performance evaluation of the vf graph matching algorithm. In *Proceedings of the International Conference on Image Analysis and Processing (CIAP)*, pages 1172 1177, 1999.

[20] L.P. Cordella, P. Foggia, C. Sansone, and M. Vento. Fast graph matching for detecting cad image components. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, volume 2, pages 1034 1037, 2000.

[21] L.P. Cordella, P. Foggia, C. Sansone, and M. Vento. A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 26(10):1367 1372, 2004.

[22] Mathieu Delalandre, Tony Pridmore, Ernest Valveny, Hervé Locteau, and Eric Trupin. Building synthetic graphical documents for performance evaluation. In *Graphics Recognition. Recent Advances and New Opportunities*, pages 288 298. Springer Berlin/Heidelberg, 2008.

[23] Philippe Dosch and Josep Lladós. *Vectorial Signatures for Symbol Discrimination*, chapter Vectorial Signatures for Symbol Discrimination, pages 154 165. Springer Berlin / Heidelberg, 2004.

[24] Philippe Dosch and Ernest Valveny. Report on the second symbol recognition contest. In Wenyin Liu and Josep Lladós, editors, *Graphics Recognition. Ten Years Review and Future Perspectives*, volume 3926 of *Lecture Notes in Computer Science*, pages 381 397. Springer Berlin / Heidelberg, 2006.

[25] Anjan Dutta, Josep Lladós, and Umapada Pal. A bag-of-paths based serialized subgraph matching for symbol spotting in line drawings. In *Proceedings of the Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA)*, pages 620 627, 2011.

[26] Anjan Dutta, Josep Lladós, and Umapada Pal. A symbol spotting approach in graphical documents by hashing serialized graphs. *Pattern Recognition (PR)*, 46(3):752 768, 2013.

[27] A. El-Yacoubi, M. Gilloux, R. Sabourin, and C.Y. Suen. An hmm-based approach for off-line unconstrained handwritten word modeling and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 21(8):752 760, 1999.

[28] S. Espana-Boquera, M.J. Castro-Bleda, J. Gorbe-Moya, and F. Zamora-Martinez. Improving offline handwritten text recognition with hybrid hmm/ann models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 33(4):767 779, 2011.

[29] V. Espinosa-Duro. Minutiae detection algorithm for fingerprint recognition. *IEEE Aerospace and Electronic Systems Magazine*, 17(3):7 10, 2002.

[30] Andreas Fischer, ChingY. Suen, Volkmar Frinken, Kaspar Riesen, and Horst Bunke. A fast matching algorithm for graph-based handwriting recognition. In Walter G. Kropatsch, Nicole M. Artner, Yll Haxhimusa, and Xiaoyi Jiang, editors, *Proceedings of the International Workshop on Graph-Based Representations in Pattern Recognition (GbRPR)*, volume 7877 of *Lecture Notes in Computer Science*, pages 194 203. Springer Berlin/Heidelberg, 2013.

[31] Thomas Gartner, Peter A. Flach, and Stefan Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Proceedings of the Conference on Learning Theory (COLT)*, pages 129 143, 2003.

[32] Jaume Gibert. *Vector Space Embedding of Graphs via Statistics of Labelling Information*. PhD thesis, Universitat Autónoma de Barcelona, 2012.

[33] Jaume Gibert, Ernest Valveny, and Horst Bunke. Graph embedding in vector spaces by node attribute statistics. *Pattern Recognition (PR)*, 45(9):3072 3083, 2012. ce:title Best Papers of Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA 2011) /ce:title .

[34] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 518 529, San Francisco, CA, USA, 1999.

[35] R. Giugno and D. Shasha. Graphgrep: A fast and universal method for querying graphs. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, volume 2, pages 112 115, 2002.

[36] T. Gnanasambandan, S. Gunasekaran, and S. Seshadri. Molecular structure analysis and spectroscopic characterization of carbimazole with experimental (ft-ir, ft-raman and uv-vis) techniques and quantum chemical calculations. *Journal of Molecular Structure (JMS)*, 1052(0):38 49, 2013.

[37] Steven Gold and Anand Rangarajan. A graduated assignment algorithm for graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 18(4):377 388, 1996.

[38] Lin Han, Francisco Escolano, Edwin R. Hancock, and Richard C. Wilson. Graph characterizations from von neumann entropy. *Pattern Recognition Letters (PRL)*, 33(15):1958 1967, 2012.

[39] Z. Harchaoui and F. Bach. Image classification with segmentation graph kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1 8, June 2007.

[40] Y. He and A. Kundu. 2-d shape classification using hidden markov model. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 13(11):1172 1184, November 1991.

[41] Tamás Horváth, Thomas Gartner, and Stefan Wrobel. Cyclic pattern kernels for predictive graph mining. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, KDD 04, pages 158 167, New York, NY, USA, 2004. ACM.

[42] Khalid Hosny. Fast computation of accurate zernike moments. *Journal of Real-Time Image Processing (JRTIP)*, 3:97 107, 2008.

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

[44] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the ACM Symposium on Theory of Computing (ACMSTOC)*, pages 604 613, 1998.

[45] David W. Jacobs. Grouping for recognition, 1989.

[46] David W. Jacobs. Robust and efficient detection of salient convex groups. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 18(1):23 37, January 1996.

[47] Thorsten Joachims. Text categorization with suport vector machines: Learning with many relevant features. In *Proceedings of the European Conference on Machine Learning (ECML)*, pages 137 142. Springer-Verlag, 1998.

[48] Shafiq Rayhan Joty and Sheikh Sadid-Al-Hasan. Advances in focused retrieval: A general review. In *Proceedings of the IEEE International Conference on Computer and Information Technology (ICCIT)*, pages 1 5, 2007.

[49] D. Justice and A. Hero. A binary linear programming formulation of the graph edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 28(8):1200 1214, 2006.

[50] R. Kaushik, P. Shenoy, P. Bohannon, and E. Gudes. Exploiting local similarity for indexing paths in graph-structured data. In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 129 140, 2002.

[51] G. Lambert and J. Noll. Discrimination properties of invariants using the line moments of vectorized contours. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, volume 2, pages 735 739, 1996.

[52] Georg Lambert and Hua Gao. Line moments and invariants for real time processing of vectorized contour data. In *Image Analysis and Processing*, volume 974 of *Lecture Notes in Computer Science*, pages 347 352. Springer Berlin / Heidelberg, 1995.

[53] Javier Larrosa and Gabriel Valiente. Constraint satisfaction algorithms for graph pattern matching. *Mathematical Structures in Computer Science (MSCS)*, 12(4):403 422, 2002.

[54] Josep Lladós, E. Martí, and Juan José Villanueva. Symbol recognition by error-tolerant subgraph matching between region adjacency graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 23(10):1137 1143, 2001.

[55] Hervé Locteau, Sébastien Adam, Éric Trupin, Jacques Labiche, and Pierre Héroux. Symbol spotting using full visibility graph representation. In *Proceedings of the International Workshop of Graphics Recognition (GREC)*, 2007.

[56] D G Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence (AI)*, 31(3):355 395, March 1987.

[57] Shijian Lu, Linlin Li, and Chew Lim Tan. Document image retrieval through word shape coding. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 30(11):1913 1918, 2008.

[58] Bin Luo, Richard C. Wilson, and Edwin R. Hancock. Spectral embedding of graphs. *Pattern Recognition (PR)*, 36(10):2213 2230, 2003.

[59] M.M. Luqman, T. Brouard, J.-Y. Ramel, and J. Lladós. A content spotting system for line drawing graphic document images. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, pages 3420 3423, August 2010.

[60] M.M. Luqman, J. Ramel, J. Llados, and T. Brouard. Subgraph spotting through explicit graph embedding: An application to content spotting in graphic document images. In *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, pages 870 874, 2011.

[61] Muhammad Muzzamil Luqman, Jean-Yves Ramel, Josep Lladós, and Thierry Brouard. Fuzzy multilevel graph embedding. *Pattern Recognition (PR)*, 46(2):551 565, 2013.

[62] Kurt Mehlhorn. *Graph algorithms and NP-completeness*. Springer-Verlag New York, Inc., New York, NY, USA, 1984.

[63] B. Messmer and H. Bunke. Automatic learning and recognition of graphical symbols in engineering drawings. In Rangachar Kasturi and Karl Tombre, editors, *Graphics Recognition Methods and Applications*, volume 1072 of *Lecture Notes in Computer Science*, pages 123 134. Springer Berlin / Heidelberg, 1996.

[64] B.T. Messmer and H. Bunke. A decision tree approach to graph and subgraph isomorphism detection. *Pattern Recognition (PR)*, 32(12):1979 1998, 1999.

[65] B.T. Messmer and H. Bunke. Efficient subgraph isomorphism detection: a decomposition approach. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 12(2):307 323, 2000.

[66] Stefan Muller and Gerhard Rigoll. Engineering drawing database retrieval using statistical pattern spotting techniques. In *Graphics Recognition Recent Advances*, volume 1941 of *Lecture Notes in Computer Science*, pages 246 255. Springer Berlin / Heidelberg, 2000.

[67] N. Nayef and T.M. Breuel. Statistical grouping for segmenting symbols parts from line drawings, with application to symbol spotting. In *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, pages 364 368, 2011.

[68] Nibal Nayef and Thomas M. Breuel. A branch and bound algorithm for graphical symbol recognition in document images. In *Proceedings of the International Workshop on Document Analysis System (DAS)*, pages 543 546, 2010.

[69] Michel Neuhaus and Horst Bunke. An error-tolerant approximate matching algorithm for attributed planar graphs and its application to fingerprint classification. In *Proceedings of the International Workshop on Statistical, Structural and Syntactic Pattern Recognition (SPR +SSPR)*, pages 180 189, 2004.

[70] Thi-Oanh Nguyen, S. Tabbone, and A. Boucher. A symbol spotting approach based on the vector model and a visual vocabulary. In *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, pages 708 712, July 2009.

[71] J.C. Niebles and Li Fei-Fei. A hierarchical model of shape and appearance for human action classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1 8, 2007.

[72] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics (TSMC)*, 9(1):62 66, January 1979.

[73] Marcello Pelillo. Relaxation labeling networks for the maximum clique problem. *Journal of Artificial Neural Network (JANN)*, 2(4):313 328, August 1996.

[74] Marcello Pelillo, Kaleem Siddiqi, and Steven W. Zucker. Matching hierarchical structures using association graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 21(11):1105 1120, 1999.

[75] R. Plamondon and S.N. Srihari. Online and off-line handwriting recognition: a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 22(1):63 84, 2000.

[76] Ronald Poppe. A survey on vision-based human action recognition. *Image and Vision Computing (IVC)*, 28(6):976 990, 2010.

[77] Rashid Qureshi, Jean-Yves Ramel, Didier Barret, and Hubert Cardot. Spotting symbols in line drawing images using graph representations. In Wenyin Liu, Josep Lladós, and Jean-Marc Ogier, editors, *Graphics Recognition. Recent Advances and New Opportunities*, volume 5046 of *Lecture Notes in Computer Science*, pages 91 103. Springer Berlin / Heidelberg, 2008.

[78] Toni M. Rath and R. Manmatha. Word image matching using dynamic time warping. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 521 527, Los Alamitos, CA, USA, 2003. IEEE Computer Society.

[79] Zhou Ren, Junsong Yuan, Chunyuan Li, and Wenyu Liu. Minimum near-convex decomposition for robust shape representation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 303 310, November 2011.

[80] Kaspar Riesen, Michel Neuhaus, and Horst Bunke. Bipartite graph matching for computing the edit distance of graphs. In Francisco Escolano and Mario Vento, editors, *Proceedings of the International Workshop on Graph-Based Representations in Pattern Recognition (GbRPR)*, volume 4538 of *Lecture Notes in Computer Science*, pages 1 12. Springer Berlin Heidelberg, 2007.

[81] Verónica Romero, Alicia Fornés, Nicolás Serrano, Joan Andreu Sánchez, Alejandro H. Toselli, Volkmar Frinken, Enrique Vidal, and Josep Lladós. The {ESPOSALLES} database: An ancient marriage license corpus for off-line handwriting recognition. *Pattern Recognition (PR)*, 46(6):1658 1669, 2013.

[82] Paul L Rosin and Geoff AW West. Segmentation of edges into lines and arcs. *Image and Vision Computing (IVC)*, 7(2):109 114, 1989.

[83] M. Rusinol. *Geometric and Structural-based Symbol Spotting. Application to Focused Retrieval in Graphic Document Collections.* PhD thesis, Universitat Autònoma de Barcelona, 2009.

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

[85] M. Rusinol and J. Lladós. *Symbol Spotting in Technical Drawings Using Vectorial Signatures*, chapter Symbol Spotting in Technical Drawings Using Vectorial Signatures, pages 35 46. Springer Berlin / Heidelberg, 2006.

[86] M. Rusinol and J. Lladós. A performance evaluation protocol for symbol spotting systems in terms of recognition and location indices. *International Journal on Document Analysis and Recognition (IJDAR)*, 12(2):83 96, 2009.

[87] M. Rusinol, J. Lladós, and G. Sánchez. Symbol spotting in vectorized technical drawings through a lookup table of region strings. *Pattern Analysis and Applications (PAA)*, 13:1 11, 2009.

[88] L. Schomaker. Advances in writer identification and verification. In *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, volume 2, pages 1268 1273, 2007.

[89] A. Shokoufandeh, D. Macrini, S. Dickinson, K. Siddiqi, and S.W. Zucker. Indexing hierarchical structures using graph spectra. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 27(7):1125 1140, 2005.

[90] Christine Solnon. Alldifferent-based filtering for subgraph isomorphism. *Artificial Intelligence (AI)*, 174(12-13):850 864, 2010.

[91] H. Sossa and R. Horaud. Model indexing: the graph-hashing approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 811 814, June 1992.

[92] H.M. Stark and A.A. Terras. Zeta functions of finite graphs and coverings. *Journal on Advances in Mathematics (JAM)*, 121(1):124 165, 1996.

[93] Minsoo Suk and Tai-Hoon Cho. An object-detection algorithm based on the region-adjacency graph. *Proceedings of the IEEE*, 72(7):985 986, July 1984.

[94] Martin Szummer and Tommi Jaakkola. Partially labeled classification with markov random walks. In *Advances in Neural Information Processing Systems*, pages 945 952. MIT Press, 2002.

[95] S. Tabbone, L. Wendling, and K. Tombre. Matching of graphical symbols in line-drawing images using angular signature information. *International Journal on Document Analysis and Recognition (IJDAR)*, 6(2):115 125, 2003.

[96] Michael Reed Teague. Image analysis via the general theory of moments. *Journal of the Optical Society of America (JOSA)*, 70(8):920 930, August 1980.

[97] Karl Tombre and B. Lamiroy. Pattern recognition methods for querying and browsing technical documentation. In *Proceedings of the Iberoamerican Congress on Pattern Recognition (CIARP)*, 2008.

[98] Wen-Hsiang Tsai and King-Sun Fu. Error-correcting isomorphisms of attributed relational graphs for pattern analysis. *IEEE Transactions on Systems, Man and Cybernetics (TSMC)*, 9(12):757 768, 1979.

[99] Wen-Hsiang Tsai and King-Sun Fu. Subgraph error-correcting isomorphisms for syntactic pattern recognition. *IEEE Transactions on Systems, Man and Cybernetics (TSMC)*, 13(1):48 62, 1983.

[100] J. R. Ullmann. An algorithm for subgraph isomorphism. *Journal of ACM (JACM)*, 23(1):31 42, 1976.

[101] Chao Wang, Lei Wang, and Lingqiao Liu. Improving graph matching via density maximization. In *Proceedings of the IEEE Conference on Computer Vision (ICCV)*, December 2013.

[102] Ching-Huei Wang and SargurN. Srihari. A framework for object recognition in a visually complex environment and its application to locating address blocks on mail pieces. *International Journal of Computer Vision (IJCV)*, 2(2):125 151, 1988.

[103] S. Watanbe. *Pattern recognition: human and mechanical.* Wiley, 1985.

[104] R.C. Wilson and E.R. Hancock. Structural matching by discrete relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 19(6):634 648, 1997.

[105] Xifeng Yan, Philip S. Yu, and Jiawei Han. Graph indexing: a frequent structure-based approach. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, pages 335 346, 2004.

[106] Xingwei Yang, L. Prasad, and L.J. Latecki. Affinity learning with diffusion on tensor product graph. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 35(1):28 38, 2013.

[107] Xu Yang, Hong Qiao, and Zhi-Yong Liu. Partial correspondence based on subgraph matching. *Neurocomputing*, 122:193 197, 2013.

[108] Shijie Zhang, Meng Hu, and Jiong Yang. Treepi: A novel graph indexing method. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, pages 966 975, 2007.

[109] Wan Zhang and Liu Wenyin. A new vectorial signature for quick symbol indexing, filtering and recognition. In *Proceedings of the International Conference of Document Analysis and Recognition (ICDAR)*, volume 1, pages 536 540, September 2007.

[110] Daniel Zuwala and Salvatore Tabbone. *A Method for Symbol Spotting in Graphical Documents*, pages 518 528. Springer Berlin / Heidelberg, 2006.