Universitat Autònoma de Barcelona

escola d'enginyeria

5956.- Graph-Based Word Spotting by Inexact Matching Techniques

Memòria del Projecte Fi de Carrera
d'Enginyeria Informàtica
realitzat per *Pau Riba Fiérrez*
i dirigit per
Josep Lladós Canet i per *Alicia Fornés Bisquerra.*
Bellaterra, 8 de Juny de 2015

Els sotsignants, Josep Lladós Canet i Alicia Fornés Bisquerra, professors de l'Escola d'Enginyeria de la UAB,

CERTIFIQUEN:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en Pau Riba Fiérrez

I per a que consti firma la present.

Signat: Josep Lladós Canet i Alicia Fornés Bisquerra
Bellaterra, 8 de Juny de 2015

A tots els que m'han hagut de patir
al llarg d'aquest camí.

Acknowledgements

Fa 7 anys, vaig realitzar el meu Treball de Recerca a l'institut, aquell treball, va ser la meva entrada al món dels grafs. En aquell moment, no m'hauria pensat que 6 anys després em tornaria a endinsar en aquest món des d'un altre vessant. És probable, que la meva professora de matemàtiques de l'institut, l'Imma, sigui la culpable que finalment, quan vaig haver de decidir el meu Projecte Final de Carrera, em decidís per aquest. Així doncs volia aprofitar per agrair-li la introducció que va suposar.
Principalment, volia donar les gràcies al Josep Lladós i l'Alicia Fornés, els dos tutors d'aquest projecte. Els hi volia agrair l'entusiasme que van mostrar quan tot aquest treball va començar, així com tota l'ajuda rebuda al llarg d'aquests mesos. Especialment donar-los les gràcies per obrir-me les portes en el món de la recerca.
Un agraïment molt especial al David, al Jon i a l'Anjan que tot i estar molt enfeinats acabant la seva tesi, em van donar un cop de mà en els primers compassos d'aquest projecte.
Aquest projecte ha estat possible gràcies a una beca de col·laboració convocada pel Ministeri d'Educació, Cultura i Esport.
Finalment, donar les gràcies als meus pares, familiars, amics, companys de pis i a la Sònia pel suport rebut.

Berga, Juny de 2015.

Contents

List of Figures

List of Tables

List of Algorithms

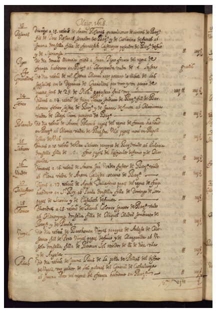
Chapter 1

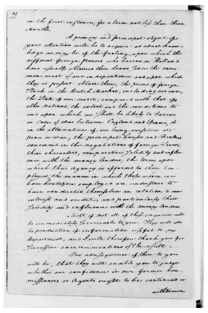
Introduction

This chapter gives a brief introduction to the developed project. It defines basic concepts and explains the planification. Firstly, there is an introduction to Historical Handwriten documents. Secondly, word spotting techniques and graphs representations are presented. Afterwards, the motivation and goals of the work are set. Then, a study of the viability and the temporal planification is explained. Finally, the document structure is drawn.

1.1 Historical Documents

Since the invention of writing, the written language has been an important way of communication for humans. Through handwritten documents, researchers can study the history, the manners and customs of the people, the relations between them, etc. Figure 1.1 presents a few examples of ancient documents.


(a) Llibre de les Esposalles


(b) George Washington Papers

(c) Abbey Library of Saint Gall

Figure 1.1: Examples of old handwritten documents.

Today, only a little part of the cultural heritage has been digitised but in a few years it will increase in order to make the documents accessible for everyone. At the same time the number of digitised documents increase, the need to transcribe, extract and search information through the computers is more necessary. All this techniques belong to the *Document Image Analysis* research field, in this case to *Historical Documents*.

In the case of the preservation of historical handwritten document collections, Document Image Analysis is of key importance for archives, museums and libraries. Their goal is not only the digitisation of paper documents, but also the extraction of the information that these documents contain towards the creation of digital libraries. Since the manual transcription by human experts is prohibitive, the challenge is to enable the automatic extraction of information through document image analysis techniques.

Word spotting, explained in the next section, applied to historical handwritten documents can be very useful for researchers who have to study those manuscripts. These techniques can be the core of new tools for the new technologies available such as smartphones or tablets.

1.2 Word Spotting

Nowadays, there is a lot of information on paper documents or on videos where the data is not in a readable format for computers. When these documents are in good machine printed fonts against good clean backgrounds an *Optical Character Recognition* (OCR) will properly work but OCR will poorly work with handwritten texts.

Word spotting is a content-based retrieval strategy where, due to the impossibility of a recognition process with enough quality, leans to a visual object detection approach. The key idea of word spotting relies upon representing word images with robust features and a subsequent classification scheme. It consists in locating a particular keyword within a document.

One of the first successful solutions was proposed by Manmatha *et al.* [21, 32].

Word spotting will allow the users to look for information in handwritten documents without the transcription in machine readable format. For instance, researchers such as historians, will be able to apply these techniques in historical documents.

Furthermore, handwritten word spotting (HWS) had been used to speed up the transcription of documents [32]. The system compares the word images against each other in order to create equivalence classes. Afterwards, the user will provide ASCII transcriptions for a representative word for each one of the top m classes. Providing the ASCII representation of some classes will result in a faster transcription process of the whole text.

Handwritten word spotting can be classified depending on the input query word provided in order to spot in the documents.

- *Query-By-Example*: The input is an example of the key word, for instance a word image cropped from the document.
- *Query-By-String*: The input key word is a string type written by the user using a keyboard.

- *Query-By-Sketch*: The key word is written by the user.

This work is focused in *Query-By-Example* handwritten word spotting for historical documents.

1.3 Graph Representations

A graph is a structural representation based on describing relations (edges) between different objects (nodes). For example, using a graph, it is possible to represent: Social Network where the nodes are the users and the edges represent the kind of relation between them; or visual objects where the edges represent the relations between their parts which are represented by nodes. Formally, a graph can be defined as:

Definition 1.3.1 (Graph)**.** *Let L be a finite or infinite set of labels for nodes and edges. A graph G is a four-tuple $G = (V, E, \mu, \nu)$ where*

- *V is the finite set of nodes,*
- *$E \subseteq V \times V$ is the set of edges,*
- *$\mu : V \to L$ is the node labelling function,*
- *$\nu : E \to L$ is the edge labelling function.*

where $|V|$ denotes the number of vertices and $|E|$ the number of edges.

Graphs are representations offering a robust paradigm able to deal with many-to-many relationships among visual features and their parts. The use of graph matching is an effective solution to deal with visual recognition. However, it is necessary to define a stable graph construction in order to handle problems where the graphs have to be compared.

1.4 Motivation

Traditionally, word spotting is based on appearance-based features. In this work we aim to explore structural features as an alternative or complementary representation offering a richest description. The motivation for representing the handwritten words using graph-based methods is to try to keep information of the two dimensional structure of handwritten text. This representation aims to avoid the loss of information provided by the appearance-based representation in one dimensional scalar vector of features. The main motivation is that the nature of handwriting suggests that the structure is more stable than the pure appearance of the handwritten strokes. This is specially important when dealing with the elastic deformations of different handwriting styles.

Although the use of graphs seems to be a natural solution, most existing word spotting techniques use statistical representations (e.g. SIFT, HOG) of the word images [4, 40]. As stated in the comparison of statistical versus structural representations for handwritten word spotting reported in [20], the main disadvantages of structural approaches are the time complexity and scalability to large document collections. Although some methods [12] only use the graph nodes (avoiding the edges), and other approaches [42] propose an embedding using a bag of graphlets (codebook of small graphs, with order 2 or 3), these approaches are still far away from being able to cope with large databases in an efficient way. Dealing with large datasets collections will be another important motivation.

Finally, an extra motivation, from an application point of view, is to develop an application of the proposed techniques through the new technologies and devices, such as mobiles and tablets.

1.5 Objectives

The main objective of this work is to show the performance of structural methods for word spotting in historical handwritten documents. This method is based on graphs in order to add structural information to the word descriptor. Moreover, a graph hashing structure will be presented in order to reduce the time and memory complexity for large document collections. Finally, an application for android devices will be shown as a demonstration of the usability of the proposed approach.

The enumeration of the objectives are:

- To understand some of the algorithms usually used in word spotting.
- To comprehend the state of the art graph representations and matching techniques.
- To define a stable graph representation for handwritten words.
- To propose a robust method for word spotting based on structural representation. Our goal is to propose a method able to deal with the handwriting style of historical documents and to be robust to the variability among instances of the same word.
- To propose a fast graph indexation.
- To evaluate the designed technique.
- To program an android application.

1.6 Requirements, Resources and Viability

As a project focused on research, the requirements are strongly related to the objectives. The objective is not to obtain specific results or a new technique which performs better than the others in the literature but propose an alternative.

The resources used in this project are:

- *Personal computer*: The project has been developed in a personal computer with Windows.
- *MATLAB R2013b*: During this project, multiple scripts and functions have been developed using this language.
- *C compiler*: Some functions have been developed using this language and then called from MATLAB using MEX-files. The objective to develope some functions using another language is to boost the time performance.
- *Image dataset*: the Marriage License Books [36] have been used as a ground truth to test the developed methods. This dataset is available for research purpose and can be downloaded from the Internet.
- *Computation cluster*: Some experiments have been performed using the cluster of the Computer Vision Center (CVC).
- *Qgar software*[1]: This software provides useful tools for Document Analysis purposes.
- *Android device*: Different Android versions have been tested. However we recommend to be newer than Android 4.1.2 Jelly Bean.
- *Android Studio*: A free integrated development environment (IDE) for developing on the Android platform. The programming language that has been used is Java.
- *Version control*: Git is a distributed revision control system that have been used to track the different versions of the code.

Some of the resources needed during this project are easy to acquire, for example the image dataset, the Qgar Software, the Android Studio and Git are free. The Android device that is used during this work is the personal smartphone that can be obtained by a price between 200€ and 300€. The remaining resources were provided by the Computer Vision Center (CVC). Therefore the project is viable.

1.7 Temporal planning

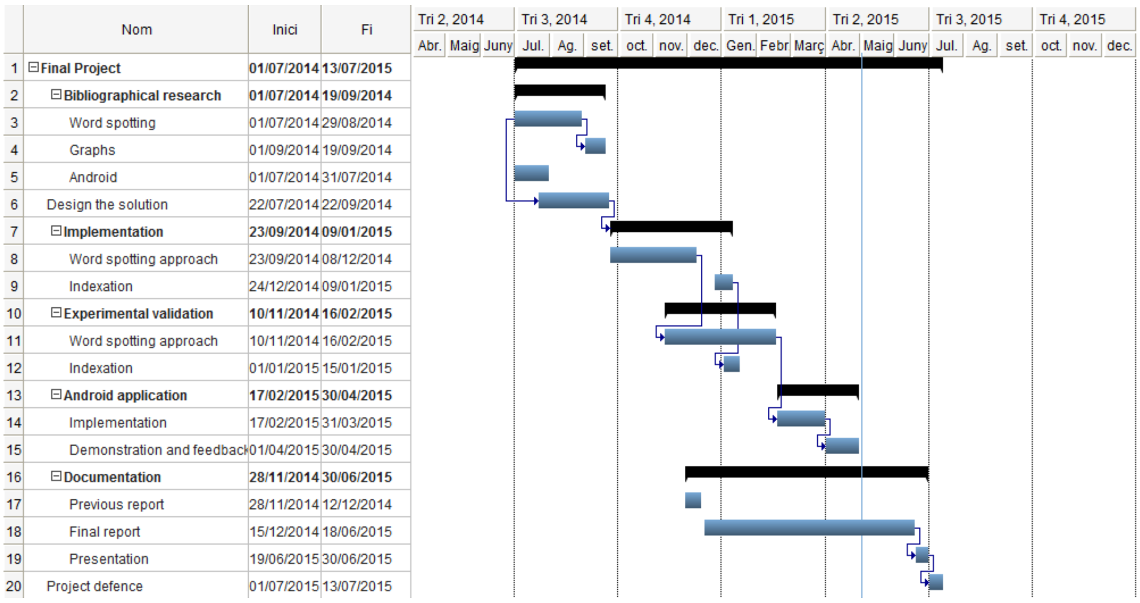
The whole project is divided in different tasks:

- *Bibliographical research*: Read and study articles from the literature to learn and understand the state of the art of word spotting and graph representations. Moreover, learn the common way to implement an Android application.

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

- *Design the solution*: First state the hypothesis to decide how to face the problem and then think possible solutions.
- *Implementation*: Implement the solutions designed in the previous stage.
- *Experimental validation*: Validate all the methods proposed during this project.
- *Android application*: Develop an android application and prepare demonstrations to potential users.
- *Documentation*: Prepare all the documentation. It is composed by the previous and final report and the presentation.
- *Project defence*: Public defence of the project.

Figure 1.2 shows the Gannt diagram representing the planning form the beginning of the project to the defence. This project is expected to be finished in June. However, the planning has changed a lot. At the beginning, this work was meant to be finished in February. Although the planning was being accomplished, we decided to extend it to June in order to develop the Android application and try to publish the research done.



	Nom	Inici	Fi
1	**⊟Final Project**	**01/07/2014**	**13/07/2015**
2	**⊟Bibliographical research**	**01/07/2014**	**19/09/2014**
3	Word spotting	01/07/2014	29/08/2014
4	Graphs	01/09/2014	19/09/2014
5	Android	01/07/2014	31/07/2014
6	Design the solution	22/07/2014	22/09/2014
7	**⊟Implementation**	**23/09/2014**	**09/01/2015**
8	Word spotting approach	23/09/2014	08/12/2014
9	Indexation	24/12/2014	09/01/2015
10	**⊟Experimental validation**	**10/11/2014**	**16/02/2015**
11	Word spotting approach	10/11/2014	16/02/2015
12	Indexation	01/01/2015	15/01/2015
13	**⊟Android application**	**17/02/2015**	**30/04/2015**
14	Implementation	17/02/2015	31/03/2015
15	Demonstration and feedback	01/04/2015	30/04/2015
16	**⊟Documentation**	**28/11/2014**	**30/06/2015**
17	Previous report	28/11/2014	12/12/2014
18	Final report	15/12/2014	18/06/2015
19	Presentation	19/06/2015	30/06/2015
20	Project defence	01/07/2015	13/07/2015

Tri 2, 2014
Tri 3, 2014
Tri 4, 2014
Tri 1, 2015
Tri 2, 2015
Tri 3, 2015
Tri 4, 2015
Abr. Maig Juny Jul. Ag. set. oct. nov. dec. Gen. Febr Març Abr. Maig Juny Jul. Ag. set. oct. nov. dec.

Figure 1.2: Gannt diagram.

1.8 Document structure

This dissertation is organized as follows. Chapter 1 is the introduction, planning and requirements of the project. Chapter 2 is devoted to analyse the state of the art related to word spotting and graph matching. Besides, Chapter 3 explains the developed word spotting approach. Chapter 4 proposes a graph indexation framework capable to deal with subgraph matching in large graphs. Then, Chapter 5 explains the validations that have been done during the project for the previous approaches. Chapter 6 explains the developed android application. Finally, Chapter 7 draws the conclusion and the future work lines are proposed. Before the bibliography is drawn, the list of publications in international conferences and workshops is presented. All the publications are derived from the research developed during the elaboration of this project.

Chapter 2

State of the art

Graph-based representations are experiencing a growing usage in visual recognition and retrieval due to their representational power in front of classical appearance-based representations in terms of feature vectors. In this chapter, an introduction to graph matching is presented. Afterwards, in order to compute the edition cost between two graphs, the graph edit distance is explained. However, a suboptimal approximation will be then introduced to avoid the high cost in terms of time of the first algorithm. Finally, the state of the art of word spotting techniques is explained, focusing on the different strategies and the different kind of descriptors.

2.1 Graph Matching

2.1.1 Introduction

Dealing with graphs in pattern recognition problems has the drawback that it is necessary to treat with inexact graph matching techniques due to the deformations which are present in the real world, for instance, in handwritten text. These techniques will make graphs to allow deformations and make them a useful tool for this kind of problems. Knowing the different deformations and noise present in the images, allows to define a distance and therefore, have a value representing the similarity between different graphs.

Graph matching is one of the most important challenges of graph processing. Generally speaking, the problem consists in finding the best correspondence between the sets of vertices of two graphs preserving the underlying structures and the corresponding labels and attributes. Graph matching plays an important role in many applications of computer vision and pattern recognition, and several algorithms have been proposed in the literature [7].

The following sections explain the state of the art of the inexact graph matching techniques.

2.1.2 Graph edit distance

Graph edit distance [41] is the process to evaluate the similarity of two different graphs. One option is to create a distance function following the idea of string edit distance but taking more dimensions. This method is called graph edit distance.

The idea of the graph edit distance algorithm is to compute the different edition operations to transform the source graph into the target one. Every step will add some cost to the final edition distance. This set of operations is not unique, because of that, the minimum cost path needs to be computed.

The different cost operations used in this project are:

- Insertions: Of a node or an edge. Denoted by $(\varepsilon \to v)$ where v is a node and ε is the empty element.
- Deletion: Of a node or an edge. Denoted by $(u \to \varepsilon)$ where u is a node and ε is the empty element.
- Substitution: Of a node or an edge. Denoted by $(u \to v)$ where u and v are two nodes.

The formal definition of this method is:

Definition 2.1.1 (Graph Edit Distance)**.** *Let $g_1 = (V_1, E_1, \mu_1, \nu_1)$ be the source and $g_2 = (V_2, E_2, \mu_2, \nu_2)$ the target graphs. The graph edit distance between g_1 and g_2 is defined by*

$$d(g_1, g_2) = \min_{(e_1,\dots,e_k) \in \Upsilon(g_1,g_2)} \sum_{i=1}^{k} c(e_i).$$

where $\Upsilon(g_1, g_2)$ denotes the set of edit paths transforming g_1 into g_2, and c denotes the cost function measuring the strength $c(e_i)$ of the edit operation e_i.

Algorithm 2.1 is the A^*-based method for optimal graph edit distance computation. Note that the edit operations of the edges are implied by edit operations of their adjacent nodes. As heuristic functions, $g(p)$ denotes the optimal path to that edition step and $h(p)$ is used to denote the estimated cost from that step to the final edit operation

Although the method finds an optimal edit path between two graphs, the computational complexity of the edit distance algorithm is exponential in the number of nodes of the involved graphs.

In the next section a suboptimal solution to cope with the complexity problem of this method is described.

2.1.3 Suboptimal graph edit distance

A suboptimal approximation to graph edit distance called bipartite graph matching was proposed by Riesen *et al.* [33]. The algorithm is based on the assignment

Algorithm 2.1 Graph edit distance algorithm

```
Input: Non-empty graphs $g_1 = (V_1, E_1, \mu_1, \nu_1)$ and $g_2 = (V_2, E_2, \mu_2, \nu_2)$, where
$V_1 = \{u_1, \ldots, u_{V_1}\}$ and $V_2 = \{v_1, \ldots, v_{|V_2|}\}$.
Output: A minimum cost edit path from $g_1$ to $g_2$ e.g. $p_{min} = \{u_1 \to v_3, u_2 \to \varepsilon, \ldots, \varepsilon \to v_2\}$
Initialize $OPEN$ to the empty set
for each node $w \in V_2$ do
    Insert the substitution $\{u_1 \to w\}$ into $OPEN$
Insert the deletion $\{u_1 \to \varepsilon\}$ into $OPEN$
loop
    Remove $p_{min} = argmin_{p \in OPEN}\{g(p) + h(p)\}$ from $OPEN$
    if $p_{min}$ is a complete edit path then
        return $p_{min}$ as the solution
    else
        Let $p_{min} = \{u_1 \to v_{i1}, \ldots, u_k \to v_{ik}\}$
        if $k < |V_1|$ then
            for each $w \in V_2 \setminus \{v_{i1}, \ldots, v_{ik}\}$ do
                Insert $p_{min} \cup \{u_{k+1 \to w}\}$ into OPEN
            Insert $p_{min} \cup \{u_{k+1 \to \varepsilon}\}$ into OPEN
        else
            Insert $p_{min} \cup \bigcup_{w \in V_2 \setminus \{v_{i1}, \ldots, v_{ik}\}} \{\varepsilon \to w\}$ into $OPEN$
```

problem solution. In this section the assignment problem is defined and a solution is presented. Finally, this solution is used in order to find the suboptimal graph edit distance.

The assignment Problem

Definition 2.1.2 (The Assignment Problem)**.** *Let us assume that there are two sets A and B together with an $n \times n$ cost matrix C of real numbers, where $|A| = |B| = n$. The matrix of elements $C_{i,j}$ corresponds to the costs of assigning the i-th element of A to the j-th element of B. The assignment problem can be stated as finding a permutation $p = p_1, \ldots, p_n$ of the integers $1, 2, \ldots, n$ that minimizes $\sum_{i=1}^{n} C_{i,p_i}$.*

This problem can be reformulated as finding an optimal matching in a complete bipartite graph and is therefore also referred to as bipartite graph matching problem. Although solving this problem with a brute force algorithm has an exponential complexity, there exists a method which is known as Munkres' algorithm [26] that solves the assignment problem in polynomial time (Alg. 2.2).

In summary, algorithm 2.2 has as an input the assignment cost matrix C defined in 2.1.2 and the output corresponds to the optimal permutation (assignment pairs with minimum cost).

Note that in the description of the method, there are markers for rows or columns of the matrix C, covered or uncovered, and markers for the zero elements, starred or primed. The solution will be provided by the starred zeros at the end of the algorithm.

Graph edit distance computation by means of Munkres' algorithm

Assuming the source graph $g_1 = (V_1, E_1, \mu_1, v_1)$ and the target graph $g_2 = (V_2, E_2, \mu_2, v_2)$ of equal size, i.e. $|V_1| = |V_2|$, one can use algorithm 2.2 in order to map the nodes of V_1 to the nodes of V_2 with the optimal node substitutions costs.

The constrain that both graphs to be matched are of equal size is too restrictive. However, Munkres' algorithm can be applied to rectangular matrices. This solution first finds the $min\{|V_1|, |V_2|\}$ node substitutions which minimize the total costs. Then the costs of $max\{0, |V_1| - |V_2|\}$ node deletions and $max\{0, |V_2| - |V_1|\}$ node insertions are added to the minimum cost node assignment such that all nodes of both graphs are processed.

In this project, the extension proposed by Riesen *et al.* is used. It uses a new cost matrix C that allows insertion or deletions to occur not only in the larger, but also in the smaller of the two graphs under consideration.

Definition 2.1.3 (Cost Matrix). *Let $g_1 = (V_1, E_1, \mu_1, v_1)$ be the source and $g_2 = (V_2, E_2, \mu_2, v_2)$ be the target graphs with $V_1 = \{u_1, \dots, u_n\}$ and $V_2 = \{v_1, \dots, v_m\}$, respectively. The cost matrix C is defined as*

$$C = \left[\begin{array}{cccc|cccc}
c_{1,1} & c_{1,2} & \cdots & c_{1,m} & c_{1,\varepsilon} & \infty & \cdots & \infty \\
c_{2,1} & c_{2,2} & \cdots & c_{2,m} & \infty & c_{2,\varepsilon} & \ddots & \vdots \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \infty \\
c_{n,1} & c_{n,2} & \cdots & c_{n,m} & \infty & \cdots & \infty & c_{n,\varepsilon} \\
\hline
c_{\varepsilon,1} & \infty & \cdots & \infty & 0 & 0 & \cdots & 0 \\
\infty & c_{\varepsilon,2} & \ddots & \vdots & 0 & 0 & \ddots & \vdots \\
\vdots & \ddots & \ddots & \infty & \vdots & \ddots & \ddots & 0 \\
\infty & \cdots & \infty & c_{\varepsilon,n} & 0 & \cdots & 0 & 0
\end{array}\right]$$

where $c_{i,j}$ denotes the cost of a node substitution, $c_{i,\varepsilon}$ denotes the cost of a node deletion $c(u_i \to \varepsilon)$, and $c_{\varepsilon,j}$ denotes the cost of a node insertion $c(\varepsilon \to v_j)$.

Studying the cost matrix definition, it is obvious that the first quadrant of our matrix represents all possible node substitutions, the diagonal of the upper right quadrant represents the cost of all possible node deletions, and the diagonal of the lower left represent the costs of all possible node insertions. Each node of the graph can be deleted or inserted only once, as a consequence, any non-diagonal element of the right-upper and left-lower part is set to ∞. The bottom right corner represents the substitution $\varepsilon \to \varepsilon$ so it is set to zero.

Until now the proposed method does not consider the edges of the graphs. To perform a better approximation, a modification of the cost matrix defined at 2.1.3 has to be done. For all entries $c_{i,j}$ representing the cost of a node substitution

Algorithm 2.2 Munkres' algorithm for the assignment problem

```
Input: A cost matrix $C$ with dimensionality $n$.
Output: The minimum cost node or edge assignment.
for each row $r$ in $C$ do
    Substract its smallest element from every element in $r$
for each column $c$ in $C$ do
    Substract its smallest element from every element in $c$
for all zeros $z_i$ in $C$ do
    Mark $z_i$ with a star if there is no starred zero in its row or column.
STEP 1:
for each column containing a starred zero do
    Cover this column.
if $n$ columns are covered then
    GOTO Done
else
    GOTO STEP 2
STEP 2:
if $C$ contains an uncovered zero $Z_0$ then
    Find an arbitrary uncovered zero $Z_0$ and prime it
    if There is no starred zero in the row of $Z_0$ then
        GOTO STEP 3
    else
        Cover this row, and uncover the column containing the starred zero
        GOTO STEP 2
else
    Save the smallest uncovered element $e_{min}$
    GOTO STEP 4
STEP 3: Construct a series $S$ of alternating primed and starred zeros as follows:
Insert $Z_0$ into S
while In the column of $Z_0$ exists a starred zero $Z_1$ do
    Insert $Z_1$ into $S$
    Replace $Z_0$ with the primed zero in the row of $Z_1$. Insert $Z_0$ into $S$
Unstar each starred zero in $S$ and replace all primes with stars.
Erase all other primes and uncover every line in C
GOTO STEP 1
STEP 4:
Add $e_{min}$ to every element in covered rows.
Subtract it from every element in Uncovered columns.
GOTO STEP 2
DONE:
Assignment pairs are indicated by the positions of starred zeros in the cost
matrix.
```

$c(u_i \to v_j)$, we have to add the minimum sum of edge edit operation costs implied by node substitution $u_i \to v_i$. Considering the two sets of adjacent edges E_{u_i} and E_{v_i} of the nodes u_i and v_i respectively, a cost matrix can be defined similarly to the definition 2.1.3 and an optimal assignment of the elements E_{u_i} to the elements E_{v_i} according to the algorithm 2.2 can be performed. This procedure leads to the minimum sum of edge edit costs implied by a given node substitution. Clearly, for the entries $c_{i,\varepsilon}$ and $c_{\varepsilon,j}$ the cost of deletion or insertion respectively of all the edges is added.

Note that the solution provided by this method is suboptimal for the graph edit distance problem. This is due to the fact that the edit operation for each node is considered individually. The approximate edit distance values obtained by this procedure are equal to, or larger than the exact distance values, since this suboptimal approach finds an optimal solution in a subspace of the complete search space.

2.2 Word Spotting

2.2.1 Introduction

Lots of techniques have been proposed to deal with word spotting in the literature. Word spotting methods can be divided in different categories. First of all, Section 2.2.2 explains the strategies that can be used to apply this technique. Secondly, the representation of the image is defined. Section 2.2.3 explains the state of the art for statistical descriptors which are the most used in the literature in the kind of problems faced during this work.

2.2.2 Strategy

In the literature there are two strategies to extract the features: segmentation-based and segmentation-free approaches. The strategies are divided whether they are applied to each word or to the whole page.

- *Segmentation-based approaches* [31] require each document image to be segmented at word level, taking advantage of the knowledge of the document structure. The main problem of these approaches is that word classification is strongly influenced by over or under segmentations. Another problem is the noise that can appear from the contiguous words.

- *Segmentation-free approaches* [14,23,39] are not affected by this problem. The document image is usually divided into patches and the query word image is classified regarding each patch (e.g. using a sliding window). Since all the regions are compared, these methods can be computationally costly in terms of time. Therefore, the best option depends on the application.

There is not a best option. Because both options have their pros and cons, the strategy depends on the application. The proposed work for word spotting will be a segmentation-based approach.

2.2.3 Descriptors

In word spotting a key decision is to choose the feature representation of the images. Word Spotting usually uses statistical descriptors in order to achieve the goal of locating words in images. Two families can be differentiated, namely *appearance-based* and *object-based* primitives.

Appearance-based methods extract descriptive features from all the image pixels in terms of the photometry. Different arrangements can be considered when analysing the pixels, so an implicit spatial information is encoded. A typical implementation is inspired by spatial pyramid methods where the descriptors are extracted on a regular grid and different scales.

Almazan et. al [2] divide the images in equal-sized cells. For each one a HOG descriptors is computed combined with an exemplar-SVM framework. Gatos *et. al* [14] perform a template matching of block-based images descriptors. Rothacker *et. al* [38] localize the descriptors on a regular grid and uniform scale. The feature vector is constructed using a dense SIFT descriptor. Almazan *et. al* [3] adopt the Fisher Vector (FV) representation computed over SIFT descriptors densely from the word image. Other methods use column-wise feature descriptors. Frinken *et. al* [13] compute global and local features in each column. Rodriguez-Serrano *et. al* [35] combine Marti and Bunke [24], Zoning [6] and LGH [34] features. These methods train first the models, using the information of the entire image. Once the model is trained, the images are compared and the candidates are ranked using a similarity measure, commonly a Dynamic time Warping (DTW) or Hidden Markov Model (HMM-based) similarity.

Object-based methods segment local interest points from the image, and extract features on each individual object. As in other pattern recognition domain, typical interest points in images are key points like corners or crossings, edges, skeletons or regions.

The Blurred Shape Model (BSM) descriptor [10] defines their interest points using high density regions. In the Shape Context descriptor proposed by Belongie *et. al* [5] for handwritten characters, the interest points are the pixels of the contour. Zernike moments [18] construct descriptors using a set of complex polynomials, which form a complete orthogonal set over the interior of the unit circle. Traditional Zoning methods [15] divide the image in a grid, and each cell contributes with statistics of its content to a position of a feature vector.

Chapter 3

Word Spotting framework

In this chapter our structural approach will be presented step by step. The word spotting technique proposed in this project has two important parts. First of all, it is needed to transform the original image of the word into a graph. Secondly, the distance between a query and the set of precomputed graphs is calculated using the bipartite graph matching explained in the Chapter 2. These two parts define our word spotting framework.

3.1 Graph definition

A good representation of the word image is very important to be able to codify the most characteristic features of each word in order to differentiate the classes in the graph matching process. It is also important to keep the graphs stable against the deformations of handwriting but robust enough to represent the topology of words in terms of their constituent primitives. This project proposes a representation based on graphemes graphs of the handwritten language.

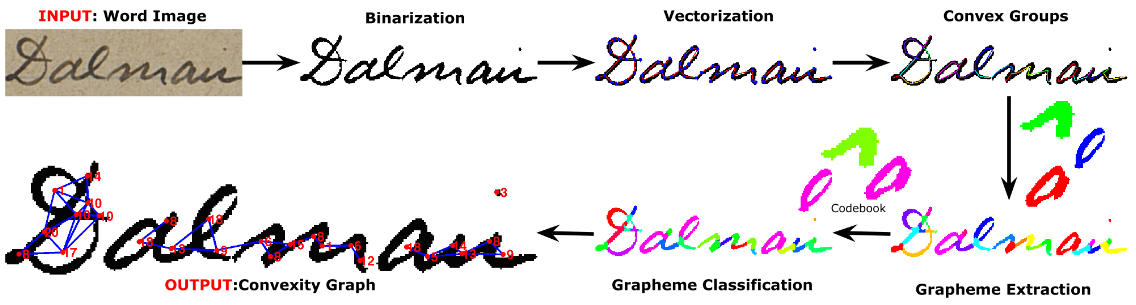


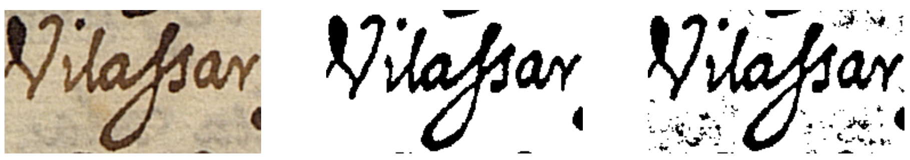
Figure 3.1: Graph definition process for the input word Dalmau

Figure 3.1 shows an overview of the graph creation given a word image. Let us describe now all the steps in detail.

3.1.1 Binarization

The first step of the graph definition is the binarization of the image. This part of the graph definition is very important due to the noise present in the images. A bad binarization process will introduce lots of noise to our image which will result in a no desired graph.

Two different binarization methods have been explored. On one hand, a global thresholding method has been tried (Otsu [28]) which takes into consideration all the image pixels together. However, some parts will be wrongly binarized due to a change of noise in the background or variations in the grey levels in different sections of the image. On the other hand, a local thresholding method (Niblack [27]) has been applied. This algorithm, performs a good binarization around the word, however the performance decreases in the parts of the background where the noise becomes more relevant. Figure 3.2 shows a comparison between both methods.



(a) Original Image (b) Otsu's method (c) Niblack's method

Figure 3.2: Binarization examples.

Finally, the chosen binarization is the Otsu's method. This method starts from a grey level image and ends in a binary image. The algorithm assumes that the image contains two classes of pixels and then calculates the optimum threshold separating both classes (black and white). The Niblack's method produces too much noise in the background of the image that will decrease the performance of our approach. This method, calculates a pixel-wise threshold using a rectangular sliding window on a grey level image. Modifying the parameters, for example, changing the windows size, the results will improve a lot and will be very similar to Otsu's one (see Fig. 3.3). However, taking into account the size of our image, we are losing the local threshold that characterizes this method.

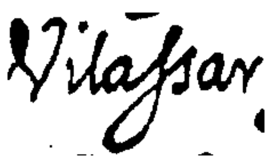


Figure 3.3: Niblack's method with bigger window.

3.1.2 Grapheme decomposition

The objective of this section, is to choose the elements of the words that will define our handwritten text. This elements, roughly speaking are called graphemes.

Definition 3.1.1 (Grapheme)**.** *Grapheme or Glyph is the smallest unit used in describing the writing system of a language.*

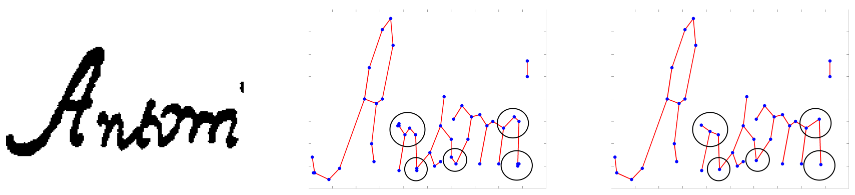
In handwritten text, it is very important to define how to determine the graphemes of the word. Different techniques has been used in the literature such as Daher *et al.* [8] who uses a decomposition based on the detection of the medial axis and chooses as decomposition points those of minimum thickness, the junction points and those previously visited.

In this work, we propose a novel glyph extraction based on the convexities found in the vectorization graph of the image.

Vectorization

The vectorization process consists in a polygonal approximation of the skeleton of the strokes. The vectors compounding the polygonal approximation are represented with a graph. For this project, the vectorization proposed by Rosin and West [37] have been used. This vectorization is available in the *Qgar Software*.

Once the first vectorization process has finished, we have a first representation of the image. However, lots of nodes may be generated together and they do not give information. These amount of nodes in a small piece of the word, which are connected with an edge, can be considered as repeated vertices. Therefore, they are not necessary and they can be represented as only one node. A post-process has been used to join all the repeated vertices. Figure 3.4 shows all the vectorization process. The places where the nodes have been joined appear with circle in the image.



(a) Binary Image (b) Roslin-West Vectorization (c) Post-Processed Vectorization

Figure 3.4: Vectorization example.

Convexity

As it has been stated, the graphemes are extracted from the convex groups of nodes that are found in the vectorization. Let us define what is a convexity given a set of points or nodes.

Definition 3.1.2 (Convex)**.** *An object is convex if for every pair of points within the object, every point on the straight line segment that joins the pair of points is also within the object.*

To extract the convexities, different steps are performed on the vectorization graph. First of all, a pre-process is done in order to separate every branch of the graph. Afterwards, all the possible convex groups for every branch are checked. We need to test every line between the nodes of the groups we are evaluating and save the larger group possible.

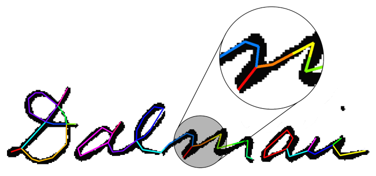


Figure 3.5: Convexities.

Once this process is finished we have several convex groups. However, not all of them are interesting for our purpose, for instance, convexities which are completely contained inside other groups are directly deleted. Finally, every pair of convex groups with intersection, are separated by the middle point of their common part. A new node is created if it is needed. This last step is not performed if one of the groups is a loop. The final result of this process is shown in Figure 3.5.

Once the convexities of the graph are obtained, the graphemes can be extracted. First of all, we need to perform dilatations from the convex groups, afterwards, the logic operation AND between the binarized image and the dilated convexity will give us the graphemes.

3.1.3 Codebook

We represent each grapheme with a codeword obtained using a shape descriptor (See section 5.1.2 for more details of the descriptor used in this work). This codeword is classified using a set of precomputed clusters.

In order to compute the clusters, lots of codewords have been extracted and stored in a huge dataset. After that, a k-means has been performed to obtain the different equivalence classes. The number of classes has been set empirically to 20. We will label the set of clusters as codebook, and every codeword will be classified into this codebook.

Table 3.1 shows the classification of the graphemes in clusters. First of all, notice that the cluster 11 corresponds mostly to noise or dots like the one of the “i”. Another issue to notice is the similarity between some clusters like 1, 5 and 9. Their graphemes are very similar and some of them are confused. However, during the matching process, the distance between their centroids is used to weight these similarities.

Finally, the graph is generated. All the graphemes are converted to vertices with the codebook index as an attribute. Two vertices have an edge between them if their convex groups are touching by at least one node. The number of vectorization vertices in common between theses two convex groups, are the attribute of the edge.

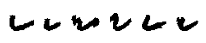
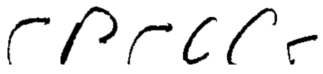
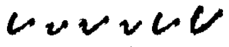
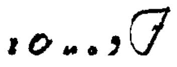
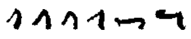
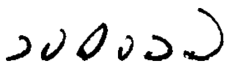
Cluster	Examples	Cluster	Examples
1		2	
3		4	
5		6	
7		8	
9		10	
11		12	
13		14	
15		16	
17		18	
19		20	

Table 3.1: Qualitative results of grapheme classification.

3.2 Graph matching for Word Spotting

Once the graphs have been computed for all the database, a comparison between them is needed. For this purpose, the bipartite graph edit distance, which has been explained in Chapter 2, is used. Figure 3.6 shows the pipeline of the proposed Word Spotting framework and illustrates the result desired for this Section.

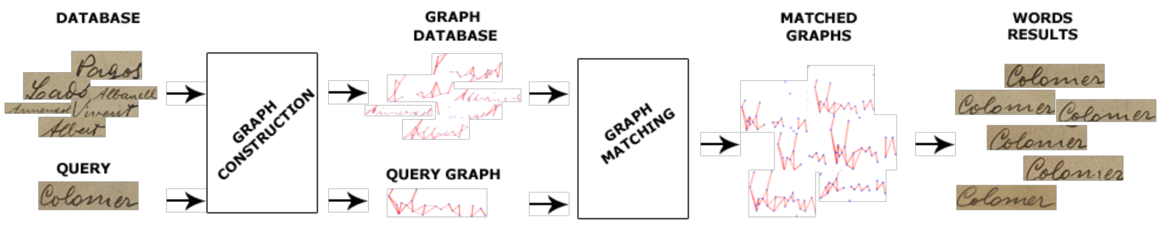


Figure 3.6: Graph matching for Word Spotting scheme.

All the values of the weights and the constant values have been set empirically and are application dependant. All the cost computations are scaled into the range $(0, 1)$, and the sum of the weights involved in the partial computation of any cost is 1.

Firstly, the definitions of the edit operations are needed. Let $G_w = (V_w, E_w, L_{V_w}, L_{E_w})$ be the graph of the query word and $G_t = (V_t, E_t, L_{V_t}, L_{E_t})$ be the target graph with $V_w = \{u_1, \ldots, u_n\}$ and $V_t = \{v_1, \ldots, v_m\}$, respectively.

3.2.1 Insertion and Deletion Costs

Two different operations have to be considered. However, despite the idea of these two steps is different, from the point of view of the cost computation for word spotting purpose both operations can be considered symmetric (both can be seen as deletions in one graph or the other). This costs will define two matrix that are part of the cost matrix C defined at 2.1.3.

$$C_{insertion} = \begin{bmatrix} c_{\varepsilon,1} & \infty & \cdots & \infty \\ \infty & c_{\varepsilon,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \infty \\ \infty & \cdots & \infty & c_{\varepsilon,n} \end{bmatrix} \quad C_{deletion} = \begin{bmatrix} c_{1,\varepsilon} & \infty & \cdots & \infty \\ \infty & c_{2,\varepsilon} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \infty \\ \infty & \cdots & \infty & c_{n,\varepsilon} \end{bmatrix}$$

Intuitively, the cost is computed in terms of the local configuration of the node defined by the incident edges. If the node is strongly connected, the cost will be higher than for example a simple node that appears disconnected. For these costs, the attribute of the node will not be taken into consideration and only a baseline cost for the edit operation will be used. Thus, the insertion and deletion cost have three terms defined as:

$$c(\varepsilon \to v_j) = c(u_i \to \varepsilon) = we_0 C_{weightEdges} + we_1 C_{edges} + we_2 t_{vertices}$$

where,

- we_i are weighting factors.
- $C_{weightEdges}$ is the sum of the attributes of the edges incident to the node being deleted. It indicates how much the node is sharing part of its grapheme with the neighbouring ones.
- C_{edges} is a measure of the density of the node computed as the ratio between the number of incident edges and the total number of graph nodes.
- $t_{vertices}$ is a constant value experimentally set as a baseline cost for the edit operation.

3.2.2 Substitution Cost

The rest of the matrix 2.1.3 is defined as follows.

$$C_{substitution} = \begin{bmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,m} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n,1} & c_{n,2} & \cdots & c_{n,m} \end{bmatrix}$$

The different costs of this matrix are the substitutions between nodes of both graphs. This cost is computed in terms of the position of the nodes, their label according to the codebook and the similarity of the local structure. Formally the substitution cost is defined as:

$$c(u_i \to v_j) = wn_0 D_{i,j} + wn_1 C_{desc} + wn_2 C_{local\_structure}$$

where,

- wn_i are different weights.
- $D_{i,j}$ is the euclidean distance between the nodes u_i and v_j position. This distance is normalized by the maximum node position of the both graphs.
- C_{desc} is the L_1 distance between the corresponding shape descriptor of the node graphemes. The value C_{desc} is computed with the distance between the centroids of the k-means clustering when the codebook is extracted.
- $C_{local\_structure}$ is the edit operation cost on the incident edges. This cost is explained with more detail below these lines.

The $C_{local\_structure}$ cost is computed using the Bipartite Graph Matching algorithm. In order to apply this algorithm, the edges have to be considered as the vertices where we have to apply the edit operations. Hence, a matrix of edit costs between the adjacent edges of both nodes must be defined. Denoting this matrix as Ce, it has the same structure as C defined at the Definition 2.1.3.

In this case, the cost of edge insertion and deletion is a constant named t_{edges}. The substitution costs are computed in terms of the edge attributes, i.e. weight, angle and length for the both edges to substitute.

$$c(e_i \to f_j) = we_0 C_{weight} + we_1 C_{angle} + we_2 C_{length}$$

where,

- we_i are weighting factors.
- C_{weight} is the difference between the weight of both edges.
- C_{angle} is the angle between them.
- C_{length} is equivalent to $1 - e_{short}/e_{long}$ where e_{short} denotes the length of the shorter edge and e_{long} the length of the longer one.

Chapter 4

Graph Indexing

Retrieving a query graph from a large dataset of graphs has the drawback of the high computational complexity required to compare the query and the target graphs. Despite the existence of many suboptimal methods for graph matching, the scalability to large scale scenarios is still a challenge. In this chapter, we propose a fast indexation formalism for graph retrieval. A binary embedding is defined as hashing keys for graph nodes. Combining this indexation approach with the word spotting framework explained in Chapter 3 will allow to compare the structural word spotting approach proposed with the statistical ones in terms of time complexity.

4.1 Introduction

In this work, we have proposed a word spotting approach for ancient documents able to deal with noise and deformations. However, the proposed framework asks for cropped images of the query word. From another point of view, we can imagine our collection of documents as a huge non-connected graph where we want to perform an inexact (sub)graph isomorphism in order to find the candidates to be similar to our query. However, it is a known NP-Complete problem and it will be unfeasible in terms of time complexity for large graphs.

Definition 4.1.1 (Subgraph Isomorphism). *Let $H = (V_H, E_H)$ and $G = (V, E)$ be graphs. A subgraph isomarphism from H to G is a function $f : V_H \to V$ such that if $(u, v) \in E_H$, then $(f(u), f(v)) \in E$.*

In this chapter, we propose a graph hashing approach inspired by the ideas of binary encoding for content-based image retrieval (CBIR). We propose to extend the attributes associated to graph nodes by an embedding function describing the local context of the node with a vector. The local context is the structure of the connected (sub)graph around this node. From now on, we will denote the local context as the (sub)graph centred at the node of radius k, where the radius means the length of the path to the farthest node. Figure 4.1 shows the local context of a graph with $k = 3$. This vector of attributes is converted to a binary code applying a binary-valued hash function.

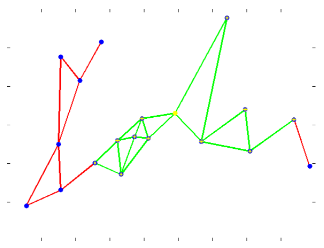


Figure 4.1: Local Context (k=3).

Therefore, graph retrieval is formulated in terms of finding target (sub)graphs in the database whose nodes have a small Hamming distance from the query nodes. This indexation based on binary codes can be easily computed with bitwise logical operators (XOR) taking advantage of the hardware benefits. In spite of the amount of comparison needed, these kind of operations are very fast.

Although this project is focused on word spotting on historical documents, the proposed indexation can be applied to other fields. Through the indexation we will have a set of candidates to perform a more accurate matching. In our case, the candidates will be words where we will perform the approach proposed in Chapter 3.

4.2 Binary Embedding Formulation

In this section, we describe the main contribution of this chapter consisting in the encoding of the local topological context of graph nodes by binary vectors. It allows to construct a fast indexing scheme in terms of the Hamming distance.

An embedding is an injective function that transforms an object X into another object Y in a certain space in such a way that its connectivity or algebraic properties are preserved. X is said to be embedded into Y. In other words, an embedding is the mapping of one set into another.

This section presents the two necessary steps to perform the embedding, first of all the embedding function and afterwards the indexing process using the generated vector.

4.2.1 Topological node features

An embedding function for a set of graphs can be defined as $\phi : \mathcal{G} \rightarrow \mathbb{R}^n$ where this function transforms a graph $G \in \mathcal{G}$ to an n-dimensional feature vector. Hence, the distance between two graphs can be computed by a distance in a metric space, and the problem of graph classification can be solved by a statistical learning approach. However, we propose an embedding function for each node which, as has been stated, codifies the local structure. For this purpose the Morgan index will be introduced.

The *Morgan index* M is a node feature, originally used to characterize chemical structures [25], that computes the node context in terms of its local context. This index is iteratively computed for each node $v \in V$ as follows:

$$M(v,k) = \begin{cases} 1 & \text{if } k = 0; \\ \sum_u M(u,k-1) & \text{otherwise.} \end{cases}$$

where u is a vertex adjacent to v. The Morgan index of order k associated to a given node v is denoted as $M(v,k)$ and counts the number of paths of length k incident to node v and starting somewhere in the graph. In order to compute the Morgan index, an interesting property of the adjacency matrix can be used. Let A be the adjacency matrix of any graph G then, the (i,j)th entry of the matrix A^n denotes the number of paths of length n from the node v_j to the node v_i. Therefore, the Morgan index of order k for a node v_i is equivalent to the sum of the cells of the i-th row of the matrix A^k, formally $M(v_i,k) = \sum_j A^k(i,j),\ j = 1 \ldots |V|$.

Taking advantage of Morgan index properties and inspired by the topological node features proposed by *Dahm et al.* [9] we define *local context* of a node v as a node embedding function computed in terms of the topological information of a subgraph centred at v and radius k. From now on, the local context is defined in terms of the Morgan index as follows.

Definition 4.2.1 (Local context)**.** *Let $M_l(v,k)$ be the Morgan index of node v, order k and label l which counts the number of paths of length k incident to node v and starting at nodes labelled as l. Following the introduced notation, the* local context *of a node v is defined as:*

$$\nu(v) = [M_{l_1}(v,1), \ldots, M_{l_1}(v,K), M_{l_2}(v,1), \ldots, M_{l_2}(v,K), \ldots, M_{l_{|\Sigma_V|}}(v,K)],$$

where K is the maximum length of the paths incident in v that is considered. Thus, every graph node is attributed by a $K \cdot |\Sigma_V|$ feature vector characterizing the number of paths incident at v of lengths up to K and starting at nodes for all the possible labels in Σ_V.

The context vector $\nu(v)$ is converted to a binary code $\hat{\nu}(v) = \{0,1\}^{K \cdot |\Sigma_V|}$ in terms of a list of corresponding threshold values T_i for each vector position. These values are application dependent, for instance can be set to the mean of all the vectors for each dimension.

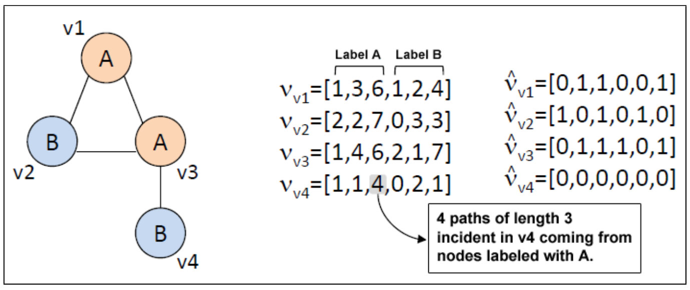

v1
A
B
A
v2
v3
B
v4
Label A
Label B
ν_{v1}=[1,3,6,1,2,4]
ν_{v2}=[2,2,7,0,3,3]
ν_{v3}=[1,4,6,2,1,7]
ν_{v4}=[1,1,4,0,2,1]
$\hat{\nu}_{v1}$=[0,1,1,0,0,1]
$\hat{\nu}_{v2}$=[1,0,1,0,1,0]
$\hat{\nu}_{v3}$=[0,1,1,1,0,1]
$\hat{\nu}_{v4}$=[0,0,0,0,0,0]
4 paths of length 3 incident in v4 coming from nodes labeled with A.

Figure 4.2: Example of the binary code computation from a graph.

Figure 4.2 illustrates the computation of the binary codes. In this example, the codes associated to nodes have length 6 ($K = 3$ and $|\Sigma_V| = 2$). The threshold value is set to the mean of each $M_l(v,k)$.

4.2.2 Indexing

Once the embedding for every node has been computed, an indexation is needed. The indexation proposed in this work, is based on *focused retrieval*, specifically *focused graph retrieval* which can be defined as:

Definition 4.2.2 (Focused Graph Retrieval)**.** *Let G_q be a query graph and $\{G_1, \ldots, G_T\}$ a database of graphs. The problem consists in finding inexact subgraph matchings between the query and the target graphs, in other words, it is defined as finding the subgraphs of G_i similar to G_q.*

The proposed approach can be understood in terms of a visual retrieval application as a pre-process which retrieves the candidate regions of the graph where the query image is likely to appear. However, this process does not assure that the retrieved region contains the query image.

The objective of this indexation is to use the binary embedding of nodes context in an inverted file indexing structure. An inverted file indexing stores a mapping from content to its location in the database. The mapping is defined as a lookup table $H : \{0,1\}^b \rightarrow \{v_i\}_{v_i \in V}$ that indexes a binary vector of length b and return a list of nodes with the same binary code, in other words, the local context of those nodes is similar to the nodes of the query graph.

Finally, a last step is needed to change the inverted file indexing into the subgraph matching process desired. Through the lookup table H we achieve the retrieval of individual nodes, therefore, it is needed to implement a node verification to decide the matching. The lookup table H can be seen as a hashing function that instead of returning the similar nodes of the database which are similar to the input binary code, returns subgraphs where these target nodes appear. In order to do so, a *partition* P of a graph G can be defined as a decomposition of the graph in n small subgraphs, $P(G) = \{g_1, \ldots, g_n\}$, where $g_i \subseteq G$.

Definition 4.2.3 (Indexation Function)**.** *Let G_q be a query graph and $\{G_1, \ldots, G_T\}$ be a database of graphs. The indexation function $H : V_q \rightarrow S$ where V_q is the set of nodes of G_q and S the set of subgraphs of $\{G_1, \ldots, G_T\}$. For each node of the query graph $v \in V_q$, the indexation function H returns the subgraphs of the database, after a partition has been previously defined, containing this vertex $H(v) = \{g_i\}$, where g_i is a subgraph of one of the target graphs $\{G_1, \ldots, G_T\}$ and $i = 1, \ldots N$ where N is the number of hashed subgraphs.*

The definition of the partition under which the database of graphs is decomposed in small graphs is application dependent. The subgraphs g_i can be seen as voting bins, according to a Hough-based principle. Thus, the final result consists of the subgraphs receiving a high number of votes.

In this project, H is defined as the Hamming distance between the binary codes of the nodes. Computing the Hamming distance between two vectors consists in computing the XOR and counting the number of 1's in the resulting vector. This computation can be computed very fast on modern CPU's, with logic operations being part of the instruction set. A fast hashing process like Locality Sensitive Hashing (LSH) [16] can be added to speed up the indexation. Afterwards, those with small Hamming distance vote in the pre-defined voting bins.

Chapter 5

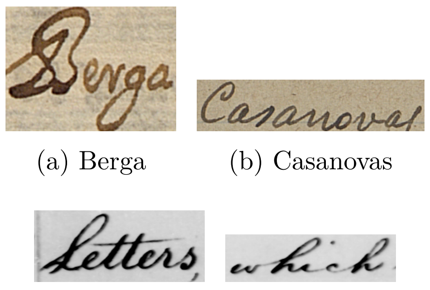
Experiments

Different experiments have been done in order to evaluate the performance of the different techniques proposed in this work. First of all, the Word Spotting approach is tested against different methods in the literature. Finally, the indexation proposed in the Chapter 4 is used to find the candidates in the whole page graph and avoid unnecessary comparisons.

5.1 Experimental setup

For this project, we assume that the words are already segmented (Fig. 5.1), since word segmentation in old handwritten documents is out of scope of this work [22].

There are different datasets available in the literature but the one that has been used for testing our approach is the *Esposalles Database* [36].


(a) Berga (b) Casanovas

(c) Letters (d) which

Figure 5.1: Word examples

5.1.1 Dataset

We have used the Marriage Licenses Books conserved at the Archives of the Cathedral of Barcelona. These manuscripts, called *Llibre d'Esposalles* [36], consist of 244 books written between 1451 and 1905, and include information of approximately 550,000 marriages celebrated in over 250 parishes (Fig. 5.2).

Each marriage record contains information about the couple, such as their names and surnames, occupations, geographical origin, parents information, as well as the corresponding marriage fee that was paid (this amount depends on the social status of the family). Each book contains a list of individual marriage license records and the corresponding tax payments (analogous to an accounting book) of two years and it was written by a different writer. Information extraction from these manuscripts is of key relevance for scholars in social sciences to study the demographical changes over five centuries.

[illegible]

(a) License from 1618 (b) License from 1729

Figure 5.2: Examples of marriage licenses from different centuries.

5.1.2 Grapheme Descriptor

The Blurred Shape Model has been chosen as a shape descriptor. This descriptor is used to generate the codebook from the graphemes used to label the nodes.

Blurred Shape Model

The Blurred Shape Model (BSM) descriptor [10] is an Object- based method. It is an improved version of the Zoning descriptor and encodes the probability of pixel densities of image regions. First, the image is divided into a grid of n x n equal-sized sub-regions. Then, each cell in the grid receives votes from the shape points in it and also from the shape points in the neighbouring sub-regions. In order to describe a symbol that can suffer from irregular deformations, the key points are defined by high gradient magnitude pixels. Therefore, each shape point contributes to a density measure of its cell and its neighbouring cells. This contribution is weighted according to the distance between the point and the centroid of each region. Finally, the descriptor is normalized within the range [0..1].

5.1.3 Settings

The number of classes used for the codebook generation has been set to $K = 20$.

For the cost matrix computation, the values of the weights have been empirically set as follows: and $wn_0 = 2/5$, $wn_1 = 1/5$, $wn_2 = 2/5$, $we_0 = 1/5$, $we_1 = 2/5$ and

$we_2 = 2/5$. Finally, the value of the baseline constants have been set to $t_{vertices} = t_{edges} = 0.5$.

5.1.4 Metrics

So far, we have explained the different methods developed but until now, none has been evaluated. In order to perform an evaluation, we need to choose the different metrics that will help us to decide whether or not the proposed approach works correctly.

Let *True Positive (TP)* be the set of correctly retrieved elements, *False Positive (FP)* be the set of badly retrieved elements from the dataset and *False Negative (FN)* be the relevant elements that have not been well retrieved.

In order to evaluate the performance, some metrics have been chosen.

- **Precision:** Is the probability that a (randomly selected) retrieved document is relevant. Precision is defined as

  $$Precision = \frac{TP}{TP + FP}$$

  In other words, it can be defined

  $$Precision = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

  Let *P@n* be the precision at *n*, it is obtained by computing the precision at a given cut-off rank, considering only the n topmost results returned by the system.

- **Recall:** Is the probability that a (randomly selected) relevant document is retrieved. Recall is defined as:

  $$Recall = \frac{TP}{TP + FN}$$

  In other words, it can be defined

  $$Recall = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

- **Mean Average Precision (mAP):** mAP is computed using each precision value after truncating at each relevant item in the ranked list. For a given query, let $r(n)$ be a binary function on the relevance of the n-th item in the returned ranked list, the mean average precision is defined as follows:

  $$mAP = \frac{\sum_{n=1}^{|\text{ret}|} P@n \times r(n)}{|\text{rel}|}$$

where *ret* is the set of relevant objects with regard to the query and *ret* is the set of retrieved elements from the dataset.

5.2 Word Spotting Performance

This first experiment consists in evaluating the performance of our approach in order to retrieve the results from a set of pre-segmented words without the indexation.

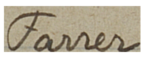
The evaluation dataset consists of 27 pages. All the words are correctly segmented and we have 6,544 word snippets with 1,751 different transcriptions from the database. All the words having at least three characters and appearing at least ten times in the collections were selected as queries. There are 514 queries corresponding to 32 different words. This experiment is equal to the one performed in [20].

Two different evaluations have been tested, first the graph is avoided and a *Histogram of Codebooks* is computed, secondly the performance is boosted with the structural information provided by the *Graph of Codebooks*. These two evaluations, allows us to understand the importance of the structural information.

5.2.1 Histogram of Codebooks

In order to reduce the complexity of graphs, a histogram of the codebooks is computed. This section presents the results using distance between these histograms. No structural information is stored, however spatial information is codified through a spatial pyramid technique. The results shown are the best obtained with different levels for the spatial pyramid which consist on a grid of 3 horizontal partition against 10 vertical.

Table 5.1 shows the qualitative results of the Histogram of Codebooks (HoC). We can see that lots of mistakes are done and that it cannot be considered as a good method for word spotting.

Query: 

Results:

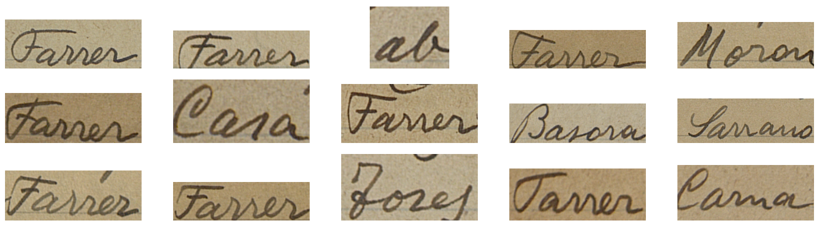


Table 5.1: Qualitative results for the query Farrer (HoC).

For this experiment two typical distances for histograms have been used:

- *Euclidean*: The Euclidean Distance between two points in Euclidean space is the length of the straight line connecting them.

$$d(p,q) = d(q,p) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2},$$

  where $p = (p_1, p_2, \ldots, p_n)$ and $q = (q_1, q_2, \ldots, q_n)$ are two points in the Euclidean n-space.

- *Cosine*: The Cosine Similarity is a measure of similarity of two vectors that measures the cosine between them. This similarity does not take into account the magnitude of the vectors.

$$S_C(p,q) = cos(\theta) = \frac{p \cdot q}{\|p\|\|q\|} = \frac{\sum_{i=1}^{n}(q_i \cdot p_i)}{\sqrt{\sum_{i=1}^{n}(p_i)^2}\sqrt{\sum_{i=1}^{n}(q_i)^2}}$$

  where $p = (p_1, p_2, \ldots, p_n)$ and $q = (q_1, q_2, \ldots, q_n)$. Cosine distance is defined as $D_C(p,q) = 1 - S_C(p,q)$. However, it is not a distance following the mathematical definition. Cosine distance does not have the triangular inequality for example.

Table 5.2 shows quantitative results using different distances to sort the results. From this table we can say that the distance computation is a very important task in the process. Table 5.3 shows his quantitative results against other techniques. This result confirms that this is not a good approach.

Distance	mAP
Euclidean	0.79
Cosine	10.78

Table 5.2: Distance Comparison (HoC).

5.2.2 Graph of Codebooks

Using the same codebooks but adding the structural information is a nice way to improve the results.

Table 5.3 shows the quantitative results and compares them to some other methods in the literature whereas Figure 5.3 presents the precision-recall curve of the Graph of Codebooks (GoC) approach changing the number of accepted results. The proposed approach outperforms most of the methods representing classical families of approaches in the literature (statistical, structural, pseudo-structural). We must notice that the aim of this work is to propose graph matching as a valid alternative for word spotting, in front of the more widespread techniques usually inspired by statistical pattern recognition.

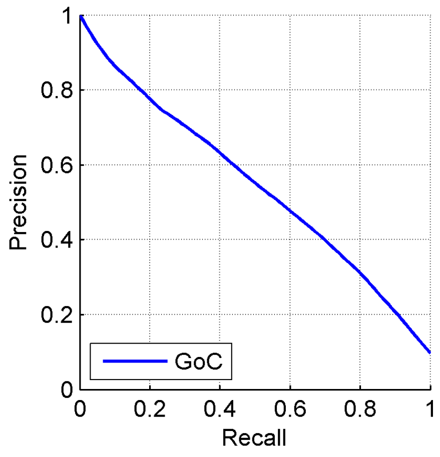


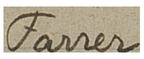
Figure 5.3: Precision-Recall Curve

Method	mAP
DTW [20]	19.20
Graph-based [43]	24.60
BoVW [20]	30.00
Loci-based [11]	40.06
nrHOG [1]	56.06
HoC	10.78
GoC	51.62

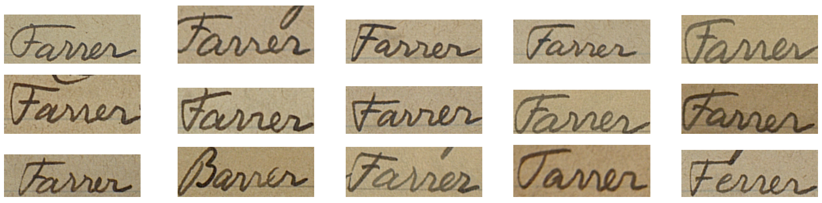
Table 5.3: Word Spotting results.

Some qualitative results are shown in Table 5.4. It is interesting to notice that most words have been correctly retrieved. This example takes the name *Farrer* as a query. The system correctly retrieves the first 11th words, whereas the 12th retrieved word corresponds to the name *Barrer* and the 15th corresponds to *Ferrer*. These words indeed are very similar to the query: it has the same character length, and only one different letter. Jumping to farther positions (from 19th to 42nd), more mistakes appear and some examples appear in the Table. However, they are still very similar to the word *Ferrer*.

Query:



Results:



Mistakes:

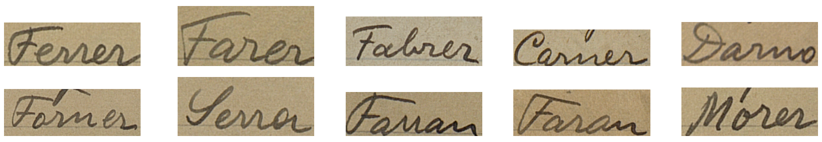
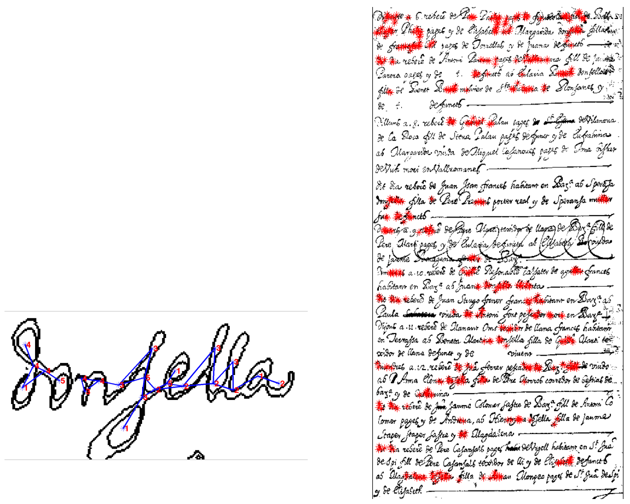


Table 5.4: Qualitative results for the query Farrer (GoC).

5.3 Word Spotting indexed for large documents

In this experiment, the graph indexation based on binary embedding explained in Chapter 4 has been applied to boost the word spotting method in terms of time. It has been used with the aim of dealing with large datasets, reducing the complexity inherent to graph matching. For this experiment, we have selected 11 pages, containing 3,609 words. From them, we have randomly chosen 5 instances for each one of the 8 different words, obtaining a total of 40 query words.


(a) A query word and its corresponding graph

(b) A full page and the locations where query nodes are detected

Figure 5.4: Qualitative results.

For each one of the query words, the fast matching is used to select the candidate regions in the pages. In this way, only the areas that receive a minimum amount of matches are considered as candidate regions, which are then matched using the proposed word spotting approach in Chapter 3. In our experiments, thanks to the indexation and detection of the candidate regions, for each query, we can reduce from 3,609 to an average of 1,254 the number of regions to compare. From these 1,254 candidate regions, an average of 40 regions contain the desired word, whereas 10 words have been missed. In summary, the indexation allows to reduce by 3 the number of target graphs to be matched to the query word graph, showing that the graph-based method can be used in large datasets. Figure 5.4 shows the result of applying the indexation process to one of the pages, figure 5.4a is the query word with its graph whereas image 5.4b shows the selected nodes in one of the pages. In the second image, it is easy to see some regions where the nodes accumulate, this regions will be our candidate subgraphs.

Table 5.5 presents the Precision and Recall of the indexation step and the mean Average Precision of our word spotting approach. Note that in this case, we only compute the matching distance between the query word and the candidate regions to compute the mean Average Precision.

Looking only to the *mAP* column, we notice that two queries are generating a lower

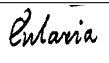
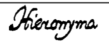
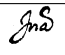
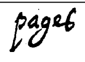
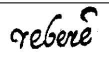
Query	Transcription	Precision	Recall	mAP
	Eularia	0.0080	0.8462	0.7959
	Hieronyma	0.0118	0.7875	0.9329
	Jua$	0.0149	0.5389	0.8490
	defunct	0.0271	0.7886	0.6372
	donsella	0.0420	0.8215	0.9454
	pages	0.0590	0.9352	0.9463
	rebere$	0.0645	0.7676	0.9815
	viudo	0.0133	0.6455	0.9231
	Total	**0.0301**	**0.7664**	**0.8764**

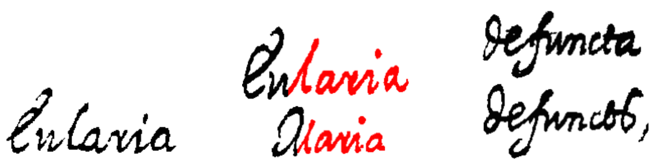
Table 5.5: Word Spotting results based on graph indexing.

value than the others. The special queries are *Eularia* and *defunct.* Studying these particular cases, some typical problems appear.

- *Binarization Problem*: The binarization step can provoke degradations or noise introduction that can affect a lot the generation of the graph. This problem appears in one of the queries *Eularia* (Fig. 5.5a).

- *Sharing Parts*: Two different words that share most of their letters may have a smaller cost than the correct word with a different writing style. For example, *Maria* is retrieved when searching the query *Eularia* (Fig. 5.5b) due to the similarities of their shapes.

- *Lexical Variations*: This problem is similar to the last one. In this case, the word appears with different lexical variations, for example for plural or feminine. The query *defunct* has a lower performance than the others due to this fact (Fig. 5.5c). Theoretically both words are not the same, but for some applications it may be useful to retrieve.



(a) Binarization (b) Shared letters (c) Lexical variations

Figure 5.5: Problems of the word spotting.

Chapter 6

E-Crowds

Word spotting combined with new technologies are a useful tool for students, researchers and a lot of people from different fields. Nowadays, word spotting is getting more importance through tablets and mobile phones. In this chapter a prototype application able to perform searches in historical documents is presented.

6.1 Introduction

Today, the new technologies are changing the way people read. The improvements on devices such as e-books, tablets and smartphones have changed the way in which humans interact with information. These devices have been transforming the reading process. The information flows bidirectionally and generates different narratives depending on the interactions. For example, users can jump between information items following hyper-links, add annotations, get contextual information or search in dictionaries clicking at words, etc.

Printed books, document or handwritten texts must be digitalized and preprocessed to enable the interactions. For example, a word spotting approach can enable to perform searches inside the documents. This chapter, presents a prototype of application that shows the uses that can be given to the developed techniques. The proposed prototype uses the collection compiled from the Marriage Licenses Books from the Cathedral of Barcelona explained in Section 5.1.1.

The application can be a very useful tool in historical research. Search centred at people, for example, can be important for family history and genealogical research [17, 29]. Queries of this type, for example a name, allow the study of the historical context (i.e. family, job, locations, etc.) of that person. Using word spotting techniques an scholar will be able to search their family or people with the same job and extract useful information. The same task without using this kind of techniques may require significant time and effort. The scholar must read lots of useless information and have to do many cross-referencing between different data sources.

Android is a mobile operating system (OS) based on the Linux kernel. It is currently developed by Google and it is the most common operating system for mobile devices around the world. In July 2013, it has had over one million Android applications (”apps”) published in the Google Play Store.

This chapter, proposes an application of the methods developed on Chapter 3. The proposed application can run on Android devices and integrates a browser to read the documents and the functionality of retrieving the queried words. The application also includes an option to combine different query words. It has been named E-Crowds[1].

6.2 System Architecture and Components

6.2.1 Devices

This software has been developed in a smartphone Sony Xperia S with Android 4.1.2 Jelly Bean. The specifications are the following: 1.5 GHz Dual Core processor, 1GB of RAM, Screen size 5 inches and Screen resolution at 1280 x 720 pixels.

In order to test the application, further devices have been used.

- Sony Xperia Z with Android 4.4.4 KitKat. The specifications are the following: 1.5 GHz Quad Core, 1GB of RAM, Screen size 4.3 inches and Screen resolution at 1920 x 1080 pixels.
- LG Nexus 5 with Android 5.0 Lollipop. The specifications are the following: 2.26 GHz Quad-core, 2GB of RAM, Screen size 4.95 inches and Screen resolution at 1920 x 1080 pixels.
- Samsung Galaxy Note 8 with Android 4.4.2 KitKat. The specifications are the following: 1.6 GHz Quad-core, 2GB of RAM, Screen size 8.0 inches and Screen resolution at 1280 x 800 pixels.

6.2.2 Preprocess

The application proposed has to rely on a robust method that extracts and represents the information of the documents of the dataset, i.e. the words that are contained. We propose to divide this process in two independent steps: first, a segmentation process to extract the individual words of the documents, and then, apply the word spotting approach to the extracted word.

Different methods have been recently proposed to segment words from handwritten documents [19,30]. However, the analysis and comparison of these techniques is out of the scope of this project. For the development of this project, the application relies on the ground-truth information to segment the words of the dataset.

The application must be provided by an Scalable Vector Graphics (SVG) file for each image with the information of the bounding boxes. This file has to contain two different tags with information:

[1]Demonstration video: `https://www.youtube.com/watch?v=4OM_GOyBIUM`

- Tag *svg* to get the size of the image where the bounding boxes have been computed. The format should be:

```
<svg xmlns:svg="http://www.w3.org/2000/svg"
     xmlns="http://www.w3.org/2000/svg"
     version="1.0" width='2690' height='3794'
     id='svg_document' style="display:inline">
```

- Tag *rect* to get the bounding box for the word. The format should be:

```
<rect id="identifier" x="565" y="792" width="185"
     height="92"></rect>
```

With these files we have the regions for every word and using these regions, the graph representation is computed. Afterwards, the distance matrix for all the words of the image has to be stored, this distance is obtained following the method explained in Chapter 3. Finally, the application must receive a java object with the distances previously sorted row-wisely and the matrix of the indices. It is recommended to keep at least 30 columns to enable searches of this number.

6.2.3 Interface

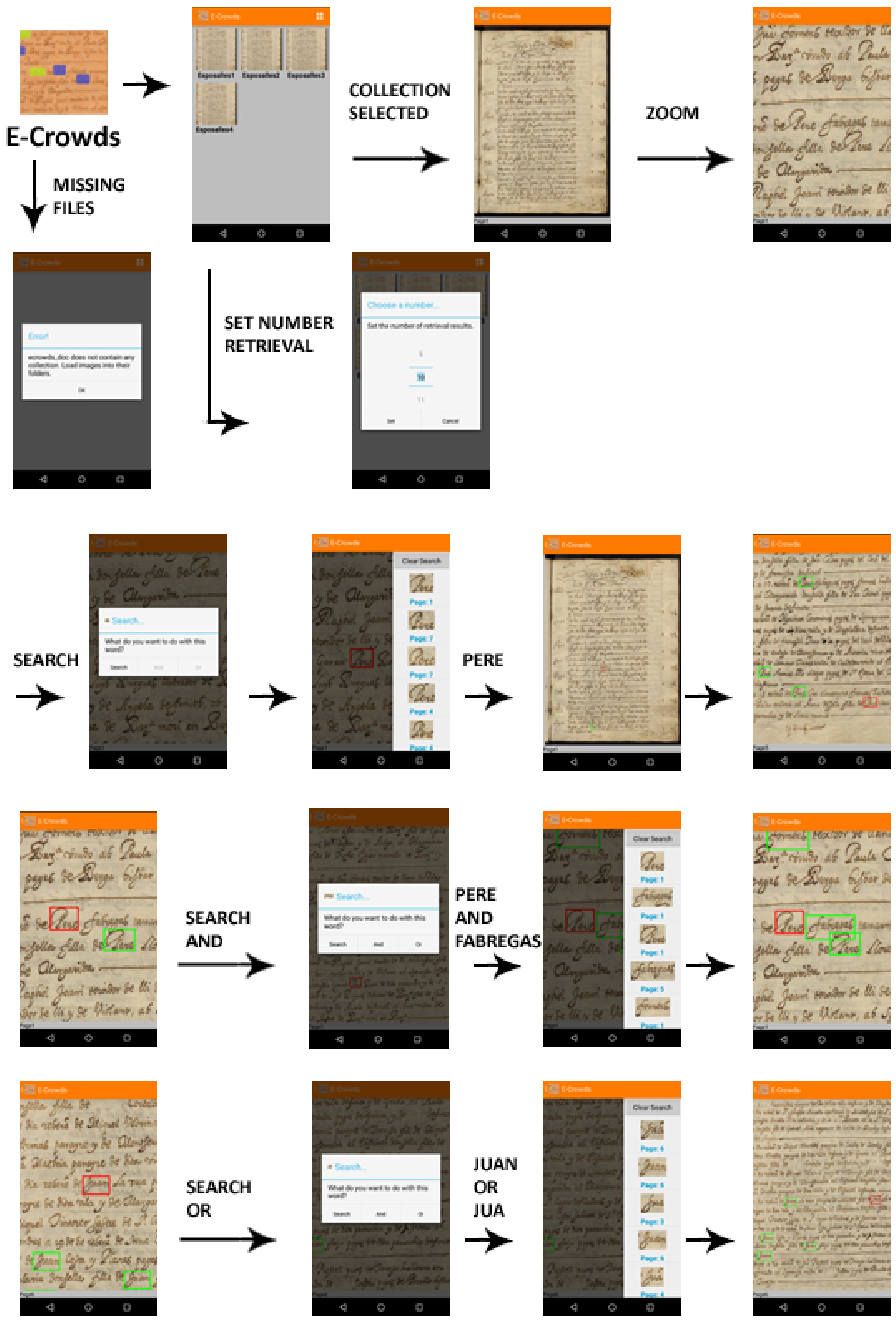
Figure 6.1 presents the interface of the application. E-Crowds application will keep a set of different document collections organized in the device. These documents have to be located in a folder accessible from the application. Inside this directory, every collection, will have its own folder with the images and the preprocessed data.

Firstly, when the users open the application touching the icon, two possible options appear. On one hand, if there are no files into the device and error message is thrown. On the other hand, the application runs normally and all the possible collections are presented in a gallery. In the same screen, there is a button to select the number of retrieved objects that will be take into consideration for each query word.

When a collection is selected, the documents are shown. At that view, the user can perform the functionalities explained in section 6.2.4. When the user has done a search, a vertical bar appears in the right side of the screen showing all the results and a button to clear the search. This bar can be hidden and shown by a swipe movement of the user. Finally the retrieved words are marked in the page with red and green boxes. The red one indicates that this is the word with a better score in that page.

6.2.4 Functionalities

Once the document collection has been chosen, the application has two main functionalities: *browsing* and *searching.*



Error!

ecrowds_doc does not contain any collection. Load images into their folders.

OK

Choose a number...

Set the number of retrieval results.

9
10
11

Set Cancel

Search...

What do you want to do with this word?

Search And Or

Figure 6.1: E-Crowds Interface.

1. *Browsing*: The user can browse through the document collection by touching the screen. The next or the previous page is shown by a swiping movement to the left or right 6.4a. The user will always know the page that is currently displayed because the page number is shown in the bottom left corner of the screen. Zoom in and zoom out are performed by 2-finger press, pinch open zooms into content 6.4b and pinch close zooms out of content 6.4c. When the image is zoomed, the swipe movement is disable to allow the user to focus to different parts of the document by a drag movement 6.4d. Finally, a double tab gesture 6.4f will reset the zoom. This set of gestures follow the standard gesture language of touch devices like tablets and smartphones.

2. *Searching*: The system allows Query By Example searches. The user can select a word in the main panel by a long press 6.4e in the screen. Then, the system shows a dialogue asking for confirmation. If the user agrees, the system shows, in the right side panel, the list of retrieved words in the document collection.

   Once a word has been searched, the user can apply two different logic operations, *and* or *or*, in order to search more words related to the previous one. The idea behind is that the user can refine the search: after searching a first word, the user can add more queries to the search, emulating a coarse-to-fine search. With the *and* operation, the second word is searched within the results, whereas with the *or* operation, the user is adding an alternative search to the results. These two options are described in detail next.

   - *And* (refined search): With this option, the user can search for different words (e.g. name, surname) at the same time in order to focus the search in one person for example. The objective of this operation is to retrieve regions of the document that are both likely to contain the query words that are related between them.

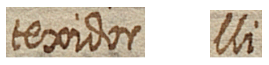

(a) texidor (b) lli

Figure 6.2: Refined search: teixidor and lli.

   - *Or* (alternative search): This functionality allows to perform a search of two words at the same time. It can be very useful in case the same word is written with different spellings or abbreviations, like “Barcelona” and “Barna”.

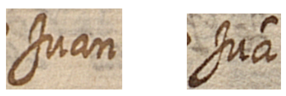
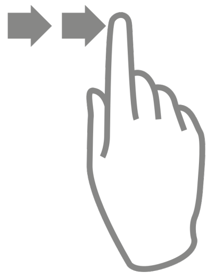
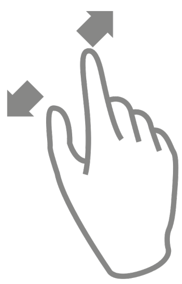
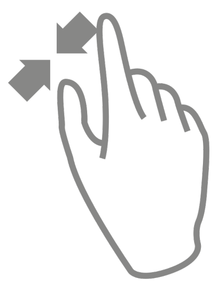
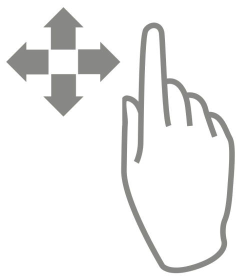

(a) juan (b) jua$

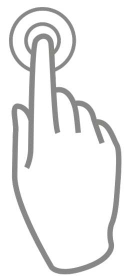
Figure 6.3: Spelling Variations: juan or jua$.

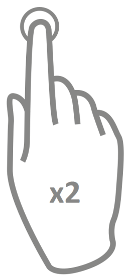

(a) Swipe


(b) Pinch Zoom In


(c) Pinch Zoom Out


(d) Drag


(e) Hold



(f) Double Tab

Figure 6.4: Android Gestures.

6.2.5 Methodology

Regarding the *Search* option, once the word has been detected, the corresponding indices and scores are stored in the retrieved list.

We define $S_1, S_2 \in \mathbb{R}^N$ as the sorted list of similarity scores between the query words w_1 and w_2, where N is the number of retrieved words. In the case of the *Or* operation we want to indistinctly retrieve results matching either with w_1 or w_2. For that, we simple merge S_1 and S_2 into a single list $S \in \mathbb{R}^{2N}$, which is again sorted and returned as the final result. In the case of the *And* we want to find regions that match both w_1 and w_2 with and additional condition: they have to be related, for instance, they belong to the same record or they are close in the document. In other words, we want to maximize the similarity of the retrieved regions with w_1 and w_2, but minimize the distance between these regions. For this purpose we evaluate all possible combinations between regions retrieved in S_1, S_2, and we weight this combinations according to the distance between them using the Gaussian function.

$$f(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

f computes the probability that both regions are related according to the distance between them. Function f uses a normal distribution with centre μ and standard deviation σ as parameters. This function has a maximum when $x = \mu$. However, our scores are ordered from the lowest to the highest and we need to obtain the lower value when $x = \mu$. To do so, we will weight the scores using $f(\mu, \mu, \sigma) - f(d(i, j), \mu, \sigma)$

that will take values between 0 and 1. We compute a matrix $S' \in \mathbb{R}^{N^2}$ of combined results using the following equation:

$$S'_{ij} = \frac{S_{1,j} + S_{2,j}}{2}(f(\mu, \mu, \sigma) - f(d(i,j), \mu, \sigma))$$

where $d(i,j)$ is the euclidean distance between regions i and j. For words that are in different pages, it can be considered to have an infinite distance between them. Afterwards, we can compute the final list $S \in \mathbb{R}^{2N}$ with the minimum scores for the rows and columns. Finally those elements with infinite score are deleted from the list S and it is sorted and returned as the final result.

6.3 Reviews

The application has been shown to different experts that can be interested in the use of this technologies for their daily work. Three groups of people have been chosen.

6.3.1 Archivists

A demonstration has been done to some archivists from the *Escola Superior d'Arxivística i Gestió de Documents* of the UAB. They are very interested in this kind of techniques and they qualified it as a very nice application for their work. However, some suggestions have been done.

- To add a *Query-by-String* search. Some times the users of the archives do not know where they can find one example of a queried word.
- To allow a search through a huge amount of collections. It would be a method to find the collection that they want to read. For example, they mention that the users of the archives usually do not know where they have to search the information they are looking for.

Another demonstration has been done in the *Arxiu Comarcal del Berguedà* which is the 6th archive with more digitalized documents in Catalonia. The archive director Xavier Pedrals has given us some feedback about the application. He thinks that the application is very interesting and useful. However, the amount of digitalized documents in the archives is very small in comparison to the non digitalized documents. One of the possible problems that he has pointed is that the documents that are used in the application have a very good and similar calligraphy for all the pages, whereas in some documents, the writer and the writing style change very frequently. He proposed that this kind of tools has to be focused on large document collections with a common or similar styles for all the pages. For example, he has shown us some documents from *Actes del Ple de l'Ajuntament de Berga* that can be good candidates to use with those techniques because they are highly demanded by the users, the writing style and the information that can provide.

6.3.2 Demographers

Demographers work with lots of documents in order to study the world population. In their opinion, this application is a great tool with lots of future possibilities. The main points of their comments are:

- To create a tutorial to show how to use the application.
- To show a variable number of retrieved results to avoid losing true positives or adding true negatives. This number should vary depending on the scores of the retrieved words.
- To mark using a different colour the word that has been searched at the beginning.
- To add more documents to the application.

6.3.3 Historians

The last group of potential user are the Historians. The application has been presented to Dr. Dolors Santandreu. She has found the application to be very useful to search for relations and also to create indices of names. Furthermore, she has explained some of the problems she had found during her thesis. One example was the time lost looking for some people's names that had appeared a few pages ago and she needed to relate with her current page.

Dolors has been very interested in this kind of technology. She has compared the idea of searching words for the appearance used in word spotting with the same technique used manually by the historians when they are not able to understand some words. They have to look for a similar word in the text around.

6.3.4 Conclusions

After showing the application to different experts, some conclusions can be extracted. Firstly, we had noticed that most of them are not used to this kind of techniques. Some courses or formation should be provided to let them know what they can expect form these tools. During the demonstration we have noticed that they are very interested in using them.

Another point that is important for all the experts is related to the scalability of the application. Their first thought without any explanation is to use the application with their own documents. However, after explaining all the previous steps that are needed they understood that it is not feasible to be done without a previous study.

Chapter 7

Conclusions

The final chapter of this dissertation is devoted to summarize the contributions of this work. We will find the final conclusions and contributions for every part of the project, the objectives accomplished during the development of the work and the future work lines and improvements.

7.1 Final conclusions

In this dissertation we have proposed a whole research project which goes from the documentation about the state of the art to the application of the developed technique. We have proposed a word spotting method based on graph-representations and a fast indexation approach.

Some conclusions can be extracted from this work. The most important one is that the proposed word spotting approach has been shown to be comparable to statistical ones in terms of performance.

For the first part of this work, we have shown that graphemes based on convexities can be stable under the deformations of handwriting. For the indexation approach, the main contribution of the proposed framework is the definition of a binary embedding for graph nodes based on the called local context. The node context has been defined as the topology of the paths of order k incident in the node and coming from nodes of a given label. A hashing architecture has been designed using binary codes as indexation keys.

In terms of a retrieval problem, high recall values are obtained, although the precision is low. The time complexity of this indexation is linear in terms of the number of nodes of the database. It leads us to conclude that a graph indexation scheme as it is proposed is very useful to compute inexact subgraph matchings in large-scale scenarios as a filtering step aiming to prune the database, so a more accurate matching method can be computed afterwards only in the retrieved subgraphs. We have demonstrated that compact structural descriptors are useful signatures for handwriting recognition, despite the variability of handwriting. Finally, combining both approaches, the experimental results demonstrate that our structural approach is comparable to statistical approaches in terms of performance and time requirements.

The last part of this project tries to show the practical utility of the proposed method. An application prototype for android devices has been developed. That section has shown a real application useful for lots of different users.

All the work proposed has been presented in international conferences and workshops.

7.2 Accomplished objectives

All the objectives have been accomplished. This project was meant to be focused on research, however, at the beginning of the work, we could not imagine that it will lead us to publish this work.

The performance of structural methods for the word spotting process in old handwritten documents has been shown. Moreover, lots of knowledge has been learnt, from word spotting state of the art to graph matching techniques.

7.3 Future work and improvements

This project opens a wide range of possibilities to keep working on. Future work will focus on the evaluation of the stability of graph-based representations in large multi-writer document collections. This work line can be useful to test if this method is still stable changing the handwriting style or even detect if a writer changes along a document. This work line can lead to different kinds of applications, for example, a writer verification or identification framework.

Other work lines that can be opened to improve the proposed approach are the modification of some modules of the framework. This is an important step in order to avoid noise introduction in the pre-process steps that can make the framework to fail in the retrieval step.

- *Binarization*: The binarization step has been noticed to be of key importance. Wrongly binarized words introduce noise to our approach.
- *Grapheme definition*: New graphemes can be defined. This new definition can lead us to a more stable representation able to better distinguish between classes.
- *Grapheme extraction*: One option is to extract the graphemes from the grey scale image without a binarization. This modification will reduce the noise produced in the binarization step.
- *Grapheme descriptor*: Another descriptor, such as *Shape Context*, may be a useful tool to improve the performance.
- *Cost Matrix*: To change the cost matrix for the graph edit distance to be more discriminant for the word spotting purpose.

- *Binary embedding*: The binary node embedding proposed can be improved adding information referring to the length of the edges or the distance between the labels of the graph.

We would like to test this approach with more databases and compare with other state of the art techniques.

Regarding the application, there is the possibility to connect it to a server where all the hard computations can be done in a fast way. In that server, the preprocess for lots of collections can be stored to enable the users to read every document at real time all around the world.

List of Publications

[1] David Fernández, Pau Riba, Alicia Fornés, and Josep Lladós. On the influence of key point encoding for handwritten word spotting. In *14th International Conference on Frontiers in Handwriting Recognition*, 2014.

[2] Pau Riba, Jon Almazán, Alicia Fornés, David Fernández, Ernest Valveny, and Josep Lladós. e-crowds: a mobile platform for browsing and searching in historical demography-related manuscripts. In *14th International Conference on Frontiers in Handwriting Recognition*, 2014.

[3] Pau Riba, Josep Lladós, and Alicia Fornés. Handwritten word spotting by inexact matching of grapheme graphs. In *13th International Conference on Document Analysis and Recognition (Accepted)*, 2015.

[4] Pau Riba, Josep Lladós, Alicia Fornés, and Anjan Dutta. *Large-Scale Graph Indexing Using Binary Embeddings of Node Contexts*, volume 9069 of *Lecture Notes in Computer Science*, pages 208–217. Springer, 2015.

Bibliography

[1] Jon Almazán, Alicia Fornés, and Ernest Valveny. Deformable hog-based shape descriptor. In *12th International Conference on Document Analysis and Recognition*, pages 1022–1026. IEEE, 2013.

[2] Jon Almazán, Albert Gordo, Alicia Fornés, and Ernest Valveny. Efficient exemplar word spotting. In *In Proceedings of the British Machine Vision Conference*, 2012.

[3] Jon Almazán, Albert Gordo, Alicia Fornés, and Ernest Valveny. Handwritten word spotting with corrected attributes. In *IEEE International Conference on Computer Vision*, pages 1017–1024. IEEE, 2013.

[4] Jon Almazán, Albert Gordo, Alicia Fornés, and Ernest Valveny. Word spotting and recognition with embedded attributes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(12):2552–2566, Dec 2014.

[5] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, 2002.

[6] Horst Bunke, Samy Bengio, and Alessandro Vinciarelli. Offline recognition of unconstrained handwritten texts using hmms and statistical language models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):709–720, 2004.

[7] Donatello Conte, Pasquale Foggia, Carlo Sansone, and Mario Vento. Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(03):265–298, 2004.

[8] Hani Daher, Djamel Gaceb, Véronique Eglin, Stéphane Bres, and Nicole Vincent. Ancient handwritings decomposition into graphemes and codebook generation based on graph coloring. In *International Conference on Frontiers in Handwriting Recognition*, pages 119–124. IEEE, 2010.

[9] Nicholas Dahm, Horst Bunke, Terry Caelli, and Yongsheng Gao. A unified framework for strengthening topological node features and its application to subgraph isomorphism detection. In W.G. Kropatsch, N.M. Artner, Y. Haxhimusa, and X. Jiang, editors, *Graph-Based Representations in Pattern Recognition*, volume 7877 of *Lecture Notes in Computer Science*, pages 11–20. Springer Berlin Heidelberg, 2013.

[10] Sergio Escalera, Alicia Fornés, Oriol Pujol, Petia Radeva, Gemma Sánchez, and Josep Lladós. Blurred shape model for binary and grey-level symbol recognition. *Pattern Recognition Letters*, 30(15):1424–1433, 2009.

[11] David Fernández, Pau Riba, Alicia Fornés, and Josep Lladós. On the influence of key point encoding for handwritten word spotting. In *14th International Conference on Frontiers in Handwriting Recognition*, 2014.

[12] Andreas Fischer, Ching Y. Suen, Volkmar Frinken, Kaspar Riesen, and Horst Bunke. A fast matching algorithm for graph-based handwriting recognition. In *Graph-Based Representations in Pattern Recognition*, pages 194–203. Springer, 2013.

[13] Volkmar Frinken, Andreas Fischer, R. Manmatha, and Horst Bunke. A novel word spotting method based on recurrent neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(2):211–224, 2012.

[14] Basilios Gatos and Ioannis Pratikakis. Segmentation-free word spotting in historical printed documents. In *10th International Conference on Document Analysis and Recognition*, pages 271–275. IEEE, 2009.

[15] Sebastiano Impedovo, Giuseppe Pirlo, Raffaele Modugno, and Anna Ferrante. Zoning methods for hand-written character recognition: An overview. In *International Conference on Frontiers in Handwriting Recognition*, pages 329–334. IEEE, 2010.

[16] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pages 604–613. ACM, 1998.

[17] Douglas J. Kennard, Andrew M. Kent, and William A. Barrett. Linking the past: discovering historical social networks from documents and linking to a genealogical database. In *Proceedings of the 2011 Workshop on Historical Document Imaging and Processing*, pages 43–50. ACM, 2011.

[18] Alireza Khotanzad and Yaw Hua Hong. Invariant image recognition by zernike moments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(5):489–497, 1990.

[19] Jayant Kumar, Le Kang, David Doermann, and Wael Abd-Almageed. Segmentation of handwritten textlines in presence of touching components. In *International Conference on Document Analysis and Recognition*, pages 109–113. IEEE, 2011.

[20] Josep Lladós, Marçal Rusiñol, Alicia Fornés, David Fernández, and Anjan Dutta. On the influence of word representations for handwritten word spotting in historical documents. *International Journal of Pattern Recognition and Artificial Intelligence*, 26(05), 2012.

[21] R. Manmatha, Chengfeng Han, and Edward M. Riseman. Word spotting: A new approach to indexing handwriting. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 631–637. IEEE, 1996.

[22] R. Manmatha and Jamie L. Rothfeder. A scale space approach for automatically segmenting words from historical handwritten documents. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1212–1225, 2005.

[23] Simone Marinai. Text retrieval from early printed books. *International Journal on Document Analysis and Recognition*, 14(2):117–129, 2011.

[24] Urs-Viktor Marti and Horst Bunke. Using a statistical language model to improve the performance of an hmm-based cursive handwriting recognition system. *International journal of Pattern Recognition and Artificial intelligence*, 15(01):65–90, 2001.

[25] H. L. Morgan. The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service. *Journal of Chemical Documentation*, 5(2):107–113, 1965.

[26] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial & Applied Mathematics*, 5(1):32–38, 1957.

[27] Wayne Niblack. *An introduction to digital image processing.* Strandberg Publishing Company, 1985.

[28] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *Automatica*, 11(285-296):23–27, 1975.

[29] Thomas L. Packer and David W. Embley. Cost effective ontology population with data from lists in ocred historical documents. In *2nd International Workshop on Historical Document Imaging and Processing*, pages 44–52. ACM, 2013.

[30] Vassilis Papavassiliou, Themos Stafylakis, Vassilis Katsouros, and George Carayannis. Handwritten document image segmentation into text lines and words. *Pattern recognition*, 43(1):369–377, 2010.

[31] Tony M. Rath and R. Manmatha. Word image matching using dynamic time warping. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages II–521. IEEE, 2003.

[32] Tony M. Rath and R. Manmatha. Word spotting for historical documents. *International Journal of Document Analysis and Recognition*, 9(2-4):139–152, 2007.

[33] Kaspar Riesen and Horst Bunke. Approximate graph edit distance computation by means of bipartite graph matching. *Image and Vision Computing*, 27(7):950–959, 2009.

[34] José A Rodrıguez and Florent Perronnin. Local gradient histogram features for word spotting in unconstrained handwritten documents. In *11th International Conference on Frontiers in Handwriting Recognition*, 2008.

[35] José A Rodríguez-Serrano and Florent Perronnin. A model-based sequence similarity with application to handwritten word spotting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2108–2120, 2012.

[36] 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*, 2013.

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

[38] Leonard Rothacker, Marçal Rusinol, and Gernot A. Fink. Bag-of-features hmms for segmentation-free word spotting in handwritten documents. In *12th International Conference on Document Analysis and Recognition*, pages 1305–1309. IEEE, 2013.

[39] Marçal Rusiñol, David Aldavert, Ricardo Toledo, and Josep Lladós. Browsing heterogeneous document collections by a segmentation-free word spotting method. In *International Conference on Document Analysis and Recognition*, pages 63–67. IEEE, 2011.

[40] Marçal Rusiñol, David Aldavert, Ricardo Toledo, and Josep Lladós. Efficient segmentation-free keyword spotting in historical document collections. *Pattern Recognition*, 48(2):545–555, 2015.

[41] Alberto Sanfeliu and King-Sun Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-13(3):353–362, May 1983.

[42] Peng Wang, Véronique Eglin, Christophe Garcia, Christine Largeron, Josep Lladós, and Alicia Fornés. A coarse-to-fine word spotting approach for historical handwritten documents based on graph embedding and graph edit distance. In *22nd International Conference on Pattern Recognition*, pages 3074–3079. IEEE, 2014.

[43] Peng Wang, Véronique Eglin, Christophe Garcia, Christine Largeron, Josep Lladós, and Alicia Fornés. A novel learning-free word spotting approach based on graph representation. *11th IAPR International Workshop on Document Analysis Systems*, pages 207–211, 2014.

Firmat: Pau Riba Fiérrez
8 de Juny de 2015

Resum

Al llarg d'aquest projecte s'ha desenvolupat un nou mètode de word spotting (localització de paraules) en què es té molt en compte l'estructura de les paraules a buscar. Aquestes tècniques consisteixen a trobar paraules escrites a mà, a partir d'un exemple. La tècnica presentada s'ha desenvolupat per utilitzar-la en documents antics. Seguidament, es presenta una indexació per tal d'accelerar el procés de cerca. Aquesta indexació consisteix a trobar ràpidament un conjunt de candidats on aplicar tècniques de word spotting en grans col·leccions de documents. Finalment, es mostra un exemple d'aplicació de les tècniques desenvolupades en una aplicació per a dispositius Android.

Resumen

A lo largo del proyecto se ha desarrollado un nuevo método de word spotting (localización de palabras) en el cual se tiene muy en consideración la estructura de las palabras a buscar. Estas técnicas consisten en encontrar palabras escritas a mano partiendo de un ejemplo. La técnica presentada se ha desarrollado utilizándola en documentos antiguos. Seguidamente, se presenta una indexación con el objetivo de acelerar el proceso de búsqueda. Esta indexación consiste en encontrar rápidamente un conjunto de candidatos donde aplicar técnicas de word spotting en grandes colecciones de documentos. Finalmente, se muestra un ejemplo de aplicación de la técnica desarrollada en una aplicación para dispositivos Android.

Abstract

Along this project a new method for word spotting (location of words) has been developed. This method has in mind the structure of the words to search. These techniques consist in finding handwritten words from a given example. The presented technique has been meant to be used in old documents. Afterwards an indexation process is presented to speed up the search step. This indexation is used to find a set of candidates in large document collections in order to apply word spotting techniques. Finally, an example application of the developed techniques is proposed for Android devices.