Sparse representations over learned dictionary for document analysis

Thanh Ha Do

UNIVERSITÉ DE LORRAINE

Département de formation doctorale en informatique **École doctorale IAEM Lorraine**

Sparse Representations over Learned Dictionary for Document Analysis

THÈSE

présentée et soutenue publiquement le 4 Avril 2014

pour l'obtention du

Doctorat de l'université de Lorraine

(spécialité informatique)

par

DO Thanh Ha

Composition du jury

Rapporteurs :	Christian Viard-Gaudin	Professeur, Institut Universitaire de Technologie de Nantes
	Rolf Ingold	Professeur, Université de Fribourg
Examinateurs :	Jean-Marc Ogier	Professeur, Université de La Rochelle
	Laurent Wendling	Professeur, Université Paris Descartes
Directeur de thèse :	Salvatore Tabbone	Professeur, L'université de Lorraine
Co-directeur de thèse :	Oriol Ramos Terrades	Professeur associé à l'UAB Barcelone

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Loria Laboratoire lorrain de recherche en informatique et ses applications

Mis en page avec la classe thloria.

Dedicated to my parents,
to Dung, to Ha Ngan.

Acknowledgments

I would like to acknowledge with much appreciation the Ministry of Education of Viet Nam, who supported a finance for my study. A special thanks goes to my university, Ha Noi University of Science - VNU, who gave me the permission to study at LORIA in France.

A special gratitude I give to my supervisor, Mr. Salvatore Tabbone, whose contribution in orientations, suggestions and encouragement. He has dedicated so much time and patience during the years of my study. I thank him not only for the invaluable advice, for the valuable expertise that he shared with me but also for the support which allowed me to make this thesis in the best working conditions.

Furthermore, a special thanks goes to my co-supervisor, Mr. Oriol Ramos Terrades, for his scientific advice as well as his availability despite the distance. The discussions with him allows me to understand better the research and to get more confidence.

I would like to express my deepest appreciation to Mr. Christian Viard-Gaudin and Mr. Rolf Ingold for accepting to review my thesis. I am also grateful to Mr. Jean-Marc Ogier, Mr. Laurent Wendling for accepting to be part of the jury.

I would like to thanks all the members of the team QGAR for their friendship, their help during my study within the team, especially grateful to Philippe Dosch for his outstanding technical support. Thanks to all my colleagues with whom I shared the office for the good atmosphere.

I am also very grateful to all my friends who, closely or remotely helped and encouraged me at the convenient moments. I thank them for all the time precious that we spent together.

In addition, I would also like to thank my husband and my daughter for their love, kindness and support that they have shown during the past three years of my study. Last but not least, many thanks to my parents for their endless love and support.

Abstract

In this thesis, we focus on how sparse representations can help to increase the performance of noise removal, text region extraction, pattern recognition and spotting symbols in graphical documents. The main goal is to provide new algorithms and applications of sparse representation in redundant dictionaries for graphical images, by addressing the problems from various perspectives.

To do that, first of all, we give a survey of sparse representations and its applications in image processing. Then, we present the motivation of building learning dictionary and efficient algorithms for constructing a learning dictionary. The techniques used to solve the sparsity problem over learning dictionary are also shown.

After describing the general idea of sparse representations and learned dictionary, we bring some contributions in the field of symbol recognition and document processing that achieve better performances compared to the state-of-the-art. These contributions begin by finding the answers to the following questions.

- The first question is how we can remove the noise of a document when we have no assumptions about the model of noise found in these images? To address the first question, we believe that there is a link between the model of noise and the reconstruction error of the signal in the learning dictionary. Therefore, we propose to calculate the model of noise automatically from the database based on the normalized-correlation between pairs of noisy and non-noisy images, and then using this value as the value of reconstruction error in the basis pursuit denoising algorithm with a learned dictionary. The efficiency of the proposed method has been also approved experimentally on different datasets for different resolutions and different kinds of noise. All experimental results show that the proposed method outperforms existing ones in most of the cases.

- The second question is how sparse representations over learned dictionary can separate the text/graphic parts in the graphical document. In fact, we have been strongly motivated by the good performance of the morphological component analysis (MCA) method when applied for separating textures and cartoons and the text detection from scenic images. In MCA method, the signal is discriminated with other based on the comparison between theirs sparse representations in two dictionaries. However, when working with graphic images, text characters are in different sizes included in the same document, and they touch either themselves or the graphics; therefore prior methods as MCA cannot be used efficiently in this case. As a results, we have developed the assumption of MCA by proposing the strategy using multi-learned dictionaries for separating the text regions from graphical part instead of two dictionaries. The experimental results show that the proposed method could be a good choice for the segmentation problem with complex graphical documents while it overcomes the restrictions of the existing methods on some documents only.

- This result encourages us to continue with the challenge in symbol recognition. Once again, we desire to answer the question how we can apply the sparse representation for symbol recognition? Now, the difficulty arises when it seems there is no connection between symbol recognition and sparsity, and in the best of our knowledge, there is not previous works in the graphics community using sparse representations to describe graphical symbols. Fortunately, after tireless research, we find the bridge between the literature of sparse representation and the visual vocabulary construction. More specifically, we apply the learned dictionary algorithm for learning a visual vocabulary based on local descriptors of symbols. Next, we propose the original way to construct a vector model for every symbol

based on its sparse representation in the vocabulary, and adapt the tdf-if approach to the sparse representations. The obtained results have approved that using sparse representation in the constructing vector model can help to improve the retrieval performance. We hope that this work will open a new range of applications for the symbol recognition since in our method other kind of local descriptor can be used.

We complete this thesis by proposing an approach of spotting symbols that use sparse representations for the coding of a visual vocabulary. This approach also uses learning techniques to adapt such visual vocabulary to the intrinsic properties of the document datasets. It allows achieving a representation being sparser than the one obtained by using a pre-fixed basis instead. The contribution done in this work focuses on the symbol retrieval process. The proposed approach follows a two steps architecture including the recall and the refining steps. The main goals of this architecture are to speed up the retrieval process using sparse representation and indexing techniques and to leave more computational expensive matching methods only for those regions in which the queried symbol may appear. The first experiments on the SESYD dataset for a symbol spotting application seems to agree, and the obtained results are promising.

Key words: sparse representations, learned dictionary, learning algorithms, removal noise, separation text/graphic, symbol recognition, symbol spotting, visual words.

Résumé

Dans cette thèse, nous nous concentrons sur comment les représentations parcimonieuses peuvent aider à augmenter les performances pour réduire le bruit, extraire des régions de texte, reconnaissance des formes et localiser des symboles dans des documents graphiques. Le but principal est de fournir de nouveaux algorithmes et d'applications de représentation parcimonieuses par des dictionnaires redondants pour des images graphiques, en abordant les problèmes selon des perspectives diverses.

Pour ce faire, tout d'abord, nous donnons une synthèse des représentations parcimonieuses et ses applications en traitement d'images. Ensuite, nous présentons notre motivation pour l'utilisation de dictionnaires d'apprentissage avec des algorithmes efficaces pour les construire. Les techniques employant à résoudre le problème de parcimonie sur le dictionnaire d'apprentissage sont présentées aussi.

Après avoir décrit l'idée générale des représentations parcimonieuses et du dictionnaire d'apprentissage, nous présentons nos contributions dans le domaine de la reconnaissance de symboles et du traitement des documents en les comparant aux travaux de l'état de l'art. Ces contributions s'emploient à répondre aux questions suivantes:

- La première question est comment nous pouvons supprimer le bruit des images où il n'existe aucune hypothèse sur le modèle de bruit sous-jacent à ces images? Pour aborder cette première question, nous croyons qu'il y a un lien entre le modèle de bruit et l'erreur de reconstruction du signal sur le dictionnaire d'apprentissage. Donc, nous proposons de calculer automatiquement le modèle de bruit à partir de la corrélation normalisée entre les paires d'images bruitée et non bruitée, et ensuite en utilisant cette valeur comme la valeur d'erreur de reconstruction dans l'algorithme BPDN (basis pursuit denoising) avec un dictionnaire d'apprentissage. L'efficacité de la méthode proposée a été aussi approuvée expérimentalement sur un ensemble de données en différentes en résolutions et en type de bruit. Tous les résultats expérimentaux montrent que la méthode proposée surpasse dans la plupart des cas les travaux existants.

- La deuxième question est comment les représentations parcimonieuses sur le dictionnaire d'apprentissage peuvent être adapter pour séparer le texte du graphique dans des documents? Notre contribution a été motivée par l'approche MCA (Morphological Component Analysis) qui a été appliquée avec succès dans la séparation de textures et des dessins animés et la détection de texte d'images scéniques. Dans la méthode MCA, le signal est séparé d'un autre par la comparaison de leurs représentations parcimonieuses sur deux dictionnaires. Cependant, dans les images graphiques, différentes tailles de caractères de texte peuvent être inclus dans un même document et ils peuvent se toucher ou toucher le graphisme; par conséquent, les méthodes antérieures comme MCA ne peuvent pas être utilisées efficacement dans ce cas. Ainsi, nous nous sommes appuyés sur MCA pour proposer une stratégie utilisant des multi-dictionnaires d'apprentissage au lieu de deux dictionnaires uniquement pour séparer la partie de texte de la partie de graphique. Les résultats expérimentaux montrent que la méthode proposée pourrait être un bon choix pour le problème de segmentation texte/graphique dans documents complexes surmontant les limitations des méthodes existantes et leurs applications à de simple documents.

- Ces résultats nous encourage à continuer avec le défi de la reconnaissance de symbole. Nous désirons répondre à la question de comment nous pouvons appliquer la représentation parcimonieuse à reconnaissance de symboles. Cette fois, une difficulté surgie car il n'y a aucune

connexion triviale entre la reconnaissance de symboles et la représentation parcimonieuse; et à notre connaissance, il n'y a pas de travaux précédents dans la communauté graphique utilisant des représentations parcimonieuses pour décrire les symboles graphiques. Notre contribution est de proposer une solution qui passe par l'établissement d'un lien entre la représentation parcimonieuse et la construction d'un vocabulaire visuel. Plus spécifiquement, nous appliquons l'algorithme d'apprentissage pour apprendre un vocabulaire visuel basé sur des descripteurs locaux définis sur le symbole. Ensuite, nous proposons une façon originale de construire un modèle vectoriel pour chaque symbole à partir de sa représentation parcimonieuse dans le vocabulaire et adaptons l'approche tdf-if aux représentations parcimonieuses. Les résultats obtenus ont approuvé cette représentation parcimonieuse et montre que le modèle vectoriel peut aider à améliorer la performance de la recherche. Nous pensons que ce travail ouvrira la voie à de nouvelles applications de reconnaissance de symbole étant donné que d'autres descripteurs peuvent être appliqué pour la description locale.

Nous complétons cette thèse en proposant une approche de localisation de symboles dans les documents graphiques qui utilise les représentations parcimonieuses pour coder un vocabulaire visuel. Cette approche utilise aussi les techniques d'apprentissage pour adapter un vocabulaire visuel aux propriétés intrinsèques des ensembles de données du document. Ceci permet d'atteindre une représentation plus parcimonieuse que la reprśentation obtenue par utilisation de bases prédefinies. Dans ce travail, notre contribution se concentre sur le processus de localisation des symboles. L'approche proposée se décompose en deux étapes : l'étape de rappel et de raffinage. Le but principal de cette architecture est d'accélérer le processus de localisation en utilisant les représentations parcimonieuses et les techniques d'indexation et de réserver les traitements coûteux en d'appariement uniquement pour les régions sur lesquelles le symbole requête peut apparaître. Les premières expériences sur l'ensemble de données SESYD pour l'application de localisation de symboles semblent être cohérents et les résultats obtenus sont prometteurs.

Mots clés: représentations parcimonieuses, dictionnaire d'apprentissage, algorithme apprentissage, réduction du bruit, séparation texte/graphique, reconnaissance de symboles, localisation de symboles, mots visuels.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The exponential development of the storage capacity of computers over the last few decades allowed to multiply the digital resources and so to facilitate numerous tasks. However, in front of this increase, raises a significant challenge of finding the relevant information on the huge amount of data. The need to develop fast and efficient methods is thus compulsory.

In the field of document analysis, the process of retrieving the information has been done based on the analyses of the structural information contained in the document. In general, the structural information in digital documents are partitioned into a hierarchy of physical components, such as pages, columns, paragraphs, text lines, words, tables, figures, etc.; a hierarchy of logical components, for example titles, authors, affiliations, abstracts, sections, etc.; or both. Depending on each kind of structural information, and the forms to represent the document layout, there exist different layout analysis algorithms. For example, with the physical layout analyses, algorithms can be categorized into three classes including top-down approaches [127, 5], bottom-up approaches [131, 88, 176, 59, 75] and hybrid approaches [138]. With the logical layout representations and analyses, some typical algorithms can be mentioned including the model-based system [181, 89], rule-based system [58], or frame-based system [32], etc. More details about document structure analysis algorithms can be found in the survey of Haralick [68], Nagy [126], Jain *et al* [76], and Mao [111].

Our work in this domain focuses on the technical documents. In particular, we develop pre-processing techniques to upgrade the quality of the document image or to reduce the noise generated from the input devices; we study a new method in text/graphic segmentation, in describing graphical symbols; and finally a new approach for the spotting problem have been also proposed. However, instead of developing the methods mentioned above, we approach these problems in different direction that have been developing widely in the computer vision committee when dealing with scene images, but not really in technical documents. It is the approaches using sparse representation over learned dictionary. In general, in the domain of the sparse representation, elementary atoms chosen in a family functions called dictionary represent the relevant information. As the results, searching this information means finding these atoms. However, how obtaining an ideal dictionary adapted to all images in the large database, how finding good sparse representations over this dictionary are fundamental questions and have a successful history. One of the first successes, considered the key to open the door to a huge jungle, is the discovery of wavelet orthogonal bases and local time-frequency dictionaries. The successes of searching good sparse representations over redundant dictionaries have helped to improve the performance not only of image denoising, but also of the source/image separation and pattern recognition.

Image Denoising

A successful denoising method using sparsity is first introduced by Elad *et al.* [43, 42]. In that approach, the authors used the assumption that patch of *clean image* can be approximated by a sparse linear combination of elements from a dictionary A. Another patch-based approaches can be found in the works of Buades *et al.* [16], Roth *et al.* [144]. In general, denoising a patch $h \in \mathbb{R}^L$ with a learned dictionary $A \in \mathbb{R}^{L \times M}$ corresponds to solve the sparse decomposition problem [169, 20] with h is the patch of noisy image and A is either the pre-defined dictionary as wavelets, ridgelets, curvelets,..., or a learned dictionary adapted to the patches of images. Following [43, 41, 107, 106, 105], the energy of noise ϵ can be chosen according to the (supposed known) standard deviation σ of the noise. The value of ϵ is proportional to both the noise variance and image size [41, 161, 109]. Mairal *et al.* [107, 106] used the cumulative distribution function F_m of the χ^2_m distribution and choose $\epsilon = \sigma^2 F_m^{-1}(\tau)$, where $F_m^{-1}(\tau)$ is the inverse of F_m.

However, in the specific applications about noise reduction on images, we usually do not know precisely the model of noise found in images, or in other words, the noise variance is unknown. In addition, the noise in documents is different compared to the noise in natural images that are generated by devices like digital cameras or similar. Thus, we cannot use the noise variance to decide the value for ϵ. In Chapter (3), we propose an energy noise model which allows us to easier set the threshold required for noise removal even if the noise model is unknown.

Image Separation

Image separation problem is the extention of the source separation problem being fundamental in processing acoustic signals. In general, suppose that the observed signal h is a superposition of two different sub-signals h_1, h_2, or

$$h = h_1 + h_2$$

where h_1 is sparsely generated using the model with the dictionary A_1 and h_2 is sparsely generated using other model with the dictionary A_2. If the optimal solutions $(\bar{x}_1, \bar{x}_2)$ are obtained by solving the Equation (1.1),

$$(\bar{x}_1, \bar{x}_2) = \min_{x_1, x_2} \|x_1\|_0 + \|x_2\|_0 \text{ subject to } \|h - A_1 x_1 - A_2 x_2\|_2 \leq \epsilon_1 + \epsilon_2 \tag{1.1}$$

then, the solutions to the separation problem are calculated by $\bar{h}_1 = A_1 \times \bar{x}_1$, $\bar{h}_2 = A_2 \times \bar{x}_2$. This is the basis idea of *Morphological Component Analysis* (MCA) algorithm. The success of the separation MCA algorithm is the roles of dictionaries A_i in a discriminant between content types, preferring the component h_i over the other parts. Figure (1.1) presents one example how sparse representation is used to solve the image decomposition problem.

Clearly, in MCA algorithm, one of the important questions is what are the proper dictionaries for these kinds of contents. To identify such dictionaries, we need to know the contents of images. In fact, in [41] the separation of the content of images has been done based on the assumption that images are combined linearly of cartoon and texture parts and the pre-defined dictionaries are chosen by experience of the authors. Following the works in [160, 44, 14, 159], candidate pre-defined dictionaries for texture part can be (local) discrete cosine transform, or Gabor transform, while the candidate for cartoon parts include the bi-orthogonal wavelet transform, isotropic à trous algorithm, the local ridgelet transform, or the curvelet transform. Because of making a choice from one transform to another is usually done by experience, so it is not adapted to various kind of images. This limitation can be overcame by using MCA with learned dictionaries. In [41],

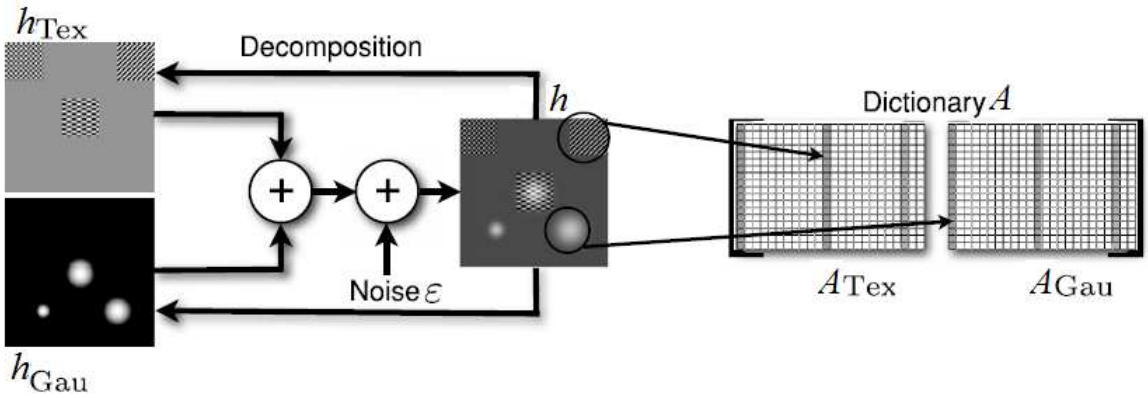


Figure 1.1: Illustration of the image decomposition problem with sparse representation (extracted from [161])

the dictionaries are learned by applying K-SVD algorithm on the corrupted patches of images itself. The separation is very successful, and one of the other remark is that this results are achieved without the need to pre-specify the dictionaries for the two parts. Learned dictionary combined with sparsity is also found in the work of Pan *et al.* [135] and Zhao *et al.* [192].

However, the performance of above methods depends strongly on the size of the patch. In fact, if the size of the patch is too large, its sparse representation vector is large. It means the computing cost time will increase. If this size is too small, it could not contain enough information for discrimination. Therefore, to overcome this shortcoming, we propose a method in the Chapter (4) using multi-resolution learned dictionaries for separating text parts from graphical ones. To the best of our knowledge, it is the first times multi-resolution learned dictionaries have been used for the separation task. In general, the proposed method is a patch-based approach using the assumption that the representations of text candidate patches in text learned dictionaries are sparse, but they are not sparse enough in graphic learned dictionaries.

Classification

There are some previous works that use sparsity over learned dictionary to enter in classification tasks and then improve the classification performance. To the best of our knowledge, the first approach is the work of *Raina et al* with the *self-taught learning* (STL) method [141]. In [141], the dictionary is learned from an unlabeled dataset, then the sparse coding coefficients obtained when coding elements of the labeled dataset serve as features which are fed into an SVM classifier (see Algorithm (1)). STL algorithm is very efficient in the case that the training and testing sets are aligned.

Algorithm 1 STL algorithm

1: $(A, X) \leftarrow$ **TrainDictionary**(H)
2: Learn a classifier C by a linear SVM with input is X
3: $X_{test} \leftarrow$ **Lars**(H_{test}, A) % Lars is a function used to find a sparse solution (see Chapter 2)
4: Classify the set X_{test} by C

Another noticeable framework for invariant classification is the framework proposed by Bar

Dataset	$\theta[^0]$	Scale	STL	HIA
10 digits	± 50	1	59.1	86.0
	± 50	± 0.2	55.6	83.6
3 Textures	–	–	76.4	94.7
4 Textures	–	–	75.6	91.2

Table 1.1: Classification accuracy for the digits and texture data [%]. Only testing samples are randomly transformed for the digits (such transformations are natural in the texture dataset) (extracted from [6])

et al. [6]. This approach integrates the ideas of the sparse representation theory and hierarchical structures. In fact, the authors used a log-polar mapping to convert rotated and scaled patterns into shifted patterns in the new space where they operate for learning the dictionary and adding hierarchy. This approach is the extension of the *self-taught learning* (STL) method [141] to a hierarchical architecture of dictionary learning. Authors approved that this hierarchical approach performs better than the layered one, and by using learned dictionary instead of pre-defined one. The results for classifying the handwritten digits and texture images are improved (see Table (1.1)).

There is other state-of-the-art algorithm based on discriminative dictionary learning models that gives good performances in classification tasks [114, 115, 112]. The authors in [114, 115, 112] proposed a formulation for learning a dictionary tuned for a classification task which is called the supervised dictionary learning. It is a discriminative approach that effectively exploits the corresponding sparse signal decompositions in image classification tasks. This work afforded an effective method for learning a shared dictionary and multiple (linear or bilinear) decision functions. The experiments done on MNIST [91] and USPS [73] handwritten digit datasets shows promising results for this approach. In fact, this approach achieved state-of-the-art results on MNIST with a 0.54 percent error rate, which was similar to the 0.60 percent error rate in [142], and 2.84 percent error rate on USPS which was slightly behind the 2.4 percent error rate in [65].

The authors in [182] presented a method that computes a spatial-pyramid image representation based on sparse representation of local features for image classification. This method uses a selective sparse representation instead of traditional vector quantization to extract salient properties of local descriptors. In addition, sparse representation is used to operate local max pooling on multiple spatial scales to incorporate translation and scale invariance. An encouraging result of this paper is the local of features used instead of an image or patches of image in the sparsity framework. This method is approved and works well when combined with simple linear SVMs for improving the scalability of training, the speed of testing, and the classification accuracy.

The mentioned methods above usually working on the images, the patches extracted from images, or even being developed to adapt to local features of images as in [182]. However, in the best of our knowledge, these methods gave no characteristic of the invariance of sparse representations. It means if the image/patch is changed under some transformations such as scale, rotation, degradation, etc., then its corresponding sparse representation is not similar (or almost similar) to the sparse representation of original image in the same dictionary. In addition, in the context of symbol recognition, the description of the symbol needs the invariance criteria under the affine transformations. Thus, in the Chapter (5), our working hypothesis is that 'if a learned dictionary is optimized by taking into account the data properties derived from a descriptor, thus it not only specifically adapted to the descriptor but also provides the optimal approximation of the descriptor'. Therefore, we propose a new approach in contributing the vector models

of the symbols from their sparse representations over learned dictionary of descriptors. Our approach have been approved that it is not only keeping the invariant criteria under the affine transformations but also improving the performance of the symbol retrieval system.

Contributions and Organization of the thesis

The contributions presented in this thesis follow the lines of how good sparse representations over redundant dictionaries can help to increase the performance of noise reduction, text/graphical separation, pattern recognition, and in localization of elements into graphic documents such as architectural or electric planes. Before presenting our contributions in the field of symbol recognition, document processing and the good performances achieved by our algorithms compared to the state-of-the-art, we review in this thesis the motivation of constructing the learned dictionary instead of using pre-defined dictionary and efficient algorithmic tools for building a learning dictionary. The methods used to solve the sparsity problem over learned dictionary are also shown.

Our works in this thesis will be reviewed in following organization:

- The first part includes one chapter, chapter number (2), which describes the background of sparse representation and related works involving to the sparsity. In this chapter, the definitions about the sparse representation and learned dictionary are presented. Detail reviews of state-of-the-art algorithms on finding the sparse representation, on building the dictionary are also shown and discussed carefully. Numerical experiments are performed with the purpose of evaluating the complexity and finding out the advantage as well as disadvantage of these algorithms for each particular problem.

- The second part is a low processing for document filtering and text/graphic separation:

  - We propose in chapter (3) an algorithm for de-noising document images using sparse representations. Following a training set, this algorithm learns not only the main document characteristics, but also the noise included into the documents. In this perspective, we propose to model the noise energy based on the normalized cross-correlation between pairs of noisy and non-noisy documents.
  - A new approach to extract text regions from graphical documents is presented in chapter (4). In the proposed method, we first empirically construct two sequences of learned dictionaries for the text and graphical parts respectively. Then, we compute the sparse representations of all different sizes and non-overlapped document patches in these learned dictionaries. Based on these representations, each patch can be classified into the text or graphic category by comparing its reconstruction errors. Same-sized patches in one category are then merged together to define the corresponding text or graphic layers, which are combined to create a final text/graphic layer. Finally, in a post-processing step, text regions are further filtered out using some learned thresholds.

- The third part is our dedications in symbol recognition and spotting symbols:

  - In chapter (5), we propose a new approach for symbol description based on the combination between shape contexts of interest points (SCIP) descriptor with sparse representation over learned dictionary. More specifically, a dictionary is learned from training dataset being the SCIPs in symbols. Then, each column of this dictionary

is considered as a visual word. Next, a vector model for the symbol is constructed based on sparse representations of its SCIPs in the learned dictionary and the tf-idf approach adapted to the sparsity.

- We propose an approach to deal with the problem of symbol spotting for graphical documents using sparse representations in chapter (6). In the proposed method, a dictionary is learned from a training database of local descriptors defined over the documents. Following their sparse representations, interest points sharing similar properties are used to define interest regions. Using an original adaptation of information retrieval techniques, a vector model for interest regions and for a query symbol is built based on its sparsity in a visual vocabulary where the visual words are columns in the learned dictionary. The matching process is then performed comparing the similarity between vector models.

Publications

The results shown in this thesis are reported from our following publications:

1. T-H Do, S. Tabbone and O. Ramos-Terrades, Noise suppression over bi-level graphical documents using a sparse representation, CIFED 2012, Bordeaux, France.

2. T-H Do, S. Tabbone and O. Ramos-Terrades, Text/graphic separation using a sparse representation with multi-learned dictionaries, ICPR 2012, Tsukuba, Japan.

3. T-H Do, S. Tabbone and O. Ramos-Terrades, New Approach for Symbol Recognition Combining Shape Context of Interest Points with Sparse Representation. ICDAR 2013, Washington DC, USA.

4. T-H Do, S. Tabbone and O. Ramos-Terrades, Document Noise Removal using Sparse Representations over Learned Dictionary, DocEng 2013, Florence, Italy.

5. T-H Do, S. Tabbone and O. Ramos-Terrades, Spotting Symbol using Sparsity over Learned Dictionary of Local Descriptors, the 11th IAPR International Workshop on Document Analysis Systems, 2014, Tours, France.

6. T-H Do, S. Tabbone and O. Ramos-Terrades, Sparse Representation over Learned Dictionary for Visual Vocabulary Construction, in preparation.

Related Works

Chapter 2

Sparsity and Learning Dictionary

Contents

This chapter describes the background of sparse representation and related works involving to the sparsity. In particular, the definitions about the sparse representation and learned dictionary are presented. Detail reviews of state-of-the-art algorithms on finding the sparse representation, on building the dictionary are also shown and discussed carefully. Numerical experiments are performed with the purpose of evaluating the complexity, finding out the advantage as well as disadvantage of these algorithms for each particular problem and to set the choice for our work.

2.1 Sparse Representation

A signal $h \in \mathbb{R}^L$ is strictly or exactly sparse if most of its entries are equal to zero, or the cardinality of the support of the signal $\#\{1 \leq i \leq L | h_i \neq 0\}$ is much less than L. A $k-$sparse signal is a signal that has exactly k nonzero entries. We can present a signal as a linear combination of k columns (or atoms) of a given over-complete dictionary A, such as

$$h = Ax = \sum_{i=1}^{k} a_i x_i \tag{2.1}$$

Mathematically, if $A = \{a_1, a_2, \ldots, a_M\} \in \mathbb{R}^{L \times M}, M \gg L$ is a full$-$rank matrix, then the underdetermined linear system of equations (Equation (2.1)) will have infinitely many different sets of values for the x's (solutions) that satisfy it simultaneously.

For example, the following system

$$\begin{cases} x_1 + 3x_2 - 2x_3 = 5 \\ 3x_1 + 5x_2 + 6x_3 = 7. \end{cases}$$

The solution set $x = \{x_1, x_2, x_3\}$ to this system can be described by the equations: $x_1 = -7x_3 - 1$ and $x_2 = 3x_3 + 2$ with x_3 is the free variable while x_1 and x_2 are dependent on x_3. The different options for the free variables may lead to different descriptions of the same solution set: If x_1 is the free variable, and x_2 and x_3 are dependent, then we have $x_2 = -3/7x_1 + 11/7$ and $x_3 = -1/7x_1 - 1/7$. Thus, each free variable gives the solution set.

In above example, the sets of such these x can be described using mathematical language. However, from the application point of view, one of the main tasks in dealing the above system of equations is to find the proper x that can describe h well comparing with others. In fact, this task is the computing and optimizing a signal approximation by choosing the best dictionary sub-set columns. As the results, to gain this well-defined solution, a function $f(x)$ is added to assess the desirability of a would-be solution x, with smaller values being preferred:

$$(P_f) : \min_x f(x) \text{ subject to } Ax = h$$

If $f(x)$ is the l_0 pseudo-norm $\|x\|_0$ (number nonzero elements in vector x), then the problem (P_f) becomes finding the sparse representation x of h satisfying:

$$(P_0) : \min_x \|x\|_0 \text{ subject to } Ax = h \tag{2.2}$$

In general, solving Equation (2.2) is often difficult (NP-hard problem) and, therefore, is computationally intractable. It is necessary to find sufficiently fast algorithms computing sub-optimally, but '*good enough*' solution. One of the choices is using greedy algorithms, such as Matching Pursuit (MP) [110], Orthogonal-MP (OMP) [137], Weak-MP [167], and Thresholding algorithm. In general, a greedy algorithm is an algorithm that follows solving the problem heuristic of making the locally optimal single term updates with the hope of finding a global optimum. In this case, the set of active columns started from empty is maintained and expanded by one additional column of A at each iteration. The column chosen is column that maximally reduces the residual error l_2 in approximating h from the currently active columns. The residual error l_2 is evaluated after constructing an approximant including the new column; if it falls below a specified threshold, the algorithm terminates.

The other choice is to relax l_0-norm by replacing it with l_p-norms for some $p \in (0, 1]$ or by smooth functions such as $\sum_i \log(1 + \alpha x_i^2)$, $\sum_i x_i^2/(\alpha + x_i^2)$, or $\sum_i (1 - exp(-\alpha x_i^2))$. The exciting algorithm of this family is the FOcal Underdetermined System Solver (FOCUSS) by Gorodnitsky and Rao [64]. In this algorithm, the l_p-norm (for some fixed $p \in (0, 1]$) is represented as a weighted l_2-norm by using Iterative-Reweighed-Least-Squares (IRLS) method.

Another popular strategy is to replace the l_0-norm by the l_1-norm proposed by Donoho *et al* [35]

$$(P_1) : \min_x \|W^{-1}x\|_1 \text{ subject to } Ax = h \tag{2.3}$$

The matrix W is a diagonal positive-definite matrix. A natural choice for the entry in W is $w(i, i) = 1/\|a_i\|_2$. Let $\tilde{x} = W^{-1}x$, then Equation (2.3) is re-formulated as

$$(P_1) : \min_{\tilde{x}} \|\tilde{x}\|_1 \text{ subject to } h = AW\tilde{x} = \tilde{A}\tilde{x} \tag{2.4}$$

Figure 2.1: Left: Minimun $l-1$ solution of $Ax = h$ for $M = 2$. Right: Geometry of l_1 minimization for $M = 3$.

in which $\tilde{A}$ is the normalized version of A. Equation (2.4) is the classic basis pursuit format, and the solution x can be found by de-normalizing $\tilde{x}$. Thus, (P_1) usually be used with a normalized matrix.

Because the purpose is to find the sparse representation in the solution, so, when convex l_0-norm to l_1-norm then basis pursuit has to ensure to recover sparse solutions and that can be explained using a geometric interpretation of basis pursuit as following: Let P is the affine subspace of $\mathbb{R}^M$ of coordinate vectors that recover $h \in \mathbb{R}^L$

$$P = \{x \in \mathbb{R}^M : Ax = h\}$$

A basis pursuit finds in P an element $\bar{x}$ of minimum l_1-norm. It can be found by inflating the l_1-norm ball B_τ by increasing τ until B_τ intersects P.

$$B_\tau = \{x \in \mathbb{R}^M : \|x\|_1 \leq \tau\} \subset \mathbb{R}^M$$

This geometric configuration is depicted for $M = 2$ and $M = 3$ in Figure (2.1). Thus, the optimal solution $\bar{x}$ is likely to have zeros or coefficients close to zeros when it is computed by minimizing an l_1- norm.

The solution for (P_1) problem can by found by some existing numerical algorithms, such as Basis Pursuit by Linear Programming [20], IRLS (*Iterative Reweighed Least Squares*) (for $p = 1$) [28]. Following [109], basis pursuit is computationally more intense than a matching pursuit, but it is a more global optimization that yields representations that can be more sparse.

If there exists some appropriate conditions on A and x, like

$$\|x\|_0 \leq \frac{1}{2}(1 + \frac{1}{\max_{i \neq j} \frac{|a_i^T a_j|}{\|a_i\|_2 \|a_j\|_2}})$$

then Basic Pursuit and OMP give the unique solution of (2.4) and it is also the unique solution of (P_0).

Sometimes, the exact constraint $h = Ax$ is changed by relaxed one by using the quadratic penalty function $Q(x) = \|Ax - h\|_2^2 \leq \epsilon$, with $\epsilon \geq 0$ is the error tolerance. Thus, an error-tolerant version of (P_0) is defined by:

$$(P_0^\epsilon) : \min_x \|x\|_0 \text{ subject to } \|Ax - h\|_2 \leq \epsilon$$

In (P_0^ϵ) the l_2-norm used for evaluation the error of $Ax - h$ can be replaced by other options, as l_1, l_2, or l_∞. We can see one of advantages of this change through noise removal. Assuming

that signal h has noise e with finite energy $\|e\|_2^2 \leq \epsilon^2$, $h = Ax + e$. Solving (P_0^ϵ) can help us to find the solution $\bar{x}$ that bases on it we can find the unknown denoised signal $\bar{h}$ by $\bar{h} = A\bar{x}$.

Similarly, when relaxing l_0-norm to a l_1-norm, we get (P_1^ϵ) known in the literature as *basis pursuit denoising* (BPDN)

$$(P_1^\epsilon) : \min_x \|x\|_1 \text{ subject to } \|Ax - h\|_2 \leq \epsilon \tag{2.5}$$

The solution to (P_1^ϵ) is precisely the solution to the following unconstrained optimization problem

$$(Q_1^\lambda) : \min_x \lambda\|x\|_1 + \frac{1}{2}\|Ax - h\|_2^2 \tag{2.6}$$

where the Lagrange multiplier λ is a function of A, h and ϵ. In the statistical machine learning community, the problem (Q_1^λ) is used for regression over a comprehensive set of features being the columns of A. Its goal is to find a simple linear combination of a few features that could explain the vector output of a complex system h. Thus, solving (Q_1^λ) not only provides a way to get such regression, but it also selects a small subset of features. (Q_1^λ) is well-known under the name LASSO (Least Absolute Shrinkage and Selection Operator) that was conceived by Friedman, Hastie and Tibshirani [170]. The LASSO team and Efron also proposed an effective algorithm, called LARS (Least Angle Regression Stagewise) [40] that guarantees the solution path of (Q_1^λ) is the global optimizer.

The generalized version of (Q_1^λ) has the form in Equation (2.7) where $\rho(.)$ is any '*sparsity-promoting*' function. For example, when $\rho(x) = |x|$, $\mathbf{1}^T\rho(x) = \|x\|_1$, we get Equation (2.6).

$$\min_x \lambda\mathbf{1}^T\rho(x) + \frac{1}{2}\|Ax - h\|_2^2 \tag{2.7}$$

Minimization Equation (2.7) can be treated using various classic iterative optimization algorithms, as Steepest-Descent, Conjugate-Gradient, or interior-point algorithms. However, in the case for high-dimensional problem, these methods perform very poorly [41]. Thus, a new family of numerical algorithms has been developed, called Iterative-Shrinkage algorithms. Some of algorithms in this family include Stage-wise Orthogonal-Matching-Pursuit (StOMP) algorithm [26], EM and Bound-Optimization approaches [57, 55], IRLS-based shrinkage algorithm, and Parallel-Coordinate-Descent(PCD) algorithm [41].

2.2 Pursuit Algorithms

In this section, we will describe deeper algorithms mentioned in Section (2.1). Following the problem that we want to deal with (P_0), (P_0^ϵ); (P_1), (P_1^ϵ); or (Q_1^λ), we divide algorithms into three classes including the greedy matching pursuit, the basis pursuit, and the l^1 lagrangian pursuit, respectively.

2.2.1 Greedy Matching Pursuits

Matching pursuit algorithm introduced by Mallat and Zhang [110] computes signal approximations from the over-complete dictionary. In this method, the column (or atom) of the dictionary is selected iteratively one by one.

Let $A = \{a_1, a_2, ..., a_M\} \in \mathbb{R}^{L\times M}$ is an over-complete dictionary having a unit norm and A includes M linearly independent columns. The matching pursuit begins by projecting the

signal $h = \{h_1, h_2, ..., h_L\}$ on a column $a_i \in A$ and computing the residue r^1 with $r^0 = h$.

$$\begin{aligned} h &= a_i \times \sum_{j=1}^{L} h_j a_i[j] + r^1 \\ &= \langle h, a_i \rangle a_i + r^1 \end{aligned}$$

Because all columns in A are linearly independent, so r^1 is orthogonal to a_i. It means

$$\|h\|^2 = |\langle h, a_i \rangle|^2 + \|r^1\|^2$$

To minimize residue $\|r^1\|^2$, we need to choose the column a_i such as $|\langle h, a_i \rangle|$ is the maximum. This procedure is iterated by sub-decomposing the residue r^1. For instance, assuming m-*th* order residual r^m is already calculated for $m \geqslant 0$, then in the next iteration, we need to choose the column a_{i_m} that maximize $|\langle r^m, a_{i_m} \rangle|$.

Weak-Matching-Pursuit is matching pursuit but rather than searching for the largest inner-product value $|\langle r^m, a_{i_m} \rangle|$ the selected column a_{i_m} is the column that satisfies Equation (2.9) where $\alpha \in (0, 1]$.

$$|\langle r^m, a_{i_m} \rangle| \geqslant \alpha \sup_{j=1,...,M} |\langle r^m, a_j \rangle| \tag{2.9}$$

Orthogonal Matching Pursuit improves the matching pursuit approximations by orthogonalizing the directions of projection. The selected column a_{i_m} now have to be orthogonal to the previously selection columns $\{a_{i_k}\}_{k=1,...,m}$. This can be done by projecting the residues on an orthogonal family $\{u_k\}_{1 \leq k < m}$ computed from $\{a_{i_k}\}_{1 \leq k < m}$ using Gram-Schmidt algorithm. Let $u_0 = a_{i_0}$, then Gram-Schmidt algorithm computes u_m by Equation (2.10).

$$u_k = a_{i_k} - \sum_{l=0}^{k-1} \frac{\langle u_l, a_{i_l} \rangle}{\|u_l\|^2} u_l \tag{2.10}$$

and the residue r^k is projected on u_k instead of a_{i_k}

$$r^k = \frac{\langle r^k, u_k \rangle}{\|u_k\|^2} u_k + r^{k+1} \tag{2.11}$$

Summing the Equation (2.11) for $0 \leq k < m$, and with the $r^0 = h$ yields:

$$h = \sum_{k=0}^{m-1} \frac{\langle r^k, u_k \rangle}{\|u_k\|^2} u_k + r^m$$

Let P_{V_k} be the orthogonal projector on the space V_k generated by $\{u_k\}_{0 \leq k < m}$, then for any $k \geq 0$ the residual r^k is the component of h that is orthogonal to V_k:

$$r^m = h - \sum_{k=0}^{m-1} \frac{\langle r^k, u_k \rangle}{\|u_k\|^2} u_k \tag{2.12}$$

The orthogonal matching pursuit select a_{i_m} that maximize $|\langle r^m, a_{i_m} \rangle|$ with r^m is calculated using Equation (2.12).

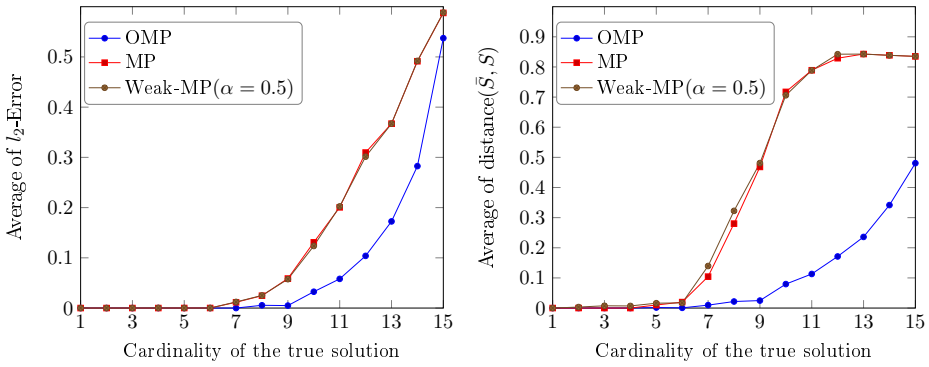


Figure 2.2: Performance of the greedy matching pursuits algorithms

Numerical Demonstration of Greedy Matching Pursuits We conclude the discussion on the greedy matching pursuits algorithms by comparing their behaviors on a simple case. To do that, a random dictionary of the size 50×100 with entries drawn from the normal distribution, and a sparse vectors x with *independent and identically-distributed* (iid) random supports of cardinalities in the range $[1; 15]$ are created. Using l_2-norm to normalize the columns of random dictionary, we get the normalize dictionary $A \in \mathbb{R}^{50\times100}$ and a vector $h = Ax$ is computed once A, x is generated. Now, we consider A, h as the input for the MP, OMP or weak-MP algorithm and perform to seek the solution $\bar{x}$ that is the most close to original x.

To verify the performance of the greedy matching pursuit algorithms, two measures, named l_2-error, and the support recovery are used. The l_2-error is computed as the ratio $\frac{\|\bar{x}-x\|^2}{\|x\|^2}$ that indicate the l_2-proximity between the two solutions: the approximation solution $\bar{x}$ and the ideal solution x. The distance between the supports of the two solutions is calculated as in Equation (2.13) where $\bar{S}, S$ are the supports of $\bar{x}, x$, respectively (we recall that the support of a sparse representation solution is its number of non-zero entries).

$$\text{distance}(\bar{S}, S) = \frac{\max\{|\bar{S}|\ , |S|\} - |\bar{S} \cap S|}{\max\{|\bar{S}|\ , |S|\}} \tag{2.13}$$

If the two supports are the same, the distance is zero and if a distance is close to 1, then the two supports are entirely different.

We have done this experiment 200 times and calculated the average values. Figure (2.2) presents the results. Overall, all algorithms perform well for low-cardinalities, and the best performing method is the OMP. The difference between the MP and Weak MP is quite slight.

2.2.2 Basis Pursuit

Basis Pursuit As we know from the previous section, a matching pursuit performs a local optimization. However, the basis pursuit performs a more global optimization. To do that, the convex optimization as in Equation (2.4) is rewritten as a linear programming problem. Before showing how to recast the convex optimization as linear programming, we will review a standard-form linear programming problem that is introduced by *Gill*

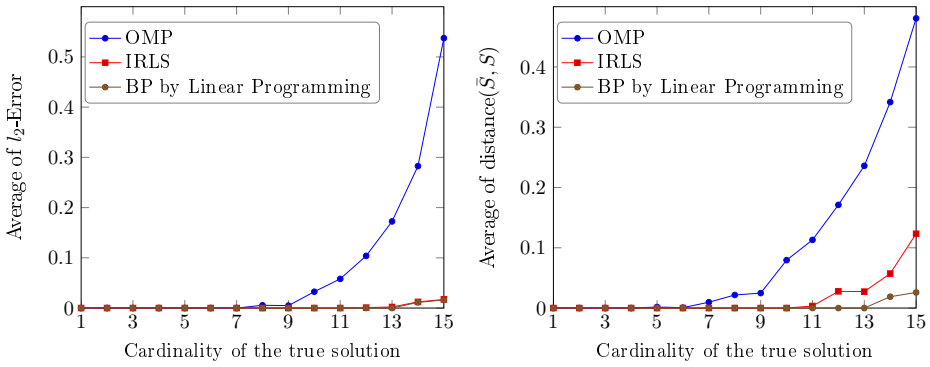


Figure 2.3: Performance of IRLS and BP (using Matlab's Linear-Programming) algorithms

et al [63]. Following [63], linear programming problem is a constrained optimization over positive vector $d = \{d_1, d_2, ..., d_L\} \in (\mathbb{R}^+)^L$. Let $b = \{b_1, b_2, ..., b_K\} \in \mathbb{R}^K, K < L$, $c = \{c_1, ..., c_L\} \in \mathbb{R}^L$ be a non-zeros vector, and $\Phi \in \mathbb{R}^{K \times L}$ is a matrix. Linear programming finds $d \in (\mathbb{R}^+)^L$ such as d is the solution of the minimization problem

$$\min_{d \in (\mathbb{R}^+)^L} \sum_{i=0}^{L-1} d_i c_i \text{ subject to } \Phi d = b$$

Now, coming back to the basis pursuit optimization and without loss of generality, let A be a normalized matrix, we have

$$\min_x \|x\|_1 \text{ subject to } Ax = h \tag{2.14}$$

If $x \in \mathbb{R}^M$ is decomposed into 2 slack variables $t, v \in \mathbb{R}^M$ such as $x = t - v$, and defining $\Phi = (A, -A) \in \mathbb{R}^{L \times 2M}$, $c = 1$, $d = (t, v) \in \mathbb{R}^{2M}$, and $h = b$ then Equation (2.14) is rewritten as linear programming since

$$\sum_{i=0}^{2M-1} d_i c_i = \|x\|_1 \text{ and } \Phi d = At - Av = h$$

Once the basis pursuit is reformulated as a linear programming, it can be solved using modern interior-point methods, simplex methods that are better than the greedy algorithms as they obtain the global solution of a well-defined optimization problem.

Numerical Demonstration of Basis Pursuit To evaluate the solution of the Basis Pursuit, two relaxation-based algorithms are performed and compared using the same example and the measures mentioned in Section (2.2.1). The first one is the IRLS for $p = 1$, and the second one is the linear-programming by Matlab.

Figure (2.3) shows evidence that the IRLS and BP solver linear-programming by Matlab provide a better approximation to the problem in Equation (2.4) in both evaluated measures of quality compared to OMP. However, Table (2.1) presents that computing the solution of Equation (2.4) by the BP and IRLS take much more time compared to the OMP (the best algorithm in greedy matching algorithms).

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Average
OMP	0.249	2.883	0.139	0.087	0.105	3.162	0.722	0.206	0.205	0.210	0.661	3.243	0.611	5.742	1.415	**2.955**
IRLS	13.923	12.781	22.167	13.629	13.324	14.383	14.063	13.640	13.993	13.744	17.193	15.290	15.305	18.444	18.901	**15.385**
BP by Linear Pro	36.592	5.373	4.210	4.516	4.639	3.7900	4.396	4.3300	4.338	4.491	4.861	4.221	4.415	4.506	4.243	**6.595**

Table 2.1: Time $\times 10^{-2}$ (seconds) performance of difference methods corresponding to the cardinality of the true solution

2.2.3 l^1 Lagrangian Pursuit

The Equation (2.6), that is called hereafter by the name l^1 lagrangian pursuit or lagrangian basis pursuit, is often preferred because of its close connection to convex quadratic programming, for which many algorithms are available. Several algorithms include the gradient pursuits [12], the *Gradient Projection for Sparse Reconstruction* (GPSR) [56], and the *Stage-wise Orthogonal Matching Pursuit* (StOMP) method [37]. In [12], a greedy element selection is done similarly as done in MP and OMP. However the costly orthogonal projection is accomplished by applying directional optimization schemes that are based on the gradient, the conjugate gradient, and an approximation to the conjugate gradient. The authors in [56] *Figueiredo et al* deal with the l^1 lagrangian pursuit problem by reformulating (Q_1^λ) as the bound-constrained quadratic program, and then solve it using a Barzilai-Borwein gradient projection algorithm originally developed in the context of unconstrained minimization of a smooth nonlinear function. StOMP [37] allows to find the approximate sparse solution of underdetermined systems with the property either that the dictionary A is random or that the non-zeros in x are randomly located, or both.

Another approach that is widely used by many researchers to deal with (Q_1^λ) is using an iterative procedure based on shrinkage (also called soft thresholding).

Soft thresholding is an iterative algorithm that solves (2.6) with a soft thresholding to decrease the l_1-norm of coefficients x, and a gradient descent to decrease the value of $\|h - Ax\|$.

1 *Initialization* Choose $x_0 = 0$, let $k = 0$;

2 *Gradient step* Update

$$\bar{x}_k = x_k + \gamma(A^T h - A^T A x_k) \tag{2.15}$$

where $\gamma \leq 2\|A^T A\|^{-1}$

3 *Soft thresholding* Compute the components $x_{k+1}[p]$ from $\bar{x}_k$

$$x_{k+1}[p] = \rho_{\gamma\lambda}(\bar{x}_k[p]) \text{ where } \rho_{\gamma\lambda}(a) = a\max(1 - \frac{\gamma\lambda}{|a|}, 0) \tag{2.16}$$

4 *Stop* If $\|x_k - x_{k+1}\| \leq \epsilon$ then stop the iterations, otherwise set $k = k + 1$ and go back to *Gradient step*

The condition $\gamma \leq 2\|A^T A\|^{-1}$ is the convergence condition.

Backprojection If Equation (2.16) is replaced by an orthogonal projection on the support of x, named $\tilde{\Lambda}$ as

$$x_{k+1}[p] = \begin{cases} 0 & \text{if } p \notin \tilde{\Lambda} \\ \bar{x}_k[p] & \text{if } p \in \tilde{\Lambda} \end{cases}$$

then approximation error is reduced by a backprojection that recovers $x_{\tilde{\Lambda}}$ from $\bar{x}$. The convergence is then guaranteed and depends on $A_{\tilde{\Lambda}}$ being columns of A at the index $\tilde{\Lambda}$.

IRLS-based iterative shrinkage algorithm starts with $x_0 = 1$ and replaces Equation (2.16) by

$$x_{k+1}[p] = \bar{x}_k[p]\frac{x_k[p]^2}{\gamma\lambda|x_k[p]| + x_k[p]^2}$$

PCD iterative-shrinkage algorithm replaces the gradient step by

$$\bar{x}_k = x_k + \text{diag}(A^TA)^{-1}(A^Th - A^TAx_k)$$

and then the iterative equation (Equation (2.16)) becomes

$$x_{k+1}[p] = (1-\gamma)x_k[p] + \gamma\rho_{\text{diag}(A^TA)^{-1}\lambda}(\bar{x}_k[p])$$

Numerical Demonstration of l^1 Lagrangian Pursuit To present the behavior of some methods used to solve l^1 lagrangian pursuit, we have done a synthetic test on three algorithms, named the soft thresholding, the IRSL-based and the PCD, to minimize the following function in the case $\lambda = 1$

$$\min_x \lambda\|x\|_1 + \frac{1}{2}\|h - Ax\|_2^2$$

In this experiment, to reduce the computing time in the multiplication A to its adjoint, we use a Hadamard matrix to construct A as suggested by Elad in [41] because of the availability of a Fast-Hadamard Transform operation. A is constructed by $A = PHQ$ where $H \in \mathbb{R}^{2^{18}\times 2^{18}}$ is a Hadamard matrix, $Q \in \mathbb{R}^{2^{18}\times 2^{18}}$ is the diagonal matrix and $P \in \mathbb{R}^{2^{14}\times 2^{18}}$ is then identity matrix. The ideal solution x_0 is a sparse vector with 1500 *iid* entries in random locations. Once A and x_0 are created, the vector h is computed by $h = A \times x_0 + v$ where v is a random vector with *iid* entries, drawn from a Gaussian distribution with zero mean and standard−deviation equals 0.2.

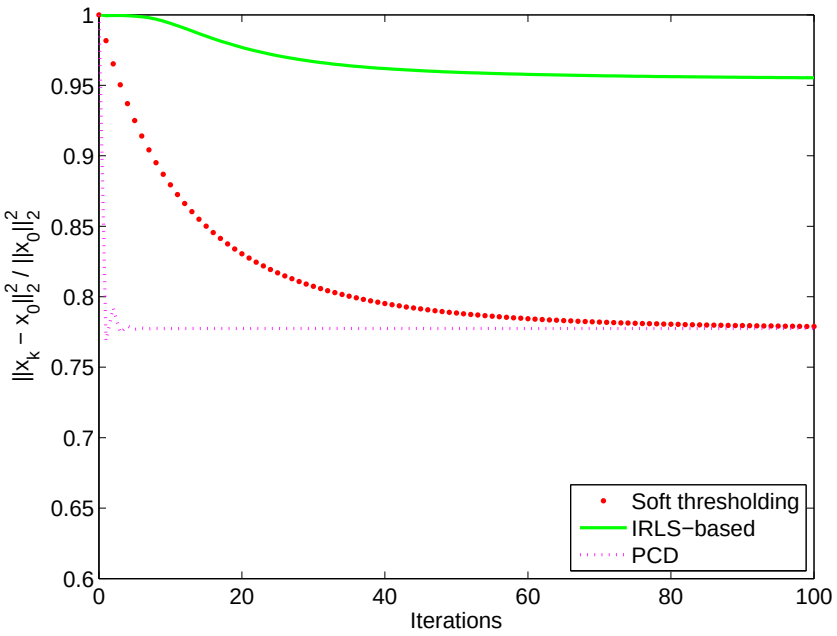


Figure 2.4: The quality of the obtained solutions

The quality of the obtained solutions is compared using the ratio $\frac{\|x_k - x_0\|_2^2}{\|x_0\|_2^2}$ that indicates the l_2 proximity between the approximation x_k at the iteration number k$-th$ and the ideal solution x_0. The value of this ratio should below 1 and close to 0 to indicate that the relative error in the solution is small. Figure (2.4) presents the results obtained by three methods with 100 iterations. As we can see, the weakest performing method among three is IRLS-based algorithm and the best performing one is PCD. In fact, IRLS-based algorithm tends to get stuck while PCD provides nearly perfect convergence results after 10 iterations.

2.3 Learning Dictionaries

The effect in finding the sparse representations of signals depends directly on a dictionary A, and the best dictionary is the dictionary leading to the sparsest representation. Hence the larger the dictionary, the higher the computation cost time for calculating the sparse representation. Therefore, there is a trade-off between the size of the dictionary and the computation time. Based on detailed theoretical analysis about the representation of the signal with a fixed number of non-zeros from the transform coefficients (or best M-term approximation), a category of redundant dictionaries has been proposed in the literature and two well-known redundant dictionaries being ridgelets and curvelets.

Ridgelets and curvelets are multiscale orientation-selective transforms that were developed to overcome the weakness of the wavelet transform in the case dealing with lines, curves, and edges in images. They have had considerable success in a comprehensive range of image processing applications, including denoising [158, 147, 70, 19], component separation [159, 71], detection [81, 72], inpainting [44, 52], and blind source separation [13, 15]. Other constructions belonging to this family of multiscale orientation-selective transforms have been investigated as the contourlets [34, 33, 50], platelets [179], complex directional wavelet transforms [53, 54, 174], shearlets [90], bandlets [139, 140], dual-tree complex wavelet transforms [87, 152], directionlets [175], grouplets [108].

One reason for the success of these pre-constructed dictionaries is the availability of fast transforms algorithms, which are available in noncommercial software packages. However, they are limited to the images that they are designed to handle. Thus, they are difficult to be used for a new family of images. For example, in the case of graphical symbols images. This motivates in researching to find out another approach for obtaining dictionaries from the underlying empirical data that can be able to adapt better to any family of images.

The initial study is taken by *Olshausen et al* [132, 133]. Later work by Lewicki [93], Lesage and Engan [46], Delgado [31], Gribonval [92], Aharon [2], and others. This section describes the core of learning methodology for constructing dictionary A and four well-known learning algorithms including the K-SVD by *Aharon et al.*, the Method of Optimal Directions (MOD) by *Engan et al.*, the online learning dictionary (OLD) by *Marial et al.*, and the RLS-DLA by *Skretting and Engan.*

2.3.1 Core Idea for Learning Dictionary

In a general learning methodology, a family S signals $\{h_j\}_{j=1}^{S}$ is considered as the training database. The goal is to find a dictionary $A \in \mathbb{R}^{L \times M}$ in which each signal $h_j \in R^L$ has an optimal sparse approximation $\bar{h}_j \backsimeq Ax_j$ satisfying $\|\bar{h}_j - h_j\|^2 \leq \epsilon$, or finding:

$$\min_{A,x_j} \sum_{j=1}^{S} \|x_j\|_1 \text{ subject to } \|h_j - Ax_j\|_2 \leq \epsilon, \text{ for all } j = 1, .., S \tag{2.17}$$

If we choose to constrain the sparsity, then Equation (2.17) is rewritten as following:

$$\min_{A,x_j} \sum_{j=1}^{S} \|h_j - Ax_j\|_2^2 \text{ subject to } \|x_j\|_0 \leq k_0, \text{ for all } j = 1, .., S \tag{2.18}$$

In terms of matrix factorizations, if we consider $H \in \mathbb{R}^{L\times S}$ as the matrix formed from all the database vectors column-wise, and similarly, all the corresponding sparse representations into a matrix $X \in \mathbb{R}^{M\times S}$, then in the case $\epsilon = 0$, the problem of discovering an underlying dictionary (Equation (2.18)) is thus the same as the problem of discovering a factorization of the matrix Y as AX where X has sparse columns.

This dictionary can be obtained by the learning process. This process iteratively adjusts A via two main steps: *sparse coding stage* and *update dictionary stage*. In the sparse coding stage, all sparse representations $X = \{x_j\}_{j=1}^{S} \in \mathbb{R}^{M\times S}$ of $H = \{h_j\}_{j=1}^{S} \in \mathbb{R}^{L\times S}$ are found by solving equation (2.5) on condition that A is fixed. Any methods mentioned in Section (2.2) can be used. In the update dictionary stage, an updating rule is applied to optimize the sparse representation of all examples. In general, a way of updating the dictionary in the second step is different from each learning algorithm to others.

2.3.2 The K-SVD Algorithm

In the K-SVD algorithm, proposed by [2], the updating rule is to make a modification in the dictionary's columns. At this step, each columns a_{j_0} of A is updated sequentially such that the residual error (2.19) is minimized, where X and $\{a_1, ...a_{j_0-1}, a_{j_0+1}, ..., a_M\}$ are fixed,

$$\begin{aligned} \|H - AX\|_F^2 &= \|H - \sum_{j=1}^{M} a_j x_j^T\|_F^2 \\ &= \|(H - \sum_{j\neq j_0} a_j x_j^T) - a_{j_0} x_{j_0}^T\|_F^2 \\ &= \|E_{j_0} - a_{j_0} x_{j_0}^T\|_F^2 \end{aligned} \tag{2.19}$$

In the description (2.19), $x_{j_0}^T \in \mathbb{R}^S$ is the j_0-*th* row in X. Because X and all the columns of A are fixed excepted the column a_{j_0}, so $E_{j_0} = H - \sum_{j\neq j_0} a_j x_j^T$ is fixed. It means that the minimum error $\|H - AX\|_F^2$ depends only on the optimal a_{j_0} and $x_{j_0}^T$. This is the problem of approximating a matrix E_{j_0} with another matrix which has a rank 1 based on minimizing the *Frobenius* norm. The optimal solutions $\tilde{a}_{j_0}, \tilde{x}_{j_0}^T$ can be given by the SVD (*Singular Value Decomposition*) of E_{j_0} of rank r_1, namely

$$\tilde{a}_{j_0}\tilde{x}_{j_0}^T = Q_1 \tilde{U} Q_2$$

where $Q_1 = \{q_1^1, ..., q_n^1\}, U = \text{diag}(\sigma_1, \sigma_2, ..., \sigma_{r_1}), Q_2 = \{q_1^2, ..., q_l^2\}$ is the singular value decomposition (SVD) of E_{j_0}: $E_{j_0} = Q_1 U Q_2$; and $\tilde{U}$ is the same matrix as U except that it contains only one singular values σ_1 (the other singular values are replaced by zero). This means $\tilde{a}_{j_0} = q_1^1$ and $\tilde{x}_{j_0}^T = \sigma_1 q_1^2$. However, the new vector $\tilde{x}_{j_0}^T$ is very likely to be filled, implying that the number of non-zeros in the representation of X is increased, or the condition about the sparsity of X can be broken.

This problem can be overcome as follows. Define the group of indices where $x_{j_0}^T$ is nonzero:

$$\omega_{j_0} = \{i | 1 \leq i \leq S, x_{j_0}^T(i) \neq 0\}.$$

and a matrix $\Omega_{j_0} \in \mathbb{R}^{S \times |\omega_{j_0}|}$ with $\Omega_{j_0}(\omega_{j_0}(i), i) = 1$ and zeros elsewhere. Let

1. $x_{j_0}^R = x_{j_0}^T \Omega_{j_0}$, $x_{j_0}^R \in \mathbb{R}^{|\omega_{j_0}|}$
2. $E_{j_0}^R = E_{j_0} \Omega_{j_0}$, $E_{j_0}^R \in \mathbb{R}^{L \times |\omega_{j_0}|}$

and return to equation (2.19), this equation is equivalent to the minimization of

$$\|E_{j_0}\Omega_{j_0} - a_{j_0} x_{j_0}^T \Omega_{j_0}\|_F^2 = \|E_{j_0}^R - a_{j_0} x_{j_0}^R\|_F^2 \tag{2.20}$$

Note that the solution of (2.20) $\tilde{x}_{j_0}^R$ has the same support as the original $\tilde{x}_{j_0}^T$, and the optimal values $\tilde{x}_{j_0}^R, \tilde{a}_{j_0}$ can be obtained by finding the SVD of a subset of the columns of the error matrix $E_{j_0}^R$ of rank r_2: $E_{j_0}^R = GDV^T$. The solution for $\tilde{a}_{j_0}$ is defined as the first column of G, and the coefficient vector $\tilde{x}_{j_0}^R$ as the first column of V multiplied by d_1, with $D = \text{diag}(d_1, d_2, ..., d_{r_2})$. More details about the K-SVD algorithm can be found in algorithm (2)

Algorithm 2 Learning algorithm K-SVD

INPUT: $A_{(0)} \in \mathbb{R}^{L \times M}$; $H = \{h_i\}_{i=1}^S$; $k = 0$;
1. Initialize: Normalization the columns of matrix $A_{(0)}$;
2. Main Iteration
$k = k + 1$;
while ($\|H - A_{(k)} X_{(k)}\|_F^2$ is not small enough) **do**
- Solve (P_1^ϵ) to find all sparse representation $\{x_i\}_{i=1}^S$ of $\{h_i\}_{i=1}^S$
for ($j_0 = 1$ to M) **do**
- Define: $\omega_{j_0} = \{i | 1 \leq i \leq S, x_{j_0}^T(i) \neq 0\}$
- Calculate E_{j_0} via equation $E_{j_0} = H - \sum_{j \neq j_0} a_j x_j^T$
- Let $E_{j_0}^R$ is the submatrix of E_{j_0} on the columns corresponding to ω_{j_0}
- Calculate SVD of $E_{j_0}^R$: $E_{j_0}^R = GDV^T$.
- Updating : $a_{j_0} = s_1$ and $x_{j_0}^R = d_1 v_1$ with $G = \{g_1, ..., g_L\}$, $V = \{v_1, ..., v_{|\omega_{j_0}|}\}$, $D = \text{diag}(d_1, d_2, ..., d_{r_2})$
end for
end while
OUTPUT: The result $A_{(k)}$

2.3.3 The MOD Algorithm

In the Method of Optimal Directions (MOD) algorithm or iterative least squares dictionary learning algorithm (ILS-DLA), proposed by *Engan et al* [45, 47], the updating rule is to make the mean of the set of norm residuals as small as possible. Assuming that $A_{(k)}$ is the dictionary obtained after *k-th* iteration, MOD update the dictionary by taking the residuals and finding an adjustment matrix $B \in \mathbb{R}^{L \times M}$ such as a minimization of the problem (2.21).

$$\begin{aligned}\min_B \sum_j \|e_j - Bx_j\|_2^2 &= \min_B \sum_j \|h_j - \bar{h}_j - Bx_j\|_2^2 \\ &= \min_B \sum_j \|h_j - A_{(k)}x_j - Bx_j\|_2^2 \\ &= \min_B \sum_j \|h_j - (A_{(k)} + B)x_j\|_2^2 \\ &= \min_B \|H - (A_{(k)} + B)X_{(k)}\|_F^2\end{aligned} \tag{2.21}$$

In Equation (2.21), e_j is the residual vector $e_j = h_j - \bar{h}_j$ where $\bar{h}_j = A_{(i)} \times x_j$ and x_j is the sparse representation of h_j in the dictionary $A_{(k)}$. Once B is found, then $A_{(k+1)} = A_{(k)} + B$.

Analyzing Equation (2.21) we have

$$\begin{aligned}\min_B \sum_j \|e_j - Bx_j\| &= \min_B \|H - (A_{(k)} + B)X_{(k)}\|_F^2 \\ &= \min_B\{\text{trace}([H - (A_{(k)} + B)X_{(k)}]^T[H - (A_{(k)} + B)X_{(k)}])\} \\ &= \min_B\{\text{trace}(((A_{(k)} + B)X_{(k)}X_{(k)}^T((A_{(k)} + B)^T) - 2 \times \text{trace}((A_{(k)} + B)X_{(k)}H^T)\} \\ &= \min_B\{\text{trace}(BX_{(k)}X_{(k)}^TB^T) + 2 \times \text{trace}(B(X_{(k)}X_{(k)}^TA_{(k)} - X_{(k)}H^T))\}\end{aligned} \tag{2.22}$$

Taking the derivative of (2.22) with respect to B and setting to zero yields $BX_{(k)}X_{(k)}^T = HX_{(k)}^T - A_{(k)}X_{(k)}X_{(k)}^T$. Thus, we have

$$A_{(k+1)} = A_{(k)} + B = HX_{(k)}^T(X_{(k)}X_{(k)}^T)^{-1} \tag{2.23}$$

Specifically, more details about the MOD algorithm can be found in algorithm (3)

Algorithm 3 Learning algorithm MOD

INPUT: $A_{(0)} \in \mathbb{R}^{L\times M}$; $H = \{h_i\}_{i=1}^S$; $k = 0$;
1. Initialize: Normalization the columns of matrix $A_{(0)}$;
2. Main Iteration
$k = k + 1$;
while ($\|H - A_{(k)}X_{(k)}\|_F^2$ is not small enough) **do**
- Solve (P_1^ϵ) to find all sparse representation $\{x_i\}_{i=1}^S$ of $\{h_i\}_{i=1}^S$
- Updating : $A_{(k+1)} = A_{(k)} + B = HX_{(k)}^T(X_{(k)}X_{(k)}^T)^{-1}$

end while
OUTPUT: The result $A_{(k)}$

If H is infinitely large, an alternative of ILS-DLA (or MOD), named LS-DLA algorithm can be used, in which the dictionary update step is formulated as following

$$A_{(k)} = (\beta_{(k)}H_{(k-1)}X_{(k-1)}^T + H_{(k)}X_{(k)}^T)(\beta_{(k)}X_{(k-1)}X_{(k-1)}^T + X_{(k)}X_{(k)}^T)^{-1} \tag{2.24}$$

In Equation (2.24), $H_{(k)}$ is the subset of training vectors used at the iteration number $k-th$. We can see if the training set H is the complete finite, then $H_{(k)} = H$, $\beta_{(k)} = 0$, and updating equation (2.24) is the same as the updating equation of MOD (Equation (2.23)).

2.3.4 The Online Learning Dictionary Algorithm

In the online dictionary learning (ODL) algorithm [113], the update rule processes each training vector h in each iteration, or the updated dictionary is calculated from Equation (2.24) with $\beta_{(k)} = 1$, and x is sparse representation of h in $A_{(k-1)}$.

$$A_{(k)} = (H_{(k-1)}X_{(k-1)}^T + hx^T)(X_{(k-1)}X_{(k-1)}^T + xx^T)^{-1}$$

In fact, the computationally demanding calculation of $A_{(k)}$ is done using the warm restart algorithm. Let t_j, s_j be the $j-th$ columns of the $(X_{(k-1)}X_{(k-1)}^T + xx^T)$, $(H_{(k-1)}X_{(k-1)}^T + hx^T)$ matrices, then column number j of updated dictionary $A_{(k)}$, named $a_{(k)}^j$ is calculated as following:

$$u_j = a_{(k-1)}^j + (s_j - A_{(k-1)}t_j)\frac{1}{t_j[j]}$$
$$a_{(k)}^j = \frac{u_j}{\max(\|u_j\|_2, 1)}$$

2.3.5 The RLS-DLA algorithm

The recursive least squares dictionary learning algorithm (RLS-DLA) [155], as ODL algorithm performs an update rule based on processing just one training vector h in each iteration. The improvement of RLS-DLA compared to LS-DLA is that the updated dictionary is calculated directly without using neither matrix $(\beta_{(k)}H_{(k-1)}X_{(k-1)}^T + hx^T)$ nor the matrix $(\beta_{(k)}X_{(k-1)}X_{(k-1)}^T + xx^T)$. More preciously, the updated rule is done as following:

$$A_{(k+1)} = A_{(k)} + \beta_{(k)}r_{(k)}u_{(k)}^T \tag{2.25}$$

where $r_{(k)}$ is the representation error: $r_{(k)} = h - A_{(k)}x$, vector $u_{(k)}$ is calculated by $u_{(k)} = \beta_{(k)}^{-1}C_{(k)}x$. The C-matrix is computed as following:

$$C_{(k)} = \beta_{(k-1)}^{-1}C_{(k-1)} - \alpha_{(k-1)}u_{k-1}u_{k-1}^T, \tag{2.26}$$

with $C_{(0)}$ is the identity matrix, and $\alpha_{(k-1)} = \dfrac{1}{(1 + x^T u_{(k-1)})}$.

2.3.6 Numerical Demonstration of Learning Algorithms

In this section, we will do synthetic experiments to compare several learning algorithms: the MOD, the K-SVD, the ODL, and the RLS-DLA algorithm. In the synthetic experiment, first a random dictionary with *iid* Gaussian entries is created. Normalizing the columns of this dictionary following the l_2-norm, we got a true dictionary $A \in \mathbb{R}^{L\times M}$. Next, a training dataset including $S = 8000$ training signals are generated such as each training signal is a random combination of six columns of A with coefficients drawn from the normal distribution. A random zero-mean Gaussian noise with $\sigma = 0.1$ is also added to each signal and the learning algorithms are run with 100 iterations. In the RLS-DLA algorithm, the value of $\beta_{(k)}$ is increased from $\beta_{(0)} = 0.99$ to 1 as following

$$\beta_{(k)} = \begin{cases} 1 - (1 - \beta_{(0)})(1 - k/\tau)^3 & \text{if } k \le \tau \\ 1 & \text{if } k > \tau \end{cases}$$

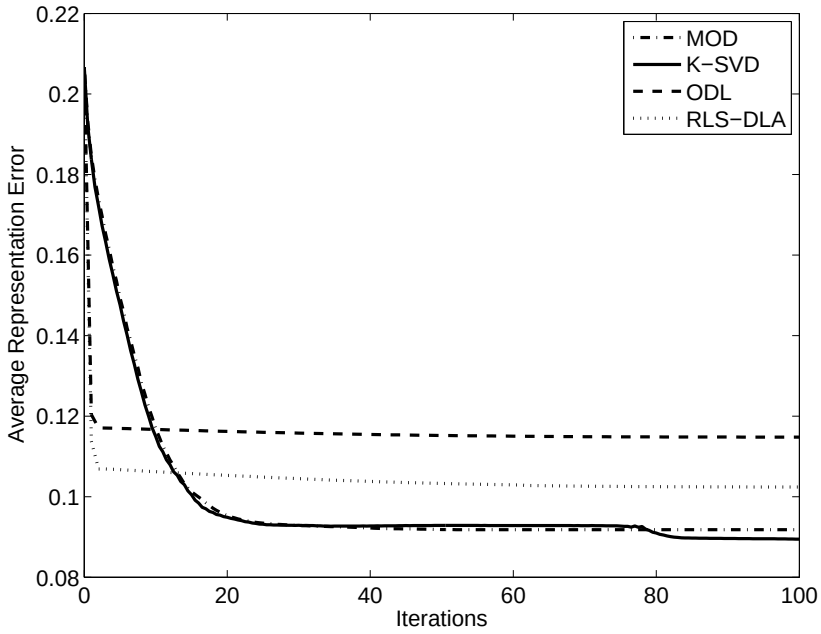


Figure 2.5: Average representation errors obtained at each iteration

where the parameter $\tau = 50 \times \frac{M}{S}$ is the argument value set by experiment. The algorithms are compared to each other using average representation error measure calculated at each iteration as following

$$\frac{1}{S} \times \sum_{j=1}^{S} \|h_j - Ax_j\|_2^2$$

Figure (2.5) shows an average representation errors obtained at each iteration of all compared algorithms. From this figure, we can see that two algorithms K-SVD and MOD behave similarly and comparably, with a small advantage to the K-SVD. We can remark from this figure that these two algorithms perform better than ODL and RLS-DLA also. In addition, in the ODL and RLS-DLA algorithms, the process of updating the dictionary is based on calculating each training signal in each iteration while this process in K-SVD is done on each column of the dictionary. Moreover, the number of columns in the dictionary is much smaller than the number of training signals. Thus, the computing time in ODL and RLS-DLA is higher than K-SVD. Besides, from Equations (2.25) and (2.26) it is clear that the performance of RLS-DLA also depends on the value of $\beta_{(k)}$.

2.4 Conclusion

In this chapter, we have concentrated on finding numerical solutions of sparse representations, by either greedy techniques or relaxation ones. Obviously, each family of methods has its own advantages and disadvantage, and choosing between the various options depends on the practical applications. In addition, this study also proves that the sparse characteristic of numerical solutions depends strongly on the dictionary. The limitations of pre-constructed dictionary as applying on any family of images has raised the motivations in researching that lead to a new family of dictionary generated. The new family of dictionary is named learned dictionaries.

We present and analyze also in this chapter the successes and previous works on constructing learned dictionary. Besides, basic experiments to demonstrate the behaviors of methods have also be done and discussed. Based on these experiments, we notice that both OMP and K-SVD algorithms provide better performances than other algorithms in most of the cases. Although, comparing with the basis pursuit algorithms, OMP does not provide a better approximation to the solution, but its computing cost time is lower. Moreover, by changing the stopping rule of OMP like accumulating nonzero elements in the solution vector until the reconstruction error is less than ϵ, the OMP can be used to find the approximate solutions instead of exact ones. This minor change is very easy to implement and use. In conclusion, from the merits of OMP and K-SVD algorithms, we decide to use these algorithms in this thesis.

Contributions on Document Analysis

Chapter 3

Denoising Graphical Documents

Contents

In this chapter, we propose a new algorithm for denoising document images using sparse representations. Using a training set, this algorithm is able to learn the main document characteristics and also, the kind of noise included into the documents even if the model of noise is unknown. In this perspective, we propose to model the noise energy based on the normalized cross-correlation between pairs of noisy and non-noisy documents. Experimental results on several datasets demonstrate the robustness of the proposed method compared to the state-of-the-art.

3.1 Introduction

The performance of many pattern recognition techniques applied on images depends on an accurate control of the noise included into the image. In the case of document image analysis applications, the kind of noise is different compared to the noise found in natural scenes images that are generated by devices like digital cameras or similar.

Therefore, the problem of document de-noising has been tackled since the very beginning. There is a vast literature on methods proposing solutions as summarized in Figure (3.1) that can be divided into two basic approaches being spatial filtering methods and transform domain filtering methods. Spatial filters are traditional approaches to remove noise from images and they can be further classified into non-linear and linear filters. Spatial filters use a low pass filtering on groups of pixels with the assumption that the noise occupies the higher region of the frequency spectrum. In fact, spatial filters remove noise to a reasonable extent but at the cost of blurring sharp edges which in turn makes the edges in pictures invisible, and linear filters as mean filter perform poorly in the presence of signal-dependent noise. In addition, the wiener filtering [74] method needs a priori information about the noise; whereas a practical application may not have

the required information about the variance of the noise or the kind of noise. To overcome the drawback of spatial filters, a variety of other filters such as weighted median [184], conditioned rank selection [69], and relaxed median [66] have been developed. However, they only work well with the presence of impulse noise and neither of them is efficient for other types of noise like white noise or noise arising from printing, photocopying and scanning processes. This kind of noise not only generally causes undesirable document appearance, but also has a bad influence on document processing performance.

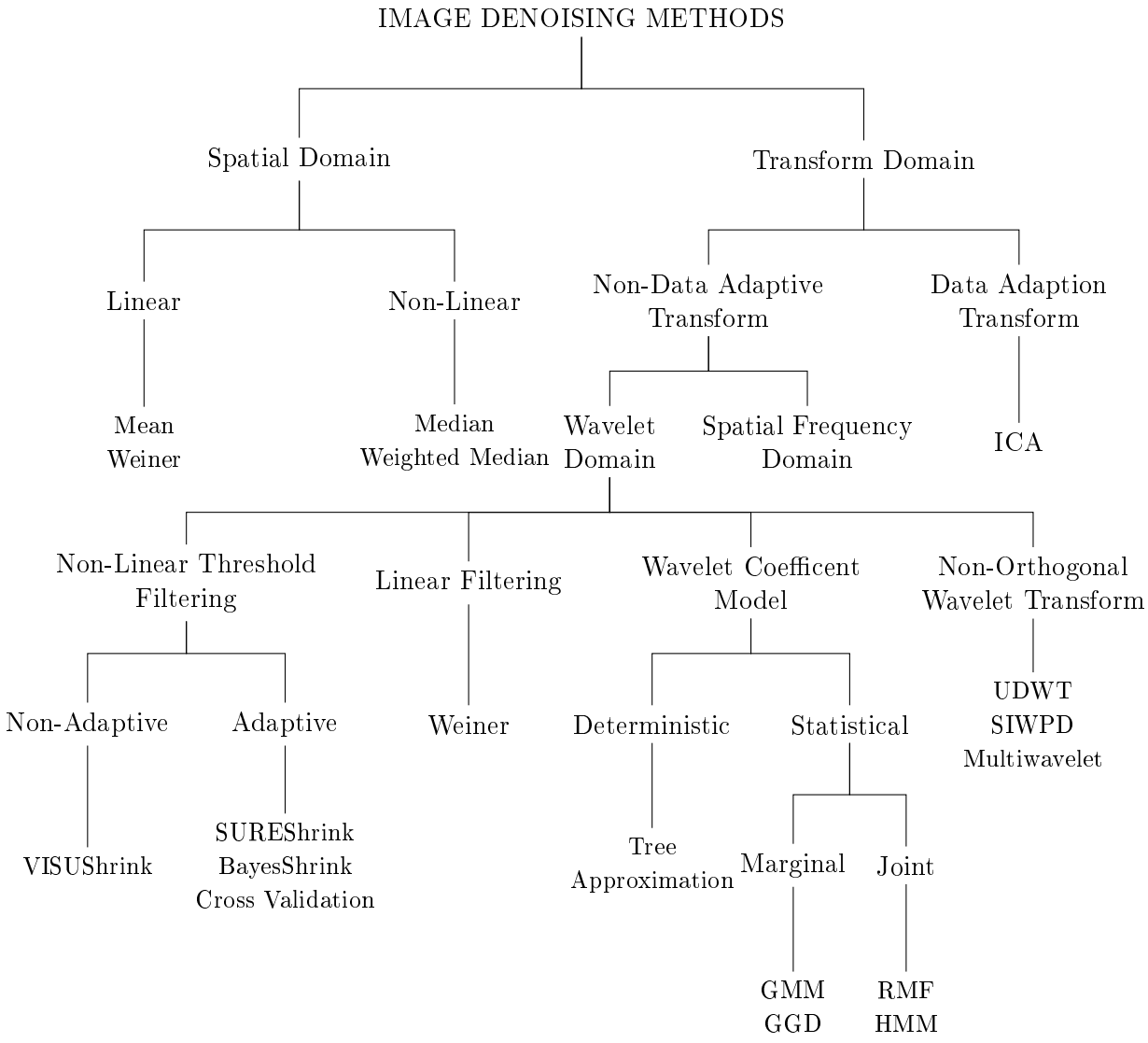


Figure 3.1: Classification of Image Denoising Methods (extracted from [124]).

The transform domain filtering methods are classified into data adaptive and non-adaptive transforms depending on the choice of the basic functions. More detail discussion about the advantage and dis-advantage of each transform domain filtering method can be found in the work of Motwani *et al.* [124]. In the transform domain, Multi Resolution Analysis (MRA) methods [158] with sparse transforms recently obtain good performance in de-noising gray-scale images with white noise. Sparse transforms and MRA methods are applied to a wide range of image pro-

cessing problems as image compression, image restoration and image denoising [158]. These methods have proven to perform well in terms of Mean Square Error (MSE) measure as well as Peak Signal-to-Noise Ratio (PSNR) measure for essentially white noise. Moreover, sparse transforms, like curvelets, contourlets, wedgelets, bandelets, or steerable wavelets, have also been successfully applied in document images for removing noisy edges, showing that sparse representation can effectively be used for denoising purposes. Sparse transforms represent images as linear combinations of atom functions of a given particular dictionary. However, the overall performance of these methods depends on two factors: first of all, a *a priori* knowledge about images which drives the choice of dictionary functions and secondly, the kind of noise found in document images. Recently, curvelets transform has been applied in document denoising with a relative high degree of success [71]. In that approach, the authors take advantage of directional properties of curvelets transform to denoise degraded graphic documents. The results obtained in that work show an improvement in removing noise for document images comparing with other state-of-the-art methods. However, curvelet is one of the pre-defined dictionaries, and therefore can only work well with some kinds of noise and cannot be adapted to arbitrary noise models.

In this chapter, we address the task of document denoising by proposing a method that overcomes the difficulties found in denoising methods based on MRA. On the one hand, we apply the K-SVD algorithm to learn a proper set of atom functions adapted to both document characteristics and document noise. On the other hand we propose an energy noise model that allows us to easier set the threshold required for noise removal even if the noise model is unknown. To evaluate the proposed method, we have compared it with three existing methods being median filter [29], morphological filters [116], and curvelets transform [158]. Experimental results on several datasets have demonstrated the robustness of our approach compared to the state-of-the-art.

The remainder of this chapter is organized as follows. We briefly state the problem in the framework of sparse representations in Section 3.2. Then, we recall document degradation models used in experiments in Section 3.3 and the proposed energy noise model in Section 3.4. Afterward, we discuss the experimental results in Section 3.5 and finally, we give our conclusions and further works in Section 3.6.

3.2 Problem Statement

Asumming that $h^0 \in \mathbb{R}^L$ is the original image (an ideal image without noise), and $h \in \mathbb{R}^L$ is its noisy version obtained by

$$h = h^0 \odot e$$

where e is the contaminating noise and $\odot$ is any composition of two arguments, for example $+$ for additive noise, $\times$ for multiplicative noise. This formulation is known as denoising and it is a long-standing problem in image processing. Many algorithms have been proposed to deal with the problem of noise removal, and the final aims of all algorithms of course is to remove noise from h to return as close as possible to the original image h^0. Denoising can be efficiently attacked by sparse representation over-complete dictionary. Let the finite energy of the noise be $\|e\|_2^2 \leq \epsilon^2$, then solving (P_1^ϵ) (Equation (3.1)) can help us to find the solution $\bar{x}$ that based on it we can calculate the unknown denoised image $\bar{h}$ by $\bar{h} = A\bar{x}$.

$$(P_1^\epsilon) : \min_x \|x\|_1 \text{ subject to } \|Ax - h\|_2 \leq \epsilon \tag{3.1}$$

Naturally, Equation (3.1) needs a proper choice of the dictionary A and the value of ϵ. From

Section (2.3) we know if the dictionary A is constructed from a training database, its columns therefore are generated from the underlying empirical data, rather than from some theoretical model. Thus, the dictionary can be adapted well to the image properties. We will see in details how we constructed the learned dictionary A for the noise removal in Section (3.4.1). However how to choose the best value of ϵ is a difficult problem especially when we do not know exactly what kind of noise is contained into the documents.

We will see in Section (3.4.2), for some specific noises like the white noise or Noise Spread, that some studies have been done to find the way of calculating the value of ϵ. However these studies cannot be applied for other noise models, such as a Kanungo noise, a real noise obtained by the scanning processes or an arbitrary unknown noise. Thus, the contribution in this chapter is about proposing an energy noise model which allows us to easier set the threshold ϵ using the normalized cross-correlation between pairs of noisy and non-noisy documents.

3.3 Document Degradation Models

Inspired by several authors like in [84], we have used their document noise models for evaluating the proposed method. In 1993, Kanungo *et al.* [84] introduced a statistical model for document degradation showing quite realistic results, which gave the following three reasons justifying research on noise models. First of all, noise modeling allows studying recognition algorithms in general, as a function of the perturbation of the input data. Secondly, it permits the evaluation of any algorithm depending on the degradation level. Thirdly, knowledge of the degradation model can enable us to design algorithms for image restoration. More recently, [8] has proposed the Noise Spread model, inspired on the physics of image acquisition process. According to this model the acquired image is obtained as a result of convolving the source image with the sensor function defined by the Point Spread Function with white noise.

Since the Kanungo noise model [84, 85] is a statistical model, widely used to verify the robustness of document image analysis methods to noise, we use it as noise model in this thesis. However, the extension to another model is easy as our method is enabled to denoise documents for which the model of noise is unknown. One of the advantage of this degradation model is it permits to generate degraded images controlled by the parameter models, and the qualitative results of these images range from quite realistic noisy images to unrealistic, or even highly degraded images in function of the parameters set up. For instance, in Figure (3.1), symbol images (b) and (d) provide more realistic symbol degradation than (c), (e), (f) and (g).

Bi-level images are represented by white background pixels and black foreground pixels representing the different entities of the document. The Kanungo model needs six parameters α_0, α, β_0, β, η, and k as a function of the pixel distance to the shape boundaries to degrade a binary image. α and α_0 control the probability of flipping a foreground pixel to a background pixel, in other words, these two parameters provide the probability of changing a black pixel to a white one. Similarly, parameters β and β_0 control the probability of changing a background (white) pixel with a foreground (black) pixel. In addition, these two probabilities exponentially decay on the distance of each pixel to the nearest boundary pixel. In contrast, the parameter η is a constant value added to all pixels regardless their relative position to shape boundaries. Finally, the last parameter k is the size of the disk used in the morphological closing operation. The whole process of image degradation can be summarized in the following algorithm:

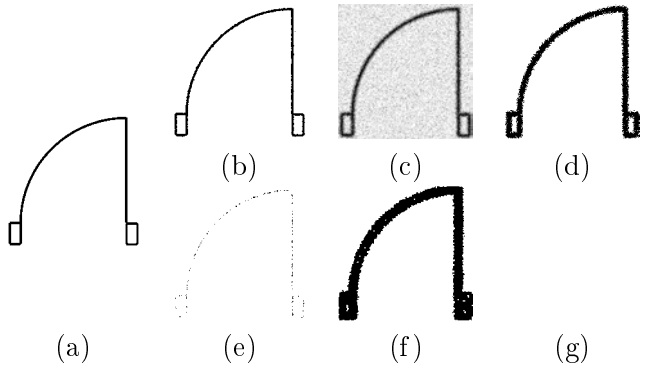


Figure 3.1: (a): Original binary symbol; from (b) to (g) examples of six levels of Kanungo noise of the GREC 2005 dataset.

Algorithm 4 Kanungo Noise

1: Use standard distance transform algorithms to calculate the distance d of each pixel from the nearest boundary pixel.
2: Each foreground pixel and background pixel is flipped with probability $p(0|1, d, \alpha_0, \alpha) = \alpha_0 e^{-\alpha d^2} + \eta$, and $p(1|0, d, \beta_0, \beta) = \beta_0 e^{-\beta d^2} + \eta$.
3: Use a disk structuring element of diameter k to perform a morphological closing operation.

3.4 Proposed Approach

3.4.1 Learning Dictionary for Documents Patches

The optimal solution of problem (P_1^ϵ) directly depends on the dictionary A used. In fact, using sparse representation with a specific choice of A has taken a lot of research attention in the past decade. Some translation-invariant redundant dictionaries, including the curvelets, contourlets, wedgelets, bandelets, and the steerable wavelets, have been used within the equation of the image denoising. These performances are good and they were considered as the best available image denoising methods in the early 2000's.

While these algorithms perform reasonably well, a more ambitious goal still continues. This ambitious goal leads to develop a dictionary adapted to the image by learning the dictionary from the noisy image itself. This may yield the dictionary A should be better adapted to document characteristics and document noise. Thus, we use one of the learning algorithms in Section (2.3) to construct the learned dictionary A.

In a learning dictionary algorithms, a training database $\{h_j\}_{j=1}^S$ is needed to construct the dictionary A that ensures all the $h_j, j = 1, ..., S$ has these optimal sparse approximations in A.

$$\min_{A,x_j} \sum_{j=1}^{S} \|x_j\|_1 \text{ subject to } \|h_j - Ax_j\|_2 \leq \epsilon, \text{ for all } j = 1, .., S$$

This database can include the whole noised images, however if the size of the image is large, for example 512×512, then the size of the dictionary is $L = 512^2, M = k \times L(k \geq 2)$, especially, in the real applications the size of an image could be 5515×6748, so the computing time to find all the sparse representations of images in training database in the *sparse coding stage* becomes prohibitive. In addition, if we use the soft thresholding algorithm to find the sparsity, then at

each iteration the $M \times M$ matrix $A^T A$ is calculated (see Equation (2.15)), and we can see that the larger of M, the more intensive the computing time. To overcome this issue, in our work, a small image patches of size $w \times w (w \ll M)$ are considered as a signals in training database, rather than on the whole image.

The dictionary A now is obtained by using learning dictionary algorithm with training database including patches extracted from noisy image. From the analysis about 4 well-known learning algorithms (Section (2.3)), named K-SVD, MOD, ODL and RLS−DLA, in both the performance and the computing time, we choose K-SVD algorithm in our thesis to construct the learned dictionary A with OMP algorithm used to find the sparse representations in the *sparse coding stage*.

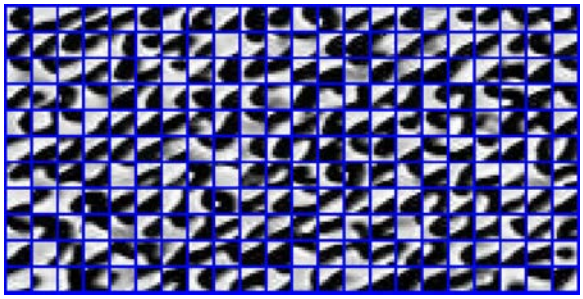


Figure 3.2: Learned dictionary obtained by using the K-SVD algorithm on patches of the size 8×8 extracted from DIBCO images after 50 iterations (used in the experiment result).

Figure (3.2) presents an example of a dictionary learned from examples extracted from noisy images with K-SVD algorithm.

3.4.2 Energy Noise Model

For denoising images using basis pursuit denoising, we need to decide which kind of dictionary A is used along with the best value ϵ in Equation (2.5). From Section (3.4.1) we know how to learn a dictionary A adapted to noisy data. In this Section, we explain how to choose the best value ϵ when we apply Equation (3.1) at the image patches of size $w \times w$.

However, before describing the proposed method, we will present some appropriate strategies that are developed to estimate ϵ for additive Gaussian noise, and Noise Spread.

Variable ϵ for Additive White Gaussian

Denoising in MRA methods assumes that images have been corrupted by an additive white noise of variance σ^2. For such noisy images, the energy of noise is proportional to both the noise variance and size image:

$$\epsilon = c \times L \times \sigma^2$$

with L is the size of the image and c is a constant. We will explain here after the theory analysis about how to calulate this value of ϵ: so far we assume that $A = \{a_i\}_{i \in \Gamma} \in \mathbb{R}^{L \times M}$ with $\Gamma = \{1, ..., M\}$ is a dictionary with M unit norm columns. Let $\{a_i\}_{i \in \Lambda}, \Lambda \subseteq \Gamma$ is a subset columns in A. The orthogonal projection of h on the space $\mathbb{V}_\Lambda$ generated by these columns is

$$h_\Lambda = \sum_{i \in \Gamma} a_i x_i \text{ with } a_i \neq 0 \text{ only for } i \in \Lambda$$

The orthogonal projection on the space $\mathbb{V}_\Lambda$ satisfies $h_\Lambda = h^0_\Lambda + e_\Lambda$, so

$$\|h^0 - h_\Lambda\|^2 = \|h^0 - (h^0_\Lambda + e_\Lambda)\|^2 = \|h^0 - h^0_\Lambda\|^2 + \|e_\Lambda\|^2 \tag{3.2}$$

We can see that the signal approximation error $\|h^0 - h_\Lambda\|^2$ decreases when $|\Lambda|$ increases and when $|\Lambda|$ increases, it means the noise energy $\|e_\Lambda\|^2$ also increases. Thus, to minimize Equation (3.2) we have to find a projection support Λ that balances these two terms $\|h^0 - h^0_\Lambda\|^2$ and $\|e_\Lambda\|^2$. However, there are 2^M possible subset Λ in Γ. Following [109], in these 2^M subsets, there exists some particular subsets Λ that $\|e_\Lambda\|^2$ may potentially take much larger values than $|\Lambda|\sigma^2$. Barron *et al* [9] approved that for any subset Λ of Γ then

$$\|e_\Lambda\|^2 \leq (\lambda^2 \sigma^2 \log_e M)|\Lambda|$$

It yields

$$\|h^0 - h_\Lambda\|^2 = \|h^0 - h^0_\Lambda\|^2 + \|e_\Lambda\|^2 \leq \|h^0 - h^0_\Lambda\|^2 + (\lambda^2 \sigma^2 \log_e M)|\Lambda|$$

and minimizing $\|h^0 - h_\Lambda\|^2$ is equivalent to find the Λ_λ that minimizes Equation (3.3)

$$\Lambda_\lambda = \arg\min_{\Lambda \subset \Gamma}(\|h^0 - h^0_\Lambda\|^2 + (\lambda^2 \sigma^2 \log_e M)|\Lambda|)) \tag{3.3}$$

However, minimizing Equation (3.3) depends on h^0, which is unknown. Thus, a crude estimator $\|h - h_\Lambda\|^2 = \|h\|^2 - \|h_\Lambda\|^2$ is used and it yields

$$\begin{aligned} \|h - h_\Lambda\|^2 - \|h^0 - h^0_\Lambda\|^2 &= (\|h\|^2 - \|h_\Lambda\|^2) - (\|h^0\|^2 - \|h^0_\Lambda\|^2) \\ &= (\|h\|^2 - \|h^0\|^2) - (\|h_\Lambda\|^2 - \|h^0_\Lambda\|^2) \end{aligned}$$

Because $h = h^0 + e$ and $h_\Lambda = h^0_\Lambda + e_\Lambda$, so

$$(\|h\|^2 - \|h^0\|^2) - (\|h_\Lambda\|^2 - \|h^0_\Lambda\|^2) = (\|h^0 + e\|^2 - \|h^0\|^2) - (\|h^0_\Lambda + e_\Lambda\|^2 - \|h^0_\Lambda\|^2)$$

In addition, e is a white noise of variance σ^2 so $E\{\|e_\Lambda\|^2\} = |\Lambda|\sigma^2$ and the expected error is $(L - |\Lambda|)\sigma^2$ which is of the order of $L \times \sigma^2$ if $|\Lambda| \ll L$. So, the threshold ϵ in the case of white noise is naturally chosen by $c \times L \times \sigma^2$ with c is a constant.

Variable ϵ for Noise Spread

For Noise Spread model [157], authors in [71] empirically found that the optimal energy of noise depends on the noise spread relation:

$$\epsilon = c \times NS$$

with NS derived from the degradation model (blur and threshold) equals:

$$NS = \frac{\sqrt{2\pi} \times \sigma \times \zeta}{LSP(ESP^{-1}(\Theta))} \tag{3.4}$$

In the description (3.4), Θ is the global thresholding; LSP is the line spread function or one dimensional point spread function (PSF) and ESP is the cumulative marginal of the functional form of PSF. The Point Spread Function (PSF) is the function describing the imaging system response to a point input, and is analogous to the impulse response. It is usually chosen to be circularly symmetric and describable by the width parameter ζ (see Figure (3.3)). In the case of blurring, an additive independent Gaussian noise, n, with a standard deviation σ is included, thus PSF used in the blurring is defined in Equation (3.5).

$$g(x,y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} h(x,y;\xi,\eta) f(\xi,\eta) d\xi d\eta + n(x,y) \tag{3.5}$$

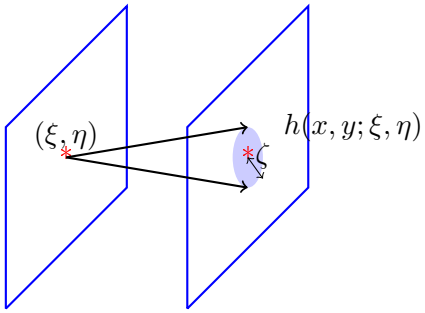


Figure 3.3: Illustration the Point Spread Function.

From the Equation (3.4), NS obviously depends not only on the standard deviation σ but also on the width ζ of the PSF. As the results, the value of $\epsilon = c \times NS$ is decided based on prior knowledge of parameters created in NS model.

Propose Method: Variable ϵ for other noise models

Neither of these two criteria can be applied to document images where noise follows a Kanungo model instead of white noise or noise spread. The reason is that in Kanungo noise model, pixels near the shape boundaries have higher probability to be affected by noise than pixels far from the shape boundaries. On the contrary, the white noise model assumes statistical independence between noise and image. In fact, the probability that a pixel is perturbed by noise depends only on the variance of model and does not depend on the position of pixel in the image. In addition, the parameters used in Noised Spread model are different from the Kanungo's ones, so we cannot use the noise spread relation proposed in [71] to decide the value for ϵ.

We have proposed an energy noise model inspired by [94]. Thus, we evaluate the noise level using the peak values of the normalized cross-correlation between noisy and cleaned documents. Let D being a training dataset including $2t$ documents $(D_i^c, D_i^n), i = 1, ..., t$ where D_i^c is a cleaned document and D_i^n is its noisy version. Define r_i as the peaks of normalized cross-correlation between D_i^n and D_i^c. Then, we define the tolerance error value by: $\epsilon = cw\bar{r}$, where c is a constant value set experimentally, w is the size of patches, $\bar{r}$ is the mean value of the peaks r_i:

$$\bar{r} = \frac{1}{t} \sum_{i=1}^{t} r_i$$

Therefore, we can summarize the procedure for document denoising using sparse representation of a learned dictionary A as follows:

1. *Create a training database* using a sliding window of size $w \times w$ to scan the corrupted image $y \in R^{M \times N}$ with scanning step set to 1 pixel in both directions. All $(M-w+1)(N-w+1)$ obtained patches $\{h_j\}_{j=1}^{l}$, $h_j \in R^{w \times w}$ are considered as the training database.

2. *Create a learned dictionary* using the K-SVD algorithm to create the learned dictionary A from $\{h_j\}_{j=1}^{l}$.

3. Combine the learned dictionary A with the sparse representation model in the purpose of denoising image, following:

   a. Find the solution of the optimization problem (3.1) for each patch h_j

   $$\hat{x_j} = \arg\min \|x_j\|_1 \text{ subject to } \|Ax_j - h_j\|_2 \leq \epsilon,$$

   b. Compute the denoised version of each patch h_j by $\hat{h_j} = A\hat{x_j}$,

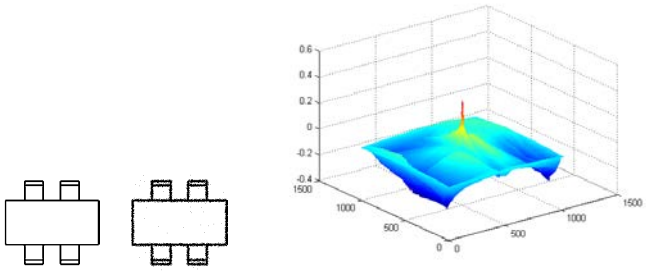
   c. Merge the denoised patches $\hat{h_j}$ to get the denoised image $\hat{y}$.

4. Binarise $\hat{y}$ to get the final result $\tilde{y}$.

3.5 Experimental Validation

We firstly evaluated our algorithm on the GREC 2005 dataset. This dataset has 150 different symbols which have been degraded using the Kanungo's method to simulate the noise introduced by the scanning process. Six sets of parameters are used to obtain six different noise levels as shown in Figure (3.1).

At each level of noise, a dataset contains 2×50 noisy and cleaned symbols is used to calculate the value of $\bar{r}$ and ϵ (Section (3.4.2)). We empirically found that the best results in denoising bi-level images are achieved when $\epsilon = cw\bar{r}$, with w is the size of corrupted patch, and c belongs to $[0.4, 1]$.



(a) (b) (c)

Figure 3.4: Normalized cross-correlation between two images.

Figure (3.4) shows one example about the normalized cross-correlation of two images (a) and (b) with its maximum value $r_i = 0.4106$; and Table (3.1) presents the best values of $\bar{r}$ corresponding to each level of degradation.

The learning dictionary was produced using K-SVD algorithm with 50 iterations, and the training dataset including all patches of the size 8×8 pixels ($w = 8$). Those patches were taken from a corrupted image h. The ratio of the dictionary is $1/4$ ($m = 4 \times n$). Here $n = 8 \times 8$ and therefore the size of dictionary is 64×256.

Level	1	2	3	4	5	6
$\bar{r}$	0.9133	0.4412	0.5629	0.3698	0.4413	0.2006

Table 3.1: The value of $\bar{r}$ at six level of noise.

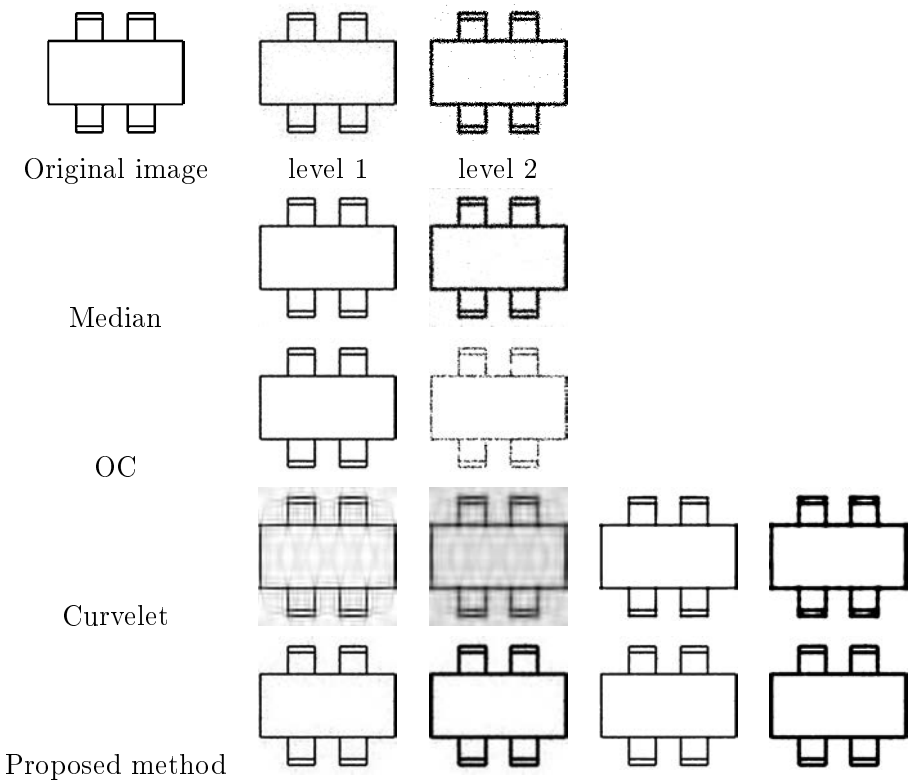


Figure 3.5: Results of denoising the noisy images with Kanungo model at levels 1 and 2 of degredation. Columns 2 and 3 are the denoised images following each method. Columns 4 and 5 are the binarized denoised images of columns 2 and 3, respectively. For the median and OC, images are already binarized in columns 2 and 3.

To examine the effectiveness of the proposed method, we compare it with three existing methods for denoising binary images: median filtering, morphological operators (opening and closing), and curvelet tranform. The median filtering is performed within a window size of 3×3. The morphological operators use a 3×3 structuring element. Curvelet transform has been verified upon the best value of $\eta = c \times \sqrt{MN} \times \sigma^2$, $\sigma \in [0.02, 0.1]$ for each noise level, where M, N is the size of the image. The criterion to choose these best values σ is the average MSE (*Mean Squared Error*).

Moreover, using the traditional MSE to estimate the quality change between the original document and the reconstructed one, all algorithms are evaluated by *Jaccard*'s similarity measure [22]. This measure is computed based on the three values a, b, c as below:

$$S = \frac{a}{a+b+c}$$

where

$$a = |\{(i,j)|y^0(i,j) = 1, \tilde{y}(i,j) = 1, 1 \leq i \leq M, 1 \leq j \leq N\}|$$

Symbols	MSE								Jaccard's measure							
	Median		OC		Curvelet		Proposed method		Median		OC		Curvelet		Proposed method	
	(a)	(b)	(a)	(b)	(a)	(b)	(a)	(b)	(a)	(b)	(a)	(b)	(a)	(b)	(a)	(b)
	2.8887	22.1582	2.1374	33.1523	0.5534	29.1777	1.0202	**20.6354**	0.9299	0.6358	0.9467	0.1726	0.9863	0.5785	0.9747	**0.6592**
	1.9473	15.5039	1.5107	22.2402	1.0967	22.7988	0.8789	**13.6172**	0.9295	0.6257	0.9441	0.1744	0.9607	0.5414	0.9676	**0.6630**
	3.9206	27.3984	3.6126	39.6582	2.9382	38.8574	2.7702	**25.7852**	0.9195	0.6242	0.9248	0.1643	0.9413	0.5496	0.9428	**0.6468**
	2.7702	15.6582	2.7552	24.1665	2.3132	21.9902	1.9297	**14.3384**	0.9200	0.6437	0.9305	0.1801	0.9437	0.5726	0.9517	**0.6716**
	2.0674	15.1836	1.7935	22.2949	1.2637	21.4375	1.2319	**13.3223**	0.9230	0.6216	0.9320	0.1465	0.9534	0.5492	0.9536	**0.6611**
	4.0371	29.5938	3.3633	44.4590	1.2529	38.6289	1.6123	**29.3594**	0.9284	0.6414	0.9390	0.1913	0.9775	0.5871	0.9709	**0.6512**
	3.6221	27.2002	2.8418	42.0830	2.6196	37.1270	1.4966	**26.0410**	0.9324	0.6500	0.9458	0.1949	0.9515	0.5845	0.9716	**0.6667**
	2.0710	16.3333	1.6361	23.8281	0.6393	23.3164	0.7285	**14.6536**	0.9300	0.6304	0.9434	0.1749	0.9782	0.5532	0.9749	**0.6626**
	2.4600	17.7080	2.3516	24.7490	2.2441	25.0537	1.7637	**16.4609**	0.9205	0.6199	0.9232	0.1797	0.9302	0.5462	0.9428	**0.6455**
	2.7253	19.4909	2.5020	28.2539	1.1725	27.1602	1.4297	**17.9434**	0.9217	0.6258	0.9272	0.1673	0.9658	0.5552	0.9584	**0.6528**
Average	2.8972	20.6229	2.45041	30.4885	1.6094	28.5548	**1.4862**	**19.2157**	0.9255	0.5677	0.9357	0.1555	0.9589	0.5030	**0.9609**	**0.5929**

Table 3.2: Summary of the denoising results in GREC 2005. (a), (b) are level 1, level 2 of noise, respectively.

$$b = |\{(i,j)|y^0(i,j) = 0, \tilde{y}(i,j) = 1, 1 \leq i \leq M, 1 \leq j \leq N\}|$$

$$c = |\{(i,j)|y^0(i,j) = 1, \tilde{y}(i,j) = 0, 1 \leq i \leq M, 1 \leq j \leq N\}|$$

a means '*right matches*', and b, c mean '*mismatches*'; $y^0, \tilde{y} \in R^{M \times N}$ are cleaned and denoised images, respectively. The maximal value of the *Jaccard* measure is one when two images are identical.

Table (3.2) shows results of denoising on ten classes of symbols with two level of noise (a) and (b) using different algorithms where (a) and (b) stand for the level 1 and level 2, respectively. The results are evaluated by MSE and Jaccard's measure. The best obtained results for (a) by MSE and Jaccard's measure are in red and green, respectively; and the best obtained results for (b) are showed in bold font.

Overall, Table (3.2) shows that our method achieves better results comparing with other methods at both level (a) and (b) of degradation. At level (b), the error values of our method are lower than others' for the MSE criterion. However, for level (a), the results of our method are quite similar with Curvelet method using MSE and Jaccard's measure, 1.482, 0.9609 in comparing with 1.6094 and 0.9589, respectively. Thus, a paired Wilcoxon signed test with a significance level of 5% is used also to check whether the difference between the results obtained by our method and the ones obtained by the other methods is significant. Table (3.3) and (3.4) show the average results obtained by the four methods on 6 levels of noise that are respectively evaluated by MSE and *Jaccard*'s measure. In these tables, an entry mark by $(-)$ indicates that the corresponding method performs worst than our method. Similarly, an entry marked by $(+)$ indicates that the corresponding method outperforms the proposed method, and an entry marked by $(=)$ indicates that results obtained by the two methods are not significantly different.

Tables (3.3) and (3.4) also show that at level 5 and level 6, none of the four methods is good enough but other methods are worse than ours. We further examine the set of noisy images at level 4 and we found that the set of noisy patches of the corrupted image cannot provide the good training data since most of patches are trivial (zeros value), making not enough discrimination between $A_{(0)}$ and $A_{(k)}$ (Algorithm (2) in Chapter (2)). This can explain why the

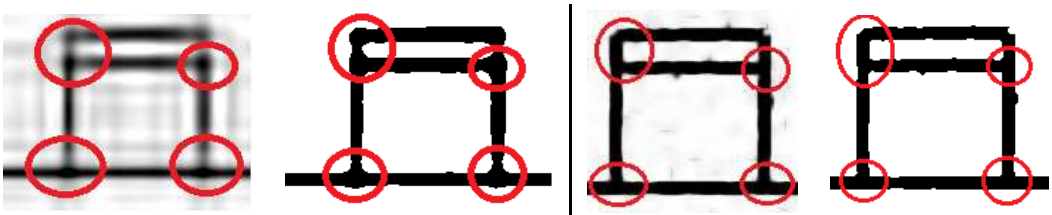
	Median	OC	Curvelet	Proposed method
Level 1	0.926 (-)	0.937 (-)	0.963 (=)	**0.963**
Level 2	0.630 (-)	0.177 (-)	0.565 (-)	**0.655**
Level 3	0.428 (-)	0.787 (-)	0.389 (-)	**0.826**
Level 4	0.088 (-)	0.001 (-)	**0.459** (+)	0.135
Level 5	0.261 (-)	0.268 (-)	0.263 (-)	**0.284**
Level 6	0.001 (-)	0.000 (-)	0.059 (=)	**0.060**

Table 3.3: Average value gained by Jaccard's similarity measure.

	Median	OC	Curvelet	Proposed method
Level 1	2.896 (-)	2.454 (-)	**1.403** (=)	1.423
Level 2	21.015(-)	30.865 (-)	28.421 (-)	**19.687**
Level 3	53.690 (-)	8.899 (-)	57.771 (-)	**8.008**
Level 4	33.599 (-)	36.819 (-)	**19.815** (+)	31.881
Level 5	111.632 (-)	107.358 (-)	110.286 (-)	**98.212**
Level 6	37.863 (-)	37.915 (-)	**35.632** (=)	35.772

Table 3.4: Average values gained by MSE.

curvelet transform method is better than the proposed method at this level of noise. Although at level 1 the Wilcoxon signed test indicates that results obtained by the curvelet and our method are not significantly different, when we zoom the denoised images we found that the intersection of edges are not well restored with the curvelet as shown in Figure (3.6). Since curvelets are smooth functions, they are not well adapted for singularity points. In addition, these measures do not take into account the structured information.



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

Figure 3.6: (a), (c): zoom of denoised images by curvelet and our method, respectively. (b), (d) denoised binary version respectively of (a) and (c).

We also verified our method on real scanned documents. In fact, we created the real datasets, used to compute the values of $\bar{r}$ or ϵ, by printing the original documents and then scanning printed documents at the different resolutions. With 15 original documents scanned at 12 different resolutions, we got 12 scanned datasets respectively in which each dataset contains 15 scanned documents. Figure (3.7) presents a zoom of one of the scanned documents and Figure (3.8) presents an example of a dictionary learned from patches extracted from real scanned documents with K-SVD algorithm.

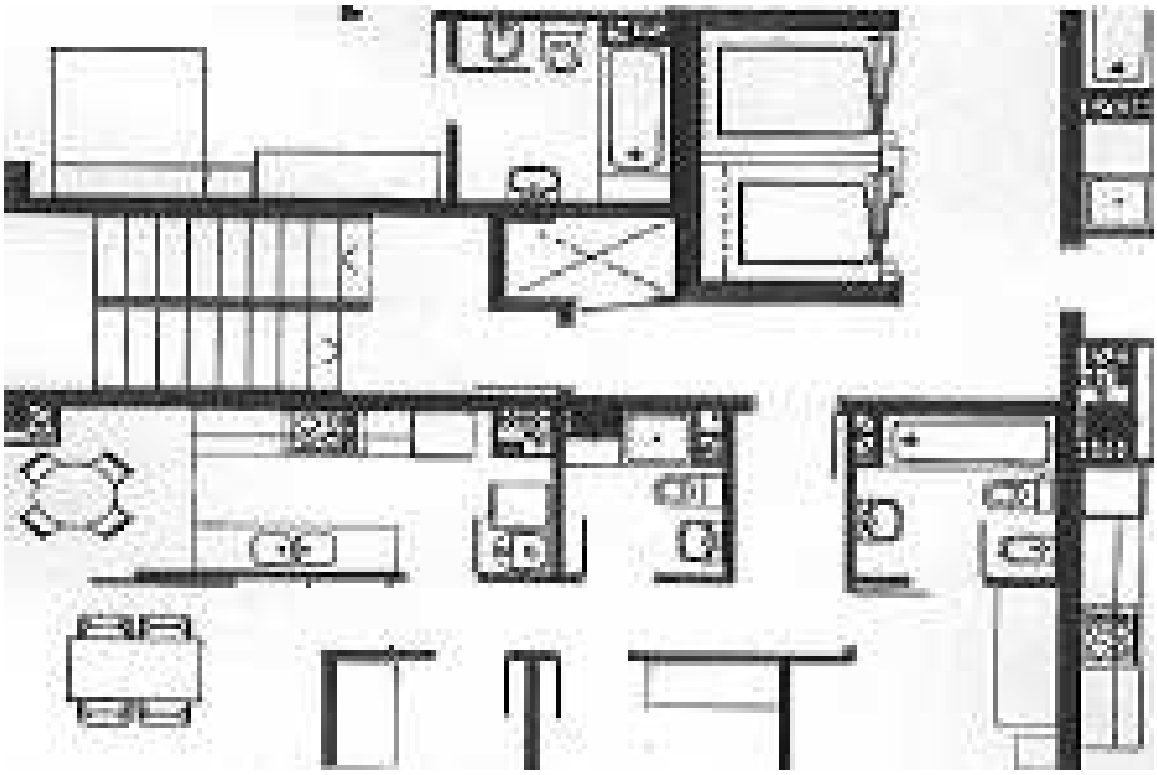


Figure 3.7: One of the scanned documents.

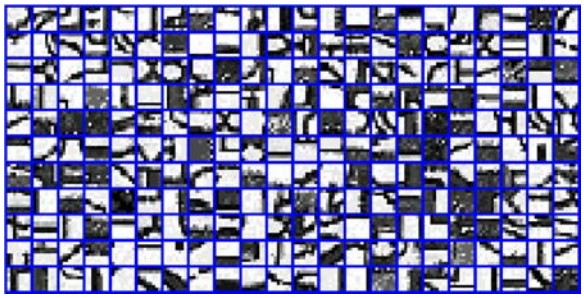


Figure 3.8: Learned dictionary obtained by using the K-SVD algorithm on patches of the size 8×8 pixels extracted from real scanned documents after 50 iterations.

The aim is to test our method on real documents with an unknown model of noise, 12 scanned images at the different resolutions are tested and evaluated using the Structural similarity (SSIM) index. SSIM [177] is also a method for measuring the similarity that is designed to improve MSE, which have proved to be inconsistent with human eye perception. SSIM metric is computed on various windows of images. The distance between two windows s_i and q_i is calculated following the Equation (3.6) where s_i, q_i are taken from the same location of noisy image and reconstructed image.

$$\mathrm{S}(s_i, q_i) = \left(\frac{2\mu_{s_i}\mu_{q_i} + c_1}{\mu_{s_i}^2 + \mu_{q_i}^2 + c_1}\right)\left(\frac{2\sigma_{s_i}\sigma_{q_i} + c_2}{\sigma_{s_i}^2 + \sigma_{q_i}^2 + c_2}\right) \tag{3.6}$$

In (3.6), $\mu_{s_i}, \mu_{q_i}, \sigma_{s_i}, \sigma_{q_i}$ are the local means and the sample standard deviations of s_i and q_i, respectively; $c_1 = (k_1 L)^2, c_2 = (k_2 L)^2$ are two variables to stabilize the division with weak denominator; L is the dynamic range of the pixel-values; and k_1, k_2 are two constants. The resultant SSIM index is a decimal with a value 1 in the case of two identical sets of data. In this paper, SSIM index is calculated on the window size 8×8, and the values of L, k_1, k_2 are respectively $100, 0.01$ and 0.03.

Images	Median	OC	Curvelet	Proposed method
1	0.5374	0.4640	0.4693	**0.6834**
2	0.6906	0.5165	0.6184	**0.7534**
3	0.5749	0.5158	0.5057	**0.7421**
4	0.6482	0.4879	0.5666	**0.7454**
5	0.6008	0.5056	0.5186	**0.7458**
6	0.6310	0.4409	0.5320	**0.7145**
7	0.6018	0.5030	0.5244	**0.7580**
8	0.6066	0.4814	0.5199	**0.7381**
9	0.6487	0.4662	0.5638	**0.7456**
10	0.6203	0.4603	0.5377	**0.7317**
11	0.6946	0.4832	0.5987	**0.7685**
12	0.7264	0.4494	0.6274	**0.7733**
Average	0.6320 (-)	0.4812 (-)	0.5485 (-)	**0.7416**

Table 3.5: The obtained results with scanned documents using SSIM measure

Table (3.5) presents the results in comparison with other methods. We can see that in each case the performance of our approach is good compared to the others. Figure (3.9) shows the denoised document of document in Figure (3.7) obtained by our approach.

The last experiment is done on the DIBCO 2009 dataset. This dataset contains images that range from grayscale to color, from printed handwritten, and from real to synthetic. The value of $\bar{r}$ for this experiment equals 0.7321 and is calculated as the same way as described above. Figure (3.2) presents one learning dictionary build by corrupted patches with size 8×8 pixels of DIBCO images. Figure (3.10)(a) gives the documents in DIBCO dataset and its versions of denoising got by our approach (Figure (3.10)(b), (3.10)(c)).

As the experiment before, the SSIM measure is used with the purpose of comparison. Table (3.6) presents the results on 5 handwritten images in DIBCO in Figure (3.10). Tables (3.5) and (3.6) present the difference results using also a paired Wilcoxon signed test with a significance level of 5%. We can observe that the difference between our approach performance compared to the others methods performance is statistically significant even in the case where the document noise is unknown.

3.6 Conclusion

A novel algorithm for denoising document images by using learning dictionary based on sparse representation has been presented in this thesis. Learning method starts by building a training

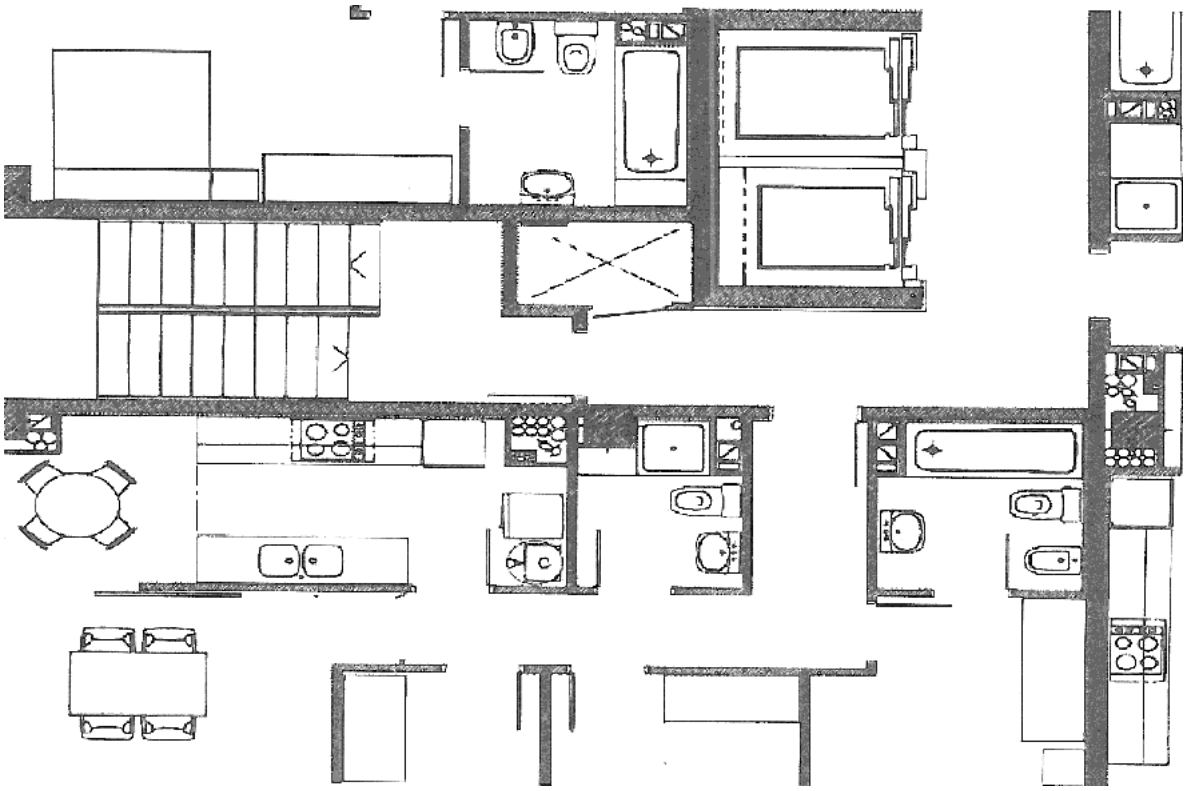
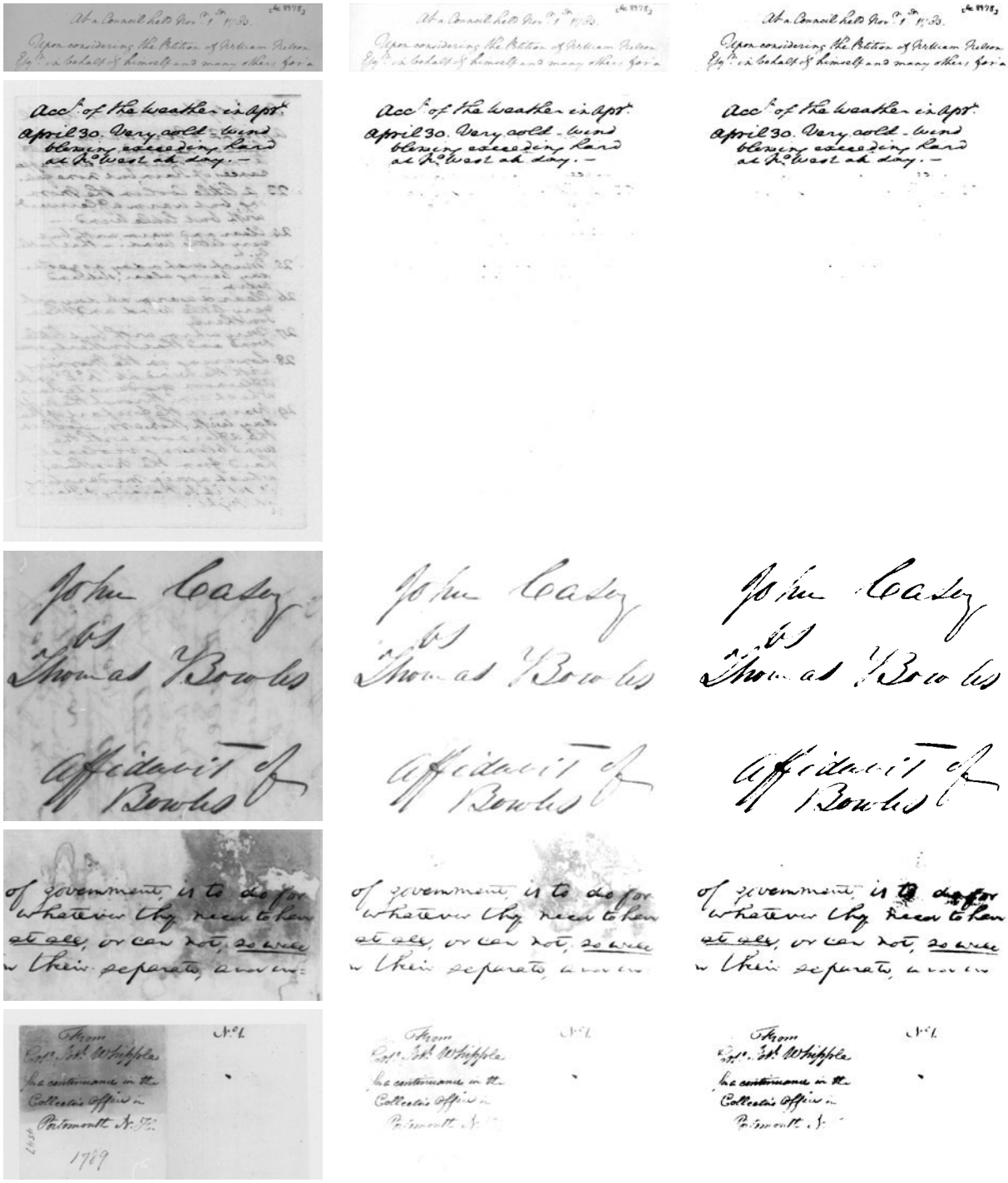


Figure 3.9: The denoised document of document in Figure 3.7 got by our approach.

database from corrupted images, and constructing an empirically learned dictionary by using sparse representation. This dictionary can be used as a fixed dictionary to find the solution of the basis pursuit denoising problem. In addition, we provide a way to define the best value of the tolerance error (ϵ) based on a measure of fidelity (cross-correlation) between two images. The efficiency of ϵ has been also approved experimentally on different datasets for different resolutions and different kinds of noise. All experimental results show that our method outperforms existing ones in most of the cases.

Images	Median	OC	Curvelet	Ours approach
1	0.6420	0.6926	0.7054	**0.9528**
2	0.4628	0.5155	0.5156	**0.9784**
3	0.5115	0.5315	0.5064	**0.8648**
4	0.4595	0.4946	0.4692	**0.8933**
5	0.6953	0.7283	0.7441	**0.9416**
Average	0.5542 (-)	0.5925 (-)	0.5881 (-)	**0.9261**

Table 3.6: The obtained results with DIBCO 2009 dataset using SSIM measure.



(a) (b) (c)

Figure 3.10: (a) noisy documents in DIBCO dataset used in Table 3.6 , (b) the denoised documents got by our approach before binarization and (c) after binarization using Otsu's method

Chapter 4

Text/Graphic Separation

Contents

The combination between the MCA method and learned dictionaries has obtained a good performance in text region segmentation. However, since the learned dictionaries have constructed from the training database being the patches of images, so the size of the patch has an important impact on learning dictionaries. In addition, when working with graphic images, text characters are in different sizes included in the same document, thus deciding a suitable size of the patch becomes impractical. Therefore, in this chapter, we have developed the MCA approach by proposing the strategy using multi-resolution learned dictionary for separating the text regions from graphical part instead of two dictionaries. In our method, we first empirically construct two sequences of learned dictionaries, one for the text and the other for the graphics. Then, we compute the sparse representations of all different sizes and non-overlapped document patches in these learned dictionaries. Based on these representations, each patch can be classified into the text or graphic category by comparing its reconstruction errors. Same-sized patches in one category are then merged together to define the corresponding text or graphic layers, which are combined to create a final text/graphic layer. Finally, in a post-processing step, text regions are further filtered out using some learned thresholds.

4.1 Introduction

Extracting text regions and other graphical objects is an important step of any automated document analysis system. The information about the type of extracting data can help to improve the accuracy rate as well as to speed up the recognition process. In addition, if the text is well segmented an OCR could be applied to define the semantic meaning of the extracted text regions. In this purpose, a great number of studies have been proposed to tackle the problem of text regions segmentation.

The existing methods of text detection can be roughly categorized into following groups:

edge based, connected component (CC)-based, stroke based, texture based, and the others. In general, edge-based methods [185, 98, 77, 51, 104] are usually efficient in extraction text regions in scene images, particularly on scene images exhibiting strong edges. However, for the same reason, a main drawback of these methods is that good edge profiles are hard to obtain under the influence of shadow or highlight. In addition, the weakness of edge based methods to other kind of image like technical documents can be found in [104] since this method is not robust for characters with different sizes included into the same document.

Connected component-based methods use a bottom-up approach by grouping small components into successively larger components until all regions are identified in the image [99, 79, 60, 171, 143]. CC-based methods directly segment candidate text components by edge detection or color clustering. The non-text components are pruned with heuristic rules or classifiers. Because of the small number of segmented candidate components, CC-based methods have lower computation cost and the located text components can be directly used for recognition. However CC-based methods cannot segment text components accurately without prior knowledge of text position and scale. Moreover, the application of methods [60, 171, 143] to cluttered graphical documents is difficult especially when texts touch the graphics.

In stroke based methods like [49, 136, 187], text is modeled as a combination of stroke components with a variety of orientations, and features of text are extracted from combinations and distributions of the stroke components. The feature, that is used to separate text from other existing elements in images, is based on a stroke feature, for example the stroke width. This feature is then utilized to recover candidate regions for text. In stroke based methods, text stroke candidates are extracted by segmentation, verified by feature extraction and classification, and grouped together by clustering. In general, these methods are easy to implement on specific applications because of their intuition and simplicity. However, complex backgrounds often make text strokes hard to segment and verify [191].

Texture-based methods [194, 67, 4] use the observation that texts in images have distinct textural properties that distinguish them from the background. Thus, texture analysis approaches are usually used. Some texture analysis approaches include Gaussian filtering, wavelet decomposition, Fourier transform, discrete cosine transform (DCT). Typically, in these approaches, features are extracted over a certain region and then identifying the existence or not of text on this region is done by applying a classifier (using machine learning techniques or heuristics). Because text regions have distinct textural properties from non-text ones, these methods can detect and localize texts accurately even when images are noisy. However, these methods are sensitive to text alignment orientation.

Because of many possible variations in text, the above approaches often do not work well under certain conditions. To deal with such variations, some researchers have developed new approaches including [17, 162, 11, 186, 122, 62, 156]. Details about some of these approaches can be found in the work of Zhang *et al* [191]. Recently, authors in [72] propose to segment technical documents into two morphological components using the Morphological Component Analysis (MCA) framework with two pre-constructed dictionaries. However, since the results are dependent on the choice of two pre-constructed dictionaries, this method is not adapted to various kind of documents.

In summary, the existing methods are not efficient when dealing with documents containing dense information. Therefore, in this chapter, we propose an alternative approach that overcomes the limitations of existing methods by using multi-learned dictionaries combined with a sparse representation. In fact, learned dictionaries were used successfully in local/global MCA with the purpose of separating textures and cartoons [41], as well as in the text detection from scenic images [135]. However, the performance of these methods depends strongly on the size of the

patch. In fact, if the size of the patch is too large, its sparse representation vector is large. It means the computing cost time will increase. If this size is too small, it could not contain enough information for discrimination. In addition, the text characters having different sizes usually include in the document, therefore, choosing a suitable size of the patch for all documents seems more difficult. To overcome this shortcoming, we propose a new approach using multi-resolution learned dictionaries for separating text parts from graphical ones. To the best of our knowledge, it is the first times multi-resolution learned dictionaries have been used for separating the text regions from graphical part.

The main idea of the proposed method is based on the assumption that the representation of text candidate patches in the learned dictionaries for texts are sparse but not sparse in the learned dictionaries for graphics. To make use of this assumption, we first empirically construct two sequences of dictionaries corresponding to two types of data (graphic or text) using the K-SVD method [41]. Then, we use these learned dictionaries combined with Orthogonal Matching Pursuit (OMP) algorithm [41] to find the sparse representations of all different sized non-overlapped patches. Next, each patch can be classified into text or graphic categories by comparing its reconstruction errors. All same-sized patches in one category are merged to make a corresponding text or graphic layers. Finally, these text/graphic layers are combined using logical operators. In a post-processing step, text regions are further filtered out using some learned thresholds.

The rest of this chapter is organized as follows. Section 4.2 states the problem in the framework of sparse representation. A discussion about how to use learning algorithm to create multi-learned dictionaries for text extraction is given in Section 4.3.1. Details of the proposed text detection method are presented in Section 4.3.2. The experimental results show that our method provides good results and outperforms other methods (Section 4.4). Finally, we give our conclusions in the Section 4.5.

4.2 Problem Statement

In text extraction, one document input containing both text and graphics needs to be processed to produce two output images, one containing text and the other containing graphics. The importance of text extraction is due to the possible existence of text's semantic meaning, which could be obtained from the extracted text by using an optical character recognition (OCR) engine and a linguistic tool and, more importantly, could facilitate the interpretation of scanned graphical documents.

Basically, graphics components contained in document images are of various types according to each specific application domain but generally they are lines, curves, polygons and circles. Meanwhile, text components consist of characters and digits that form words and sentences used to annotate the graphics. Text/graphic separation is a challenging problem because of following reasons. The first one is the graphical components as lines can be of any length, thickness, and orientation; the circles, polygons can be filled or unfilled; and the text components can vary in font styles and sizes. The second one is the existence of touching text components and crossing between text and graphics components. The third one is text strings are usually intermingled with graphics and can have any orientation.

As we have recalled in Section (4.1), several methods have been proposed to tackle this problem and they can be roughly classified into main families including morphological analysis, connected component analysis, multi-resolution analysis. However, these approaches cannot work with the difficult case of touching between text and graphics components. For example, with a graphic-rich and complex engineering document image as showed in Figure (4.1), none of the

mentioned methods in Section (4.1) provide reliable results, giving rise to a demand for a new proposal.

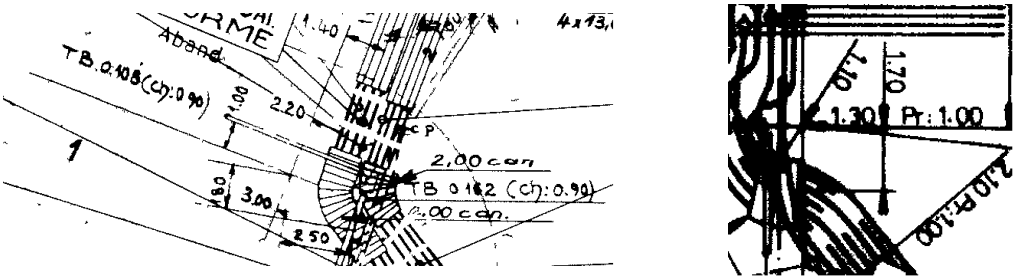


Figure 4.1: Examples of graphic-rich documents.

The proposed method in this chapter extracts text components in a totally different way from existing ones as a document image h containing text and graphics components is considered as a 2D signal, and it is the mixture of two separate 2D signals of the same size:

$$h = h_t + h_g$$

where h_t and h_g contain text and graphics components respectively. The problem of text extraction is now seen as the inverse problem of recovering h_t and h_g from h, which is similar to the blind source separation problem [82]. We assume that there are two over-complete dictionaries $A_t, A_g \in R^{L \times M}$ that satisfy the signal h_t is sparse in A_t but not sparse enough-or at least not as sparse-in A_g, and similarly, h_g is sparse in A_g but not sparse enough in A_t. In this manner, the dictionaries A_t, A_g play the role of a discriminant between h_t and h_g. This is a key observation for the success of the separation algorithm. Once such dictionaries are identified, the use of a pursuit algorithm (see Section (2.2)) searching for the sparsest representation leads to the desired separation. However, to do this, we should answer the question: What should the proper dictionaries be for these two kinds of contents?

In fact, the recent work in [72] used undecimated wavelets and curvelets as the two over-complete dictionaries A_t and A_g. However, this method did not totally separate text and graphics components because:

- There exists an overlap between the two chosen dictionaries. Particularly both dictionaries can represent the low-frequency contents efficiently and hence both consider these contents as theirs.

- Some graphics, such as arrowheads, short curve segments, have morphological characteristics being similar to those of text components. Thus, they are more likely to be represented by the dictionary defined from undecimated wavelets and may appear in the text image.

(a) (b) (c)

Figure 4.2: Examples when using MCA with undecimated wavelets and curvelets as two over-complete dictionaries A_t, A_g to extract the Text/Graphic parts: (a) original document, (b) the text component, and (c) graphic component (extracted from [72]).

Although, applying MCA to graphical documents have not separated the text and graphic parts totally as expected (see Figure (4.2)), its first success consolidates our motivation in finding better solution. As the results, we have developed the assumption of MCA by proposing a strategy using multi-learned dictionaries for separating text regions from graphical part instead of two pre-defined dictionaries. We believe that when dictionaries have learnt from the text/graphic database separately, they will better adapt to the characteristics of signals; therefore, reducing the overlap between them or increasing the performance of text extraction. In addition, using multi-resolution learned dictionaries can overcome the drawback of previous methods [41, 135] in choosing a suitable size of the patch.

4.3 Proposed Approach

4.3.1 Learned Dictionaries for Text and Graphic Parts

The effect of separation depends directly on dictionaries. If they are pre-constructed or pre-defined dictionaries, then they typically lead to fast transforms. However, they are typically limited in their ability to sparsity the images they are designed to handle. Furthermore, most of these dictionaries are restricted to images of a certain type, and cannot be used for a new and arbitrary family of images of interest, as the graphical images as shown in Section (4.2). This leads us to another approach for obtaining dictionaries that overcomes these limitations by adopting learning point-of-view.

The learning option starts by building a training database of signal instances and constructing an empirically learned dictionary, in which the generating atoms come from the underlying empirical data, rather than from some theoretical model. Such a dictionary can then be used in the application as a fixed and redundant dictionary. The details of how create the learned dictionary is shown in Section (2.3). We recall that we need to solve the following Equation (4.1)

$$\min_{A,x_j} \sum_{j=1}^{S} \|h_j - Ax_j\|_2^2 \text{ subject to } \|x_j\|_0 \leq k_0, \text{ for all } j = 1,..,S \tag{4.1}$$

where $\{h_j\}_{j=1}^{S}$ is the training database. This equation is similar to the problem (P_1^ϵ) but now, also the dictionary $A \in \mathbb{R}^{L\times M}$ is unknown and has to be found during the optimization process. This dictionary can be computed by any learning algorithm as mentioned in Section (2.3). In our work, we use K-SVD algorithm. In general, K-SVD builds the dictionary by an iterative procedure with two main steps: the sparse representation step, and the update dictionary step.

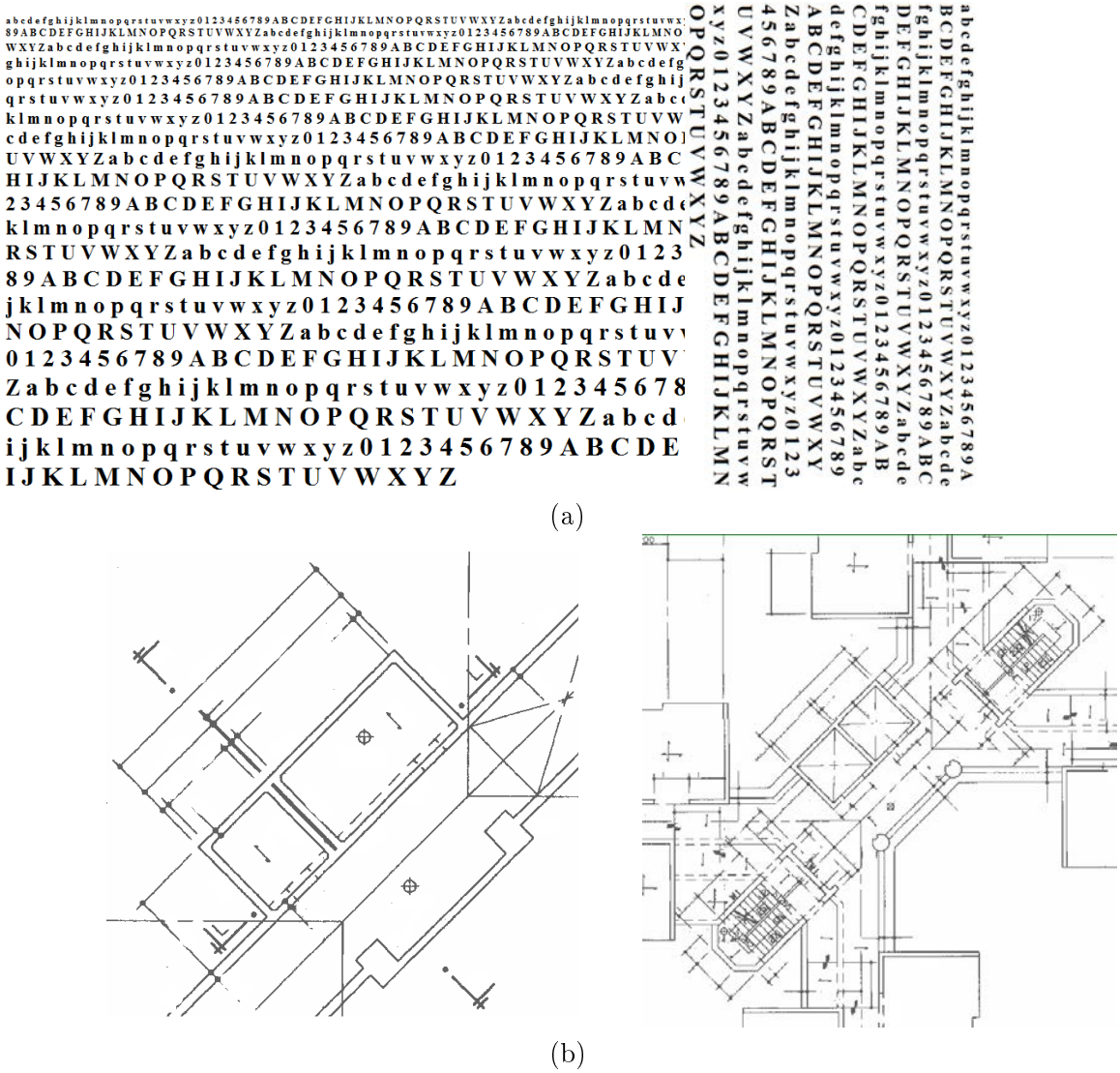


Figure 4.3: Discriminative dictionary training samples: (a) text training samples; (b) graphic training samples

In the sparse representation step, all sparse representations of training signals are computed while keeping the dictionary fixed. In the update dictionary step, columns-by-columns of the dictionary are updated such as minimizing the residually error based on calculating the *Singular Value Decomposition* (SVD) over the relevant examples.

In fact, if the size of signals is too large, their sparse representation vectors will be high dimensional, increasing the time processing. If it is too small, it could not contain enough information to be discriminant. Moreover, the text characters including in the document usually have different size, thus deciding a suitable size of the patch seem to be not practical. This are the reasons, in this chapter, instead of two over-complete dictionaries, two sequences of over-complete dictionaries are used for text and graphic separation. Each sequence has K different dictionaries. The first sequence of dictionaries provides a sparse representation for the text, while the second one provides a sparse representation for the graphic. To create the sequences of the text dictionaries $\{A_k^t\}_{k=1}^K$, we create the set of text images being composed by 26 lowercase and

$\sqrt{s_k}$	8	10	12	14	16	18	20
$\lvert Y_k \rvert$	183276	116838	80962	59260	45194	35604	28822
$\lvert Z_k \rvert$	48805	30873	21480	15600	12004	9472	7612
$L \times M$	64×256	100×400	144×576	196×784	256×1024	324×1296	400×1600

Table 4.1: The sizes of the training databases.

uppercase English letters and 10 Arabic numerals of various fonts, sizes and types and 90 degrees clockwise rotation. Then the different sliding windows w_k of the size $\sqrt{s_k} \times \sqrt{s_k}, k = 1, ..., K$ used to scan text images into non-overlapped patches $\{Y_k\}_{k=1}^K$. For the K graphic learned dictionaries $\{A_k^g\}_{k=1}^K$, some graphic images are used to construct the training database $\{Z_k\}_{k=1}^K$ using the same way we do with text images. The graphics training examples are collected from the graphic images including only the graphic component and their different resolutions. Figure (4.3) presents examples of text/graphic images used to learn dictionaries. Table (4.1) shows the sizes of the training database used to learn dictionaries.

After having two sequences of training databases $\{Y_k\}_{k=1}^K$ and $\{Z_k\}_{k=1}^K$, we use again the discriminative dictionary training algorithm K-SVD (see Section (2.3)) to construct two sequences of learned dictionaries $\{A_k^t\}_{k=1}^K$, $\{A_k^g\}_{k=1}^K$.

In the experiment, each dictionary A_k^t (or A_k^g) is a matrix with s_k rows and $4 \times s_k$ columns, the last row of Table (4.1) presents in details the size of learned dictionaries. Moreover, 10 iterations of the algorithm are run. At each iteration, we prune the two training database by keeping the best 90 % classified patches. It is hoped that the overlap between the text and background sets can then be minimized. The resultant text and background dictionaries are shown in Figure (4.4)

4.3.2 Text Regions Extraction by Learned Dictionaries

Given a graphical image $y \in \mathbb{R}^{n \times m}$, we first decompose it into K sets of non-overlapped patches by using K sliding windows $\{w_k\}_{k=1}^K$, in which w_k has the size $\sqrt{s_k} \times \sqrt{s_k}$ (see Figure (4.5)). Afterwards, for each set of patches $\{p_i^k\}_i$, we use two learned dictionaries A_k^t, A_k^g combined with OMP [41] to find all sparse representations of all patches in this set, named $\{\bar{q}_{a,i}^k\}_i$, here a stands for A_k^t and A_k^g:

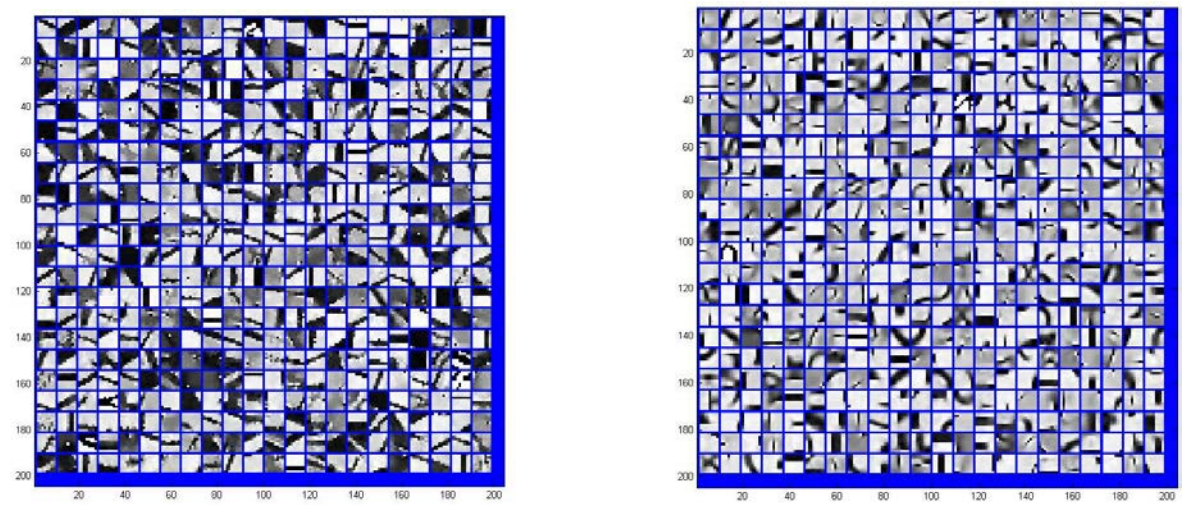
$$\bar{q}_{a,i}^k = \arg\min_{q_{a,i}^k} \|p_i^k - a q_{a,i}^k\|_2 \text{ s.t } \|q_{a,i}^k\|_0 \leq T_k$$

Then, each patch p_i^k can be classified into text or graphic by comparing its reconstruction errors in A_k^t, A_k^g using Equation (4.2).

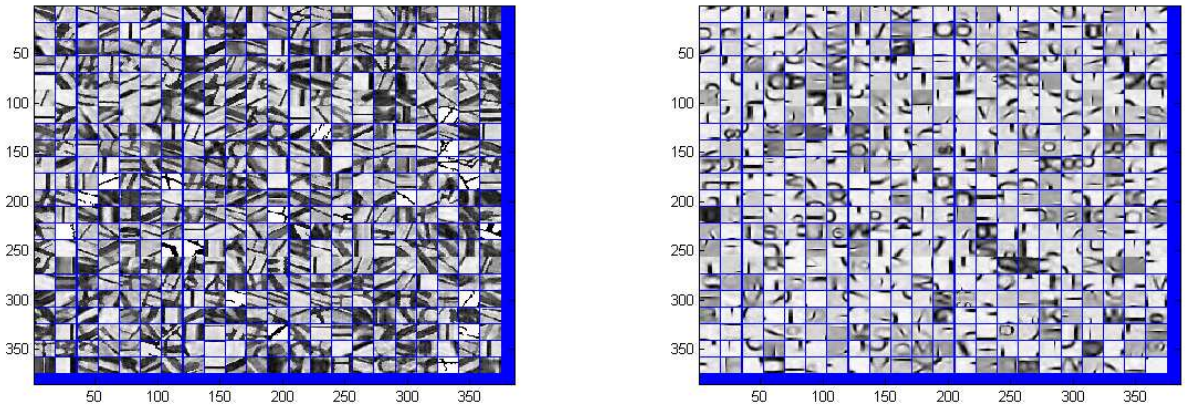
$$\epsilon_{a,i}^k = \|p_i^k - a \bar{q}_{a,i}^k\|_2 \tag{4.2}$$

If the reconstruction error of p_i^k in text learned dictionary A_k^t is smaller than its reconstruction error in the graphic learned one, A_k^g, it means that the representative of p_i^k in A_k^t is sparse and not sparse (or at least not enough sparse) in A_k^g and, p_i^k is considered as a text patch. Otherwise, it is classified as a graphic patch.

Next, all text/graphics patches are added to the text/graphics layer y_T^k, y_G^k, respectively. Binarizing y_T^k, $y_G^k, k = 1, ..., K$, and then the K text and graphic layers (K is set experimentally



(a)



(b)

Figure 4.4: A zoom of the trained dictionaries with (a) $\sqrt{s_k} = 8$, (b) $\sqrt{s_k} = 16$; Graphic (left) and Text (right)

.

to 2 for experimental results) are combined into the final text/graphic layer by using the logical operations $y_T = \wedge y_T^k$ and $y_G = \vee y_G^k$.

The post-processing phase is necessary to further filter out some text candidates. To delete some remaining graphic components, we learn thresholds defined on the behavior of the sparsity of noise components related to real true texts components. More details are given in the next section.

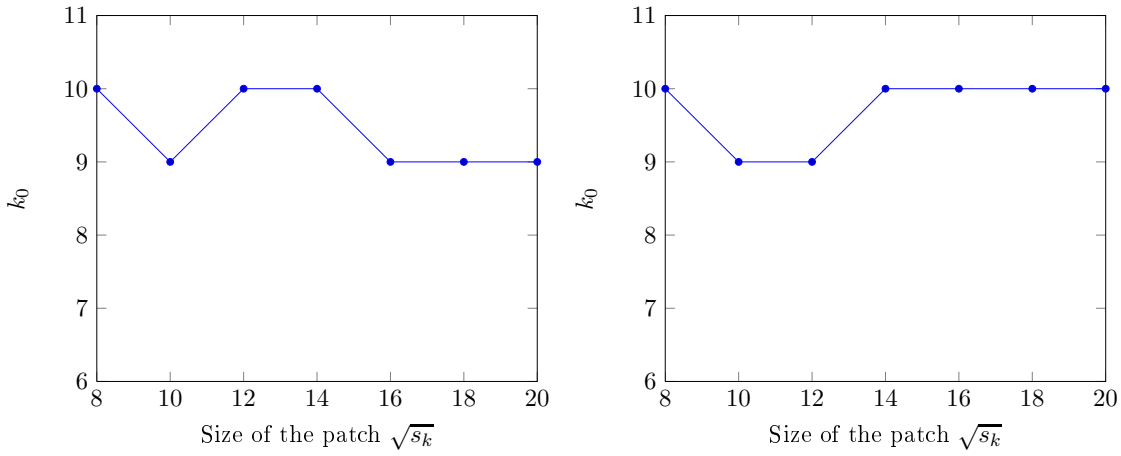


Figure 4.6: The optimal value of k_0 following the size of the patch for the graphic dictionaries (left) and text dictionaries (right) in term of average representation error

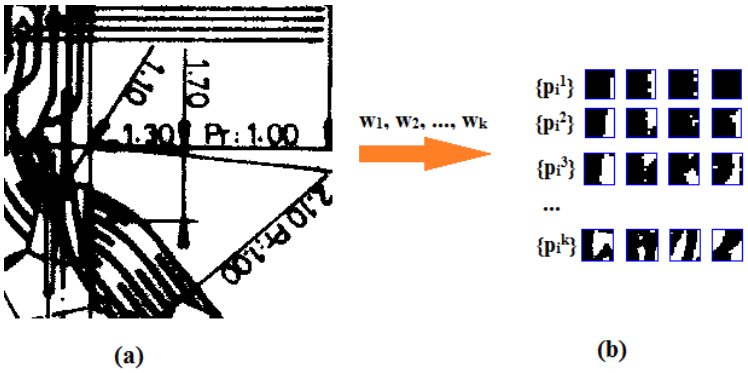


Figure 4.5: Example of decomposing input image into K sets of non-overlapped patches by using K sliding windows: (a) input image, (b) K sets of non-overlapped patches for the first four patches in each set.

4.4 Experimental Validation

In the K-SVD algorithm used to solve Equation (4.1), we have to take care on the value k_0 and the size of the patch which have an impact on the learned dictionaries. We have mentioned that if the size of the patch is too small, the information in the graphic and text patch is not so much different, so, text candidate patches can be considered as graphic patches. If the size is too large, each patch can contain the text and graphic components together, and there is not enough sparsity, either in text dictionary or in graphic dictionaries. Moreover, if we analyze the behavior of k_0 we can remark that the optimal value of this value is different following the size of the chosen patch (see Figure (4.6)).

In this perspective, we propose to use multi-learned dictionaries as describe in Section (4.3.1). We generate a set of dictionaries for $\sqrt{s_k}$ from 8 to 20 and we experimentally find that the best trade-off is the combination of two dictionaries with $\sqrt{s_k} = 8$ and 16. Figure (4.7) presents an

Algorithm 5 Text/Graphics Separation

INPUT: $y \in \mathbb{R}^{n \times m}$;
1. Create the learned dictionaries
- Collecting sets of text/graphic images,
- Using k sliding windows w_k of the size $\sqrt{s_k} \times \sqrt{s_k}$ to scan text/graphic images into non-overlapped patches $\{Y_k\}_{k=1}^{K}$, $\{Z_k\}_{k=1}^{K}$,
- Building two sequences of learned dictionaries:
for $k = 1$ to K **do**
  - $A_k^t \longleftarrow$ K-SVD(Y_k)
  - $A_k^g \longleftarrow$ K-SVD(Z_k)
end for
2. Text/Graphics separation
- Decompose y into K sets of non-overlapped patches $\{p_i^k\}_i$ using k sliding windows w_k,
for Each patch $p_i^k \in \{p_i^k\}_i$ **do**
  - finding its sparse representation $\bar{q}_{a,i}^k$ over text/graphic dictionaries a, with a stands for A_k^t and A_k^g:

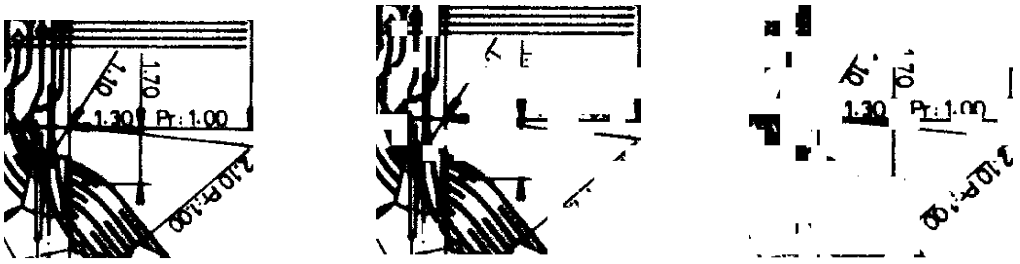
$$\bar{q}_{a,i}^k = \arg\min_{q_{a,i}^k} \|p_i^k - a q_{a,i}^k\|_2 \text{ s.t } \|q_{a,i}^k\|_0 \leq T_k$$

  - Computing the reconstruction errors of $\{\bar{q}_{a,i}^k\}_i$ in a:

$$\epsilon_{a,i}^k = \|p_i^k - a\bar{q}_{a,i}^k\|_2$$

  if $\epsilon_{A_k^t,i}^k < \epsilon_{A_k^g,i}^k$ **then**
    p_i^k is the text patch
  else
    p_i^k is the graphic patch
  end if
end for
- Combining same size text/graphic patches into text/graphic layers,
- Combining K text/graphic layers into final text/graphic layer,
- Filtering out some text candidates in text layer using learn thresholds.
OUTPUT: Text/graphic layer.

example of extracting text with the learned dictionaries.



(a) (b) (c)

Figure 4.7: Examples using two sequences of over-complete learned dictionaries to separate the Text/Graphic parts: (a) original document, (b) final graphic layer, and (c) final text layer.

In the final text layer, there are still some graphical components that are considered as noise components so far. This kind of noise will be deleted by verifying the behavior of its sparsity and compared to the one of the true texts (characters) in the text/graphical dictionaries. Figure (4.8) shows the average sparsity of the non-overlapped patches of the noise components and the text components in the text and graphic dictionaries. The figure clearly shows that, in the text dictionaries, the sparse representation of the noise component has less non-zero elements than the text. Moreover, the sparse representation of the noise component in the text dictionaries is sparser than in the graphic dictionaries. This explains why some noise components are misclassified as text candidates. In this perspective, we consider that a patch is a text if its sparsity is above a threshold *Th*. This threshold is larger than the average sparsity of the remained noise components, see Figure (4.8) (left). Figure (4.9) illustrates one example of how the chosen threshold is applied to further filter out noise components.

We compare our method with the one proposed by Thai *et al* [72] and Tombre *et al* [171], using the same quantitative measures where Nb.ch is the number of characters in each image counted by the same protocol as in [171]. From Table (4.2) we can remark that our method provides the best results for each document and is better than each dictionary used separately. Figure (4.10) presents 5 documents used in the evaluation of Table (4.2).

Img	Nb. ch.	[72]	[171]	Our method	Dic. 8×8	Dic. 16×16
Doc 1	63	53	58	61	24	50
Doc 2	92	70	71	85	38	78
Doc 3	93	77	81	86	32	83
Doc 4	121	104	104	114	53	111
Doc 5	31	22	7	23	6	19

Table 4.2: Performance evaluation: (see Figure 4.10) with T_0 set in Figure 4.6 and $T_k = 16; 32$ for $\sqrt{s_k} = 8; 16$.

4.5 Conclusions

In this chapter, we present an alternative approach for separation text/graphics from technical documents based on sparse representation. In our method, we combine multi-learned dictionaries

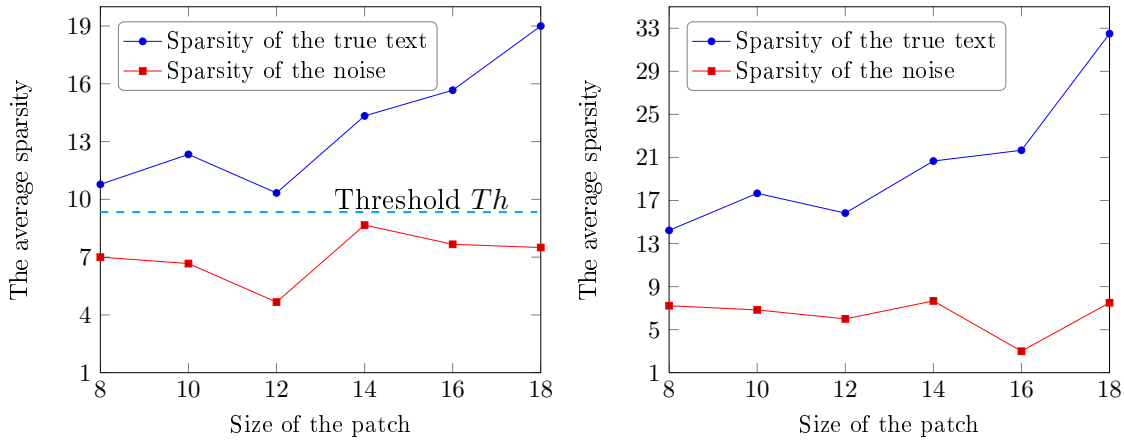


Figure 4.8: Behaviour of the sparsity of the texts and the noise components in the text dictionaries (left) and graphic dictionaries (right).

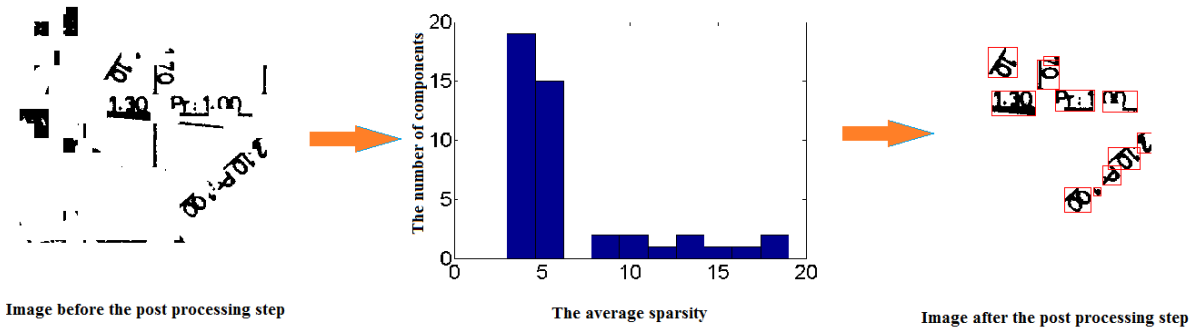
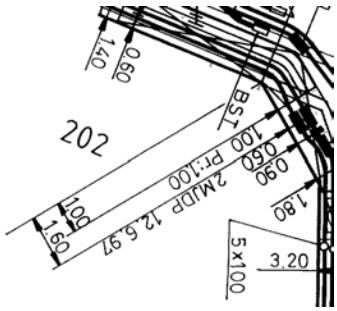
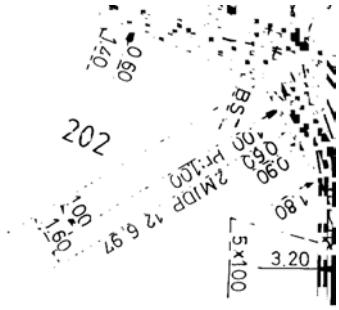


Figure 4.9: Example to illustrate how to further filter out noise components, the value of the threshold is 6.

and sparse representation. The experimental results show that this combination could be a good choice for the segmentation problem with complex graphical documents. We propose an original way to set the sparse thresholds automatically. Additionally, we overcome the restrictions of the existing methods.

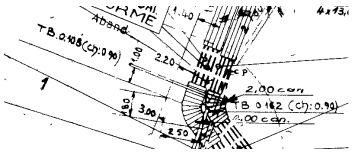


Doc 1: Original image

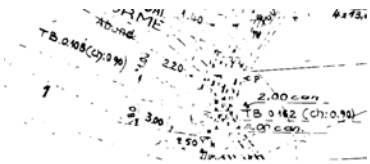


Doc 1: Text layer

Doc 1: Text extraction

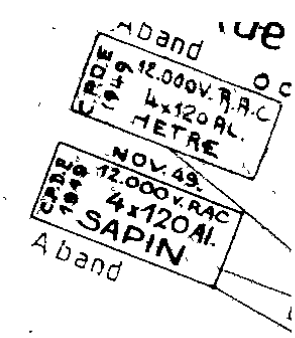


Doc 2: Original image

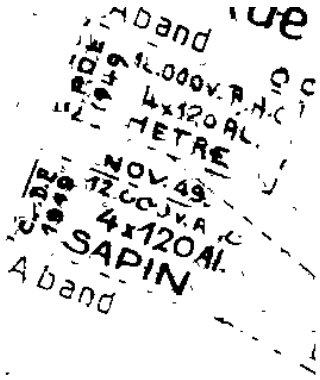


Doc 2: Text layer

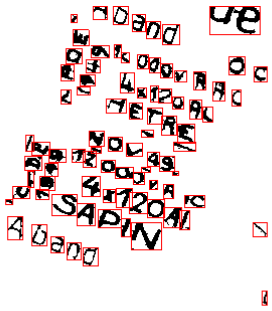
Doc 2: Text extraction



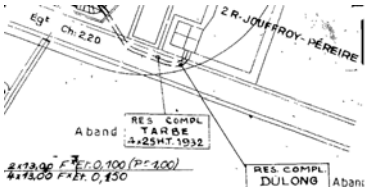
Doc 3: Original image



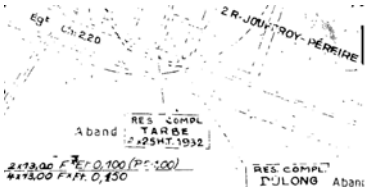
Doc 3: Text layer



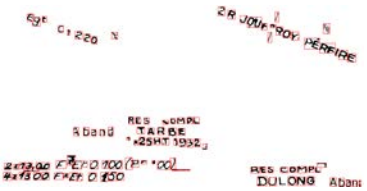
Doc 3: Text extraction



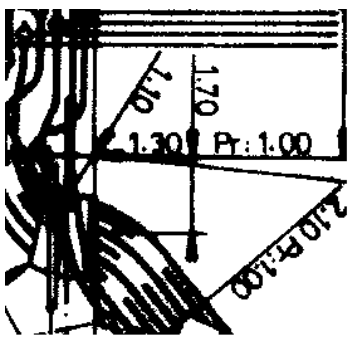
Doc 4: Original image



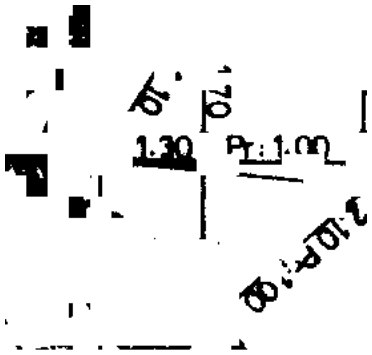
Doc 4: Text layer



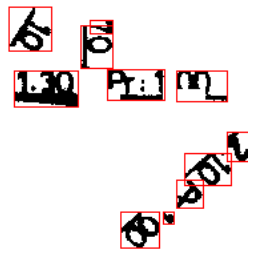
Doc 4: Text extraction



Doc 5: Original image



Doc 5: Text layer



Doc 5: Text extraction

Figure 4.10: Documents used in the evaluation of Table 4.2: Origial images (left), Text layers (midle), Text extraction (right).

Chapter 5

Symbol Recognition

Contents

Sparse representations of images or patches extracted from images have no property of invariance. It means if the image/patch is changed under some transformations such as scale, rotation, degradation, etc., then its corresponding sparse representation is not similar (or almost similar) to the sparse representation of the original image/patch in the same dictionary. In addition, the same size requirement in images is other drawback of sparsity model when working directly on images. Thus, in this chapter, our working hypothesis is '*if a learned dictionary is optimized by taking into account the data properties derived from descriptor, thus it not only specifically adapts to the descriptor but also provides the optimal approximation of the descriptor*'. Therefore, we propose a new approach based on the vector models of the symbols from the sparse representations in the learned dictionary of descriptors. More specifically, we first learn a dictionary describing shape context of interest point descriptors. Then, based on information retrieval techniques, we build a vector model for each symbol based on its sparse representation in a visual vocabulary whose visual words are columns in the learned dictionary. The retrieval task is performed by ranking symbols based on similarity between vector models. The evaluation of our method, using several datasets, demonstrates that the proposed method not only keeps the invariant criteria under the affine transformations but also improves the performance of the symbol retrieval system.

5.1 Introduction

Retrieving similar symbols from document images is still a challenging task in the document image analysis community. The goal here is giving an image example of the symbol, retrieving all the documents and the page locations where it appears. This task naturally appears in human interaction with information systems. There is a need for providing better methods than the existing ones, which can support the number of digitized document collections, and the improvement of the computation capabilities of mobile devices. Similarly related tasks, such as content-based image retrieval (CBIR) techniques, image classification or object recognition techniques, have also received a renewed interest from the computer vision community.

Symbols represent an abstraction of human thinking. Small changes in a symbol can represent a completely different class of symbols. Moreover, if we sketch symbols then finding robust descriptors for geometric distortion while remaining discriminative enough is a difficult task [21, 196, 117, 134]. Recently, the applying of the wide-spread descriptors such as SIFT [102] or HoG [27] in handwriting word spotting tasks obtains various degree of success [146, 3]. However, there is still a limitation on the performance of such descriptors when applied to multi-writer documents. Thus, further research in that direction is going to continue. Authors in [178] apply the Choquet integral to select a suitable set of decision rules that are used to classify symbols. This approach works well in the situation where little information is available for training, and databases having a fair number of categories.

In general, symbol recognition systems are composed of two phases: *symbol description* and *symbol retrieval*. The description phase consists of defining local shape descriptors, invariant to similarity transforms and robust to local symbol distortions and document noise. The retrieval phase consists of implementing matching algorithms along with indexing, or hashing, techniques to effectively retrieve the queried symbol.

In technical document, images usually are gray-scale or black and white images, and usually the only source of information used in symbol recognition tasks is shape information. In addition, the choice of suitable feature descriptors for each task is a critic step in any recognition system, which has generated a large literature [190, 168]. Symbol descriptors can be divided into different groups depending on the properties of *primitives* used to be computed, the *feature extraction method* applied and the data representation used for each *descriptor* [168]. In particular, in symbol retrieval tasks we need local descriptors being able to deal with partial symbol occlusions and non-rigid deformations.

Descriptors based on pixel primitives like moments [21, 86, 23], generic Fourier descriptor (GFD) [189] and Shape Context [10] are invariant to translation, scaling, and rotation. In addition, original symbols can be recovered from moment descriptors. They provide a global description of the whole symbol and; consequently, they assume that symbols have correctly been segmented. However in technical documents, symbols are non-isolated and they are affected by partial occlusions.

On the contrary, descriptors with primitives based on contours, closed regions, connected components or shape skeleton are able to deal with partial occlusions. Although these descriptors can easily be computed, they are usually poorly discriminant [128] or sometimes the matching process is time consuming [123]. There are also several primitives describing symbols using geometric information [103] or vector signatures [38]. In general, they are invariant under similarity transformations, but this invariance depends on a prior normalization step. In addition, they are quite sensitive to noise at vector level.

Syntactic and structural descriptors are suitable descriptors for symbol description since they represent spatial relations between symbol primitives (e.g. lines, arcs) and differences in symbols

usually comes from differences in primitives. Some examples of such descriptors are rule-based descriptions [18, 117], strings [180] and attributed relational graph (ARG) [100, 95, 148, 96]. In general, such descriptors are powerful tools for symbol description, but their performance [18, 117] are highly affected by the noise and the computation time linked to sub-graph isomorphism matching [100] is NP-hard despite some tentative to speed up the process [61].

In summary, the time complexity of matching algorithms of *structural* descriptors is still an important drawback in many symbol retrieval systems. In addition, global descriptors are difficult to be applied in documents with non-isolated symbols. Thus, descriptors locally computed around points of interest like SIFT [101] and a variant of the Shape Context [130] are more suitable for symbol retrieval tasks.

The Bag of Visual Words (BoW) framework has also mostly been used in object recognition applications [153, 97, 149], text detection [193], image classification [183, 80] and symbol description [130]. In the BoW framework, there are two addressed issues: feature description and vocabulary generation. Concerning feature description and as we have seen, SIFT and HoG descriptors are two kinds of feature vectors commonly used in computer vision applications while Shape Context descriptor [10] on interest points have shown interesting results in symbol recognition applications [130]. For visual vocabulary generation, k-means algorithm and product quantization [78] are two normal techniques. These techniques aim to partition feature vectors into clusters in which each feature vector belongs to the cluster and the cluster centroids plays the role of visual words. Vector quantization is an extreme sparse representation where only one *codevector* is assigned to a feature vector so it could be too restrictive, giving rise to often a coarse description of the feature vector [183].

In general, in sparse representations each feature vector is represented as a linear combination of several vectors or atoms. If we consider atoms as visual words, then sparse representation allows representing each feature vector by a group of visual words. As the results it allows to describe the feature vector better and reduce the size of the visual vocabulary. However, to take this advantage of sparse representation, we need to answer two questions: the first one is what are the atoms, and the second one is how many atoms are needed to present a feature vector or symbol descriptor. In the context of this chapter, atoms are the set of primary elements, also called dictionary, learned from training symbol descriptors (Shape Context of Interest Points) by applying the K-SVD algorithm. As mentioned in Chapter (2), the K-SVD algorithm is one of learning dictionary algorithms used to build a learned dictionary from a training database. The elegance and success of learning algorithms in general, and the K-SVD in particular are presented in previous chapters. However, there are still also important shortcomings. The first one is the restriction to low-dimensional signals, i.e., the size 1000 is a reasonable limit not to be surpassed [41]. An attempt to go beyond this raises a series of problems, starting with a need for an intolerable amount of training database, a very slow learning procedure, and a risk of over-fitting because of the too many free parameters in the sought dictionary [41]. This is one of the reasons in image processing algorithms in general and in the two previous chapters, learned dictionaries are used on small image patches, rather than on the whole image.

Another shortcoming is that the learned dictionary as obtained by learning algorithms operates on signals by considering only their native scale. This shortcoming is related to the above-mentioned limits on the dimensionality of the signals (which does not leave much freedom for multi-scale processing) and the computing time issue. In addition, in some applications, we desire the dictionary having specific invariance properties as shift, rotation, and scale-invariances. These imply that when the dictionary is used on a shifted/rotated/scaled version of an image, we expect the sparse representation obtained to be tightly related to the representation of the original image. However, the original learning methodology has not addressed this matter.

As the results, in this chapter and the next chapter, we propose to solve some of the above problems by using sparse representations at the '*higher*' level of information of the image. The '*higher*' level of information in our context is the description of the images. Specially, descriptors extracted from a set of training images now is considered as a family of training database, so we will create the learned dictionaries of descriptors. As the size of the descriptor is usually not high, the restriction on the dimensions overcomes. Moreover, some descriptors have invariance properties under linear transforms, thus learned dictionary built from these descriptors can have also invariance properties. The answer for the second question about how many atoms are needed to present a feature vector can be found by using sparse representation techniques. Sparse representation techniques are a collection of optimization methods seeking the minimum number of atoms needed to represent a given symbol descriptor. Some of the optimization algorithms can be found in [137, 20, 36, 195].

In this chapter, we apply sparse representation techniques over dictionary learned from a set of symbol descriptors for retrieval purposes. More specifically, we first compute shape context of the interest points in symbols and use them as training dataset for constructing a learned dictionary by means of the K-SVD algorithm. Then, we construct a vector model for every symbol based on its sparse representation in the dictionary and the tf-idf approach is adapted to the sparse representation. Finally, the retrieval task is performed by ranking symbols based on their similarity to the queried symbol and where the similarity is computed based on vector models approach.

We have organized the rest of this chapter as follows. In Section 5.2 and Section 5.3, we briefly present some fundamental backgrounds on the shape context, interest points and shape context of interest points descriptors, respectively. Our proposed method and retrieval model are presented in Section 5.5 and we report experimental results in Section 5.6. Finally, we conclude and discuss the future work in Section 5.7.

5.2 Shape Context

The *Shape Context* (SC) is one of the descriptors with higher accuracy rates in many shapes recognition tasks and was introduced in [10]. Shape boundaries, either internal or external, are sampled in n points. For each point p_i on the symbol contour, Belongie *et al* compute its coarse histogram h_i of the relative coordinates of the remaining $n-1$ points:

$$h_i(l) = \#\{c \neq p_i : (c - p_i) \in bin(l)\}, l = \overline{1, L}$$

where c are contours points expressed in log-polar coordinates and L is the number of bins of the SC histogram at point p_i. Thus, for each symbol S, its SC is the real matrix $H = \{h_1, ..., h_n\}$ with dimensions $L \times n$. Figure (5.1) presents an example of how to compute the shape context of a symbol.

Since all measurements are computed with respect to all points that are sampled from internal or external contours on the symbol, SC is therefore invariant under translation. Its invariance under scaling can be obtained by normalizing all radial distances by the mean distance among all point pairs in the symbol. Moreover, it is inherently insensitive to small perturbation of parts of the symbol, and robust to small nonlinear transformation.

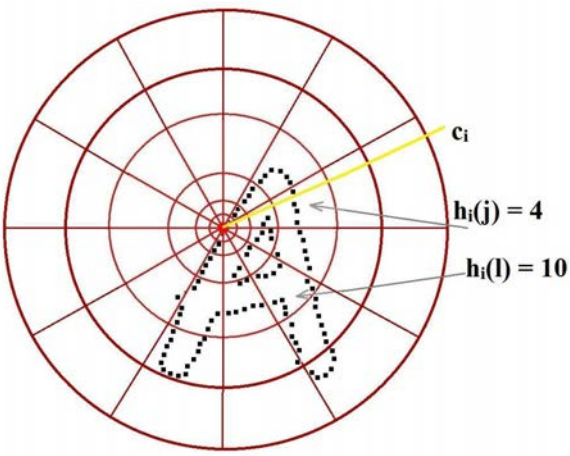


Figure 5.1: Illustration about how to compute the shape context

5.3 Interest Points

Since the beginning of the year 80's, there has been a numerous research works to identify the interest points [24, 25, 151, 102, 120]. The general principle of these methods is searching the interest points either from various resolutions of the image, or from a reasonable resolution selected according to a set of criteria [83]. In general, the approach using various resolutions is preferred and SIFT (*Scale-Invariant Feature Transform*) [102], Harris-Laplace and Harris-Affine [120] are three of the most popular and successful methods to find the interest points following this approach.

Both detectors SIFT and Harris-Laplace are invariant under scaling transforms while Harris-Affine is invariant to more complex transformations such as the viewpoint changes. However, within the framework of graphical symbols applications, we are not interested much in the viewpoint change, so we prefer to use an invariant detector SIFT and Harris-Laplace rather than Harris-Affine. In fact, the detector Harris-Laplace is slightly more successful than the detector SIFT [120]. Nevertheless, the detector SIFT is more advantageous regarding computational complexity because it is based itself on DoG (*Difference of Gaussian*), an approximation close to LoG (*Laplacian of Gaussian*) [120]. We selected in this thesis the detector SIFT even if the other methods can be valid choices.

According to the evaluation of Mikolajczyk *et al* [121], the descriptor SIFT [102] calculated in the interest points gives very good results. Furthermore, the experimental comparisons in [119] showed that maxima and minima of the function LoG produce the most stable image features compared to a range of other possible functions, such as the gradient Hessian or Harris. In [163], the analysis of the behavior of LoG on the junctions of the models showed that the LoG supplies one or several extrema near the junctions which play an important role to distinguish a model of an other one, especially for the graphic symbols. The positions of the extremas depend on the resolution in which the LoG is calculated. We will explain briefly how these positions are affected by recalling the results in [163] with the finite model defined in Equation (5.1).

$$I(x,y) = f(x)f(y)f(W-x)f(tan(\theta)x - y) \tag{5.1}$$

where W and θ are the width and the slope of the model, respectively and f is the Heaviside function defined as following:

$$f(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ 1 & \text{otherwise} \end{cases}$$

Let g, ϕ are the one−dimensional Gaussian filter and the error function at scale σ, respectively:

$$g(x, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{x^2}{2\sigma^2}}$$

$$\phi(x) = \int_{-\infty}^{x} g(t, \sigma) dt$$

then $L(x, y, \sigma)$, is the convolution of the original image $I(x, y)$ with the two−dimensional Gaussian at scale σ defined by $G(x, y, \sigma) = g(x, \sigma)g(y, \sigma)$, is calculated in following equation:

$$\begin{aligned} L(x, y, \sigma) &= G(x, y, \sigma) * I(x, y) \\ &= \phi(y)(\phi(x) - \phi(x - W)) - \int_{x-W}^{x} g(\alpha)\phi(y - \tan(\theta)(x - \alpha))d\alpha \end{aligned}$$

Following Tabbone *et al* [163] the Laplacacian of $L(x, y, \sigma)$ equals

$$\begin{aligned} \nabla^2 L(x, y, \sigma) &= \frac{\partial^2 L}{\partial x^2} + \frac{\partial^2 L}{\partial y^2} \\ &= \frac{1}{\sigma^2}(-yg(y)(\phi(x) - \phi(x - W)) + (x - W)g(x - W)(\phi(y) - \phi(y - W\tan(\theta))) \\ &\quad +(y\cos(\theta) - x\sin(\theta))g(y\cos(\theta) - x\sin(\theta)) \\ &\quad \times(\phi(y\sin(\theta) + x\cos(\theta)) - \phi((y\sin(\theta) + x\cos(\theta)) - \frac{W}{\cos(\phi)}))) \end{aligned}$$

Figure (5.2) presents a finite multi-model defined in Equation (5.1) and its Laplace of Gaussian at two scales $\sigma = 1$ and $\sigma = 5$. This figure also shows the effect of σ on the positions of the extremas. As the LoG, the DoG possesses a similar behavior. In fact, the detector SIFT supplies interest points corresponds to an extremum of the DoG that occurs at multiple scales, so it allows to avoid the arbitrary choice of a resolution. Specifically, a DoG image $D(x, y, \sigma)$ is given by

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$$

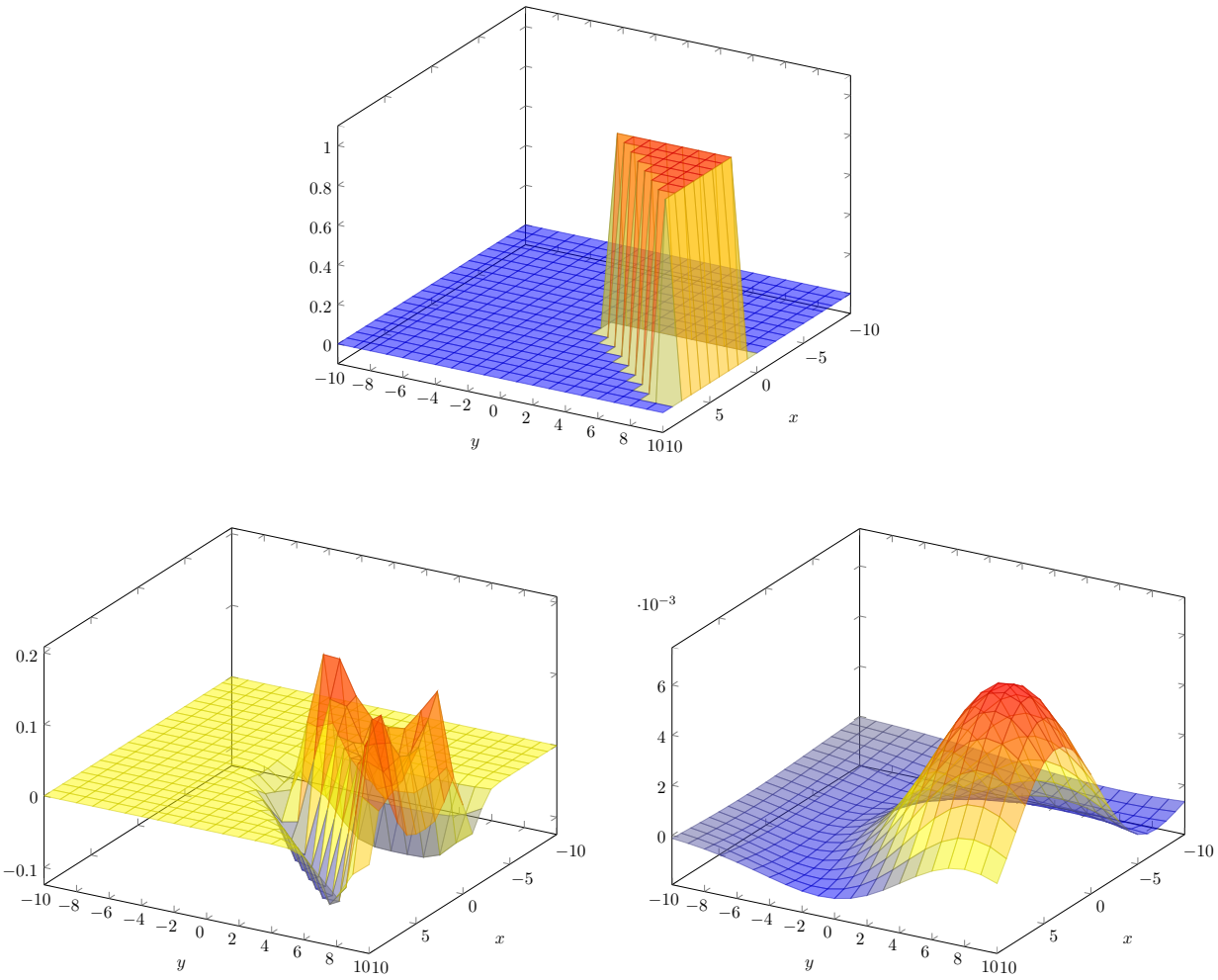


Figure 5.2: Example of a finite multi-model for $W = 10$ and $\theta = \pi/4$ (above). The corresponding Laplace of Gaussian at two scale $\sigma = 1$ (left), $\sigma = 5$ (right).

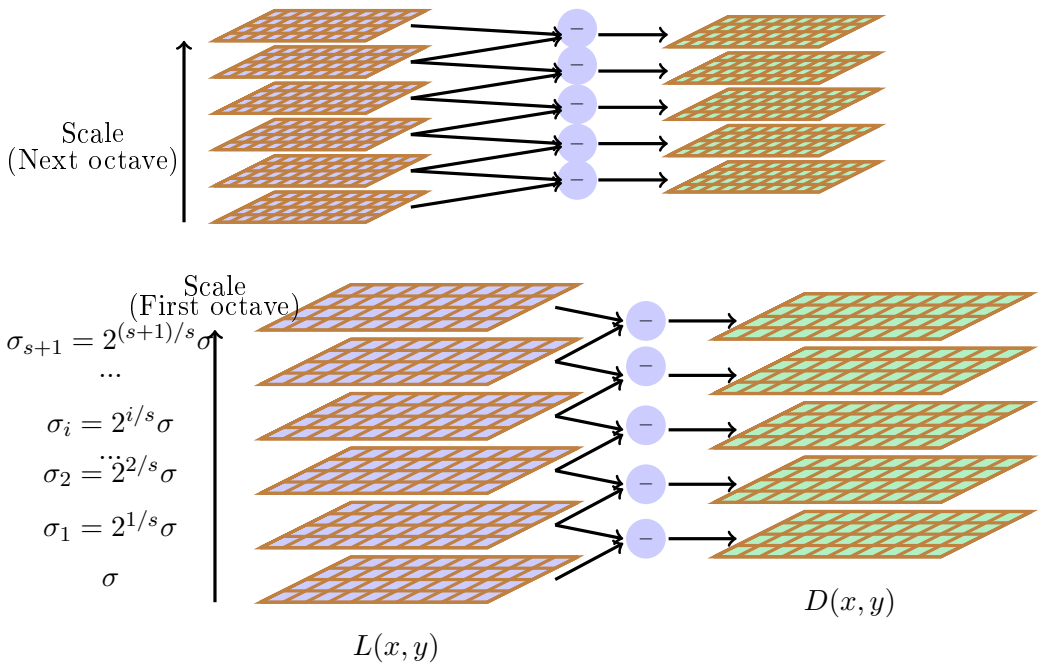


Figure 5.3: Process to create the DoG images.

Figure (5.3) presents how to create the DoG images. The scale space is separated into octaves in which octave 1 uses scale σ, octave 2 uses scale 2σ etc... In each octave, the initial image is repeatedly convolved with Gaussians to produce a set of scale space images (Figure 5.3 (left)), and then the adjacent Gaussians are subtracted to produce the DoG (Figure 5.3 (right)). After each octave, the Gaussian image is down-sampled by a factor of 2 to produce an image a quarter the size of the next level.

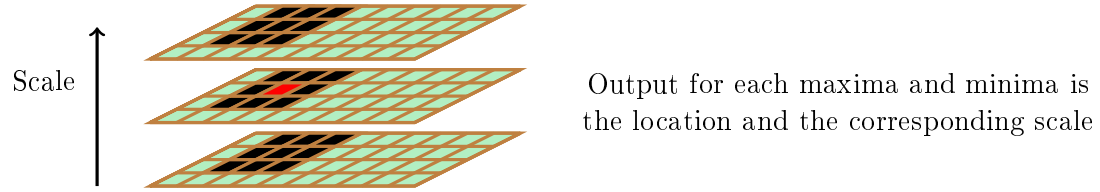


Figure 5.4: Maxima and minima of the DoG images are detected by comparing the pixel (marked by red color) neighbors in the current and adjacent images (marked by black color)

To detect the local maxima and minima of $D(x, y, \sigma)$, each sample point is compared to its eight neighbors in the current image and nine neighbors in the scale above and below (see Figure 5.4). This point is considered as an interest point candidate only if it is larger than all of these neighbors or smaller than all of them. Once the interest point candidates have been found, the next step is to filter out interest points with low contrast and with poorly localized along an edge. It is done by using 2×2 Hessian matrix and the derivative of the function being the Taylor expansion of $D(x, y, \sigma)$ calculated in the position and the scale of the considered point. The orientation of an interest point is determined by the local gradient dominating in the region around this interest point.

Thus, by using SIFT algorithm, each interest point p_i is presented by $(x_i, y_i, \delta_i, \theta_i)$ where (x_i, y_i) is the coordinates of p_i; δ_i is the resolution where p_i is detected (corresponding to σ) and θ_i it the orientation of p_i.

5.4 Shape Context of Interest Points

SC defined so far shows two main drawbacks when applied to symbol retrieval tasks. On the one hand, as most of *photometric* descriptors, we need to segment symbols well enough for having satisfactory retrieval performance. On the other hand, matching function is computationally time-demanding if the number of boundary points is large.

Inspired by the works of detecting efficiently an object from its key-points (also known as interest points) [102, 1], the authors in [130] proposed a new approach, named *Shape context of interest point* (SCIP). In their approach, SC is only defined around interest points. More specifically, interest points $IP = \{p_1, p_2, \ldots, p_r\}$ and contour points $C = \{c_1, \ldots, c_n\}$ of a given symbol are detected. Indeed, each of these interest points p_i is thus a reference point to compute the SC of a symbol. Because the IP set is rarely a subset of C for most of the cases [163], the same rotation normalization method for SC cannot be applied to SCIP. Instead, dominant orientation of interest point information for orientation normalization is used. Each interest point p_i is represented by its coordinates and the dominant orientation: $p_i = \{x_i, y_i, \vec{e_i}\}$. The relative log-polar coordinates of contour points $c_j \in C$ is denoted by $c_j^i = (\log r_{ij}, \theta_{ij})$ in which r_{ij} is the normalized distance from p_i to c_j, and $\theta_{ij} = \langle \overrightarrow{p_i c_j}, \vec{e_i} \rangle$. The coarse histogram at p_i is

computed as:

$$\bar{h}_i(l) = \#\{c_j^i \neq p_i : (c_j^i - p_i) \in bin(l)\}, l = \overline{1, L}$$

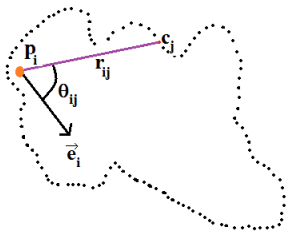


Figure 5.5: The relative log-polar coordinates of c_j with regard to p_i

Then, the SCIP descriptor is the set $\bar{H} = \{\bar{h}_1, \bar{h}_2, \ldots, \bar{h}_r\}$, where each $\bar{h}_i$ is a histogram of L bins.

5.5 Proposed Approach

Querying symbols on a dataset using SCIP descriptor needs to accurately assign each SCIP descriptor to a visual word. In the proposed approach we avoid this step by describing each SCIP descriptor by a linear combination of *visual* words being the columns of the learned dictionary A. In this section, we will build a learned dictionary from a set of SCIP descriptors providing sparse representation and then, we will explain how to built vector models permitting to query symbols in a dataset.

5.5.1 Learned Dictionary of SCIPs

As explained in Section (5.1), beside of the elegance and success of learning methodology, there are also important shortcomings to it such as the limitation about the dimension of the signal, and the lack of invariance properties under linear transforms of learning dictionaries. Thus, to overcome these shortcomings, in this chapter, we propose to use sparse representations at the '*higher*' level of information of the images. The '*higher*' level of information in our context is the description of the images. The over-complete learned dictionary A for sparse representation now is a dictionary built from a training database being the symbol descriptors (shape context of interest points). Indeed, shape context of interest points (SCIPs) $\{\bar{H}_n\}_{n=1}^N$ extracted from a set of N training symbols now is considered as a family of training database. So each SCIP $\bar{h}_i \in \mathbb{R}^L$ has an optimally sparse approximation x_i^* satisfying $\|\bar{h}_i^* - \bar{h}_i\|_2 \leq \epsilon$ with $\bar{h}_i^* = Ax_i^*$.

$$\min_{x_i, A} \sum_i \|x_i\|_1 \text{ subject to } \|\bar{h}_i - Ax_i\|_2^2 \leq \epsilon \tag{5.5}$$

In this chapter, the learned dictionary A is constructed by using the K-SVD algorithm because of advantages of this algorithm (see Chapter (2)). In fact, the initial dictionary A is constructed from a random of M descriptors in the training set $\bar{H}$ with the normalized column-wise by the l_2-norm. Then, we execute the K-SVD algorithm until a fixed number of iterations or the approximation error is smaller than a fixed value of ϵ. After applying the K-SVD algorithm on the training set $\bar{H}$, we learn a dictionary A and the sparse representations of all SCIP descriptors in $\bar{H}$.

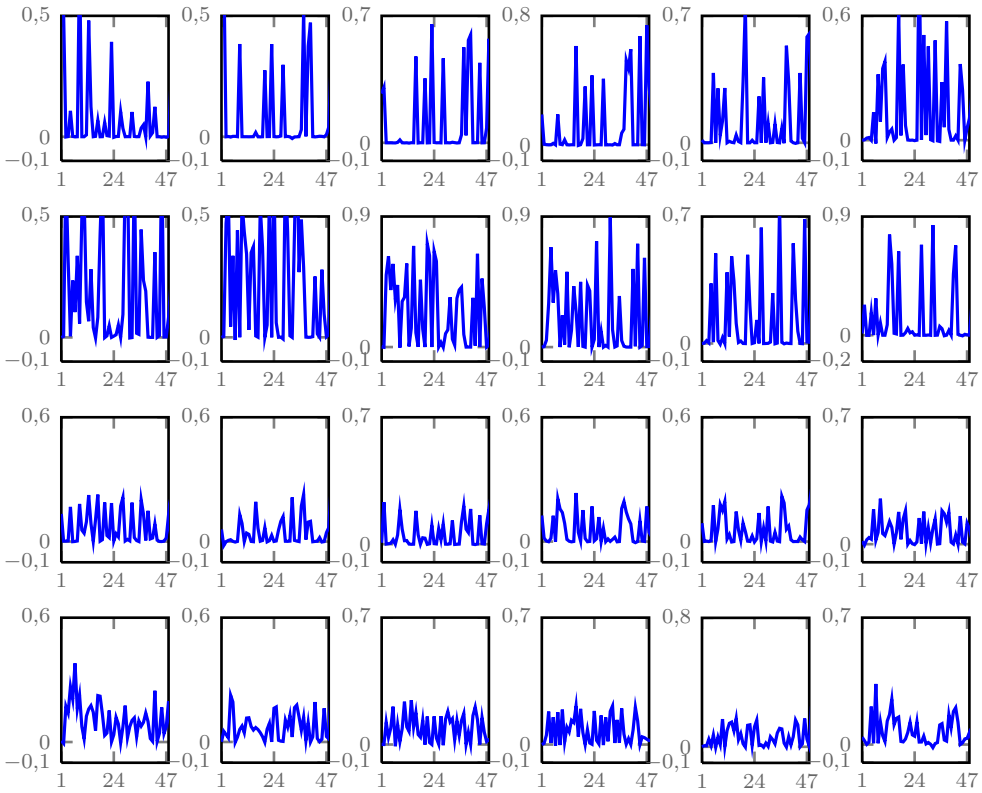


Figure 5.6: The 24 atoms (columns) of learned dictionary built from SCIP descriptors of symbols in GREC2003 database.

Figure (5.6) shows examples about columns of learned dictionary built from SCIP descriptors of symbols in GREC2003 database.

5.5.2 Visual vector model

After applying the K-SVD algorithm on the set of SCIPs descriptors $\{\bar{H}_n\}_{n=1}^{N}$ used as training dataset, we have learned a dictionary $A \in \mathbb{R}^{L \times M}$ as well as the sparse representations of all SCIPs in that dataset. In the remainder of this Section, we explain how to combine sparse representation of SCIP descriptors with the vector model framework used in retrieval tasks.

Without loss of generality, we can assume that $\bar{h} \in \mathbb{R}^{L}$ is one SCIP descriptor in $\{\bar{H}_n\}_{n=1}^{N}$ and $x \in \mathbb{R}^{M}$ is the sparse representation of $\bar{h}$ given the dictionary A obtained by applying the OMP algorithm. Instead of using vector quantization techniques and assigning a single visual word to each SCIP [154, 130], we can see $\bar{h}$ as a linear combination of visual words where each visual word is each column of the learning dictionary A. Therefore, the *sparse vector model* is the extension of vector model to sparse representations.

In general, vector model provides a representation of the discriminative words at the document level. Thus, we have a M dimensional vector of real values denoting the relative importance of words according the number of occurrences of such word in the document (*tf* factor) and its discriminative capacity (*idf* factor). Similarly, we have constructed the sparse vector model at symbol image level, and herein, documents and symbol images are identified.

For each training n, we denote v_i^n the characteristic vector of $\bar{h}_i^n$. Then v_i^n is generated based on the sparse representation x_i^n of $\bar{h}_i^n$ over the learned dictionary A. More specifically, let x_i^n be

the sparse representation of $\bar{h}_i^n$ in A, or x_i^n is the solution of following equation:

$$\min_{x_i^n} \|x_i^n\|_0 \text{ subject to } \|\bar{h}_i^n - Ax_i^n\|_2 \leq \epsilon$$

The solution x_i^n is obtained by applying the OMP algorithm. Because x_i^n is sparse, so the number of zero elements in x_i^n is larger than the number of nonzero elements. Let I be the indexes set where x_i^n is different to 0, then the characteristic vector $v_i^n \in \mathbb{R}^M$ is the $0-1$ valued vector $v_i^n(k) = 1$ if $k \in I$ and 0 otherwise. For example, if

$$x_i^n = \{\alpha_1, 0_2, \ldots, \alpha_{p-1}, \alpha_p, 0_{p+1}, \ldots, \alpha_q, \alpha_{q+1}, 0_{q+2}, \ldots, 0_M\}$$

then $I = \{1, p-1, p, q, q+1\}$ and

$$v_i^n = \{1_1, 0_2, \ldots, 1_{p-1}, 1_p, 0_{p+1}, \ldots, 1_q, 1_{q+1}, 0_{q+2}, \ldots, 0_M\}$$

Next, we define the tf and idf factors with the purpose of describing the document contents and the importance degree of terms, respectively, in a similar way to the *tf-idf* approach for building vector models in information retrieval. Thus f_k^n is the frequency of the word k in a symbol n and

$$tf_{k,n} = \frac{f_k^n}{\max_s f_s^n}$$

Observe that we can easily computed these frequencies through the characteristic vector:

$$f_k^n = \sum_i^{r_n} v_i^n(k)$$

with r_n is the number of interest points extracted from the training symbol n. The idf factor is similarly defined as usual in information retrieval systems but also adapted to the sparse representation of SCIP descriptors. The importance in distinguishing a relevant symbol from non-relevant one in a database is measured by $\log \frac{N}{l_k}$, where l_k is the number of symbols in which the word k appears, and N is the number of symbols in the database.

$$l_k = \#\{n = \overline{1, N} | f_k^n \neq 0\}$$

Therefore, the vector model for a given symbol is defined by the weighted frequency for all words k in our visual vocabulary:

$$w^n(k) = tf_{k,n} \times idf_k = \frac{f_k^n}{\max_s f_s^n} \times \log \frac{N}{l_k} \tag{5.6}$$

This means that the vector model for a given symbol n is the vector w^n in which $w^n(k)$ is the k-*th* element of w^n, or

$$w^n = (w^n(1), w^n(2), ..., w^n(k), ..., w^n(M))$$

5.5.3 Retrieval Symbol

For each query symbol s_i^q in the set of query symbols $S_q = \{s_1^q, s_2^q, \ldots, s_m^q\}$, its vector model is computed in the same way described in Section (5.5.2). We first compute its SCIP descriptor $\bar{H}_i^q = \{\bar{h}_1^q, \ldots, \bar{h}_{r_i}^q\}$. Then, using the learned dictionary A, we find the sparse representation of each element in $\bar{H}_i^q$ by applying the OMP algorithm [137, 172] to solve Equation (5.7) with the same value of ϵ used in training dictionary A in Section (5.5.1). Finally, we compute the vector model, named w_i^q, as summarized in Equation (5.6).

$$\min_{x_i} \|x_i\|_1 \text{ subject to } \|\bar{h}_i - Ax_i\|_2^2 \leq \epsilon \tag{5.7}$$

Next, the similarity of the query symbol $s_i^q \subseteq S_q$ and the symbols in the database $s^n \subseteq S$ is computed as the cosine distance between two vectors w_i^q and w^n:

$$\text{distance}(w_i^q, w^n) = \frac{\langle w_i^q, w^n \rangle}{|w_i^q| \times |w^n|}$$

where $\langle \cdot, \cdot \rangle$ is the dot product. Finally, the symbols in the database are ranked based on their similarity to the query symbol s_i^q.

5.6 Experimental Validation

One of the main challenges in symbol retrieval is the capacity of symbol retrieval system to retrieve good results when the class of the queried symbol does not exist in the dataset used for learning the visual dictionary. The experiments designed in this section are devoted to show that the proposed scheme outperforms other related approaches even in such cases.

We have designed three groups of experiments that have allowed us to judge the performance of the sparse vector model approach under several conditions of the training set. We have devoted the first group of experiments in finding the best parameters of the symbol retrieval system given the benchmark datasets considered. The second group of experiments, aim at evaluating the performance of the SCIP descriptor, comparing to other shape descriptors, on datasets in which symbols suffer different degree of geometric distortion, noise, rotation and scale transforms. In addition, we have compared its performance to the k-means algorithm if it is used to learn the dictionary. The last group of experiments seeks to evaluate the retrieval system in a more realistic configuration in which some symbol classes does not exist in the training set.

5.6.1 Datasets and Performance Evaluation

We have considered three public datasets. The first dataset is the synthetic GREC2003 dataset [1], used in many symbol recognition contest since 2003. The second dataset, called herein CVC dataset, is a handwritten version of the GREC2003 dataset, and it is created by the Computer Vision Center [117]. The third dataset is the FRESH dataset composed of real symbol instances [150].

The GREC2003 dataset includes 50 different classes of symbols with different number of instances for each class [173]. This dataset was originally created to evaluate symbol descriptors performance under different geometric distortions of symbols; rigid transforms (rotation and scale) and noise simulating document degradation. So, different tests were created depending on

[1] http://www.cvc.uab.es/grec2003/SymRecContest/

the degradation level, geometrical distortion and rigid transform applied on them. We have taken the images of such tests and grouped into three different subsets: the first one is composed of only rotated and scaled symbols, the second one is composed of geometrically distorted of instances of original symbols and the last one is composed of distorted and noisy instances symbols.

The CVC dataset of handwritten symbol is the selection of 459 handwritten symbols drawn by 14 different volunteer people from the CVC.

The FRESH dataset includes symbols extracted from aircraft electrical wiring diagrams of real world industrial drawings. Differences between symbols are due to slight details in symbols, and the number of instances of each class of the symbol is imbalance. Consequently, symbols have been grouped into *similar* symbol classes according the agreement of 6 volunteers who have participated in the ground-truth generation [150]. Figure (5.7) shows some examples of symbols in all datasets that we used in this chapter.

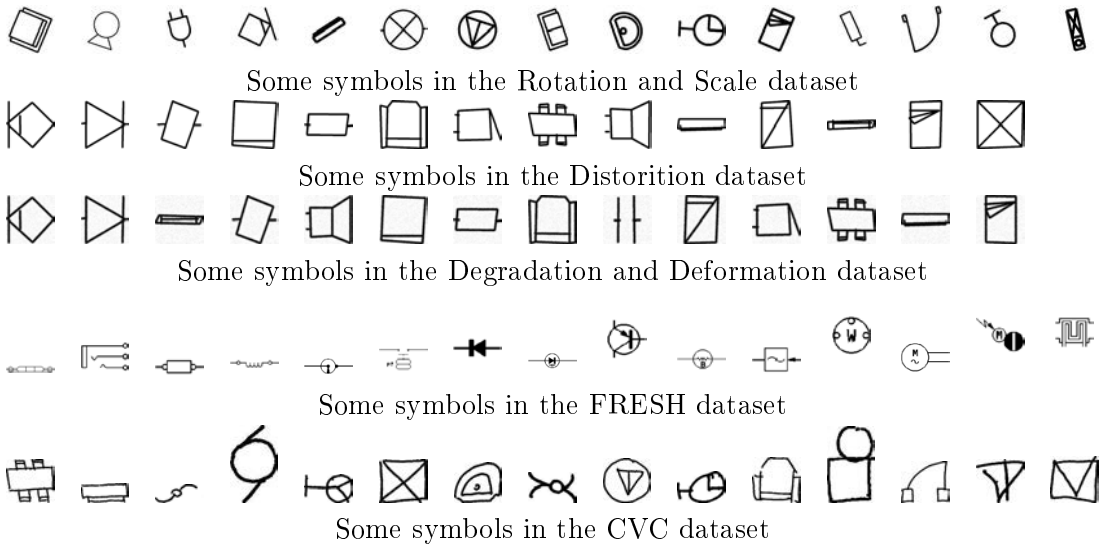


Figure 5.7: Examples about the symbols in different datasets

For evaluation purposes, we create two partitions for each dataset: a training partition and a test partition. The training partition is used for learning the descriptors dictionary and the test partition is the symbol images used as queries in the evaluation. In each dataset, the search set is the same than the training set. The obtained results is reported and compared with related methods using the *area under the precision-recall curve* (AUC-PR) metric [30]. AUC-PR metric is a simple matrix to define how algorithm performs over PR space. It can be calculated by using the trapezoidal areas created between each PR points.

We use the implementation introduced in [145] to learn the visual dictionary of SCIP descriptors, and we repeat 10 times of each experiment due to the random initialization of the K-SVD algorithm. To assess if differences on the AUC-PR values are significant, or not, from a statistical viewpoint, we perform a two-sided Wilcoxon rank sum test with 5% of significance level.

5.6.2 Study of Parameters

This group of experiments aim at finding the best parameter values of the proposed scheme: the dimension of SCIP descriptor, the approximation error and the size of visual dictionary,

respectively named L, ϵ, and M. An exhaustive search in the parameter space will require run a K-SVD algorithm for each combination of these three parameters taken a lot of time in a PC with a standard configuration. Thereby, we have fixed one of these parameters, the size of visual dictionary since we want to evaluate the discriminative power of the sparse representation of SCIP descriptors comparing related approaches given the same size of vocabulary. We have fixed the size of visual dictionary (or the number of columns in A) $M = 512$ for all the experiments done in this chapter.

Concerning the SCIP dimension, we have sampled the radial and angular parameters respectively in the following sets of values: $\{3, 4, 5\}$ and $\{12, 16\}$; thus SCIP dimension $L =$ radial $\times$ angular belongs to $L = \{36, 48, 60, 48, 56, 80\}$. Furthermore, we have sampled the approximation error ϵ, defined in the problem (5.5), in an interval of values ranging from 0.01 to 0.2.

Thereby, with each pair (L, ϵ) of the sampled parameter space, we learn a dictionary on the training dataset having the size of 395, 459, and 164 corresponding respectively to GREC 2003, CVC and FRESH dataset. For each training dataset, we compute the AUC-PR values and repeat this scheme 10 times since the K-SVD is randomly initialized. Tables (5.1), (5.2) show the average of AUC-PR values for the rotation and scale, CVC, and FRESH dataset computed over the 10 experiment repetitions. In these tables, an entry marked by (-) indicates that the corresponding sampled parameter performs worst than the best sampled parameter (in bold). Similarly, an entry marked by (+) indicates that the corresponding sampled parameter outperforms the others, and an entry marked by (=) indicates that results obtained by the both sampled parameter are not significantly different.

	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1	0.2
(12, 3)	0.533(-)	0.591(-)	0.595(-)	0.607(=)	0.611(=)	0.606(=)	0.615(=)	0.612(=)	0.560(-)	0.603(-)	0.602(-)
(12, 4)	0.345(-)	0.491(-)	0.556(-)	0.596(-)	0.605(=)	0.602(-)	0.613(=)	0.614(=)	0.609(=)	**0.620**(+)	0.106(-)
(12, 5)	0.290(-)	0.417(-)	0.558(-)	0.574(-)	0.597(-)	0.607(=)	0.608(=)	0.610(=)	0.604(=)	0.607(=)	0.614(=)
(16, 3)	0.421(-)	0.538(-)	0.580(-)	0.594(-)	0.598(-)	0.601(-)	0.604(-)	0.601(=)	0.603(-)	0.595(-)	0.610(=)
(16, 4)	0.308(-)	0.433(-)	0.542(-)	0.582(-)	0.599(-)	0.602(=)	0.612(=)	0.613(=)	0.606(-)	0.603(-)	0.619(=)
(16, 5)	0.211(-)	0.347(-)	0.504(-)	0.568(-)	0.584(-)	0.596(-)	0.593(-)	0.608(=)	0.584(-)	0.610(=)	0.616(=)

Table 5.1: Average AUC-PR values for the rotation and scale dataset.

	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09	0.1	0.2
(12, 3)	0.0766(-)	0.2062(-)	0.2360(-)	0.2710(-)	0.2920(-)	0.2561(-)	0.2498(-)	0.2860(-)	0.3100(-)	0.3026(-)	0.2958(-)
(12, 4)	0.0848(-)	0.1439(-)	0.2124(-)	0.2593(-)	0.2928(-)	0.3091(-)	0.3232(-)	0.3236(-)	0.3364(-)	0.3211(-)	0.3156(-)
(12, 5)	0.0831(-)	0.1301(-)	0.2211(-)	0.2887(-)	0.3095(-)	0.3106(-)	0.3471(-)	**0.3604**(+)	0.3470(-)	0.3348(-)	0.3378(-)
(16, 3)	0.0719(-)	0.1791(-)	0.2266(-)	0.2584(-)	0.2316(-)	0.2078(-)	0.2222(-)	0.2847(-)	0.2887(-)	0.2873(-)	0.3034(-)
(16, 4)	0.0726(-)	0.1257(-)	0.1796(-)	0.2586(-)	0.2849(-)	0.3139(-)	0.3171(-)	0.3220(-)	0.3144 (-)	0.2873(-)	0.3034(-)
(16, 5)	0.0515(-)	0.0992(-)	0.1875(-)	0.2529(-)	0.3018(-)	0.3195(-)	0.3310(-)	0.3432(-)	0.3371(-)	0.3318(-)	0.3154(-)

Table 5.2: Average AUC-PR values for the CVC dataset.

The best values about the retrieval time and the AUC-PR for all datasets are presented in Table (5.3). The (+) on the right side of the average of AUC-PR values indicates that the AUC-PR values obtained for the best configuration are significant better compared to other pairs (L, ϵ).

Dataset	L	ϵ	Av. time (s)/query	Av. AUC-PR
GREC2003	48 (12 ×4)	0.1	1.380	**0.6199** (+)
CVC	60 (12 × 5)	0.08	0.831	**0.3604** (+)
FRESH	36 (12 × 3)	0.02	20.824	**0.4018** (+)

Table 5.3: Best values for the datasets.

5.6.3 Invariance and Robustness

These experiments aim to prove that using sparse representation for the vector models of symbols is not only keeping the invariance criteria under the affine transformations but also improving the performance (for the retrieval) of the symbol retrieval system.

A learned dictionary A is optimized by taking into account the data properties derived from the descriptors, thus it not only specifically adapted to a descriptor but also provides the optimal approximation of the descriptor. If the descriptor is invariant under the transformations, its optimal approximation constructed from linear combination of some columns in A, also keeps this criterion. Results in Figure (5.8) show that the retrieved list of symbol images are almost the same regarless the angle and the scale of the query symbol.

To evaluate the performance of sparse vector model of SCIP descriptors, we compare our approach to 6 state-of-the-art descriptors reviewed in Section (5.1) for symbol recognition, namely R-signature [164], GFD [188], Zernike moments [166], SIFT [102], SC [10] and SCIP [130]. In addition, we have used two descriptors for Zernike moments descriptors related the considered order moments. The first descriptor, G_1 includes 32 low-order moments while the second descriptor, G_2, includes 32 hight-order moment. We have only considered the magnitude of Zernike moments for both descriptors G_1 and G_2 and for each one, the computed moments satisfy the following conditions:

$$G_1 = \begin{cases} 3 \leq n \leq 10 \\ |m| \leq n \\ n - |m| = 2k \\ k \in N \end{cases} \quad G_2 = \begin{cases} 10 \leq n \leq 17 \\ |m| \leq n \\ n - |m| = 4k \\ k \in N \end{cases}$$

We have applied all these descriptors on the all datasets and the results are summarized in Table (5.4).

	G_1 Zernike	G_2 Zernike	R-Signature	GFD	SC	SCIP	SIFT	SIFT+sparse	Our Approach
Rotation and Scaling	0.057	0.075	0.041	0.202	0.088	0.548	0.153	0.213	**0.620**
Distoration	0.661	0.504	0.519	0.638	0.699	0.761	0.447	0.497	**0.773**
Deformation and Degradation	0.499	0.273	0.460	**0.530**	0.220	0.292	0.203	0.234	0.457
CVC	0.064	0.015	0.053	0.025	0.002	0.220	0.021	0.029	**0.360**
FRESH	0.266	0.222	0.343	0.314	0.301	0.286	0.355	**0.443**	0.402

Table 5.4: Retrieval effectiveness with AUC-PR values in different datasets

In addition, the non-sparse version of SIFT, and SCIP, descriptors also depend on the random initialization of the k-means algorithm. Consequently, we have also repeated the experiments 10 times, as we have done for the sparse representation of the SIFT descriptor and the SCIP descriptor.

To perform this experiment we have used the learnt dictionaries in the previous experiment and their respective best values for L and ϵ. To avoid the effect of the random initialization we have used again the same 10 dictionaries learnt in the previous experiment.

Dataset	Descriptor + sparse	25%	50%	75%	100%
GREC2003	SCIP	0.610 (=)	0.592 (=)	0.610 (=)	0.620
	SIFT	0.211 (=)	0.203 (=)	0.209 (=)	0.213
CVC	SCIP	0.295(-)	0.306(-)	0.340(-)	0.360
	SIFT	0.024 (-)	0.027 (=)	0.028 (=)	0.029
FRESH	SCIP	0.386 (=)	0.387(=)	0.397(=)	0.402
	SIFT	0.429(-)	0.440(=)	0.436(=)	0.443

Table 5.5: Average of the AUC-PR values considering several percentages of training set size.

We can observe that there is a significant improvement of retrieval schemes using SCIP descriptors comparing to other retrieval schemes using symbol descriptors like Zernike moments, SC descriptors and R-Signature. Only the GFD achieves similar results than the proposed scheme in the degraded set of the GREC2003 dataset. This result can be explained by two facts. On the one hand, we have applied GFD to the whole image, since symbols in this dataset are fully segmented. On the other hand, the key-points detection step is sensitive to noise. This fact also explain the poor performance of SIFT descriptor on that datasets. Moreover, it was also observed that SIFT descriptor looses its effectiveness when working with symbols in [130].

Nevertheless, comparing our proposed method to the others based on key-points extraction, SIFT and SCIP, we achieve better retrieval performance. It therefore shows that constructing a vector model from the sparse representations of descriptors provides better results than using cluster algorithms like k-means and assigning just one visual word to each local descriptor.

Some retrieval examples are giving using our approach in Figure (5.10), (5.11). Thanks to the sparse representations over learned dictionary, we can remark that our approach is robust to scale, rotation and degradation. Figure (5.9) presents the examples of the best and worst retrieval results in all datasets in terms of the AUC-PR value.

5.6.4 Unseen symbols

One of the main difficulties in symbol recognition and symbol retrieval system is the relative few number of instances of each kind of symbol. This fact makes the task of any learning algorithm harder and consequently their performance usually drop down in general purpose symbol recognition tasks. However, the sparse representation proposed in this work is based on the learning of a dictionary of SCIP descriptors that have been computed on a training dataset composed of a given set of kind of symbols. It seems quite obvious that symbols in the training dataset have an impact in the final sparse representation, and the question is whether the learned dictionary has the capacity of describing symbols that have not been used in the learning step, *unseen symbols*. This last experiment is devoted to evaluate the discrimination capacity of symbol retrieving for unseen symbol instances.

In this experiment, only a subset of searching set used as training dataset is used in the learned dictionary. The vector models for other symbols in searching set are built as the same way we do with the query symbol using this learned dictionary. The sub-training datasets are composed of 25%, 50% and 75% of training dataset coming from the GREC2003, CVC and FRESH dataset, respectively. Indeed, symbols used for training sets are randomly selected, and we have again repeated 10 times all the experiments. The parameters used for building the dictionary are the best value of L and ϵ found in Section (5.6.2). The two-sided Wilcoxon sum rank test also is performed to asses if observed difference is statically significant or not.

Table (5.5) shows the results obtained in this experiment. We can see that, for the GREC2003

and the FRESH dataset, there are no significant differences when using a smaller training set than using the whole dataset. This is significantly important for the FRESH dataset since symbols have been grouped according to their similarity. Only the retrieval performance, in the 25% subsets and for the SIFT descriptor, seems to decrease a bit. In addition, we get the same results using SIFT with 50 % of the training dataset. Nevertheless, the behavior of SCIP and SIFT descriptor is quite similar in all cases. For the CVC dataset, the reported result seems to show a drop in the retrieval performance for the SCIP descriptor. This can be caused by the fact that the CVC dataset is composed of handwritten symbols. If we note, also that the average of the AUC-PR is quite low (0.360) also for the 100% training set, so maybe we should increase the size of the dictionary.

In conclusion, this last experiment have proven that our approach is robust even if symbols are not in the learning set.

5.7 Conclusion

In this chapter we have applied sparse representation on visual vocabulary for symbol recognition tasks. Thus, we have taken a symbol descriptor and we have adapted the sparse representation theory as well as the learning dictionary algorithm. Moreover, we have extended the traditional vector model used in retrieval tasks, to sparse representation. The approach is general and easily applied to any descriptor.

We have evaluated this approach on several reference symbol datasets in a symbol recognition task. The reported results show a stable behavior of the system depending on the symbol descriptor. It means that, although a parameter tuning has to be done in order to select the best system parameter, there is a wide range of values shared by all the datasets, given similar results. Moreover, we have also study the robustness of sparse representation to affine transforms and symbols distortions. The reported results also shown a good behavior in such cases compared to other related methods. Finally, we have studied the capacity of sparse descriptors to represent unseen symbols. The reported results also show a good behavior in this case.

In the next chapter, we will extend this approach and apply it to the symbol spotting problem in the large graphical documents where symbols can not be easily segmented.

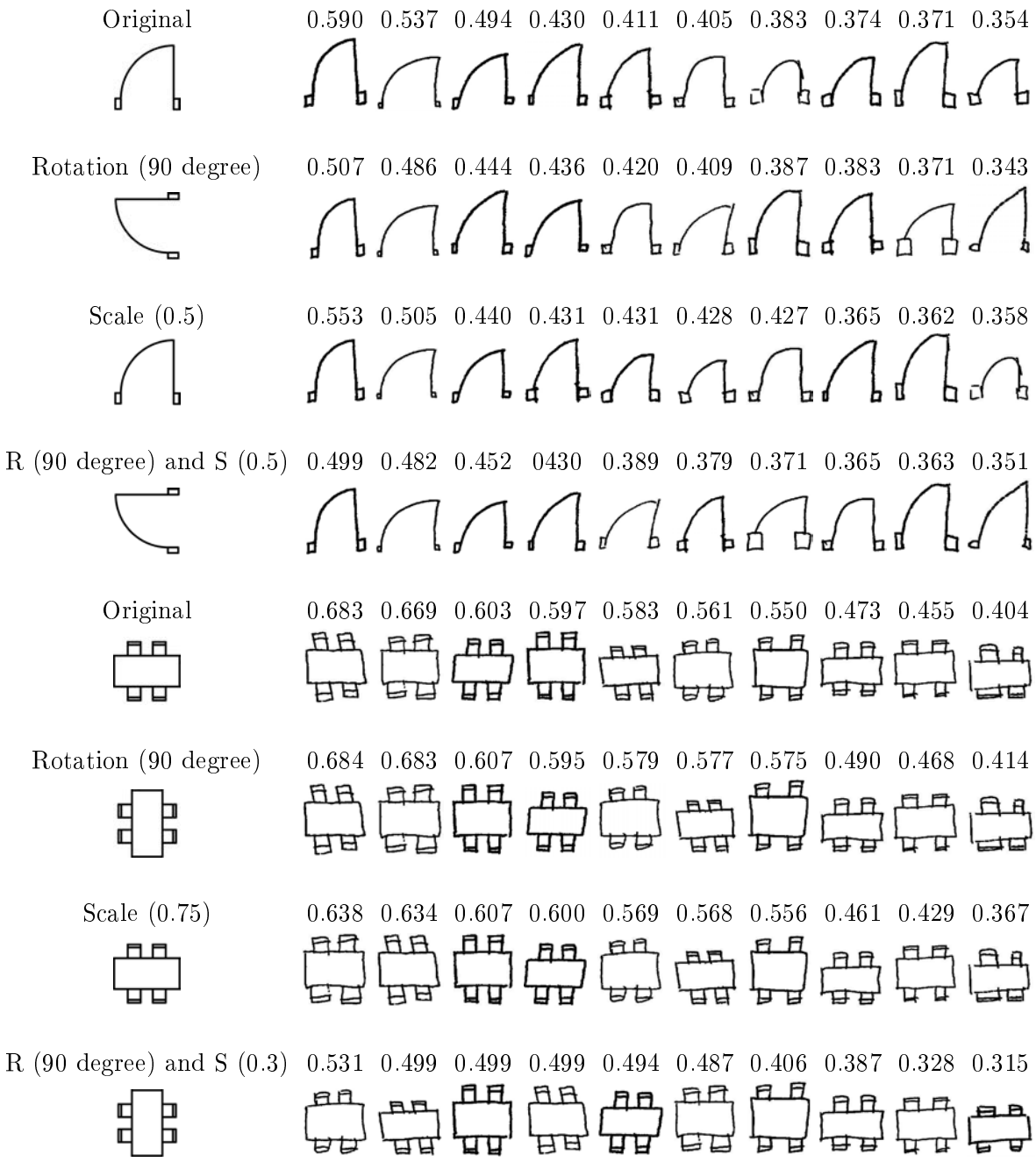


Figure 5.8: Retrieval symbols on CVC dataset when request (first column) is rotated, scaled.

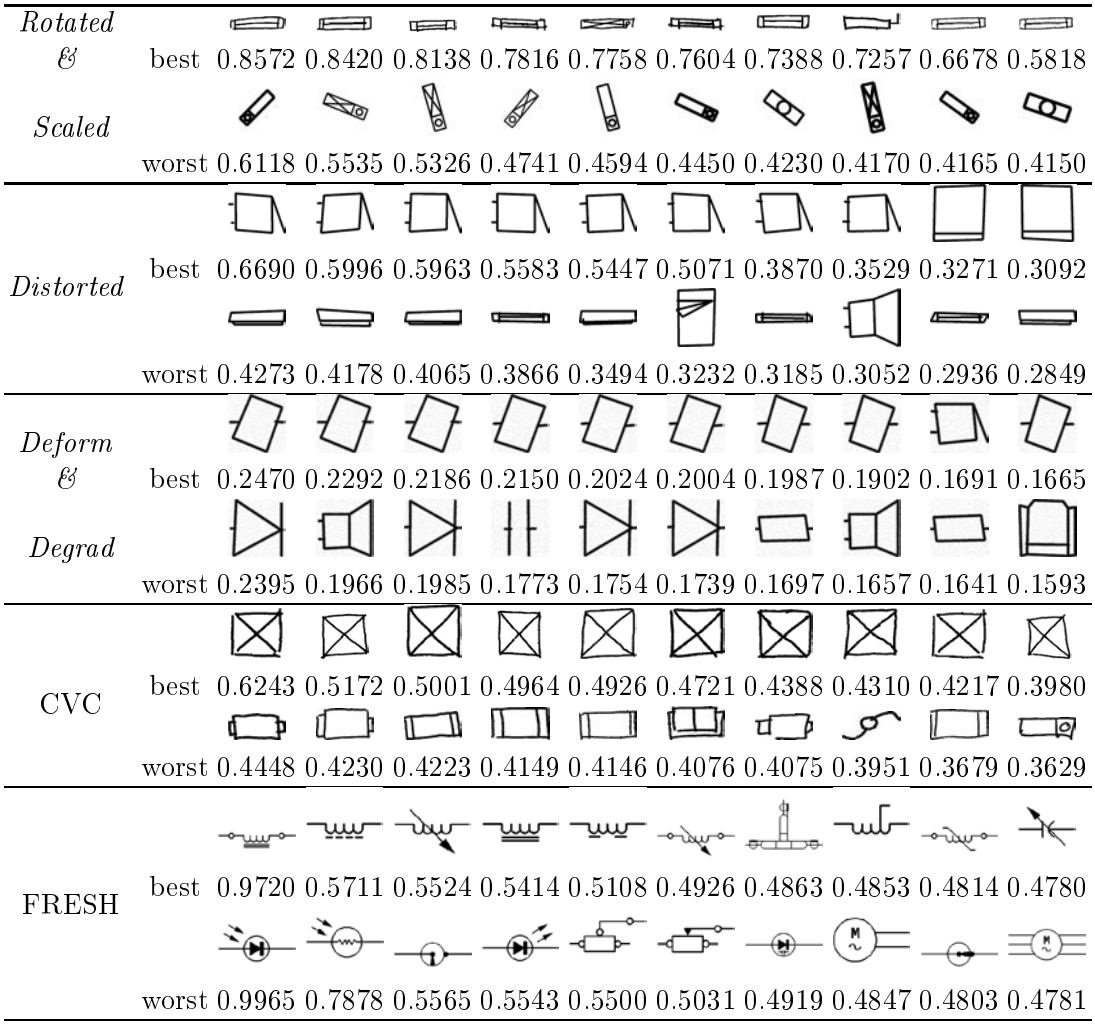


Figure 5.9: Examples of querying symbols achieving the best and worst retrieval results in terms of the AUC-PR value. Values correspond to the cosine distance.

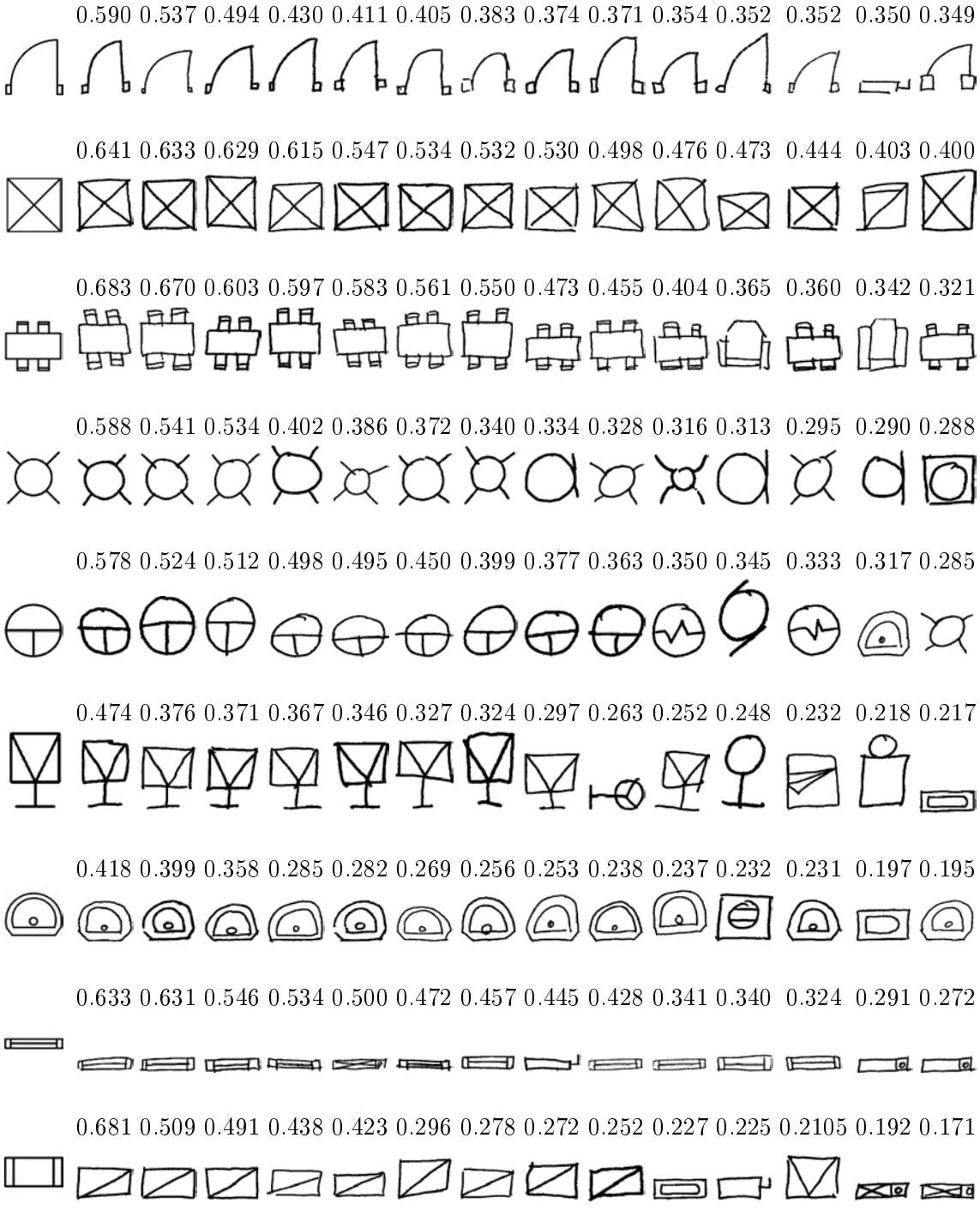


Figure 5.10: Some retrieval examples in CVC dataset: the query symbol is in the first column; other columns are the nearest matches ranked from left to right.

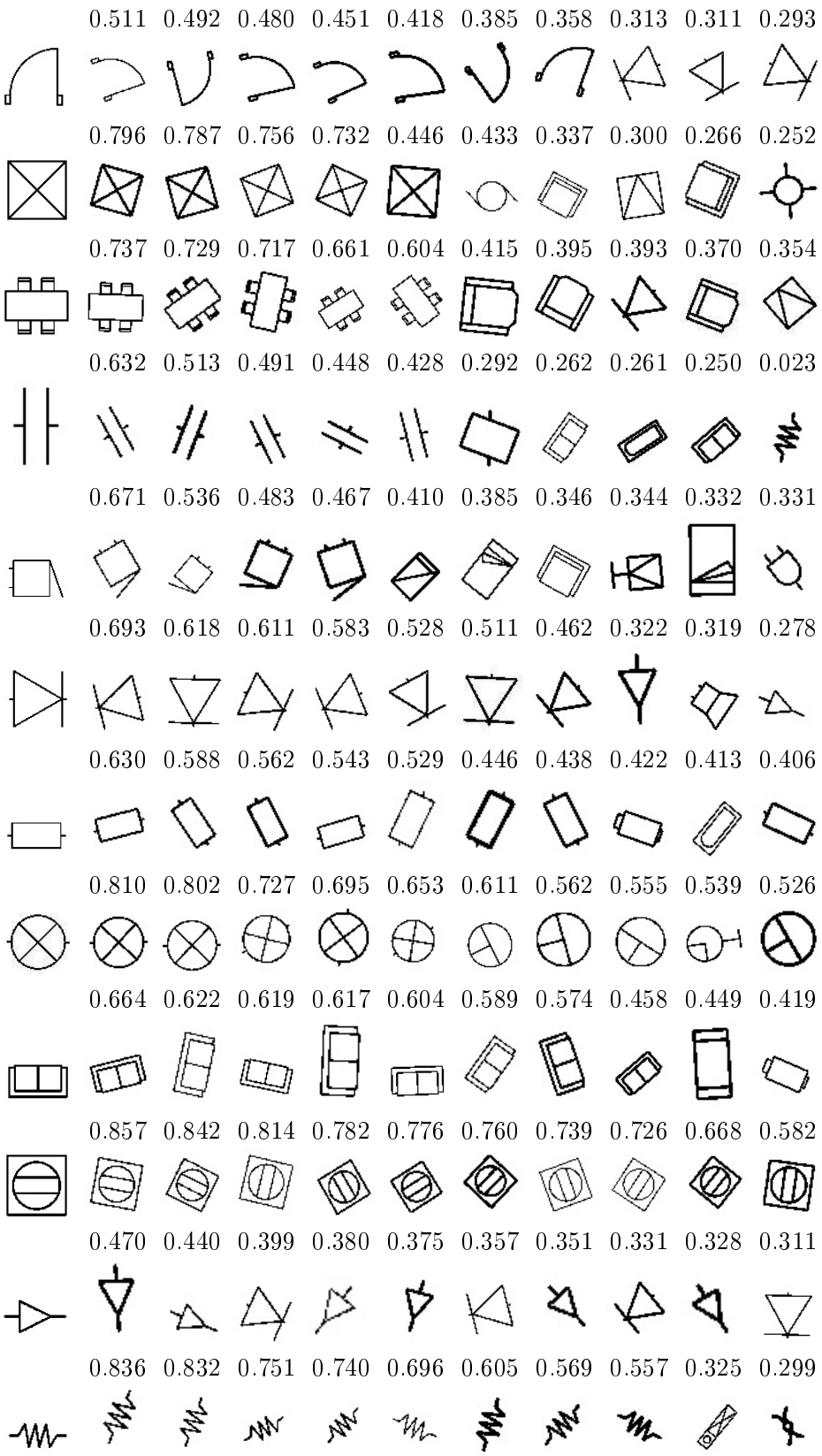


Figure 5.11: Some retrieval examples in GREC dataset: the query symbol is in the first column; other columns are the nearest matches ranked from left to right.

Chapter 6

Spotting Symbols

Contents

This chapter proposes a new approach to spot symbols into graphical documents using sparse representations. More specifically, a dictionary is learned from a training database of local descriptors defined over documents. Following their sparse representations, interest points sharing similar properties are used to define interest regions. Using an original adaptation of information retrieval techniques, a vector model for interest regions and for a query symbol is built based on its sparsity in a visual vocabulary where the visual words are columns in the learned dictionary. The matching process is performed comparing the similarity between vector models. Evaluation on SESYD datasets demonstrates that our method is promising.

6.1 Introduction

Identifying regions likely to contain a graphical symbol within graphics-rich documents (see Figure 6.1) has taken a lot of efforts in the last years among the Graphics Recognition community. Even if this problem is still an emerging topic, several works can be found. One of the first approaches was the retrieval of engineering drawings based on the use of a pseudo 2-D Hidden Markov Model [125]. The main advantage of that method is that it can still work well even with symbols embedded in documents with cluttered background. However, there is a need of training a classifier for each class of symbols, reducing the flexibility of that approach.

Structural approaches, focusing on the structural information inherent in graphical symbols have also been applied [7, 100, 118]. In that methods, the location and recognition of graphical symbols is done based on subgraph isomorphism techniques. Subgraph isomorphism is a generalization of both the maximum clique problem and the problem of testing whether a graph contains a cycle; therefore it is NP-complete. In general graph matching schemes are computationally expensive, although particular cases of subgraph isomorphism can be solved in

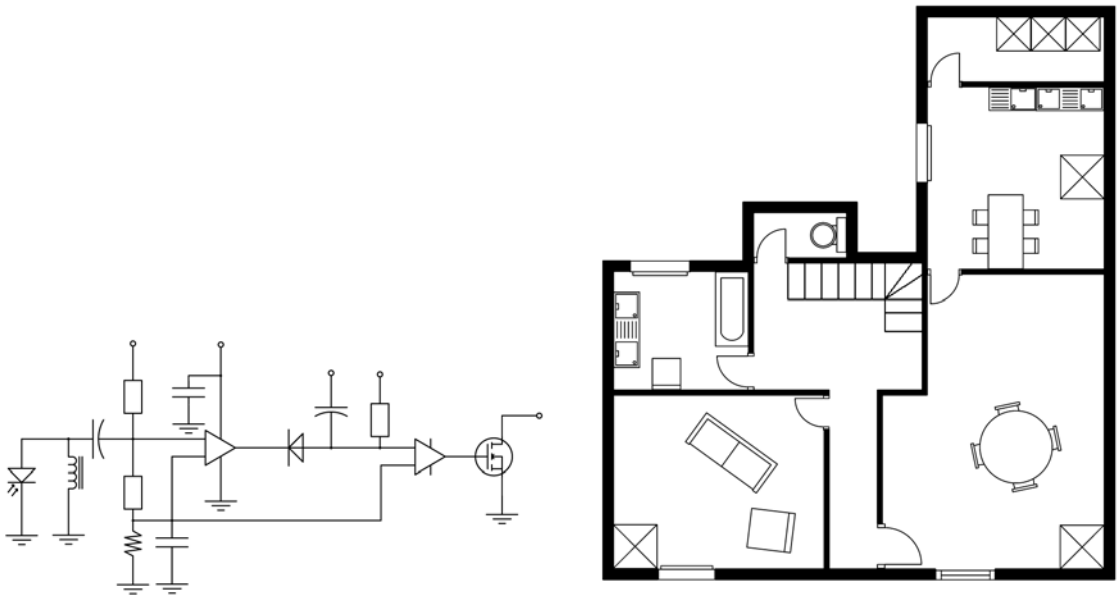


Figure 6.1: Example of the graphic documents

polynomial time [48]. So, these approaches are inadequate when facing large collections of data unless indexing strategies are used to reduce the retrieval time and to increase the potential applications of this approach [165, 39].

In general, symbol segmentation methods requires some kind of assumption about symbol class such as number of connected components, size, symbol type, to enumerate some of them. A method based on graphical documents vectorization which avoid the segmentation step was proposed in [196]. In that method, chain points where conveniently merged according to a density measure in a dendrogram framework. Consequently, since that recognition technique did not need a prior segmentation, no assumption regarding the class of symbol was required.

In this chapter we propose a method for symbol spotting in graphical documents. The proposed approach, relies on a strategy combining off-line and on-line processes. During the off-line process, we learn a dictionary of visual words providing a sparse representations of shape descriptors computed over key-points. During the on-line process, symbol retrieval is performed in two steps: the *recall* and the *refining* steps, respectively. In the recall step, we use indexing techniques like inverted files to efficiently retrieve regions of interest where symbol is more likelihood to appear. Then, in the refining step we build vector models to filter out only those regions where the queried symbol appears.

We organize this chapter as follow: in Section 6.4, we present how to use sparsity over learned dictionary to describe graphical document. Particularly, Section 6.2 and Section 6.3 describe how to calculate the local descriptor adapted to the graphical document, and how to learn a dictionary from the training set being the local descriptors. The details of symbol spotting system is addressed in Section 6.5. The first evaluations of our approach is dedicated in Section 6.6. Finally, we conclude and discuss the future work in Section 6.7.

6.2 Extension of Shape Context of Interest Points for Documents

In Chapter 5 we described the Shape Context of Interest Point (SCIP) descriptor based on the interest points. Basically, SCIP descriptor describes objects nearby an interest point and its invariance to scaling and rotation can be obtained thanks to the information about the dominant orientation of interest points. However, for the whole document since symbols in the document have not been segmented, and using interest points, the contour points being far from them provide less useful information to discriminate objects, so the same strategy of SCIP used in [130] therefore cannot be applied at document level. Instead, for each reference interest point, its neighborhood region is defined. This region has to ensure the invariance of SCIP computed inside it, so it cannot be fixed a priori. This difficulty is overcome by using the scale on which the interest point detected [129]. More details, with each interest point, the neighborhood region associated with it is an ellipse that is defined with the center at this point, and the semi-major axis, the semi-minor axis are decided depending on the scale in which this interest point is detected. Figure (6.2) presents the interest points and the corresponding neighborhood regions in the document.

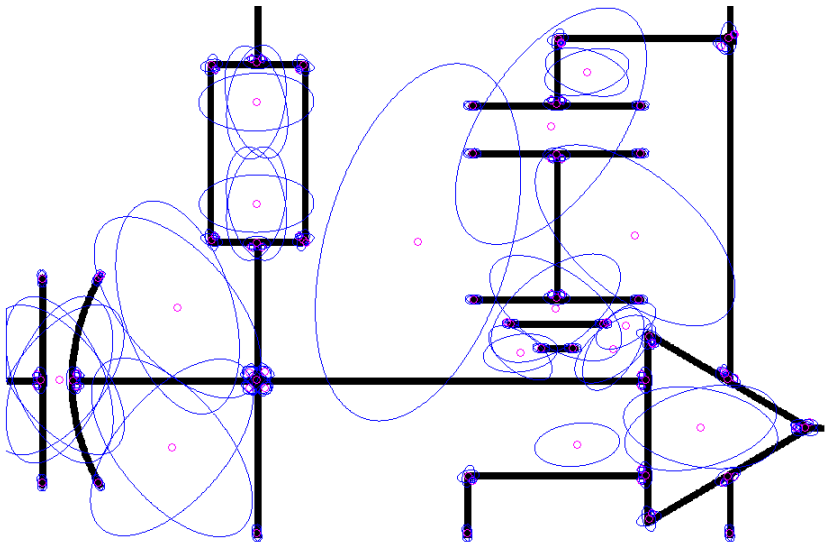


Figure 6.2: One of the scanned documents

This extension of SCIP descriptor for a document level is called ESCIP descriptor from now on. In fact, ESCIP for the neighborhood region corresponding to one interest point in the document is the SCIPs calculated on this neighborhood region. More specifically, let $\mathcal{P}^i = \{p_1^i, p_2^i, \ldots, p_r^i\}$ and $\mathcal{C}^i = \{c_1^i, \ldots, c_r^i\}$ be the interest points and the contour points of the neighborhood region $\mathcal{E}_i$ built from interest point number i-*th* of one document, respectively. Each of these interest points p_j^i is thus a reference point to compute the SCIP descriptors, so the ESCIP descriptors of $\mathcal{E}_i$ is the set $H_i = \{h_1^i, h_2^i, \ldots, h_r^i\}$, where each h_j^i is a histogram of L bins (as defined in Chapter (5), Section (5.2)) [129]. Figure (6.3) shows one interest point p_j^i, its ellipse $\mathcal{E}_i$ with the contour points inside $\mathcal{C}^i$ and ESCIP h_j^i at the point p_j^i.

If the document has m interest points, the ESCIP descriptors of the document is the set $\mathcal{H} = \{H_1, ..., H_m\}$.

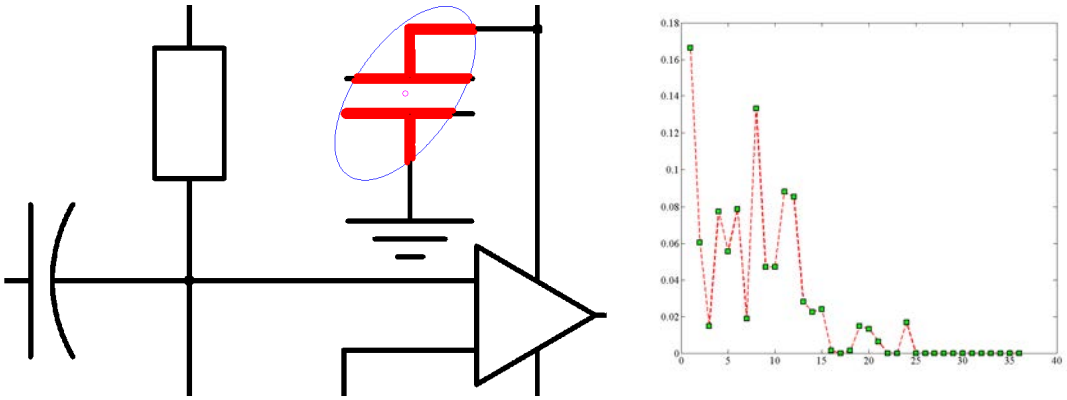


Figure 6.3: An example about ESCIP descriptor at interest point p_j^i of document

6.3 Learned Dictionary of ESCIPs

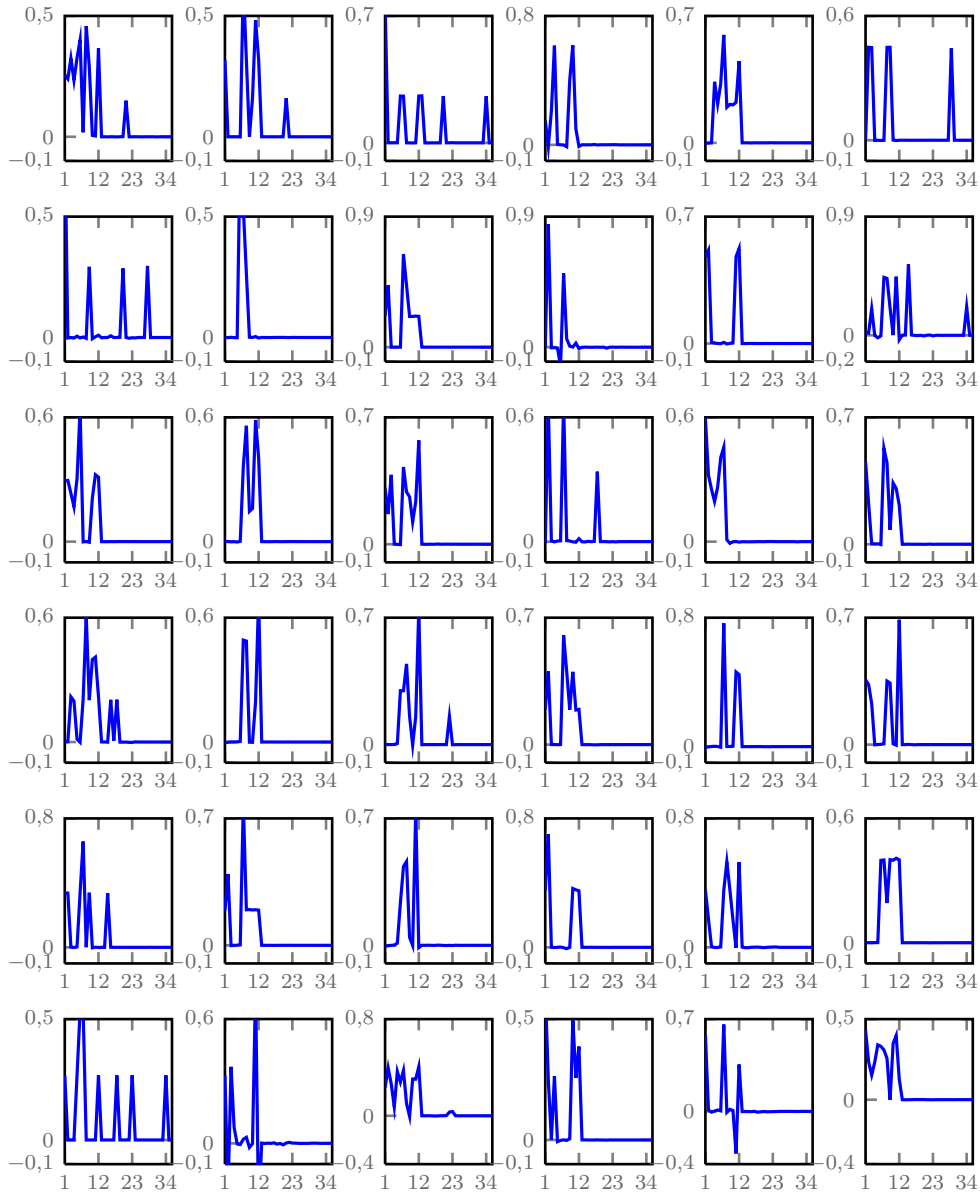


Figure 6.4: The 36 atoms (columns) of learned dictionary obtained by using ESCIP descriptors as the training dataset.

The explanation of an over-complete learned dictionary A for sparse representation is mentioned many times from the beginning of this thesis, it is the dictionary built from a family of training signals. So, in this Section, instead of repeating the learning methodology for constructing the dictionary A, we will describe what is the dictionary of ESCIP descriptors and illustrate how one signal is presented over this dictionary.

The learned dictionary of ESCIP descriptors is the dictionary built from the training dataset $\mathcal{H} = \{\mathcal{H}_1, ..., \mathcal{H}_n\}$ being the ESCIP descriptors extracted from n documents. One of the training signal is presented in Figure (6.3) (right). By applying one of the learned algorithms (see Section

(2.3)), we learned the dictionary that satisfies that each ESCIP descriptor $h_j^i \in \mathbb{R}^L$ in training dataset has an optimally sparse approximation $\bar{x}_j^i$ in this dictionary. Where $\|\bar{h}_j^i - h_j^i\|_2 \leq \epsilon$ with $\bar{h}_j^i = A\bar{x}_j^i$ and $\bar{x}_j^i$ is the solution of Equation (6.1).

$$\bar{x}_j^i = \min_{x_j^i, A} \sum_i \|x_j^i\|_1 \text{ subject to } \|h_j^i - Ax_j^i\|_2^2 \leq \epsilon \tag{6.1}$$

This problem can be solved efficiently by using K-SVD algorithm. It approximately expresses the input h_j^i as a sparse linear combination of the columns in $A = \{a_1, ..., a_K\} \in \mathbb{R}^{L \times K}$. The sparse vector $\bar{x}_j^i$ is our new representation for h_j^i.

Using a set of M learned ESCIPs bases (as in Figure (6.4)), Figure (6.5) illustrates a solution to the optimization problem, where input h_j^i being one of the ESCIP descriptors (Figure (6.3), right) is approximately expressed as a combination of 7 basis vectors (or columns in A) $a_{190}, a_{216}, a_{230}, a_{320}, a_{370}, a_{422}, a_{499}$. The ESCIP descriptor h_j^i can now be represented via the vector $\bar{x}_j^i \in \mathbb{R}^M$ with $\alpha_{190} = -0.0308$, $\alpha_{216} = 0.0151$, $\alpha_{230} = 0.0447$, $\alpha_{320} = -0.0392$, $\alpha_{370} = -0.0194$, $\alpha_{422} = -0.0751$ and $\alpha_{499} = 0.3134$.

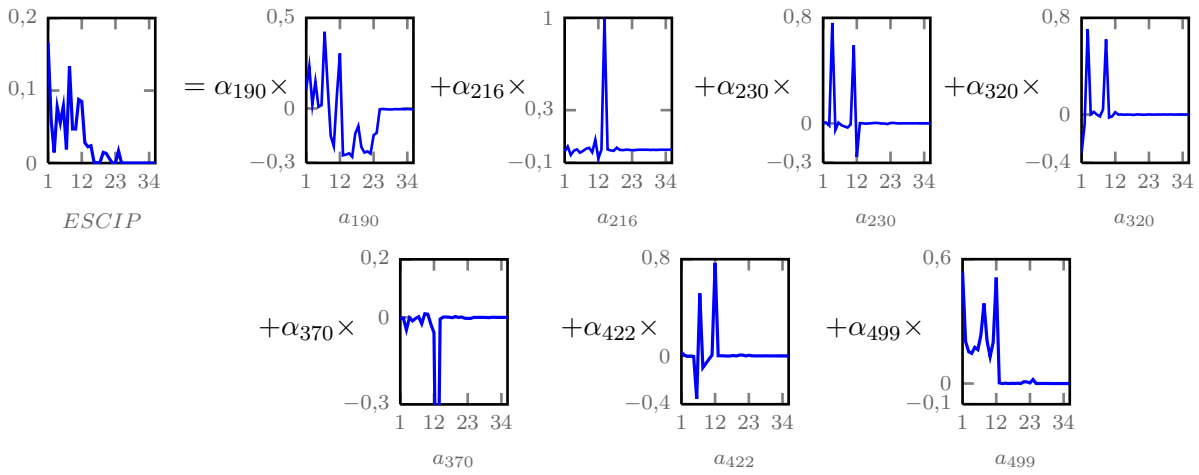


Figure 6.5: Illustration of a solution to the opimization problem over learned dictionary.

6.4 Document Indexing by Sparsity over Learned Dictionary

Generally, the number of ESCIP descriptors describing each document is large because of the large number of detected interest points. In addition, ESCIP descriptor provides a local description of symbols, so searching and then matching of all ESCIPs will waste the memory as well as the time. So, some clustering techniques in the literature are proposed to overcome this difficulty by grouping similar descriptors, to define '*visual words*' being the centroids of clusters. However, a good classification depends on which clustering algorithm is applied on what kind of data.

In this chapter, we propose another way to associate each ESCIP to some '*visual words*' being the columns of learning dictionary A built in the previous section. As mention in Section (6.3), after applying a learned algorithm, K-SVD algorithm, on all the ESCIPs $\mathcal{H}$ of n documents, we got a learned dictionary A as well as all the optimal sparse representations of ESCIPs in $A = \{a_1, ..., a_M\}, a_i \in \mathbb{R}^L$. If we consider each column of A as a '*visual word*', then A becomes a *'visual vocabulary'* and the sparsity of ESCIP in A gives us the group of '*visual words*' used.

Without loss of generality, consider h_i^s is the ESCIP number *i-th* in the document number *s-th*, and the sparse representation of h_i^s in A is:

$$h_i^s = \alpha_1^s \times a_1 + \alpha_k^s \times a_k + \alpha_t^s \times a_t + \alpha_l^s \times a_l$$

It means ESCIP of the interest point number *i-th* in the document number *s-th* is assigned to the corresponding group of visual words $W_i^s = \{a_1, a_k, a_t, a_l\}$ and the coefficients in its representations $\Delta_i^s = \{\alpha_1^s, \alpha_k^s, \alpha_t^s, \alpha_l^s\}$.

Once the document is described by visual words being the columns of learned dictionary A, an inverted file structure is used to index the content which helps to match candidate regions in each document. The inverted file structure is composed of two elements: the vocabulary and the occurrences. The vocabulary is the set of *visual words* or the set of the columns of the learned dictionary A. For each *visual word*, a list of all interest points having this word in their sparse representations over dictionary A, the corresponding documents, their groups of visual words as well as the coefficients in their representations are stored (see Figure (6.6)).

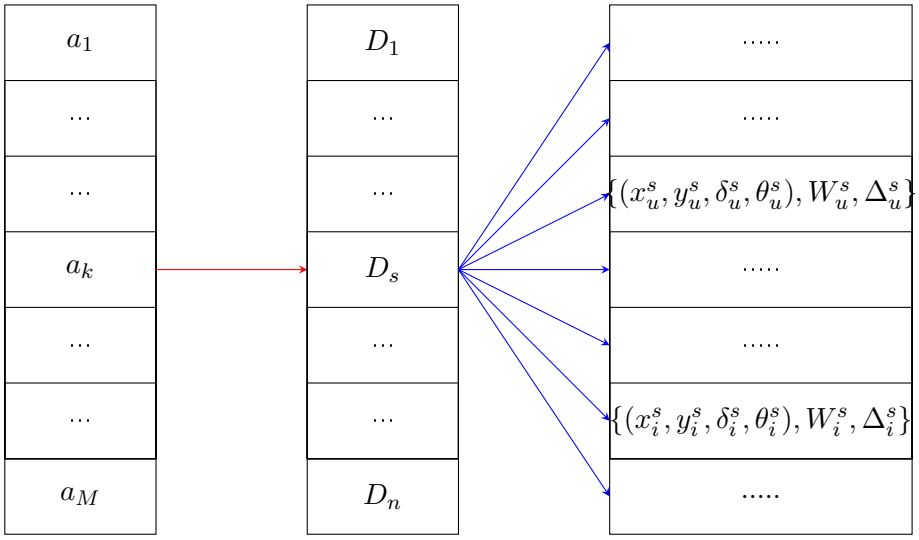


Figure 6.6: The inverted file structure

6.5 Location symbols in the graphical documents

The symbol spotting method proposed in this paper consists of a two steps process called: *recall* and *refining* steps. As it name suggests, the recall step aims at detecting regions of interest in the stored documents where the queried symbol may appear. The refining step aims to filter out the obtained regions in the previous step and keep only those where the queried symbol effectively is.

6.5.1 Symbol Recall

Given a symbol query, one of the main issues in spotting methods is to determine the regions of interest where the queried symbol may appear since symbols can be rotated or scaled. In general, the shape descriptors used are invariant to rotation and scale. Thus and for each extracted key-point in query symbol and each key-point, we propose to define a candidate region of interest

taking into account the orientation and the scale of both key-points. More specifically, given $p = \{x_p, y_p, \delta_p, \theta_p\}$ and $q = \{x_q, y_q, \delta_q, \theta_q\}$ respectively a key-point of the document dataset and a key-point of the queried symbol. They are represented by their coordinates (x, y), the scale δ and the orientation θ. The center of coordinates of the region of interest (x_r, y_r) are defined by the affine transform of the coordinates of q:

$$\begin{pmatrix} x_r \\ y_r \end{pmatrix} = \frac{\delta_p}{\delta_q} G_{\theta_p - \theta_q} \begin{pmatrix} x_c - x_q \\ y_c - y_q \end{pmatrix} + \begin{pmatrix} x_p \\ y_p \end{pmatrix}$$

where $G_{\theta_p - \theta_q}$ is the rotation matrix, (x_c, y_c) is the center of coordinates of the queried symbol, and the width w_r, the height h_r and the orientation θ_r of the region are given by:

$$h_r = h \times \frac{\delta_p}{\delta_q}; \quad w_r = w \times \frac{\delta_p}{\delta_q}; \quad \theta_r = \theta_q - \theta_p$$

where h and w are the height and the width of the queried symbol.

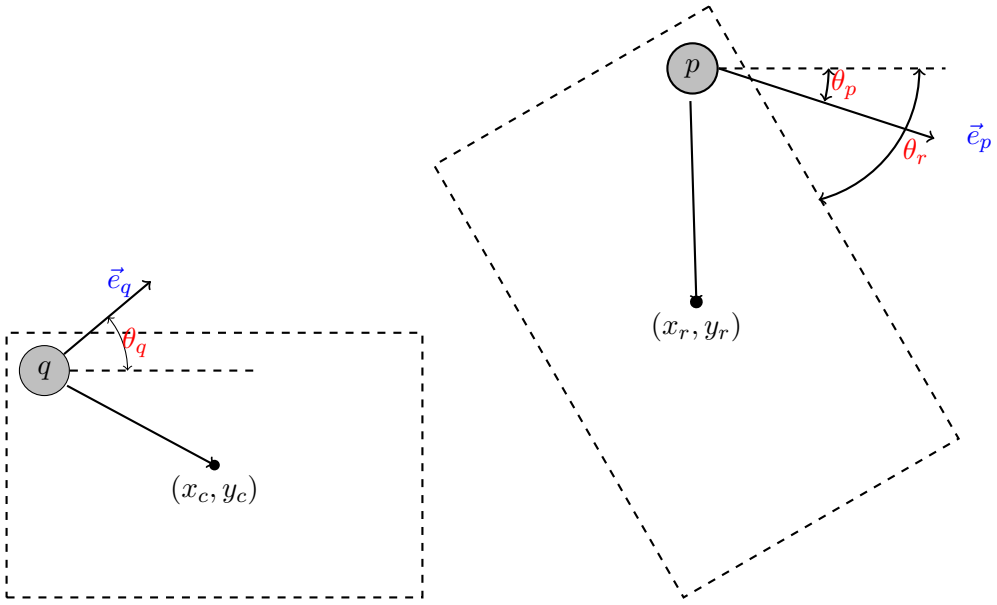


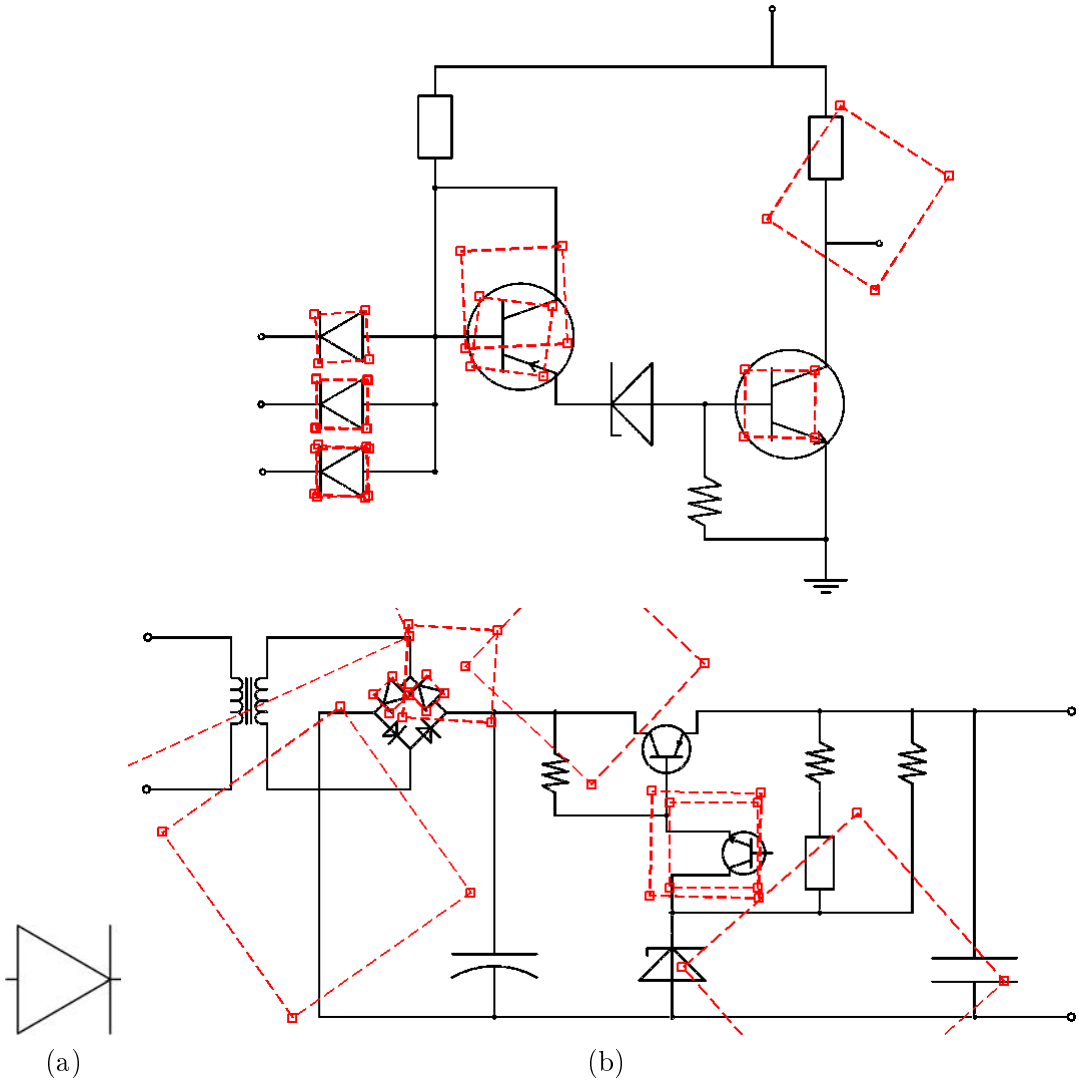
Figure 6.7: Example about how to locate an interest region in the document (right) being corresponding to the request symbol (left)

The equations above define all the possible regions of interest that we can construct from a particular symbol query. To select only those where the queried symbol may appear we introduce the notion of *similarity* property between pairs of interest points s at level c and we select only those regions satisfying this property. We recall that W_p is the set of visual words describing the descriptor h_p computed on p.

Definition 1 (Similarity property). *Two interest points p and q holds the* similarity *property at level $c \in (0, 1]$ if the following inequality is satisfied:*

$$c \times |W_q| \leq |W_p \cap W_q| \leq |W_q|$$

Intuitively, we use the similarity property to compare key-points p and q in terms of the visual words used to describe descriptors h_p and h_q. In fact, the value of $c \in (0, 1)$ controls the overlapping degree of W_p and W_q. Moreover, by setting $c = 1$ we force all the visual words used in the representation of h_q appear in the representation of h_p.



(a) (b)

Figure 6.8: (a): request symbol, (b): corresponding interest regions in the documents

Figure (6.8) presents an example of one request symbol and the interest regions corresponding found in the document with value of $c = 3/4$.

6.5.2 Symbol Refining

The symbol recall step described above permits us to reject regions of document where we can ensure that the queried symbol is not found with high confidence degree. However, using only the *similarity* property as a similarity measure, we will retrieve many false positive instances. Consequently, we need to apply a *refining* step that will permit us to keep the regions of interest where the queried symbol actually is.

Our first choice in this refining step was to apply the vector model framework adapted to sparse representations of descriptors h_p. However, we could observe that we still retrieve many false positive instances. To illustrate this point consider the example of Figure (6.9). There, p is a key-point in the document, and the red region $\mathcal{R}$ corresponds to a region of interest retrieved during the recall step. Moreover, descriptor h_p has been computed on the region defined by the cyan ellipse $\mathcal{E}$ when we had learned the dictionary A. Observe that $\mathcal{E}$ overlaps a document region (orange points) which does not belong to the queried symbol. Using the original descriptor h_p will reduce the discrimination capacities of the considered shape descriptor.

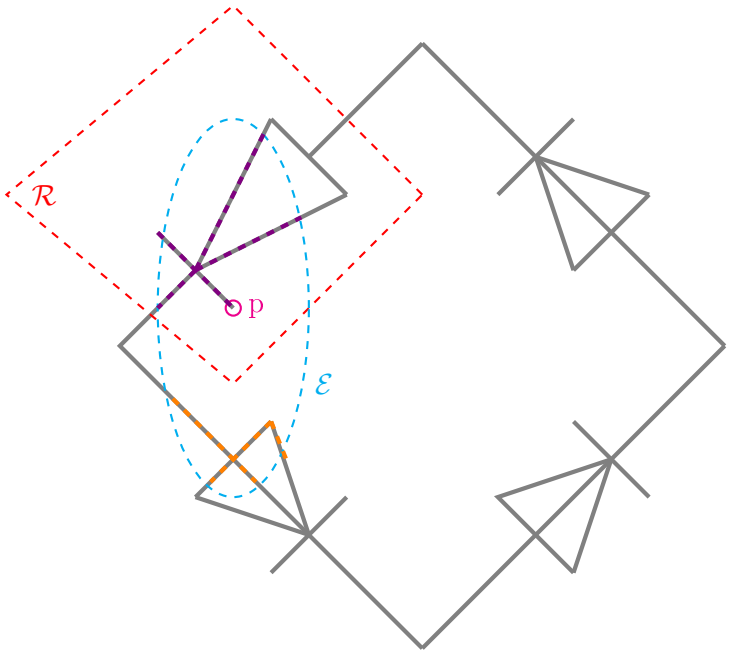


Figure 6.9: Example of situation where contour points do not belong to the interest region.

To overcome this problem, we propose to calculate descriptors h_p only in the regions of interest, namely $h_p^{\mathcal{R}}$, (violet points in Figure 6.9) and build the vector model based on the sparse representation of $h_p^{\mathcal{R}}$ given by A. For all the key-points p in $\mathcal{R}$, we compute descriptors $h_p^{\mathcal{R}}$ and their corresponding optimal sparse representations $\bar{x}_p^{\mathcal{R}}$ (by means of (P_1^{ϵ})). In general, the supports used for computing h_p and $h_p^{\mathcal{R}}$ are different and thus the sparse represention of $h_p^{\mathcal{R}}$ is different to the previously computed for h_p. Let $W_p^{\mathcal{R}}$ and $\Delta_p^{\mathcal{R}}$, respectively, be the visual and the coefficients sets of p constrained at the region $\mathcal{R}$.

The columns of the learned dictionary play the role of words in a visual word framework and the coefficients play the role of the degree of confidence for visual words. With the purpose of keeping information not only on what visual words in the dictionary are used, but also on the coefficients in the sparse representation, we use the optimal sparse representation $\bar{x}_p^{\mathcal{R}} \in \mathbb{R}^K$ of $h_p^{\mathcal{R}}$ as its characteristic vector.

Now, we can compute the tf and idf factors to build the vector model associated to the query symbol q and the region of interest $\mathcal{R}$. On the one hand, we compute the k-th word frequency $tf_k^{\mathcal{R}}$ as:

$$tf_k^{\mathcal{R}} = \frac{\sum_{p\in\mathcal{R}} \bar{x}_p^{\mathcal{R}}(k)}{\sum_{j=1}^{M}\sum_{p\in\mathcal{R}} \bar{x}_p^{\mathcal{R}}(j)}$$

where we take into account the contribution (positive or negative) of the visual word in the representation of $h_p^{\mathcal{R}}$. On the other hand, the *idf* factor indicates the importance of the word k for the discrimination between regions. To compute this value, the number of instances of a word k in the whole document have to be computed. However, we are more interested in computing this value not at document level but at symbol image level. That is, we need to know when a word k is used to describe a key-point belonging to a particular symbol.

Since we are working in a segmentation-free configuration, symbols are embedded into the documents, maybe partially occluded by other graphical entities. Then, it is not possible to compute the *idf* factor from the whole document datasets. To overcome this problem, we compute the *idf* factor from an alternative dataset composed of samples of segmented symbols. Figure (6.10) presents some symbols in the reference database including 1859 segmented symbols.

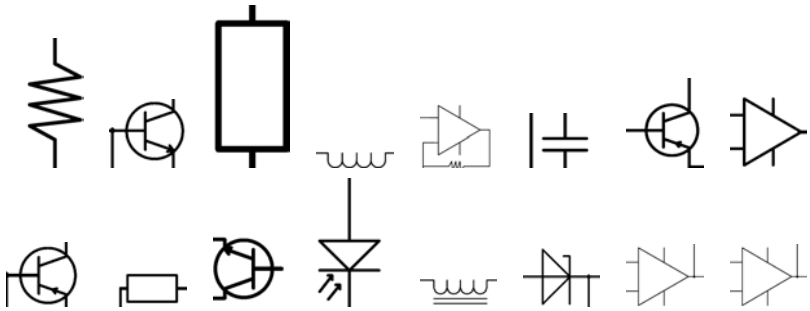


Figure 6.10: Some isolated segmented symbols in the reference database.

More details, *idf* is calculated as following:

1. For each symbol in a reference database:

   (1a). calculate its interest points, contour points and its ESCIPs descriptors H,

   (1b). calculate the sparse representation of all ESCIP $h \in H$ using the learning dictionary A, or solving equation

   $$\bar{x} = \min_x \|x\|_1 \text{ subject to } \|Ax - h\|_2 \leq \epsilon$$

   (1c). let I be the index set where $\bar{x}$ is different to 0, we define vector $v \in \mathbb{R}^M$ as being the 0-1 valued vector: $v(i) = 1$ if $i \in I$ and 0 otherwise,

   (1d). let f_k be the frequency of the word k in this symbol, then $f_k = \sum_{i=1}^{r} v_i(k)$

2. Let l_k be the number of symbols in which the word k appears and N is the number of symbols in reference database, then

   $$l_k = \text{card}\{s = 1 \div N | f_k^s \neq 0\}$$

3. Compute

   $$idf_k = \log(\frac{N}{l_k})$$

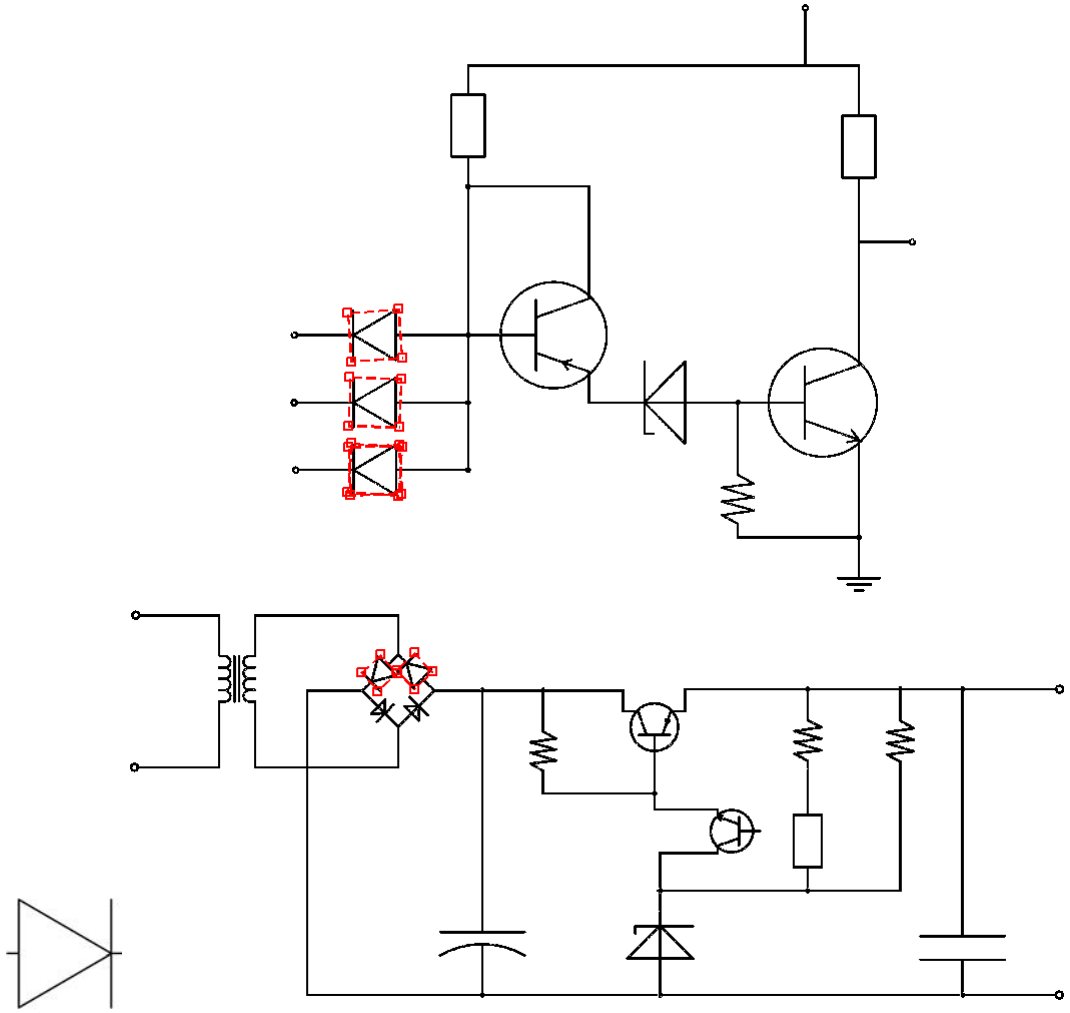
Therefore, the vector model for an interest region $\mathcal{R}$ is defined by the weighted frequency for all words k in visual vocabulary A as following:

$$v_{\mathcal{R},k} = tf_k^{\mathcal{R}} \times idf_k \tag{6.2}$$

The *refining* step is finished by comparing the vector model of the queried symbol, v_q, and the vector model of *recalled* regions, $v_{\mathcal{R}}$ as usual in information retrieval systems using the cosine distance:

$$\text{distance}(v_q, v_{\mathcal{R}}) = \frac{\langle v_q, v_{\mathcal{R}} \rangle}{|v_q| \times |v_{\mathcal{R}}|}$$

Finally, regions with low cosine distance are discarded and the others are ranked in descendent order as being the regions containing the requested symbol. Figure (6.11) shows the candidate regions found after applying refining step on images in Figure (6.8).



(a) (b)

Figure 6.11: (a): request symbol, (b): corresponding candidate regions after refining step.

6.6 Experimental Validation

In this section we present a first experimental evaluation to verify the efficiency of the proposed approach. The first experiment is performed on subset of the SESYD dataset[1] which is a collection of 15 images (of 3331×2126 pixels in average) and 6 different symbols as queries are tested.

Although numerous works have been proposed to deal with the problem of localization symbols in the documents, in the best of our knowledge there is no complete evaluation for all the existing approaches on the same database. Thus, we have compared our approach to [129] using the same symbol queries. In previous Chapter, the sparse representation of SCIP descriptor have proved to perform better than other well-know descriptors like SIFT and Zernike descriptors. The goal of the experiments carried out in this work is to evaluate if the symbol spotting method described in the previous section improve when sparse representations are used.

Both methods, the proposed one and one in [129] are based on a local description of symbols using the SCIP descriptors. Moreover, both methods use inverted files for indexing key-points and construct vector models to compare queried symbols and regions of interest of documents. The main difference between them are two-fold: First of all, in the proposed approach we use sparse representation techniques for building a visual vocabulary, while in [129] a visual vocabulary is obtained after applying the k-means algorithm. Secondly, in our approach, we first define candidate regions of interest based on the number of shared visual words between pairs of key-points and then we apply the refining step to improve the precision rate. Conversely, this recall step before computing the vector model and retrieved regions is not applied in [129].

A slight modification is performed on the SCIP descriptor by changing the SCIP support to an elliptical one (see Section (6.2)). The main idea of this modification is to provide more weight to the region close to the detected key-point in the direction of the key-point orientation and to increase in such way the discrimination capacity of the SCIP descriptor. Thus, if the scale of the key-point is σ, then the value of the semi-minor and the semi-major axes are $\frac{3}{2}\sigma$, 3σ (set by the experience), respectively.

Consequently, we have computed such extension of the SCIP descriptor on the SESYD dataset and compute a visual vocabulary A using the K-SVD algorithm. We have fixed the size of the vocabulary to $M = 512$, the number of iterations to 350 and the approximation error to $\epsilon = 0.4$. Once we have computed the vocabulary A, we have used a database including isolated segmented symbols to compute the *idf* factor needed in Equation (6.2).

To compare both methods we used the widespread precision and recall measures for retrieval tasks. The precision measure is the ratio between the number of relevant retrieved items and the number of retrieved items. On the one hand, precision rate equals to 1 means that all retrieved examples correspond to the queried symbol. That is, there are not false positives samples in the retrieved documents and location. Conversely, a lower precision rate, a higher non-relevant (false positive) items included in the results are. On the other hand, the recall rate is the number of relevant items in the collection. It measures the effectiveness of the system in retrieving the relevant items, and it equals 1 in case all the items considered as retrievals are relevant. Indeed, a low recall rate means that relevant items have been missed.

In Table (6.1) we can find the average of the obtained precision and recall rates after querying and spotting the 6 queries over the whole SESYD datasets. We can observe that precision and recall rates increase in most cases. Table (6.2) presents the computing time that corresponds to each query.

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

Symbol	Retrieval measures			
	Proposed Approach		Nguyen *et al* [129]	
classes	Precision (%)	Recall (%)	Precision (%)	Recall (%)
	100	100	100	100
	100	100	33.33	100
	54.55	52.63	50	42.85
	68.00	89.47	33.33	100
	100	100	100	100
	90	84.38	73.68	73.68
Average	**85.43**	**87.74**	65.06	86.09

Table 6.1: Spotting results for queries in the column 1.

	classes					
images						
image 1	8.6534	11.4800	12.8461	2.6261	21.4594	4.9100
image 2	8.6961	11.3504	12.8563	2.5374	21.3855	4.8034
image 3	8.6377	11.4096	12.8259	2.6336	20.7571	4.8560
image 4	8.7181	11.3959	12.8552	2.6078	20.9517	4.7784
image 5	8.6534	11.4260	12.8515	2.5890	20.7913	4.8589
image 6	8.7157	11.3998	12.8658	2.6142	20.9933	4.8065
image 7	8.6566	11.3706	12.8402	2.6295	20.9692	4.8377
image 8	8.7488	11.4102	12.8510	2.6098	21.0592	4.7998
image 9	8.6646	11.3915	12.8696	2.6343	20.9809	4.8657
image 10	8.7111	11.3695	12.8404	2.5645	20.9928	4.8055
image 11	8.6586	11.4164	12.8500	2.6580	21.0074	4.8675
image 12	8.7442	11.3737	12.8369	2.6196	21.2620	4.8178
image 13	8.6554	11.4137	12.8398	2.6101	20.8022	4.8580
image 14	8.7265	11.3629	12.8730	2.6024	21.0608	4.8028
image 15	8.6317	11.4065	12.8353	2.6136	20.5692	4.8669
Average	**8.6848**	**11.3984**	**12.8558**	**2.610**	**20.9361**	**4.8357**

Table 6.2: The computing time (seconds) that corresponds to each query.

6.7 Conclusion

This chapter presents a new approach for symbol spotting systems that uses sparse representations for coding a visual vocabulary. In addition, this approach also uses learning techniques to adapt such visual vocabulary to the intrinsic properties of the documents datasets. This allows to achieve a sparser representation than using pre-fixed basis instead.

Another contribution done in this chapter is found in the symbol retrieving process. The proposed approach follows a two steps architecture that we have called: *recall* and *refining* steps. The main goal of this architecture is to speed up the retrieving process using sparse representation and indexing techniques and to left more computational expensive matching methods only for those regions in which the queried symbol may appear.

First experiments on a benchmark dataset for a symbol spotting application are promising. In the future, we would like to examine the robustness and scalability of the proposed method on larger datasets.

Chapter 7

Conclusions

This thesis has addressed the problem of how to use the sparse representation to present document images efficiently for processing and pattern recognition tasks. It focuses specifically on sparsity over a learned dictionary obtained by applying learning algorithms on training dataset. The advantages of different learned dictionaries are compared to each other and compared to the pre-defined dictionary have been discussed and summarized in Chapter (2). In general, pre-defined dictionaries are characterized by mathematical models of data. The analytical formulations lead to the existence of structure inside dictionaries. This structure sometimes results in fast implementations or other preferable properties such as orthogonality. However, it cannot be adapted to a new and arbitrary family of images of interest as they are restricted to images of a certain type. While learned dictionary is constructed empirically from training database, so the generating atoms come from the underlying empirical data, rather than from some theoretical model. As the result, learned dictionary can adapt well with the characteristic of the data.

We have presented how learned dictionaries are suitable choice for some applications as denoising, text regions extraction, and symbol recognition and spotting symbol in graphical document. In the following, a short summary of our contributions is presented.

Sparse representation in the pre-defined dictionary has been successfully employed for denoising images under some assumptions about the model of noise found in these images. However, in practical applications, we usually do not know which model of noise on the image, so how sparsity can be used in these applications is one question that should be answered. Our work in Chapter (3) gives the answer for this question. In fact, beside proposing a using learned dictionary built from a training database of noisy patches extracted from noisy images, we also provide a way to define automatically the best value of the tolerance error based on a measure of fidelity between two images. The efficiency of the proposed method has been approved experimentally on three different datasets with different kind of noise raised from synthetic to realistic noise.

From Chapter (4), we know that the advantage of multi-resolution learned dictionaries compared to learned dictionaries is that they overcome the shortcomings about the size of patch. Indeed, if the size of the patch is too large, the computing cost time will increase. If this size is too small, it could not contain enough information for discrimination. In addition, when working with graphic images, text characters are in different sizes included in the same document, therefore, choosing a suitable size of the patch cannot be used efficiently in this case. As the results, in this chapter, we propose to use multi-resolution learned dictionaries and sparse representations for separating text/graphics in graphical document. The main idea used to separate is the comparison of the reconstruction error of the patches in two different kinds of learned dictionary. The results shown that the proposed method has high recall rate of text components, overcomes

the problem of text/graphics touching, and outperforms the existing methods. Moreover, in a post-processing step, text regions are further filtered out using some learned thresholds.

The next Chapter (Chapter (5)), we have found a way to bridge the gap between the literature of sparse representation and the visual vocabulary construction. In fact, a vector model for symbol is constructed based on its sparse representation in the learning dictionary. This dictionary is learned from a training database being the local descriptors of symbols. Since the descriptor is invariant, thus the learned dictionary keeps the characteristics. Obtained results for some datasets as GREC, CVC, and FRESH, have proven that our method is suitable for retrieval task and competitive to related state-of-the-art methods. Indeed, by describing each local descriptor as a linear combination of visual words we have achieved better system performance.

The last contribution in spotting symbol in graphical document is presented in Chapter (6). The symbol retrieving process includes two processes, called off-line and on-line. During the off-line process, we learn a dictionary of visual words providing sparse representations of local descriptors. During the on-line process, symbol retrieval is performed in two steps: the recall and the refining steps, respectively. The main goal of this architecture is to speed up the retrieving process using sparse representation and indexing techniques and to left more computational expensive matching methods only for those regions in which the queried symbol may appear.

In the following, a short-list of open issues, which we are currently investigating, are considered as topics for future research.

1. *Improve the speed of proposed de-noising method*: As presented in Chapter (3), our method works directly on noisy patches of image, so the larger size of noisy image, the bigger size of the set of noisy patches. The computing time is high in general when the image has a large size. In addition, the practical applications usually require a reasonable computing time in procedure of de-noising image. Thus, to increase the speed of the proposed method, two directions can be considered:

   - The first one is using randomly sub-set of patches extracted from noisy documents as the training dataset in building the learned dictionary with K-SVD algorithm.
   - The second one is using an alternative of K-SVD that works well under the condition that the training dataset is very large. One of these choices is LS-DLA algorithm. In this algorithm, a subset of training vectors used in each iteration, and the dictionary is updated by using Equation (2.24).

   However, in both directions, the effect depends directly on how to choose subset vectors. Thus what is the reasonable subset that ensures to create a good enough dictionaries is a necessary questions that we need to answer.

2. *Combination of sparse representations to other descriptions*: In Chapter (5), we have defined a way to bridge the gap between symbol recognition and sparse representation over learned dictionary, and we believe that there are other connections to make with sparsely and other descriptors instead of the shape context of interest point. For instance, descriptors are robust to noise as the curvature scale space (CSS) and the pixel level constraint histogram (PLCH). To address this issue, we are interested in discrete optimization theory, where we hope to find a general formulation of the proposed method in Chapter (5) for other descriptors. Another important direction we consider is to apply our approach to character recognition, where we believe that the proposed method can work well. Our belief bases on the fact that instead of using a training database of text images (as in Chapter

(4)), we can use directly the shape context of interest points on the training database of text characters.

3. *Spotting symbols in the noisy documents*: Other contribution in this thesis is spotting symbols. In the proposed method, an extension of shape context of interest points adapted to the documents is used. However, because of working on interest points, thus our method is not robust enough to noisy documents. In the future, we would like to do deeper investigations that can help us to improve the performance of the proposed method with noisy documents. In fact, we have thought about using the junction point instead of interest point. However, by using junction point, a considerable amount of research work will be required before explicit conclusions can be reached such as how to define supported regions? What is a good descriptor that can present the relationship between the junction point with other points in a supported region? Or how we can match the request symbol to a supported region based on two junction points?

Bibliography

[1] S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. *Pattern Analysis and Machine Intelligence*, 26(11):1475–1490, November 2004.

[2] M. Aharon, M. Elad, and A. Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *Signal processing*, 54(11):4311–4322, November 2006.

[3] J. Almazan, A. Gordo, A. Fornes, and E. Valveny. Efficient exemplar word spotting. In *Proceedings of the 23rd British Machine Vision Conference*, volume 11, pages 1–67, 2012.

[4] S.A. Angadi and M.M. Kodabagi. A texture based methodology for text region extraction from low resolution natural scene images. In *Advance Computing Conference*, pages 121–128, 2010.

[5] H. S. Baird, S. E. Jones, and S. J. Fortune. Image segmentation by shape-directed covers. In *Proceedings of the International Conference on Pattern Recognition*, pages 820–825, Atlantic City, NJ, June 1990.

[6] L. Bar and G. Sapiro. Hierarchical dictionary learning for invariant classification. In *Proceeding of the International Conference on Acoustics Speech and Signal Processing*, pages 3578–3581, March 2010.

[7] E. Barbu, P. Hérroux, S. Adam, and E. Trupin. Using bags of symbols for automatic indexing of graphical document image databases. In *Graphics Recognition. Ten Years Review and Future Perspectives, Lecture Notes on Computer Science*, volume 3926, pages 195–205. 2005.

[8] E. Barney. Modeling image degradations for improving ocr. In *Proceedings of the 16th European Signal Processing Conference (EUSIPCO)*, pages 1–5, 2008.

[9] A. R. Barron, L. Birgé, and P. Massart. Risk bounds for model selection via penalization. *Probability Theory Related Fields*, 113(3):301–415, February 1999.

[10] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *Pattern Analysis and Machine Intelligence*, 24(24):509–522, April 2002.

[11] L. G. Bigorda and D. Karatzas. Multi-script text extraction from natural scenes. In *The 12th International Conference on Document Analysis and Recognition*, pages 467–471, 2013.

[12] T. Blumensath and Mike E. Davies. Gradient pursuits. *Signal Processing*, 56(6):2370–2382, June 2008.

[13] J. Bobin, Y. Moudden, J.-L. Starck, and M. Elad. Morphological diversity and source separation. *Signal Processing*, 13(7):409–412, July 2006.

[14] J. Bobin, J.-L Starck, J. M. Fadili, Y. Moudden, and D. L Donoho. Morphological component analysis: An adaptive thresholding strategy. *Image Processing*, 16(11):2675–2681, November 2007.

[15] J. Bobin, J.-L. Starck, M. J. Fadili, and Y. Moudden. Sparsity and morphological diversity in blind source separation. *Image Processing*, 16(11):2662–2674, November 2007.

[16] A. Buades, B. Coll, and J.M. Morel. A review of image denoising algorithms, with a new one. *Multiscale Modeling and Simulation*, 4(2):490–530, 2005.

[17] S. S. Bukhari, F. Shafait, and T. M. Breuel. Towards generic text-line extraction. In *The 12th International Conference on Document Analysis and Recognition*, 2013.

[18] H. Bunke. Attributed programmed graph grammars and their application to schematic diagram interpretation. *Pattern Analysis and Machine Intelligence*, 4(6):574–582, 1982.

[19] E.J. Candés and D.L. Donoho. Recovering edges in ill-posed inverse problems: optimality of curvelet frames. *Annals of Statistics*, 30(3):784–842, March 2000.

[20] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1998.

[21] T. Cheng, J. Khan, H. Liu, and D. Yun. A symbol recognition system. In *Proceedings of the 7th International Conference on Document Analysis and Recognition*, pages 918–921, October 1993.

[22] S. Choi, S. Cha, and C. Tappert. A survey of binary similarity and distance measures. *Journal of Systemics, Cybernetics and Informatics*, 8(1):43–48, 2010.

[23] C. Chong, P. Raveendran, and R. Mukundan. Translation and scale invariants of legendre moments. *Pattern Recognition*, 37(1):119–129, January 2004.

[24] J.-L. Crowley. *A representation for visual information.* PhD thesis, Carnegie Mellon University, 1981.

[25] J.-L. Crowley and A.-C. Parker. A representation for shape based on peaks and ridges in the difference of low pass transform. *Pattern Analysis and Machine Intelligence*, 6(2):156–169, March 1984.

[26] I. Drori D. L. Donoho, Y. Tsaig and J.-L. Starck. Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit. Technical report, Stanford University, 2006.

[27] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, volume 1, pages 886–893, 2005.

[28] I. Daubechies, R. Devore, M. Fornasier, and C.S Gunturk. Iteratively reweighted least squares minimization for sparse recovery. *Communications on Pure and Applied Mathematics*, 63(1):1–38, October 2009.

[29] E. Davies. *Machine Vision: Theory, Algorithms and Practicalities.* Academic Press, 1990.

[30] J. Davis and M. Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine Learning*, 2006.

[31] K.-K. Delgado, J.F. Murray, B.D. Rao, K. Engan, T.-W. Lee, and T.J. Sejnowski. Dictionary learning algorithms for sparse representation. *Neural Computation*, 15(2):349–396, February 2003.

[32] D. Derrien-Peden. Frame-based system for macro-typographical structure analysis in scienti?c papers. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 311–319, Saint-Malo, France, September 1991.

[33] M. Do and M. Vetterli. The contourlet transform: An efficient directional multiresolution image representation. *Image Processing*, 14(12):2091–2106, December 2005.

[34] M. N. Do and M. Vetterli. Contourlets. *J. Stoeckler and G. V. Welland (Eds.)*, pages 1–27, 2001.

[35] D. Donoho and M. Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via ell1 minimization. *PNAS*, 100(5):2197–2202, March 2003.

[36] D. Donoho and M. Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via ell1 minimization. *PNAS*, 100(5):2197–2202, 2003.

[37] D. L. Donoho, Y. Tsaig, I. Drori, and J. Starck. Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit. *Information Theory*, 58(2):1094–1121, February 2006.

[38] P. Dosch and J. Lladós. Vectorial signatures for symbol discrimination. In *Graphics Recognition: Recent Advances and Perspectives, Lecture Notes on Computer Science*, volume 3088, pages 154–165, 2004.

[39] A. Dutta, J. Lladós, , and U. Pal. A symbol spotting approach in graphical documents by hashing serialized graphs. *Pattern Recognition*, 43(3):752–768, March 2013.

[40] B. Efron, I. Johnstone, T. Hastie, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 2003.

[41] M. Elad. *Sparse and redundant representation: From theory to applications in signal and images processing.* Springer, 2010.

[42] M. Elad and M. Aharon. Image denoising via learned dictionaries and sparse representation. In *Proceedings of the International Conference on Computer Vision and pattern Recognition*, pages 17–22, New-York, June 2006.

[43] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *Image Processing*, 54(12):3736–3745, December 2006.

[44] M. Elad, J.-L. Starck, D. Donoho, and P. Querre. Simultaneous cartoon and texture image inpainting using morphological component analysis (mca). *Applied and Computational Harmonic Analysis*, 19(3):340–358, November 2005.

[45] K. Engan. Method of optimal directions for frame design. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, volume 5, pages 2443–2446, March 1999.

[46] K. Engan, S. O. Aase, and J. H. Husoy. Frame based signal compression using method of optimal directions (mod). In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, volume 4, pages 1–4, June 1999.

[47] K. Engan, K. Skretting, and J. H. Husoy. Family of iterative ls-based dictionary learning algorithm, ils-dla, for sparse signal representation. *Digital Signal Processing*, 17(1):32–49, January 2007.

[48] D. Eppstein. Subgraph isomorphism in planar graphs and related problems. *Graph Algorithms and Applications*, 3(3):1–27, 1999.

[49] B. Epshtein, E. Ofek, and Y. Wexler. Detecting text in natural scenes with stroke width transform. In *The Conference on Computer Vision and Pattern Recognition*, pages 2963–2970, 2010.

[50] R. Eslami and H. Radha. The contourlet transform for image de-noising using cycle spinning. In *Proceedings of Asilomar Conference on Signals, Systems, and Computers*, pages 1982–1986, November 2003.

[51] N. Ezaki, M. Bulacu, and L. Schomaker. Text detection from natural scene images: towards a system for visually impaired persons. In *The 17th International Conference on Pattern Recognition*, volume 2, pages 683–686, 2004.

[52] M. J. Fadili, J.-L. Starck, and F. Murtagh. Inpainting and zooming using sparse representations. *The Computer Journal*, 52(1):64–79, 2009.

[53] F. Fernandes, R. van Spaendonck, and S. Burrus. A new framework for complex wavelet transforms. *Signal Processing*, 51(7):1825–1837, July 2003.

[54] F. Fernandes, M. Wakin, and R. Baraniuk. Non-redundant, linear-phase, semiorthogonal, directional complex wavelets. In *Proceedings of IEEE Conference on Acoustics, Speech and Signal Processing*, pages 953–956, 2004.

[55] M. A. Figueiredo and R. D. Nowak. A bound optimization approach to waveletbased image deconvolution. In *Proceedings of the International Conference on Image Processing*, volume 2, pages 782–785, September 2005.

[56] M. A. Figueiredo, R. D. Nowak, and S. J. Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *Selected Topics in Signal Processing*, 1(4):586–598, December 2007.

[57] M. A. Figueiredo and R.D. Nowak. An em algorithm for wavelet-based image restoration. *Image Processing*, 12(8):906–916, August 2003.

[58] J. L. Fisher. Logical structure descriptions of segmented document images. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 302–310, Saint-Malo, France, September 1991.

[59] L. A. Fletcher and R. Kasturi. A robust algorithm for text string separation from mixed text/graphics images. *Pattern Analysis and Machine Intelligence*, 10:910–918, 1988.

[60] L. A. Fletcher and R. Kasturi. A robust algorithm for text string separation from mixed text/graphics images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):910–918, 1988.

[61] P. Franco, J.-M. Ogier, P. Loonis, and R. Mullot. A new minimum trees-based approach for shape matching with improved time computing : application to graphical symbols recognition. *Graphics Recognition. Achievements, Challenges, and Evolution*, 2010.

[62] G. Garz, A. Fischer, H. Bunke, and R. Ingold. A binarization-free clustering approach to segment curved text lines in historical manuscripts. In *The International Conference on Document Analysis and Recognition*, pages 1322–1326, 2013.

[63] P. E. Gill, W. Murray, and M. H.Wright. *Numerical Linear Algebra and Optimization*. Addison-Wesley, 1991.

[64] I. Gonzalez and B. Rao. Sparse signal reconstruction from limited data using focuss: a re-weighted minimum norm algorithm. *Signal Processing*, 45(3):600–616, March 1997.

[65] B. Haasdonk and D. Keysers. Tangent distance kernels for support vector machines. In *Proceeding of the 16th International Conference on Pattern Recognition,*, volume 2, pages 864–868, 2002.

[66] A. B. Hamza, P. Luque, J. Martinez, and R. Roman. Removing noise and preserving details with relaxed median filters. *Mathematical Imaging and Vision*, 11(2):161–177, 1999.

[67] S.M. Hanif, L. Prevost, and P.A. Negri. A cascade detector for text detection in natural scene images. In *The 19th International Conference on Pattern Recognition*, pages 1–4, 2008.

[68] R. M. Haralick. Documentimage understanding: Geometric and logical layout. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 385–390, Seattle, WA, June 1994.

[69] R. C. Hardie and K. E. Barner. Rank conditioned rank selection filters for signal restoration. *Image Processing*, 3:192–206, 1994.

[70] G. Hennenfent and F. Herrmann. Seismic denoising with nonuniformly sampled curvelets. *Computing in Science and Engineering*, 8(3):16–25, Avril 2006.

[71] T. Hoang, E. Barney, and S. Tabbone. Edge noise removal in bilevel graphical document images using sparse representation. In *Proceedings of the International Conference on Image Processing*, 2011.

[72] T. Hoang and S. Tabbone. Text extraction from graphical document images using sparse representation. In *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, pages 143–150, Boston, USA, June 2010.

[73] J. J. Hull. A database for handwritten text recognition research. *Pattern Analysis and Machine Intelligence*, 16:550–554, 1994.

[74] A. K. Jain. *Fundamentals of digital image processing.* Prentice-Hall, 1989.

[75] A. K. Jain and B. Yu. Document representation and its application to page decomposition. *Pattern Analysis and Machine Intelligence*, 20:294–308, 1998.

[76] A. K. Jain and B. Yu. Document representation and its application to page decomposition. *Pattern Analysis and Machine Intelligence*, 20:294–308, 1998.

[77] A.K. Jain and B. Yu. Automatic text location in images and video frames. *Pattern Recognition*, 31(12):2055–2076, 1998.

[78] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011.

[79] R. Jiang, F. Qi, L. Xu, and G. Wu. Using connected-components' features to detect and segment tex. *Image and Graphics*, 11, 2006.

[80] Z. Jiang, Z. Lin, and L. S. Davis. Learning a discriminative dictionary for sparse coding via label consistent k-svd. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 1697–1704, 2011.

[81] J. Jin, J.-L Starck, D. Donoho, N. Aghanim, and O. Forni. Cosmological non-gaussian signatures detection: Comparison of statistical tests. *EURASIP Journal on Applied Signal Processing*, 15:2470–2485, 2005.

[82] C. Jutten and J. Herault. Blind separation of sources, part 1: an adaptive algorithm based on neuromimetic architecture. *Signal Processing*, 24(1):1–10, 1991.

[83] T. Kadir and M. Brady. Scale, saliency and image description. *Computer Vision*, 45(2):83–105, 2001.

[84] T. Kanungo, R. M. Haralick, and I. T. Phillips. Global and local document degradation models. In *Proceedings of the Second International Conference on Document Analysis and Recognition*, pages 730–734, October 1993.

[85] T. Kanungo, R.M. Haralick, H.S. Baird, W. Stuezle, and D. Madigan. A statistical, non-parametric methodology for document degradation model validation. *Pattern Analysis and Machine Intelligence*, 22(11):1209–1223, June 2000.

[86] A. Khotanzad and Y. Hong. Invariant image recognition by zernike moments. *Pattern Analysis and Machine Intelligence*, 12(5):489–497, 1990.

[87] N. Kingsbury. The dual-tree complex wavelet transform: A new efficient tool for image restoration and enhancement. In *Proceedings of the 9th European Signal Processing Conference*, pages 319–322, 1998.

[88] K. Kise, A. Sato, and M. Iwata. Segmentation of page images using the area voronoi diagram. *Computer Vision and Image Understanding*, 70:370–382, 1998.

[89] J. Kreich, A. Luhn, and G. Maderlechner. An experimental environment for model based document analysis. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 50–58, Saint-Malo, France, September 1991.

[90] D. Labate, W.-Q. Lim, G. Kutyniok, and G. Weiss. Sparse multidimensional representation using shearlets. In *Proceedings SPIE, Wavelets XI*, volume 5914, pages 254–262, September 2005.

[91] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

[92] S. Lesage, R. Gribonval, F. Bimbot, and L. Benaroya. Learning unions of orthonormal bases with thresholded singular value decomposition. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, volume 5, pages 293–296, March 2005.

[93] M. S. Lewicki and T. J. Sejnowski. Learning overcomplete representations. *Neural Computation*, 12(2):337–365, February 2000.

[94] J. P. Lewis. Fast normalized cross-correlation. *Vision Interface*, pages 120–123, 1995.

[95] J. Li, H. Mouchère, and C. Viard-Gaudin. An annotation assistance system using an unsupervised codebook composed of handwritten graphical multi-stroke symbols. *Pattern Recognition Letters*, 35:46–57, 2014.

[96] J. Li, C. Viard-Gaudin, and H. Mouchère. Unsupervised handwritten graphical symbol learning - using minimum description length principle on relational graph. In *The International Conference on Knowledge Discovery and Information Retrieval*, pages 172–178, 2011.

[97] Q. Li, H. Zhang, J. Guo, B. Bhanu, and L. An. Reference-based scheme combined with k-svd for scene image categorization. *Signal Processing Letters*, 20(1):67–70, 2013.

[98] X. Liu and J. Samarabandu. Multiscale edge-based text extraction from complex images. In *The International Conference on Multimedia and Expo*, pages 1721–1724, 2006.

[99] Z. Liu and S. Sarkar. Robust outdoor text detection using text intensity and shape features. In *The 19th International Conference on Pattern Recognition .*, pages 1–4, 2008.

[100] J. Lladós, E. Martí, and J. Villanueva. Symbol recognition by error-tolerant subgraph matching between region adjacency graphs. *Pattern Analysis and Machine Intelligence*, 23(10):1137–1143, October 2001.

[101] D. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the 7th IEEE International Conference on Computer Vision*, pages 1150–1157, 1999.

[102] D. Lowe. Distinctive image features from scale-invariant keypoints. *Computer Vision*, 60(2):91–110, November 2004.

[103] G. Lu and A. Sajjanhar. Region-based shape representation and similarity measure suitable for content-based image retrieval. *Multimedia Systems*, 7(2):165–174, March 1999.

[104] H. Luo and R. Kasturi. Improved directional morphological operations for separation of characters from maps/graphics. *Graphics recognition algorithms and systems*, 1389:35–47, 1998.

[105] J. Mairal. *Sparse coding for machine learning, image processing and computer vision.* PhD thesis, Lécole Normale Supérieure de Cachan, 2010.

[106] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Non-local sparse models for image restoration. In *Proceedings of the 12th International Conference on Computer Vision (ICCV)*, pages 2272–2279, 2009.

[107] J. Mairal, M. Elad, and G. Sapiro. Sparse representation for color image restoration. *Image Processing*, 17(1):53–69, January 2008.

[108] S. Mallat. Geometrical grouplets. *Applied and Computational Harmonic Analysis*, 26(2):161–180, March 2009.

[109] S. Mallat. *A wavelet tour of signal processing: The Sparse Way.* Academic Press, third edition, 2009.

[110] S. G. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *Signal Processing*, 41(12):3397–3415, December 1993.

[111] S. Mao, A. Rosenfeld, and T. Kanungo. Document structure analysis algorithms: a literature survey. In *Proceedings of the Conference on Document Recognition and Retrieval (DRR)*, pages 197–207, 2003.

[112] J. Marial, F. Bach, and J. Ponce. Task-driven dictionary learning. *Pattern Analysis and Machine Intelligence*, 34(4):791–804, April 2012.

[113] J. Marial, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 689–696, 2009.

[114] J. Marial, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Supervised dictionary learning. In *Proceeding of the ProNeural Information Processing Systems*, pages 1033–1040, 2008.

[115] J. Marial, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Supervised dictionary learning. *The Computing Research Repository*, abs/0809.3083, September 2008.

[116] P. Marrgos and R.W. Schafer. Morphological filters, part 2: Their relations to median, order-statistic, and stack filters. *Acoustics, speech, and signal processing*, 35(8):87–134, August 1987.

[117] J. Mas, J. Jorge, G. Sánchez, and J. Lladós. Representing and parsing sketched symbols using adjacency grammars and a grid-directed parser. In *Graphics Recognition. Recent Advances and New Opportunities, Lecture Notes on Computer Science*, volume 5046, pages 169–180, 2008.

[118] B. Messmer and H. Bunke. Automatic learning and recognition of graphical symbols in engineering drawings. In *Graphics Recognition Methods and Applications, Lecture Notes on Computer Science*, volume 1072, pages 123–134, 1996.

[119] K. Mikolajczyk. *Detection of local features invariant to affine transformations.* PhD thesis, Institut National Polytechnique de Grenoble, France, 2002.

[120] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. *Computer Vision*, 60(1):63–86, October 2004.

[121] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, October 2005.

[122] R. Minetto, N. Thome, M. Cord, J. Fabrizio, and B. Marcotegui. Snoopertext: A multiresolution system for text detection in complex detection in complex visual scenes. In *The 17th International Conference on Image Processing*, volume 1, pages 3861–3864, 2010.

[123] F. Mokhtarian, S. Abbasi, and J. Kittler. Robust and efficient shape indexing through curvature scale space. In *Proceedings of the British Machine Vision Conference*, pages 53–62, 1996.

[124] M. C. Motwani, M. C. Gadiya, R. C. Motwani, and Jr F. C. Harris. Survey of image denoising techniques.

[125] S. Müller and G. Rigoll. *Engineering drawing database retrieval using statistical pattern spotting techniques.* 2000.

[126] G. Nagy. Twenty years of documentimage analysis in pami. *Pattern Analysis and Machine Intelligence*, 22:38–62, 2000.

[127] G. Nagy, S. Seth, and M. Viswanathan. A prototype document image analysis system for technical journals. *Computer*, 25:10–22, 1992.

[128] J. Neumann, H. Samet, and A. Soffer. Integration of local and global shape analysis for logo classification. *Pattern Recognition Lettersx*, 23(12):1449–1457, 2002.

[129] T. O Nguyen, S. Tabbone, and A. Boucher. A symbol spotting approach based on the vector model and a visual vocabulary. In *Proceedings of the 10th International Conference on Document Analysis and Recognition*, 2009.

[130] T. O. Nguyen, S. Tabbone, and O. Ramos Terrades. Symbol descriptor based on shape context and vector model of information retrieval. In *Proceedings of the 8th IAPR International Workshop on Document Analysis Systems*, Nara, Japan, 2008.

[131] L. OGorman. The document spectrum for page layout analysis. *Pattern Analysis and Machine Intelligence*, 15:1162–1173, 1993.

[132] B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, June 1996.

[133] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1. *Vision Research*, 37(23):3311–3325, December 1997.

[134] T. Y. Ouyang and R. Davis. A visual approach to sketched symbol recognition. In *Proceedings of the 21st International Conference on Artifical intelligence*, pages 1463–1468, San Francisco, CA, USA, 2009.

[135] W. Pan, T. D. Bui, and C. Y. Suen. Text detection from scene images using sparse representation. In *Proceedings of the 19th International Conference on Pattern Recognition*, pages 1–5, 2008.

[136] Y.F. Pan, Y. Zhu, J. Sun, and S. Naoi. Improving scene text detection by scaleadaptive segmentation and weighted crf verification. In *The International Conference on Document Analysis and Recognition*, pages 759–763, 2011.

[137] Y. Pati, R. Rezaiifar, and P. Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Proceedings of the 27th Annual Asilomar Conference on Signals, Systems, and Computers*, pages 40–44, 1993.

[138] T. Pavlidis and J. Zhou. Page segmentation and classi?cation. *Graphical Models and Image Processing*, 54:484–496, 1992.

[139] E. Le Pennec and S. Mallat. Sparse geometric image representations with bandelets. *Image Processing*, 14(4):423–438, April 2005.

[140] G. Peyré and S. Mallat. A review of bandlet methods for geometrical image representation. *Numerical Algorithms*, 44(3):205–234, April 2007.

[141] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. Self-taught learning: transfer learning from unlabeled data. In *Proceeding of the 24th international conference on Matchine learning*, pages 759–766, 2007.

[142] M. Ranzato, F. Huang, Y. Boureau, and Y. LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2007.

[143] R. Raveaux, J. C. Burie, and Jean-Marc Ogier. A colour text/graphics separation based on a graph representation. In *The 19th International Conference on Pattern Recognition*, pages 1–4, 2008.

[144] S. Roth and M. J. Black. Fields of experts: A framework for learning image priors. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, 2005.

[145] R. Rubinstein, M. Zibulevsky, and M. Elad. Efficient implementation of the k-svd algorithm using batch orthogonal matching pursuit. Technical report, April 2008.

[146] M. Rusinol, D. Aldavert, R. Toledo, and J. Lladós. Browsing heterogeneous document collections by a segmentation-free word spotting method. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 63–67, 2011.

[147] B. Saevarsson, J. Sveinsson, and J. Benediktsson. Speckle reduction of sar images using adaptive curvelet domain. In *Proceedings of the International Conference on Geoscience and Remote Sensing Symposium*, volume 6, pages 4083–4085, Piscataway, 2003.

[148] K. C. Santosh, B. Lamiroy, and L. Wendling. Symbol recognition using spatial relations. *Pattern Recognition Letters*, 33(3):331–341, 2012.

[149] K. C. Santosh, L. Wendling, and B. Lamiroy. Relation bag-of-features for symbol retrieval. In *The 12th International Conference on Document Analysis and Recognition*, pages 768–772, 2013.

[150] K.C. Santosh, B. Lamiroy, and L. Wendling. Symbol recognition using spatial relations. *Pattern Recognition Letters*, 33(3):331–341, February 2012.

[151] C. Schmid, R. Mohr, and C. Bauckhage. Evaluation of interest point detectors. *Computer Vision*, 37(2):151–172, June 2000.

[152] I. Selesnick, N. G. Kingsbury, and R. G. Baraniuk. The dual-tree complex wavelet transform. *Signal Processing*, 22(6):123–151, November 2005.

[153] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *Proceedings of the 9th International Conference on Computer Vision*, volume 2, pages 1470–1477, October 2003.

[154] J. Sivic and A. Zisserman. Video google: a text retrieval approach to object matching in videos. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, volume 2, pages 1470–1477, October 2003.

[155] K. Skretting and K. Engan. Recursive least squares dictionary learning algorithm. *Signal Processing*, 58(4):2121–2130, April 2010.

[156] F. Slimane, S. Kanoun, J. Hennebert, A. M. Alimi, and R. Ingold. A study on font-family and font-size recognition applied to arabic word images at ultra-low resolution. *Pattern Recognition Letters*, 34, 2013.

[157] E. H. Barney Smith. Modeling image degradations for improving ocr. In *The 16th European Signal Processing Conference*, August 2008.

[158] J.-L. Starck, E.J. Candés, and D.L. Donoho. The curvelet transform for image denoising. *Image Processing*, 11(6):670–684, June 2002.

[159] J.-L Starck, M. Elad, and D. Donoho. Redundant multiscale transforms and their application for morphological component separation. *Advances in Imaging and Electron Physics*, 132:287–348, 2004.

[160] J. L Starck, M. Elad, and D. L. Donoho. Image decomposition via the combination of sparse representations and a variational approach. *Image Processing*, 14(10):1570–1582, October 2005.

[161] J.L. Starck, F. Murtagh, and J.M. Fadili. *Sparse Image and Signal Processing: Wavelet, Curvelet, Morphological Diversity*. Cambridge University Press, 2010.

[162] L. Sun and Q. Huo. An improved component-tree based approach to user-intention guided text extraction from natural scene images. In *The 12th International Conference on Document Analysis and Recognition*, 2013.

[163] S. Tabbone, L. Alonso, and D. Ziou. Behavior of the laplacian of gaussian extrema. *Mathematical Imaging and Vision*, 23(1):107–128, July 2005.

[164] S. Tabbone, L. Wendling, and J.-P. Salmon. A new shape descriptor defined on the radon transform. *Computer Vision and Image Understanding*, 102:42–51, 2006.

[165] S. Tabbone and D. Zuwala. An indexing method for graphical documents. In *Proceedings of the 9th International Conference on Document Analysis and Recognition*, pages 789–793, 2007.

[166] Amir Tahmasbi, Fatemeh Saki, and Shahriar B. Shokouhi. Classification of benign and malignant masses based on zernike moments. *Computers in Biology and Medicine*, 41(8):726–735, August 2011.

[167] V. N. Temlyakov. Weak greedy algorithms. *Advances in Computational Mathematics*, 12(2-3):213–227, February 2000.

[168] O. Ramos Terrades, S. Tabbone, and V. E. A review of shape descriptors for document analysis. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 227–231, 2007.

[169] R. Tibshirani. Regression shrinkage and selection via the lasso. *Royal Statistical Society. Series B*, 58(1):267–288, 1994.

[170] R. Tibshirani. Regression shrinkage and selection via the lasso. *Royal. Statist. Soc B*, 58(1):267–288, 1996.

[171] K. Tombre, S. Tabbone, L. Pélissier, B. Lamiroy, and P. Dosch. Text/ graphics separation revisited. In *Proceedings of the 5th International Workshop on Document Analysis Systems*, volume 2432, pages 200–211, 2002.

[172] J.A. Tropp and A.C. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *Information Theory*, 53(12):4655–4666, Decembre 2007.

[173] E. Valveny and P. Dosch. Symbol recognition contest: A synthesis. *Lecture Notes in Computer Science*, 3088:368–386, 2003.

[174] R. van Spaendonck, T. Blu, R. Baraniuk, and M. Vetterli. Orthogonal hilbert transform filter banks and wavelets. In *Proceedings of the Conference on Acoustics, Speech and Signal Processing*, volume 6, pages 505–508, 2003.

[175] V. Velisavljevic, B. Beferull-Lozano, M. Vetterli, and P. Dragotti. Directionlets: Anisotropic multi-directional representation with separable filtering. *Image Processing*, 15(7):1916–1933, July 2006.

[176] F. Wahl, K. Wong, and R. Casey. Block segmentation and text extraction in mixed text/image documents. *Graphical Models and Image Processing*, 20:375–390, 1982.

[177] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *Image Processing*, 13(4):600–612, 2004.

[178] L. Wendling, J. Randek, and P. Matsakis. Selection of suitable set of decision rules using choquet integral. In *Structural, Syntactic, and Statistical Pattern Recognition, Joint IAPR International Workshop*, pages 947–955, 2008.

[179] R. M. Willett and R. D. Nowak. Platelets: A multiscale approach for recovering edges and surfaces in photon-limited medical imaging. *Medical Imaging*, 22(3):332–350, March 2003.

[180] H. Wolfson. On curve matching. *Pattern Analysis and Machine Intelligence*, 12(5):483–489, May 1990.

[181] A. Yamashita, T. Amano, I. Takahashi, and K. Toyokawa. A model based layout understanding method for the document recognition system. In *Proceedings of International Conference on Document Analysis and Recognition*, pages 130–138, Saint-Malo, France, September 1991.

[182] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *Proceeding of the Conference on Computer Vision and Pattern Recognition*, pages 1063–6919, June 2009.

[183] J. Yang, K. Yu, Y. Gong, and T.S. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 1794–1801, 2009.

[184] R. Yang, L. Yin, M. Gabbouj, J. Astola, and Y. Neuvo. Optimal weighted median filters under structural constraints. *Signal Processing*, 43:591–604, 1995.

[185] Q. Ye, J. Jiao, J. Huang, and H. Yu. Text detection and restoration in natural scene images. *Visual Communication and Image Representation*, 18(6):504–513, 2007.

[186] C. Yi and Y. Tian. Text extraction from scene images by character appearance and structure modeling. *Computer Vision and Image Understanding*, 117:182–194, 2013.

[187] C. Yi and Y.L. Tian. Text string detection from natural scenes by structure-based partition and grouping. *Image Processing*, 20(9):2594–2605, 2011.

[188] D. Zhang. Generic fourier descriptor for shape-based image retrieval. In *International Conference on Multimedia and Expo*, volume 1, pages 425–428, 2002.

[189] D. Zhang and G. Lu. Shape-based image retrieval using generic fourier descriptor. *Signal Processing*, 17(10):825–848, November 2002.

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

[191] H. Zhang, K. Zhoa, Y. Song, and J. Guo. Text extraction from natural scene image: A survey. *Neurocomputing*, 122:310–323, 2013.

[192] M. Zhao, S. Li, and J. Kwok. Text detection in images using sparse representation with discrimination dictionaries. *Image and Vision Computing*, 28:1590–1599, 2010.

[193] M. Zhao, S. Li, and J. Kwok. Text detection in images using sparse representation with discriminative dictionaries. *Image and Vision Computing*, 28(12):1590–1599, December 2010.

[194] G. Zhou, Y. Liu, Q. Meng, and Y. Zhang. Detecting multilingual text in natural scene. In *The 1st International Symposium on Access Spaces (ISAS)*, pages 116–120, 2011.

[195] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67:301–320, 2005.

[196] D. Zuwala and S. Tabbone. A method for symbol spotting in graphical documents. In *Proceeding of the 7th International Workshop on Document Analysis Systems*, pages 13–15, February 2006.