وزارة التعليم العالي و البحث العلمي

BADJI MOKHTAR-ANNABA UNIVERSITY
UNIVERSITE BADJI MOKHTAR-ANNABA

جامعة باجي مختار - عنابة

Faculté des sciences de l'ingénieur

Département d'informatique

Année : 2010

MEMOIRE

Présenté en vue de l'obtention du diplôme de **MAGISTER**

Concurrence entre recherche approximative et classification pour la recherche d'images de documents Arabes dégradés

Option

TIC & Ingénierie du document

Par

Abderrahmane KEFALI

DIRECTEUR DE MEMOIRE : Mokhtar SELLAMI Professeur Université de Annaba

DEVANT LE JURY

PRESIDENT :	Halima ABIDAT-BAHI	MC	Université de Annaba
EXAMINATEURS:	Habiba BELLEILI	MC	Université de Annaba
	Hayette MEROUANI	MC	Université de Annaba

ملخص

تتكدس اليوم مجموعات كبيرة من الوثائق و المستندات بمختلف أنواعها في المكتبات و المتاحف بالإضافة إلى مؤسسات أخرى ذات طبيعة بيداغوجية، اجتماعية أو سياسية. الوثائق التاريخية الخاصة بالحضارات القديمة و الأرشيفات الوطنية تعتبر أحسن مثال على هذا الثراء الذي يمثل التراث، التاريخ و شخصية الشعوب. المحافظة إذن على هذه الوثائق و جعلها في متناول أكبر عدد ممكن من الناس تعتبر اليوم حاجة ملحة. تعد رقمنة الوثائق و حفظها بشكل صور واحدة من طرق المحافظة عليها، لكن الرقمنة وحدها ليست كافية لجعل هذه الوثائق في متناول أكبر عدد من المهتمين. في الحقيقة الوصول إلى هذه المجموعات يستلزم استراتيجيات فهرسة و بحث فعالة. في أغلب الأوقات، تتم الفهرسة يدويا، لكن إذا كان هذا ممكنا من أجل عدد قليل من الوثائق، فإن التكلفة و الجهد يصبحان كبيرين جدا في حالة مجموعات الوثائق الكبيرة. يمكن أن تكون المعالجة الآلية للكتابة بديلا في حالة الوثائق المطبوعة أو المكتوبة باليد مع مفردات محدودة، لكن ما أن تكون الوثائق ذات نوعية رديئة و مع مفردات أوسع حتى تصبح المعالجة الآلية للكتابة غير فعالة، خاصة في حالة الوثائق العربية التي تحتوي على صعوبات أخرى مرتبطة أساسا بمعالجة اللغة العربية.

العمل المقدم في هذه المذكرة يدخل في إطار مسعى المحافظة و تقييم الوثائق الموروثة التي أدرك المجمع الدولي أهميتها. اهتمامنا في هذه الدراسة ينصب على الوثائق العربية القديمة النصية لذا قمنا باقتراح نظام يسمح بالبحث عن هذه الوثائق انطلاقا من عرائض نصية، بدون اللجوء إلى التعرف على محتواها من أجل تجنب التكلفة الكبيرة و الجهد المضني للمعالجة الآلية للكتابة.

من أجل بلوغ الأهداف المسطرة، يضم النظام المقترح عدة معالجات منحدرة أساسا من مجالات تحليل الوثائق: تحويل الصورة إلى صورة ذات لونين، تجزئة الصورة، استخراج الخصائص، تشفير...الخ. و من مجال البحث عن المعلومات، خصوصا تقنيات أشجار اللواحق و البحث التقريبي.

كلمات مفاتيح: تحليل الوثائق، البحث عن الوثائق، المعالجة الآلية للكتابة العربية، الوثائق العربية القديمة، البحث التقريبي.

ABSTRACT

Currently, important numerical documentary collections exist in the libraries, museum and other institutions in pedagogic or sociopolitic nature. Historical documents of old civilizations and public archives are the typical example of such richnesses which represent the inheritance, history and dignity of nations. A method of conservation consists to digitize them and save them under image format, but this only digitization is not sufficient to makes it accessible to a grand public. Access to these collections requires efficient indexation and retrieval strategies. Such indices are usually created manually. While this approach may be feasible for small number of documents, the cost will be prohibitive for large collections. The OCR may be a promising alternative in the case of printed documents and handwriting documents with very limited lexicons. For general documents with the usually degraded image quality and with large lexicons, traditional OCR techniques are not adequate, especially for Arabic documents witch contain others difficulties relatives to Arabic writing processing.

The work presented in this manuscript integrates in the attempt of saving and valorization of patrimonial documents of witch the international community waked up of their interest. Despite we interested by old textual Arabic documents, we propose a free recognition system allowing to retrieve these documents with text queries, in order to avoid the high cost and the arduous effort of the OCR.

In order to accomplish the aimed goals, the proposed system regroups several processing coming mainly from the fields of documents analysis: binarization, segmentation, features extraction, coding…etc. and the field of information retrieval, especially suffixes trees and approximate search techniques.

Keywords: documents analysis, documents retrieval, Arabic handwritten recognition, old Arabic documents, approximate search.

RESUME :

D'importants fonds documentaires existent actuellement dans les bibliothèques, musées et autres institutions à caractères pédagogiques ou sociopolitiques. Les documents historiques des civilisations anciennes et les archives nationales sont l'exemple typique de telles richesses qui représentent le patrimoine, l'histoire et la dignité des nations. La conservation de ces documents et leur accès à un grand nombre est constituée aujourd'hui un besoin incontournable. Une méthode de conservation consiste à les numériser et les stocker sous format image, mais seule, la simple numérisation n'est pas suffisante pour les rendre accessibles à grand public. En effet, l'accès à ces collections nécessite des stratégies d'indexation et de recherche efficaces. Dans la plupart du temps, les index sont crées manuellement. Si cette approche est possible pour un petit nombre de documents, son coût et effort deviennent très élevés pour des larges collections. L'OCR peut être une alternative pour les documents imprimés ou les documents manuscrits avec un lexique limité. Dés que les documents soient dégradés et avec un lexique plus large, l'OCR devient inefficace, surtout dans le cas des documents arabes qui présentent d'autres difficultés relatives essentiellement aux traitements de l'écriture arabe.

Le travail présenté dans ce mémoire s'intègre dans la démarche de sauvegarde et de valorisation de documents patrimoniaux dont la communauté internationale a pris conscience de l'intérêt. Bien que nous nous intéressons aux images d'anciens documents arabes textuels, nous proposons un système permettant la recherche de ces documents par des requêtes textuelles, sans recourir à une reconnaissance du contenu afin d'éviter le coût élevé et l'effort ardu de l'OCR.

Afin d'atteindre les objectifs visés, le système proposé regroupe plusieurs traitements issus principalement du domaine de l'analyse de documents : binarisation, segmentation, extraction de caractéristiques, codage,…etc. et du domaine de la recherche d'information, notamment les techniques d'arbres de suffixes et la recherche approximative.

Mots clés : analyse de documents, recherche de documents, reconnaissance de l'écriture arabe, documents arabes anciens, recherche approximative.

REMERCIEMENTS

Au moment où s'achève la rédaction de ce mémoire, je tiens à remercier tous les gens qui ont contribué à le rendre possible.

Après Dieu, je tiens à exprimer toute ma reconnaissance et mon profond respect à mon encadreur Pr. Mokhtar SELLAMI, d'avoir accepté de diriger mon travail, de m'avoir fait confiance et de m'avoir laissée m'exprimer.

Un très très grand merci à Dr. Toufik SARI pour tous ce qu'il a fait pour moi. Les mots vont ni me suffire, ni pouvoir exprimer ma gratitude envers lui, parce qu'il était non seulement mon co-encadreur, mais un fleuve intarissable de valeurs humaines. Je le remercie de m'avoir proposé ce sujet de mémoire et diriger constamment de près mes travaux. Je le remercie pour ses conseils, pour sa disponibilité à tout moment, pour sa gentillesse, pour son aide illimité, pour son soutien moral, et pour la confiance qu'il m'a témoignée. Merci, Monsieur Sari, Merci et Merci.

Je voudrais remercier madame H. BAHI, maitre de conférence à l'université de Annaba de m'avoir fait l'honneur de présider mon jury.

Je veux également exprimer toute ma gratitude à madame BELLEILI et à madame MEROUANI maitres de conférences à l'université de Annaba qui ont eu la grande gentillesse de mettre leur savoir au service de mon travail en acceptant d'en être les rapporteurs.

Merci à tous ceux qui ont contribué de près ou de loin à l'élaboration de ce travail.

Liste des Tableaux

Liste des Figures

Table des matières

Introduction générale

1. Contexte et problématique

Les bibliothèques, les musées et autres institutions à caractères pédagogiques ou sociopolitiques contiennent des collections considérables de documents, le plus souvent manuscrits. Les documents historiques des civilisations anciennes et les archives nationales sont l'exemple typique de telles richesses qui représentent le patrimoine et l'histoire des nations. Les documents manuscrits arabes anciens forment une bonne partie de ce patrimoine (il existe environ 3 millions de manuscrits d'origine arabe et islamique, éparpillés à travers le monde). En effet, ces documents encourent une dégradation progressive parce qu'ils ne sont pas conservés dans de bonnes conditions et ne sont accessibles que par un petit nombre de personnes.

Les collections historiques sont d'intérêt pour beaucoup de personnes, non seulement pour les historiens, les étudiants et les élèves. Par exemple, les biologistes peuvent utiliser les notes de champs manuscrites pour comparer l'état courant d'un écosystème avec les conditions du passé. Les Paleoclimatologistes sont aussi intéressés aux notes des manuscrits historiques, qui contiennent souvent des références aux temps, lesquelles sont des indicateurs de climats au passé.

Les archives ont donc un défi important à relever : comment préserver les documents originaux et rendre accessibles au public ces millions de pages contenant des informations manuscrites, pour lesquelles il n'existe pas encore d'outils de recherche construits par les archivistes ? Depuis quelques années, les bibliothèques ont commencées à numériser les collections des documents historiques qui ont beaucoup d'intérêt pour un grand public. Parmi les projets de numérisation de grande envergure nous pouvons citer le projet DEBORA [BAR 02] ou encore les projets BAMBI [BOZ 97] et METAe [JOU 04].

La numérisation des documents permet une préservation numérique des documents, un partage des images avec un grand nombre de personnes via internet, CDs ou autres supports électroniques, un accès simultané et un feuilletage virtuel. Cependant, cette simple numérisation n'est pas suffisante pour répondre aux besoins des utilisateurs des bibliothèques numériques car les difficultés d'accès sont les mêmes que sur papier ou microfilm. Le contenu est généralement non structuré ce que rend toujours nécessaire de feuilleter (même virtuellement) une énorme quantité de pages avant de retrouver les pages contenant l'information cherchée.

Différentes solutions à ce problème (qui reposent entièrement sur des compagnons humains) sont possibles : une méthode simple pour structurer les collections de documents historiques est de les ordonner chronologiquement. Les annotations de documents avec des principaux thèmes permettent de raffiner la granularité. Un très grand niveau de détail dans l'annotation de contenu peut être accompli par la transcription. Elle permet une recherche de texte complet en utilisant un moteur de recherche textuel traditionnel. Parce que le coût de l'annotation électronique du contenu augmente considérablement avec le niveau désiré de détails et de la taille de la collection annotée, un compromis entre le détail et le coût est habituellement préféré.

Des approches automatiques d'annotation du contenu et de recherche sont donc souhaitables afin de réduire l'énorme coût de transcription humaine. Plusieurs systèmes d'indexation ont été déjà proposés mais la plupart d'entre eux ne sont efficaces que sur des images de documents récents. La reconnaissance automatique de documents historiques manuscrits peut paraître comme un choix évident, mais la reconnaissance de manuscrits a atteint un bon niveau d'exactitude dans deux domaines seulement: la reconnaissance *enligne*, lorsque les traits de stylo du scripteur sont enregistrés en temps-réel, et les applications *hors-ligne* avec des vocabulaires très limités, comme par exemple le traitement de chèques ou le tri automatique de courriers. Pour les documents anciens qui incluent des larges vocabulaires, orthographe inconsistant, en plus de la présence des dégradations produites par la chaîne de numérisation (faible contraste, variation d'éclairage…) ou celles intrinsèques aux documents (humidité, tâches d'encre, effet de transparence, …), les résultats de reconnaissance restent loin d'être satisfaisants. Tous ces problèmes constituent un nouveau défi aux chercheurs dans le domaine de traitement et d'analyse de documents. Il est donc indispensable de définir de nouveaux outils permettant d'accéder aux documents manuscrits anciens sans reconnaissance de leurs contenus.

2. Contribution

Notre travail de Magister s'inscrit dans la démarche de sauvegarde et de valorisation de données patrimoniales dont la communauté internationale a pris conscience de l'intérêt.

Dans ce travail, nous nous intéressons à la recherche des anciens documents arabes qui forment une bonne partie de notre patrimoine sans recourir à une reconnaissance du contenu afin d'éviter le coût élevé et l'effort ardu de la reconnaissance. Nous nous intéressons principalement dans ce projet aux composantes textuelles des documents Arabes. La séparation entre les zones textuelles et graphiques sort donc du cadre de notre travail.

Ce travail décrit les techniques que nous avons développées pour construire un système de recherche d'images de documents arabes anciens. Ce système implémente tous les niveaux du traitement à partir d'une collection non structurée d'images de documents arabes numérisées jusqu'à l'interface utilisateur permettant de retrouver les documents les plus pertinents. Pour ce faire, différents traitements au niveau image de document elle-même: prétraitements, segmentation, extraction de caractéristiques et codage sont développés ainsi que les algorithmes de recherche d'informations: indexation, arbres de suffixes et recherche approximative.

L'analyse automatique d'image de documents ne permet pas en général d'extraire une information ***complète*** et ***précise*** du contenu à cause de plusieurs facteurs. Afin de remédier à ce problème, nous avons développé des techniques de recherche approximative et des arbres de suffixes. Deux avantages ont été ainsi acquis, une recherche fructueuse et plus précise.

3. Organisation du mémoire

Ce mémoire est organisé autour de quatre chapitres, dans ce qui suit nous donnons une brève description de leurs contenus respectifs.

Chapitre 1 : Analyse de documents

Parce que la plupart de techniques qu'on va utiliser dans notre approche sont issues du domaine de l'analyse de documents, nous commençons notre mémoire par la présentation du problème général de traitement et d'analyse d'images de documents sans être limités par un type particulier de documents. Après avoir introduit la nécessité de l'analyse de documents, nous donnons quelques définitions relatives à la notion de document, au document numérique et au cycle de vie d'un document écrit. Ensuite, nous passons à l'analyse de documents tout en citant quelques domaines d'applications. Dans le reste du chapitre, nous décrivons le schéma général d'un système d'analyse de documents tout en détaillant les différentes étapes qui le composent. Pour chaque étape du système nous présentons un état de l'art de quelques techniques proposées dans la littérature.

Chapitre 2 : Appariement et recherche approximative

Dans ce chapitre, nous considérons le problème de la recherche approximative de mots dans un texte. La première section de ce chapitre est consacrée à la présentation des différentes définitions et notations utilisées dans la suite du chapitre. Dans la deuxième section, nous passons à la comparaison de chaînes de caractères, et à la recherche exacte d'un mot dans un texte tout en citant quelques algorithmes des plus connus. La prochaine section s'intéresse à la comparaison approximative de chaînes et les mesures de similarité. Par la suite, nous passons au problème central de ce chapitre, c'est-à-dire la recherche des occurrences approximatives d'un mot dans un texte, en détaillant quelques approches classiques. La dernière section explore les arbres et les automates de suffixes comme des outils efficaces et rapides de recherche de motifs.

Chapitre 3 : Conception

Dans ce chapitre, nous détaillons la méthodologie adoptée pour la conception d'un système de recherche de mots dans des images d'anciens documents arabes dégradés. On commence par l'exposition des caractéristiques des documents arabes anciens afin de montrer la complexité et la difficulté de traitement de ce genre de documents. Par la suite, nous présentons quelques travaux antérieurs dans le domaine de recherche d'images de documents. Nous décrivons, dans le reste du chapitre en détail, les différentes phases intervenantes dans le système, ainsi que les algorithmes choisis.

Chapitre 4 : Expérimentations et résultats

Ce chapitre a comme but d'évaluer les performances du système proposé. Dans un premier temps nous présentons l'environnement expérimental et le contexte des expérimentations. Dans un deuxième temps, nous montrons les deux bases de données utilisées pour évaluer toutes les étapes de notre système. La dernière section présente les différentes expérimentations effectuées, les résultats obtenus ainsi que les discussions de ces résultats.

Chapitre 1 : Analyse de documents

1.1. Introduction

Un très grand nombre de documents de différentes catégories : journaux, formulaires, cartes géographiques, dessins techniques, partitions musicales, documents historiques…etc. existent aujourd'hui dans les musées, les archives nationales, les bibliothèques …etc. La conservation de ces documents et leur accès au grand public devient aujourd'hui un besoin incontournable. De plus la présence de gigantesques quantités de documents qui circulent dans les postes, les banques et les entreprises, nous oblige à développer des outils pour les traiter automatiquement. Une méthode de conservation des documents consiste à les numériser et les sauvegarder ainsi sous format image. Mais seule, la numérisation n'est pas suffisante, elle doit être accompagnée d'outils informatiques permettant un accès rapide et pertinent à l'information y contenue, en profitant des progrès réalisés dans les deux domaines : électronique et informatique. Tout ceci rend indispensable une phase d'analyse de documents, c'est-à-dire le passage de l'image à des connaissances représentant le contenu.

Contrairement à l'être humain qui arrive facilement à comprendre le contenu de n'importe quel document, cette tâche pose jusqu'à aujourd'hui différents problèmes pour l'ordinateur.

L'analyse de documents consiste à chercher dans le traitement d'images, des solutions génériques à des problèmes de type document avec comme but la reconstitution du contenu du document selon une forme définie par l'application en question. Notons que l'analyse de documents fait l'objet d'un grand nombre de travaux de recherche, et les bons résultats obtenus avec certains types de documents fait sortir ce domaine des laboratoires vers des applications réelles dans l'industrie comme le cas d'analyse de chèques, traitement de formulaires et le tri des courriers postaux.

Dans ce chapitre, nous aborderons le problème d'analyse automatiques des images de documents en général sans se limiter à un type particulier de documents, ni d'une application particulière. On essaye à travers ce chapitre de donner un survol des techniques et des méthodes les plus fréquemment utilisées dans l'analyse de documents.

La suite de ce chapitre est organisée de la manière suivante : dans un premier temps nous présentons brièvement quelques définitions de la notion de document, le document numérique, et son cycle de vie. Dans un deuxième temps nous passons à l'analyse de documents comme une discipline intéressante de recherche, et nous citons quelques domaines d'applications. Nous donnons par la suite le schéma général du processus d'analyse du document en détaillant les différentes étapes, et nous terminons par une conclusion.

1.2. La notion de document

1.2.1. Définition

Plusieurs articles scientifiques abordent des problématiques liées aux documents, mais peu d'entre eux tentent de donner des définitions génériques pour ce terme. Par contre plein de définitions existent dans les dictionnaires, les encyclopédies et les répertoires. Nous

présentons dans cette section quelques définitions du mot « document » et nous commençons par une définition étymologique.

D'après le Petit Robert édition 2002 [ROB 02], le terme « document » provient du mot latin « documentum », qui veut dire "ce qui sert à instruire". On remontant dans l'origine, nous trouvons aussi le mot indo-européen « docte » qui veut dire " acquérir ou faire acquérir une connaissance".

Certaines définitions du mot « document » se limitent au document écrit, ou bien le document papier. Par exemple :

- Dans le Petit Larousse [LAR 86] : « Un document est un renseignement écrit ou objet servant de preuve ou d'information : document historique, photographique; (droit) titre qui permet d'identifier des marchandises pendant leur transport ».
- En 1989, le Larousse de poche [LAR 89] définit le document comme « Ecrit servant de preuve ou de titre: objet quelconque servant de preuve ».
- Le Petit Robert édition 1993, définit le document comme étant « un écrit servant de preuve ou de renseignement ».

Une définition plus générale donnée par Hadjar [HAD 06] dans sa thèse de doctorat est la suivante : « Un document est le support physique pour conserver et transmettre de l'information ». Dans cette définition, un document peut avoir plusieurs types (textuel, sonores, vidéo, graphique…etc.) selon le support choisi.

Bachimont [BAC 98] considère que le document est indissociable d'un support matériel. Pour lui, « un document est un objet matériel exprimant un contenu ». L'objet matériel est le support d'inscription où un contenu est exprimé. Le contenu est l'ensemble d'informations, de savoir à exprimer.

Après l'introduction de l'informatique et l'émergence des réseaux, et de l'internet, la définition du terme « document » devient plus générale. Cela est montré par les définitions suivantes :

- Un document est un ensemble de données consignées sur support papier, électronique ou autre, pouvant être utilisées pour consultation, étude ou preuve. [OFF 06]
- Un document est une œuvre fixée à un support matériel au moyen du langage ou d'autres symboles [OFF 06].
- L'ISO[1] définit le document comme « l'ensemble d'un support d'information et des données enregistrées sur celui-ci sous une forme, en général, permanente et lisible par l'homme ou par une machine »

Dans la suite de ce manuscrit, nous ne nous intéresserons qu'aux documents « papier » ou bien « écrits » tout en gardant en mémoire que tels documents peuvent contenir des parties non-textuelles (images, graphiques,…etc.).

[1] Organisation Internationale de Normalisation

1.2.2. Document numérique

Partant des définitions de Bachimont et de Hadjar présentées précédemment, un document est dit numérique (électronique) lorsque le support physique est numérique.

Parmi les définitions du document numérique, nous pouvons citer :

- « Un document électronique est la représentation d'un document sous la forme d'une structure de données informatique entreposable[2] dans la mémoire d'un ordinateur et transmissible d'un ordinateur à un autre, notons également qu'un document électronique nécessite un support de stockage pour exister » [MAR 94]
- « Un document électronique est la représentation d'un document, sous la forme d'une structure de données stockée en mémoire ou sur un support informatique, transmissible entre ordinateurs. Dans un système informatique, cette structure de données est représentée dans un fichier sous forme d'une séquence d'octets. Un document électronique peut avoir plusieurs représentations, d'où la notion de format de fichiers » [HAD 06]

1.2.3. Cycle de vie d'un document écrit

Le cycle de vie d'un document peut être résumé en deux étapes principales [SOU 02]:

- *La production* : commence à partir de la conception jusqu'à l'impression du document.
- *La consommation* : qui englobe toutes les opérations relatives à l'utilisation du document, son stockage, récupération et destruction.

En introduisant une troisième étape de *dématérialisation*, le cycle de vie se transforme en une boucle. Cette étape permet la numérisation du document papier, et le résultat peut être imprimé par la suite ce qui donne à nouveau, un document papier (Figure 1.1).

[2] Qui peut être rangé ou stocké

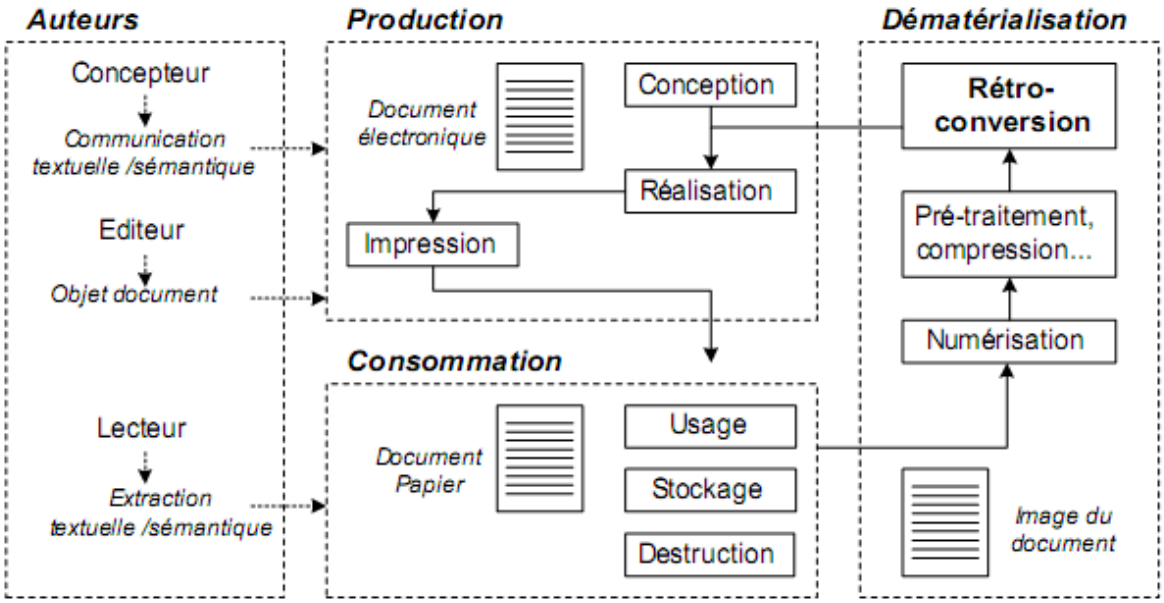


Figure 1.1: Cycle de vie des documents [SOU 02]

1.3. Analyse de documents

L'analyse de documents, ou plus précisément l'analyse d'images de documents est une discipline qui étudie les possibilités algorithmiques de reconstituer une information structurée à partir de la forme visuelle brute (à partir de son image) [BAP 98].

Pendant longtemps, les efforts dans le domaine de l'analyse de documents se sont concentrés sur la problématique de la reconnaissance optique de caractères (OCR), ainsi que sur quelques domaines très spécifiques exploitant l'OCR, comme la lecture de chèques, le tri de courriers postaux, et l'acquisition de formulaires. En effet, le principal intérêt d'un document ne se trouve pas dans sa forme physique, mais plutôt dans son contenu [BEL 01], ce qui fait sortir l'analyse de documents de la simple reconnaissance à des objectifs beaucoup plus larges pour répondre aux besoins actuels, en profitant des progrès réalisés dans des domaines proches, comme le traitement du signal, l'analyse d'images, la reconnaissance des formes, ou l'intelligence artificielle. Les chercheurs se sont attaqués à une multitude de sous- problèmes, explorant autant les fondements théoriques que les approches empiriques. Chaque tentative d'application a mis en évidence de nouvelles difficultés pratiques, et des moyens pour les contourner [BAP 98].

Aujourd'hui, l'analyse de documents possède tous les avantages d'un domaine scientifique intensif et bien organisé. Le savoir-faire accumulé est riche et varié. La technologie se concrétise par des prototypes de recherche opérationnels et des logiciels commerciaux. Chaque année, plusieurs colloques internationaux réunissent industriels et académiciens pour discuter les dernières innovations, citons:

- International Conferences on Document Image Analysis and Recognition (ICDAR);
- International Workshops on Document Analysis Systems (DAS);
- Annual Symposium on Document Analysis and Information Retrieval (SDAIR);

- Conférence Internationale Francophone sur l'Ecrit et le Document (CIFED, précédemment CNED).
- Digital librairies

Malgré les recherches intensives dans ce domaine, les résultats jusque là obtenus sont jugés non satisfaisants. L'analyse de documents reste un problème complexe à cause de plusieurs facteurs. Le premier facteur à mentionner est l'absence d'un objectif universel, simple à formuler, en plus de l'insuffisance des modèles de représentation de connaissances permettant d'orienter l'analyse. En effet, les résultats souhaités dépendent fortement de l'application visée, et des caractéristiques spécifiques à la classe de documents considérée.

L'analyse automatique de documents a plusieurs applications industrielles importantes dans plusieurs domaines, tel que : trier des courriers, traiter des formulaires, mettre au propre un brouillon, rechercher des documents à partir du contenu et non à partir de mots-clés fixés par un opérateur humain…etc.

1.4. Etapes d'analyse de documents

L'absence d'un objectif universel à toutes les applications d'analyse et la diversité de types de documents traités, rendent difficile de s'accorder sur une architecture générale d'un système d'analyse de documents. En pratique, les prototypes opérationnels sont toujours dédiés à une application spécifique.

L'architecture générale commune à la plupart des systèmes d'analyse d'images de documents peut être donnée par le schéma suivant (Figure 1.2) :

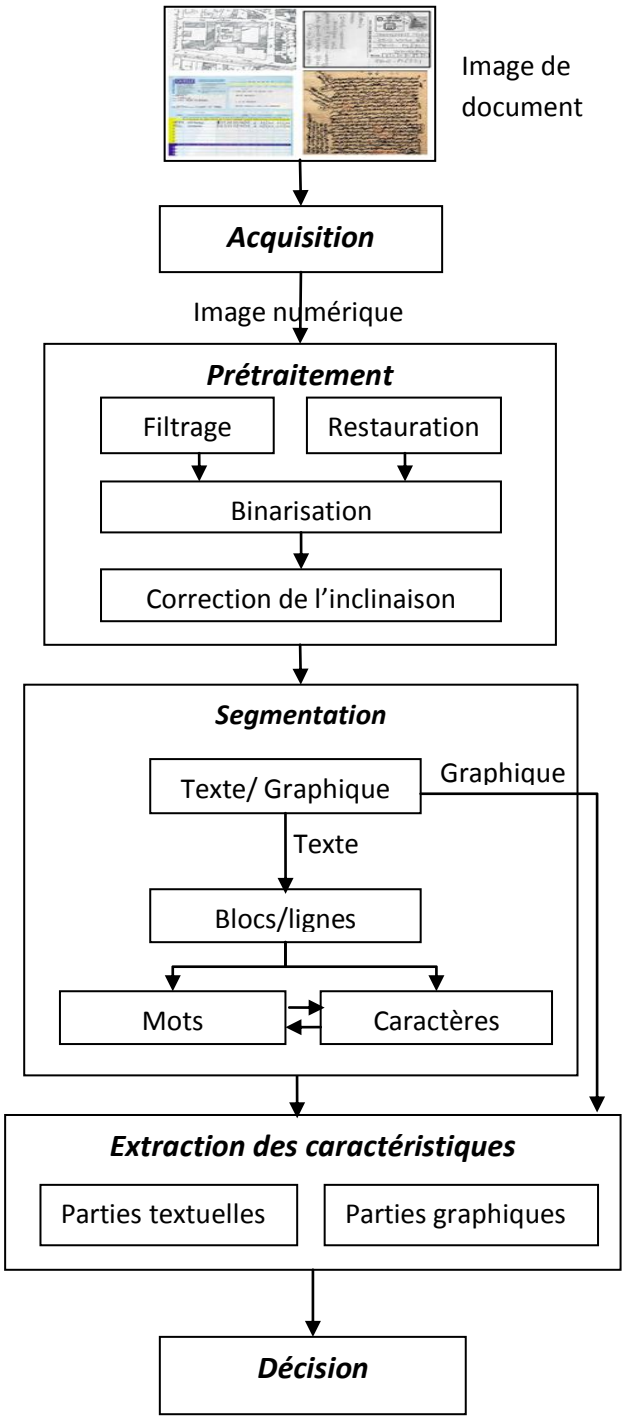


Figure 1.2: Architecture générale d'un système d'analyse de documents

Nous détaillons dans la suite de cette section chaque étape d'analyse de documents, tout en présentant un petit état de l'art de chacune de ces étapes.

1.4.1. Acquisition

La numérisation ou l'acquisition est la première étape dans le processus de traitement et d'analyse de documents. Le Réseau canadien d'information sur le patrimoine (RCIP) définit

la numérisation comme « *le processus par lequel on crée une image numérique (c'est-à-dire contenue dans un ordinateur) à partir d'un objet (document, photographie ou artefact) en trois dimensions* ».

Cette conversion est effectuée à travers un scanner ou une caméra. La numérisation doit être effectuée le plus fidèlement possible, c'est-à-dire sans perte d'informations car les traitements ultérieurs vont être appliqués sur l'image numérique et non pas sur le document original.

1.4.2. Prétraitement

Lorsque l'acquisition est réalisée, la plupart des systèmes de traitement et d'analyse d'images disposent d'une étape de prétraitement. Le prétraitement regroupe un ensemble de techniques de bas niveau ayant comme but d'éliminer le bruit superposé aux données reçues du capteur et de ne garder autant que possible que l'information pertinente, afin de préparer le terrain aux étapes suivantes dans le processus d'analyse. Le résultat de ce prétraitement est une image épurée dépourvue de bruit.

Le bruit peut être dû aux dispositifs d'acquisition, aux conditions d'acquisition (éclairage, mise incorrecte du document sur la vitre …), ou encore à la qualité du document d'origine (effet de transparence, présence des taches d'humidité et des trous …etc.)

Un grand nombre de traitements peuvent être cités dans cette section, mais l'application de ces traitements diffère d'un système à un autre selon le type du document traité, et l'objectif visé de l'application. Nous détaillons dans la suite quelques traitements.

1.4.2.1. Suppression du bruit

Comme nous l'avons dit précédemment, les images reçues du capteur sont souvent bruitées. Une étape de suppression du bruit est alors nécessaire. Le bruit s'insère dans le signal des images de différentes manières. Il peut être aléatoire et cohérent avec ce signal et dans ce cas, il ne peut être supprimé qu'en faisant appel à des connaissances a priori sur l'image. Le bruit peut être aussi périodique et donc se trouve en dehors des informations utiles. Dans ce cas, sa suppression est souvent aisée par des techniques classiques de filtrage et ne cause aucune destruction des informations utiles [BEL 92b].

Le filtrage est l'opération consistant à remplacer la valeur de chaque pixel de l'image par une valeur dépendante de celle des pixels appartenant à son voisinage. Les différents filtres peuvent être classés en 2 grandes familles : les filtres linéaires et les filtres non linéaires.

1.4.2.1.1. Filtres linéaires (par convolution)

Ce sont les filtres les plus simples et les plus faciles à implanter. Grâce à cette simplicité, de nombreux filtres, typiquement de type passe-bas (pour le lissage), ont été proposés dans la littérature et appliqués au filtrage d'image. Le filtrage est effectué en utilisant un masque de convolutions, le niveau de gris du pixel central est remplacé par la convolution du voisinage avec le masque.

Les filtres linéaires sont les plus performants en termes de réduction de bruit dans le cas de bruit à distribution gaussienne. Ceci constitue une première limitation des filtres linéaires car le bruit dans une image naturelle n'est pas toujours gaussien. Dans des nombreuses situations, il est plutôt impulsionnel ou au contraire, très concentré [TAB 98]. Les filtres linéaires utilisés classiquement en traitement d'images peuvent être classés en quatre groupes : Filtres passe bas, Filtres passe haut, Filtres laplaciens, et Filtres de gradient.

1.4.2.1.2. Filtres non linéaires

La famille des filtres non linéaires est très vaste [PIT 90]. Nous trouvons dans cette famille, les filtres morphologiques, les filtres homomorphiques, les filtres basés sur l'approche markovienne, les filtres d'ordre…etc. La classe des filtres d'ordre est couramment rencontrée en réduction de bruit [TAB 98].

Le filtre médian [TUK 71] est un cas particulier du filtrage d'ordre et est considéré comme l'un des plus simples et des plus efficaces opérateurs non linéaires en présence de bruit impulsionnel ou de non stationnarités. Au contraire des filtres passe-bas, les filtres médians tendent à diminuer l'effet du flou, en ne moyennant plus sur le voisinage mais on prend la valeur médiane sur ce voisinage. Le filtre médian permet de supprimer les impulsions parasites, par contre, ses performances se détériorent en présence de bruit à distribution bornée ou concentrée, en supprimant des informations utiles.

1.4.2.2. Restauration

Les systèmes qui traitent des documents dégradés disposent souvent d'un module de restauration qui permet de reproduire une représentation, la plus proche possible, de la qualité de l'image originale avant sa dégradation (figure 1.3).

Figure 1.3: Résultat de la restauration, (a): Image dégradée, (b) : Image restaurée

Les défauts attachés aux documents sont de différents types, et peuvent être classés en deux groupes : dégradations du fond, et dégradation de formes [DRI 07]. Nous présentons dans cette section quelques méthodes de restauration, groupées selon le type de dégradation traité.

a. Restauration du fond

Beaucoup de travaux ont été intéressés à la séparation entre le recto et le verso du document. Nous présentons ici quelques approches.

a.1. Restauration par recalage

Le principe des méthodes utilisant le recalage pour la séparation entre le recto/ verso est de mettre en correspondance les deux faces d'une même page pour en extraire le texte original associé à chaque face. Parmi les méthodes de restauration par recalage, on peut citer la méthode de Sharma [SHA 00], et celle de Dubois et Pathak [DUB 01].

a.2. Restauration par analyse de l'orientation

Certaines méthodes se basent sur l'analyse de l'orientation de l'écriture dans le document pour distinguer le texte du recto de celui du verso. Comme représentant de ce type de méthodes, nous citons la méthode de Wang et al. [WAN 03] dont l'analyse de l'orientation est basée sur la transformée en ondelettes.

a.3. Restauration par analyse des formes

La méthode de Wolf [WOL 06], comme représentant de ce type de méthodes, est basée sur une régularisation de la segmentation par les champs de Markov pour la séparation entre le recto et le verso.

a.4. Restauration par analyse de la couleur

Garain [GAR 06] propose une méthode de séparation entre le recto/verso basée sur la segmentation en régions de couleur, qui sont ensuite agrégées suivant des règles prédéfinies. Une deuxième méthode est celle de Smigiel [SMI 04] qui utilise une carte de Kohonen pour la suppression des tâches en transparence. La méthode de Leydier [LEY 04] est basée sur la sérialisation de l'algorithme des nuées dynamiques (*k-means*) pour séparer le recto du verso.

a.5. Restauration basée sur les EDPs :

Ben Halima et a1. [BEN 06] ont proposé d'utiliser les équations aux dérivées partielles (EDP), notamment les équations de diffusion non linéaire, pour la restauration des anciens documents arabes dégradés.

b. Restauration des formes

b.1. Restauration par modèle

Certaines méthodes sont basées sur la définition des modèles pour la restauration des formes reflétant plus ou moins fidèlement les dégradations. Un exemple d'un modèle est proposé par Baird [BAI 93], qui inclut un certain nombre de paramètres sur l'image. Un autre modèle proposé par Kanungo [KAN 00], simule certains défauts, comme le flou par des opérations de fermeture morphologique. Dans [ZHA 01], le modèle proposé est basé sur les paramètres extraits de l'image dégradée produite. ALLIER [ALL 03] propose l'utilisation du contour actif (snake) pour la restauration des caractères cassés.

b.2. Restauration par prototypage

Cette approche est appliquée généralement sur les caractères d'un document imprimé [DRI 07]. Elle consiste à collectionner toutes les formes similaires d'une page, ensuite on construit

le modèle de la forme idéale d'une lettre à partir de la superposition de toutes les formes dégradées de la même lettre. Une méthode de restauration des formes en utilisant le prototypage, est celle de Hobby et Baird [HOB 97].

b.3. Restauration des formes par morphologie mathématique

L'utilisation de la morphologie mathématique, notamment les opérations de dilatations et d'érosions conditionnelles, pour la restauration des formes est possible pour les petites dégradations qui concernent quelques pixels seulement. Elle permet de remplir les vides et/ou de corriger les ruptures des traits. Whichello et Yan [WHI 96] proposent une méthode qui combine la morphologie avec un suivi de contours afin de rétablir une certaine continuité des contours dégradés. Le suivi de contours fait à l'aide de masques de tailles variables risque de relier les éléments qui ne devraient pas l'être s'ils ont une distance inférieure à la taille du masque. Yu et Yan [YU 01] résolvent ce problème en ajoutant l'information sur la distance entre deux composantes connexes.

c. Restauration géométrique de la courbure

Un autre défaut qui peut apparaître lors de la numérisation des livres volumineux, est l'apparition de la courbure à proximité de la reliure du livre, ce qui réduit la lisibilité du document. Le traitement de ce défaut est alors nécessaire.

c.1. Restauration par un modèle 2D

Les méthodes de cette classe procèdent en deux étapes : calcul de la courbure et traitement géométrique pour la corriger. Zheng [ZHE 01] propose une correction basée sur une analyse des lignes et des mots de l'image. Trinh [TRI 03] combine l'analyse de la courbure des lignes de textes et des bords des pages et obtient un résultat final de meilleure qualité.

c.2. Restauration par un modèle 3D

Les méthodes de restauration, basées sur un modèle 3D, peuvent être réalisées avec une prise de vue soit normale (un appareil photographique), soit particulière (deux appareils prises sous des éclairages différents). Pour une prise de vue normale, les déformations des feuilles sont généralement modélisées par une surface cylindrique aux abords de la reliure. Pour une prise de vue particulière, une mise en correspondance est calculée entre les images pour reconstruire le relief de la page.

1.4.2.3. Binarisation (seuillage)

La binarisation ou bien seuillage est un traitement irréversible qui permet de transformer une image en niveaux de gris ou en couleur, en une image noir et blanc en fonction d'un seuil à définir. Elle a comme but de diminuer la quantité d'informations présentes dans l'image, et de ne garder que les informations pertinentes, ce qui nous permet d'utiliser des méthodes d'analyse simples vis-à-vis des images en niveaux de gris ou en couleurs.

En effet, plusieurs techniques ont été proposées dans la littérature pour la binarisation d'images en niveaux de gris, d'autres sont applicables directement sur des images en couleurs [BAD 06] [SOB 00] [TSA 02]. Nous ne nous intéressons qu'à la binarisation des documents en niveaux de gris, car la plupart des documents en couleurs, peuvent être convertis fidèlement en niveaux de gris.

Selon plusieurs travaux de recherche [ARI 01], les techniques de binarisation d'images en niveaux de gris peuvent être classées en deux catégories : seuillage global, où un seul seuil est utilisé dans toute l'image pour la diviser en deux classes (texte et fond), et seuillage local où les valeurs des seuils sont déterminées localement, pixel par pixel ou bien région par région. D'autres [SAU 00], ajoutent un troisième groupe de méthodes hybrides, ces méthodes combinent des informations globales et locales pour attribuer les pixels à l'une des deux classes.

a. Méthodes globales

a.1. Seuillage global fixe

C'est la technique de binarisation la plus simple [GUP 07], elle consiste à comparer le niveau de gris de chaque pixel x_i de l'image avec un seuil global fixe t (par exemple 127).

a.2. Méthode d'Otsu

La méthode d'Otsu [OTS 79] essaye de trouver le seuil S qui sépare l'histogramme de façon optimale en deux segments (qui maximise la variance inter-segments ou bien qui minimise la variance intra-segment). La limite de cette méthode est qu'elle n'est applicable que lorsque l'image est bimodale (l'histogramme comporte deux pics).

a.3. Méthode ISODATA

Le seuillage par ISODATA [VEL 80] consiste à trouver un seuil en séparant l'histogramme en deux classes itérativement avec la connaissance à priori des valeurs associées à chaque classe. Cette méthode commence par la division de l'intervalle de valeurs non-nulles de l'histogramme en deux parties équidistantes. Ensuite elle prend m_1 et m_2 comme les moyennes arithmétiques de chaque classe. Répéter jusqu'à la convergence, le calcul du seuil T comme l'entier le plus proche de $(m_1+m_2)/2$, et mettre à jour les deux moyennes m_1 et m_2.

a.4. Méthode de Kapur

La méthode de Kapur [KAP 85] est considérée comme une extension de la méthode de Pun [PUN 80]. A la déférence de cette dernière, la méthode de Kapur prend en compte la distribution de probabilité de l'objet P_t et la distribution de probabilité du fond $(1 - P_t)$ dans la détermination de l'entropie de division. Le seuil de binarisation est choisi comme le niveau de gris dont la somme de l'entropie de l'objet et du fond soit maximale.

a.5. Méthode de Cheng et Chen

La méthode de Cheng et Chen [CHE 98] est basée sur le principe d'entropie maximale et la c-partition floue, pour le choix du seuil de partitionnement. Dans cette méthode, le seuil de binarisation est choisi comme le niveau de gris dont la fonction d'appartenance=0.5, et donc c'est le centre de l'intervalle $[a_{opt}, c_{opt}]$, tel que a_{opt} et c_{opt} sont les valeurs de a et c maximisant l'entropie de la division.

a.6. Méthode de Li-Lee

Li et Lee [LI 93] ont proposé une méthode de seuillage d'images où le regroupement des niveaux de gris en deux classes est fondé sur la minimisation de l'entropie croisée. Notons $\mu_1(t)$ et $\mu_2(t)$ les moyennes des deux classes en fonction du niveau de gris t, et h_i

l'histogramme au niveau de gris *i*. Li et Lee ont établi que le seuil optimal *S* est calculé de façon à minimiser $\eta(t)= \eta_1(t)+ \eta_2(t)$ où:

$$\eta_1(t) = \sum_{i=1}^{t-1} i\, h_i \log(\frac{i}{\mu_1(t)}) \; et \; \eta_2(t) = \sum_{i=t}^{256} i\, h_i \log(\frac{i}{\mu_2(t)})$$

b. Méthodes locales

b.1. Méthode de Bernsen

C'est une méthode locale adaptative dont le seuil est calculé pour chaque pixel de l'image [BER 86]. Ainsi pour chaque pixel de coordonnés *(x, y)*, le seuil est donné par :

$T(x,y) = \frac{Z_{bas}+Z_{haut}}{2}$, Tel que Z_{bas} et Z_{haut} sont le niveau de gris le plus bas et le plus haut respectivement, dans une fenêtre carré $r \times r$ centré sur le pixel *(x, y)*.

b.2. Méthode de Niblack

L'algorithme de Niblack [NIB 86] calcule un seuil local à chaque pixel en glissant une fenêtre rectangulaire sur toute l'image. Le seuil *T* est calculé en utilisant la moyenne *m* et l'écart-type σ de tous les pixels dans la fenêtre (voisinage du pixel en question). Ainsi le seuil *T* est donné par : $T = m + k* \sigma$.

Tel que *k* est un paramètre qui prend des valeurs négatives (*k* est fixé - 0.2 par l'auteur).

b.3. Méthode de Sauvola

L'algorithme de Sauvola [SAU 97] est une modification de celui de Niblack pour donner plus de performances dans les documents contenant trop de variation et illumination inégale. Dans la modification de Sauvola, la binarisation est donnée par : $T = m.(1 - k.(1 - \frac{\sigma}{R}))$.

Où *R* est le porté dynamique de l'écart-type σ, et le paramètre *k* prend des valeurs positives dans l'intervalle [0.2, 0.5]. Dans ses tests, Sauvola utilise *R=128* et *k=0.5*

b.4. Méthode de Wolf

Pour s'adresser aux problèmes de l'algorithme de Sauvola (faible contraste, écart de niveaux de gris etc.), Wolf et *al.* [WOL 02] proposent de normaliser le contraste et la moyenne de niveaux de gris de l'image, et de calculer le seuil par :

$$T = (1 - k) * m + k * M + k * \frac{\sigma}{R}(m - M)$$

Tel que *k* est fixé à 0.5, *M* est le niveau de gris minimum de l'image et *R* vaut l'écart-type maximum de niveaux de gris obtenu sur toutes les fenêtres.

b.5. La méthode Nick

L'avantage majeur de cette méthode est qu'elle améliore considérablement la binarisation des images claires et les images qui présentent de faible contraste, en déplaçant vers le bas le seuil de binarisation [KHU 09]. Le calcul du seuil est réalisé comme suit :

$$T = m + k\sqrt{\frac{(\sum p_i^2 - m^2)}{NP}}$$

Tel que : *k* est le facteur de Niblack et varie entre -0.1 à -0.2 selon les besoins de l'application, *m*: le niveau de gris moyen, p_i: niveau de gris du pixel *i* et *NP* est le nombre total de pixels.

1.4.2.4. Correction de l'inclinaison (redressement)

Les traitements ultérieurs dans le processus d'analyse de documents sont très sensibles à l'inclinaison et nécessitent que l'écriture soit correctement orientée. Si ce n'est pas le cas, le système doit disposer d'un module de correction de l'inclinaison. En effet, l'inclinaison peut être introduite à l'image de deux manières possibles: soit lors de la saisie optique par un mauvais positionnement des pages par exemple, soit par le scripteur lui-même lors de l'écriture.

L'angle d'inclinaison est considéré comme l'angle produit entre les lignes de texte de l'image et la direction horizontale. La correction de l'inclinaison consiste à calculer d'abord l'angle d'inclinaison φ, puis de ré-échantillonner l'image en appliquant une rotation d'angle φ :

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos(\varphi) & \sin(\varphi) \\ -\sin(\varphi) & \cos(\varphi) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Plusieurs techniques ont été proposées dans la littérature pour la détection de l'inclinaison des images de documents et sa correction par la suite. Nous présentons dans cette section quelques méthodes de détection de l'inclinaison les plus fréquemment citées dans la littérature.

a. Méthodes basées sur la Transformée de Hough

La transformée de Hough est une technique bien connue pour la détection des formes paramétriques notamment des lignes et des courbes dans une image. Le défaut majeur de cette méthode est sa lenteur de calcul. Les déférents algorithmes qui se basent sur la transformée de Hough essaient de minimiser le temps de calcul de cette dernière.

a.1. Méthode de Sehad et al

Sehad et al [SEH 05] proposent une méthode de détection de l'angle d'inclinaison des documents arabes imprimés en réduisant le nombre de points à soumettre à la transformée de Hough. Pour ce faire, la méthode utilise les centres de liaisons inter-caractères pour la recherche de l'angle d'inclinaison (Figure 1.4). Le choix d'utiliser ces liaisons est justifié par le fait qu'elles reposent sur la même ligne, et possèdent approximativement la même hauteur et la même largeur. La méthode procède en une extraction des composantes connexes, filtrage de ces dernières, détection des liaisons en calculant l'histogramme horizontal et vertical pour chaque composante connexe, filtrage des ces liaisons, et enfin l'application de la transformée de Hough sur les points centraux des liaisons.

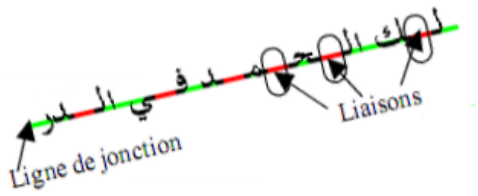


Figure 1.4: Liaisons inter-caractères

a.2. Algorithme d'Amin et Fischer

C'est un algorithme capable de traiter des textes complexes [AMI 00]. Il commence par l'étiquetage des composantes connexes, qui seront regroupées par la suite en blocs. En fonction de leurs tailles, les blocs sont divisés en trois catégories : bruit, petit ou grand. Si un bloc est suffisamment proche d'un groupe de blocs de même catégorie, il y est ajouté. A la fin du regroupement, on calcule la transformée de Hough sur les centres des blocs les plus bas dans le groupe. Les résultats sont ensuite stockés pour chaque bloc dans un histogramme pondéré de nombre de composantes connexes. Il ne reste plus qu'à en choisir le maximum.

a.3. Méthode d'Aradhya et al

Cette méthode [ARA 06] commence par la détection des composantes connexes du document. Ensuite elle élimine les caractères possédant des hampes ou bien des jambages, ainsi que les petites composantes comme les points diacritiques, les marques de ponctuation…etc. A chaque composante connexe sélectionnée, on applique l'algorithme d'amincissement défini dans [AHM 02]. Enfin, le reste de points est soumis à la transformée de Hough pour estimer l'angle d'inclinaison.

a.4. Algorithme de Yu et Jain

Yu et Jain [YU 96] proposent un algorithme robuste, capable de détecter l'angle d'inclinaison des images de documents de différents types, (articles techniques, enveloppes et textes manuscrits). La technique consiste à appliquer la transformée de Hough sur les centres des BAG[3] [YU 92] des composantes connexes afin de diminuer le grand temps de calcul de la transformée de Hough. Le problème de cet algorithme est le calcul des BAG, qui prend beaucoup de temps dans l'algorithme.

b. Méthodes basées sur le profil de projection

Les méthodes basées sur le profil de projection consistent à parcourir l'image selon des directions (des angles) *d* proches de l'horizontale, et à compter le nombre de pixels noirs selon cette direction pour chaque ligne. La qualité de la projection est estimée par son entropie : la direction la plus probable est celle qui maximise l'entropie.

b.1. Méthode de Newman et al.

Newman et al. [NEW 99] ont proposé un algorithme rapide applicable sur des images binarisées basé sur le profil de projection pour l'estimation de l'inclinaison des documents dont l'angle varie entre -10° et 10°. La méthode commence par la définition d'une fenêtre qui doit contenir quelques fragments de lignes de texte. On applique ensuite une projection sur cette fenêtre en calculant la somme de pixels noirs dans chaque ligne. Pour détecter l'angle

[3] Block Adjency Graph

d'inclinaison, on tourne l'image dans un petit intervalle de degrés, et on choisit l'angle qui maximise la variance de profiles.

c. Méthodes basées sur le regroupement des KPPV

c.1. Algorithme d'Avila et Lins

Avila et Lins [AVI 05] proposent un algorithme capable de détecter n'importe quel angle d'inclinaison dans les images de documents. L'algorithme commence par l'étiquetage des composantes connexes. Il associe ensuite à chaque bloc, son voisin le plus proche, puis l'ensemble de blocs dans la même direction que le premier et respectant certaines contraintes (de taille et de distance) sont regroupés. En calculant l'inclinaison de cette série de blocs, on obtient au final l'inclinaison du document.

c.2. Méthode FNNC (Focused Nearest-Neighbor Clustering)

Cette méthode [JIA 99] prend comme entrée, un ensemble de points caractéristiques qui peuvent être les centres des composantes connexes, ou n'importe quelles autres caractéristiques qui indiquent l'orientation dominante du document. A partir de ces points caractéristiques, on calcule l'angle d'inclinaison local en trois étapes :

1) Détermination des plus-proches voisins (3ppv choisi par les auteurs)
2) Calcul de la ligne d'inclinaison locale
3) Groupage de plus proches voisins focalisés (FNNC)

c.3. Méthode d'Okun et al

Okun et al [OKU 99] proposent une méthode d'estimation de l'angle d'inclinaison des images de documents, basée sur les moments d'ordre 1 et 2. Cette méthode est applicable sur des images de faible résolution (50 dpi), ce qui réduit considérablement le temps d'exécution et elle est capable de détecter n'importe quel angle d'inclinaison. Elle commence par la réduction de la résolution de l'image. Par la suite l'extraction des composantes connexes est nécessaire. Selon leurs tailles et leurs formes, on classe les composantes connexes en quatre classes : texte, caractère, ligne et graphique. Et enfin, l'estimation de l'angle d'inclinaison se fera en deux phases.

d. Méthodes basées sur la morphologie

d.1. Méthode de Chen et al

Chen et al. [CHE 95] proposent un algorithme pour l'estimation de l'angle d'inclinaison dans les images de documents basé sur les transformés d'ouverture (RCT) et de fermeture (ROT) morphologiques récursives [HAR 92] [CHE 95b]. L'algorithme proposé est résumé en cinq étapes comme suit :

1. *Sous-échantillonnage* **:** afin de réduire la résolution de l'image d'entrée à 100 dpi.
2. *Remplissage de l'espace inter-caractères* **:** en utilisant la RCT

3. *Elimination des ascendants et des descendants* **:** en utilisant la ROT
4. *Optimisation de la ligne* **:** afin de minimiser la somme des distances carrées entre les points et la ligne optimale.
5. *Estimation de l'inclinaison***.**

1.4.3. Segmentation

Dans la chaîne d'analyse d'images de documents, l'étape de segmentation est l'une des étapes les plus importantes car elle permet d'extraire et de séparer les entités composant le document pour l'étape de décision. La qualité des résultats obtenus à l'issue de cette opération détermine la qualité globale de tout le système d'interprétation.

La segmentation d'un document consiste à trouver sa structure physique en délimitant les différentes entités structurelles homogènes : colonnes, blocs graphiques, lignes de texte, mots, caractères,…etc. Si les documents structurés comme les pages de journaux, les formulaires, les brochures, …etc. possèdent une structuration riche, cette structure se limite dans le cas des documents manuscrits, à une hiérarchie simple : les lignes de texte et parfois les mots qui les composent [LIK 08].

Selon [JOU 06], les états de l'art séparent usuellement les techniques de segmentation de documents en deux familles :

1. ***Les approches mélangeant connaissances a priori et analyse d'images*** : Dans cette famille nous trouvons trois classes :
   - *Approches ascendantes* : les méthodes de cette classe sont guidées par les données et n'incluent donc pas de connaissances sur le modèle. Elles se basent sur l'extraction de données de bas niveaux (couleur, position...) relatives aux pixels. Les pixels sont regroupés en caractères, les caractères en mots, les mots en lignes puis en paragraphes pour former la structure physique du document.
   - *Approches descendantes* : dans ces méthodes, la segmentation est guidée par le modèle du document. Elles sont utilisées souvent pour des documents à structure bien définie.
   - *Approches mixtes* : cette classe regroupe souvent des méthodes qui embarquent en partie des approches descendantes et également des approches ascendantes.
2. ***Les approches textures :*** les outils de caractérisation de textures sont traditionnellement issus de l'analyse d'images naturelles, mais ils se montrent bien adaptés à l'analyse de structure de documents. En effet, les aspects fortement texturés du texte, des photos et des dessins rendent possible la description très fine de fontes, de polices mais aussi des éléments non textuels.

Dans cette section nous nous limitons par la segmentation texte/ graphique, la segmentation du texte en lignes, en mots, pseudo-mots et en caractères.

1.4.3.1. Segmentation texte/ graphique

Plusieurs approches ont été proposées pour la séparation entre le texte et le graphique, qui se basent sur l'utilisation des outils comme les filtres de Gabor ou les ondelettes et en s'appuyant

sur l'étude des fréquences et des orientations des différentes parties de l'image. En effet, il y a une différence importante entre les fréquences caractéristiques d'une zone de texte et de graphique ce qui peut conduire à une bonne séparation entre le texte et le graphique. La principale difficulté de ces approches comme il est signalé dans [LEE 01] est leur complexité algorithmique due à la paramétrisation des filtres utilisés pour rechercher les fréquences et les orientations.

Dans [BOU 06], les auteurs ont proposé une méthode pour la segmentation fond/ texte/ graphique d'images d'ancien manuscrits arabes à structure complexe. Cette méthode opère en deux phases : utilisation de l'analyse multi-échelles pour la segmentation avant/arrière-plan, puis segmentation texte/graphique par l'algorithme de C-moyen flous.

Dans les images binaires, la taille des composantes connexes ou leur aspect constituent un critère pour la séparation entre les composantes textuelles et les composantes graphiques. Khurshid et a1. [KHU 08] par exemple utilisent la taille des composantes connexes comme critère de segmentation texte/ graphique. Ainsi les composantes graphiques vérifient:

Aire composante > (aire moyenne × A) ET Taille composante > (taille moyenne × B) ;

Tel que : aire moyenne est la moyenne des aires de toutes les composantes connexes dans cette image. A et B sont des constantes choisies par expérience égales à 5 et 4 respectivement.

La taille des composantes connexes est utilisée aussi par Waked et al. [WAK 98] comme un critère de séparation. Dans cette méthode les tailles des composantes connexes sont comparées avec un seuil calculé auparavant en partant de la connaissance à priori que les composantes graphiques ont généralement une taille plus grande que les composantes textuelles. Dans [GUS 99], l'élimination des éléments graphiques notamment les lettrines dans les manuscrits médiévaux est basée sur des connaissances a priori sur la position de ces éléments. Des techniques basées sur la morphologie mathématique sont aussi utilisées [GRA 00]. Dans le projet DEBORA[4] [EMP 07], une simple formule permet de déduire si la composante connexe CC_i est un texte ou un graphique. Elle associe à chaque CC_i une probabilité $P(x)$, si cette dernière est inférieure à un certain seuil, alors CC_i est un composant graphique sinon il s'agit d'un texte. [AND 03] propose une segmentation texte/dessin en deux étapes. La première étape concerne l'apprentissage en utilisant un certain nombre d'images dont les zones sont pré-étiquetées (texte ou dessin). Dans la deuxième, un algorithme de segmentation de type XY-CUT[5] est utilisé pour sur-segmenter l'image. Pour chaque zone segmentée, on extrait des informations des composantes connexes. Ces informations sont données comme entrée à un perceptron multicouches. HADJAR [HAD 06] s'intéresse à la séparation texte/graphique appliquée sur des images de journaux arabes. A partir de ces derniers il extrait les plus grandes composantes connexes qui seront filtrées par la suite. Après, il calcule les densités des pixels noirs et blancs dans chaque composante connexe. Si

[4] DEBORA est l'acronyme de Digital AccEss to BOoks of the RenAissance (Accès numérique à des livres de la Renaissance).
[5] Consiste à appliquer récursivement le même algorithme sur une zone, tant qu'une condition n'est pas satisfaite. Le découpage peut, par exemple, être une division en quatre parties égales.

l'une de ces densités est supérieure à un seuil donné alors la composante connexe est un graphique.

1.4.3.2. Segmentation du texte en lignes

Une fois la séparation texte/graphique effectuée, on procède généralement à l'extraction des lignes du texte. L'extraction des lignes du texte est un préalable à tous les processus d'analyse et de reconnaissance de mots ou de caractères. Elle a comme objectif d'assigner chaque composant du texte à une ligne appropriée ; ce qui permet de préparer les données pour les traitements ultérieurs tel que la segmentation en mots et l'extraction des caractéristiques.

Cette opération est relativement facile quand le texte est régulier, non incliné, et ne comporte pas de chevauchements. Ces conditions sont sans doute réunies pour des textes imprimés mais pas souvent pour des textes manuscrits. Ces derniers sont caractérisés par une variation de la distance interligne, la présence de plusieurs lignes de base dans le même document, et les caractères de deux lignes successives peuvent se toucher ou se chevaucher, ce qui complique considérablement l'opération de segmentation en lignes.

Les techniques d'extraction des lignes de texte pour les documents imprimés se divisent en deux catégories : descendante et ascendante. Les méthodes descendantes sont intéressantes lorsque des connaissances sur la structure du document sont disponibles [DEF 95]. Ce sont les méthodes basées sur les profils de projections comme celles de [BAI 92] et [ELL 90].

Notons que la technique de projection horizontale totale présente trois inconvénients majeurs:

- Elle est sensible à l'inclinaison,
- Sensible au chevauchement,
- La présence des points diacritiques produit des faux minima.

Pour surmonter le problème d'inclinaison, Benasri et al. [BEN 99] proposent une méthode d'extraction des lignes de texte manuscrit arabe, basée sur la projection partielle et le suivi de contour partiel. Tout d'abord on segmente l'image en colonnes de largeur fixe. La projection horizontale sur la première colonne nous permet d'extraire les minima locaux qui seront filtrés par la suite pour éliminer les faux minima. Un suivi de contour partiel est appliqué par la suite en considérant les minimas restants comme points de départ. Il ne reste qu'annexer les points diacritiques à l'une des deux lignes la plus proche. L'inconvénient de cette méthode est qu'elle impose que les lignes adjacentes ne doivent pas être collées. De plus la présence de chevauchement dans une colonne peut faire disparaitre le minimum correspondant. Zahour et al. [ZAH 04] ont modifié la méthode précédente afin de résoudre le problème de collage des caractères de lignes voisines. La méthode commence par l'extraction des blocs de texte à partir des histogrammes des projections partielles. Ensuite, l'algorithme *k-means* est utilisé pour classer les blocs en trois classes : les symboles diacritiques, les tracés principaux des mots et les grands blocs résultant du chevauchement et collage des caractères des lignes voisines. L'étape suivante s'intéresse à la segmentation des grands blocs.

Les méthodes ascendantes partent des pixels de l'image et les fusionnent en composantes connexes puis en mots pour former des lignes. Nous trouvons sous cette famille le lissage en

séquence de plages [WON 82], la transformation de Hough [LIK 95], et le regroupement de composantes connexes par proximité [LIK 94] [FEL 01].

Dans [LIK 94], les auteurs ont proposé une méthode itérative d'extraction des lignes adaptée aux documents manuscrits non-contraints. La méthode commence par l'étiquetage des composantes connexes. A partir des points d'ancrage[6], les composantes connexes qui satisfont certains critères de proximité, de similarité et continuité de direction sont regroupées dans un alignement. Likforman-Sulem et al. proposent une autre méthode [LIK 95] basée sur la transformée de Hough. La transformée de Hough est utilisée ici pour la détection des lignes de texte. Un critère perceptif doit être utilisé par la suite pour valider les alignements détectés. Dans [DEF 95], une méthode de segmentation en lignes est proposée permettant de localiser puis d'extraire précisément les lignes de texte d'un document en trois étapes. La première étape se base sur une approche multi-résolution pour localiser les différentes zones de texte à partir de l'image en niveaux de gris. La binarisation est ensuite effectuée localement sur chaque zone d'intérêt, et enfin une étape de post-segmentation est nécessaire.

1.4.3.3. Segmentation en pseudo-mots

Un pseudo-mot est une composante connexe avec ses points diacritiques. Un pseudo-mot peut contenir un ou plusieurs caractères. L'extraction des composantes connexes consiste à regrouper les pixels noirs voisins (dans une image binarisée) en un seul bloc délimité par un rectangle englobant. Les points diacritiques sont ensuite attribués par analyse de proximité à l'une des composantes connexes la plus proche pour former un pseudo-mot (figure 1.5).

Figure 1.5: Attribution des points diacritiques à l'une des composantes la plus proche

1.4.3.4. Segmentation en mots

Dans le cas où l'écriture est cursive, les mots peuvent être extraits facilement à partir des documents (ils correspondent aux composantes connexes). Dans la figure 1.6, les mots apparaissent encadrés par des rectangles englobants.

[6] Ce sont des composantes connexes ayant une direction privilégiée

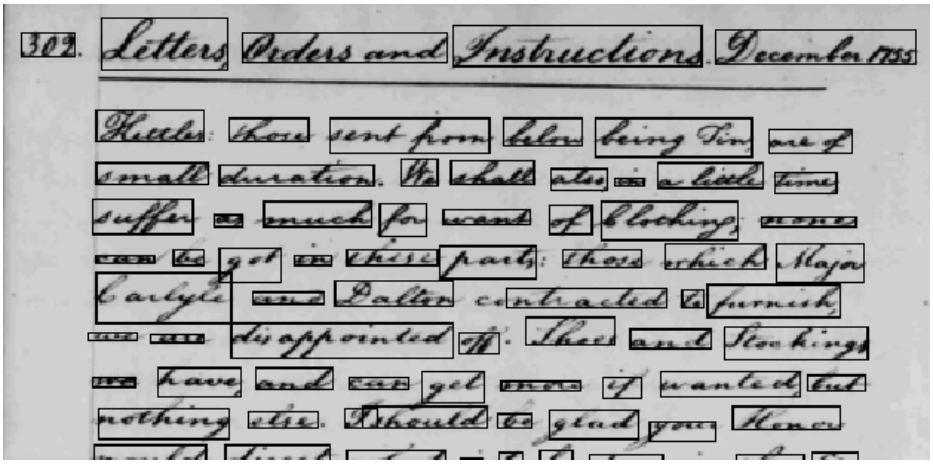


Figure 1.6: Extraction des mots dans un texte cursif

L'écriture latine imprimée est caractérisée par un espace régulier entre les caractères et les mots, ce qui nous permet de reconstruire les mots par agrégation des caractères sur des règles de proximité [SOU 02]. Deux caractères voisins C_L et C_R (respectivement à droite et à gauche) seront agrégés dans un même mot si l'espace qui les sépare n'excède pas un seuil calculé automatiquement.

Pour l'écriture arabe imprimée, les sous-mots peuvent contenir un ou plusieurs caractères (figure 1.7), mais l'espace inter-mots et inter-caractères reste régulier, ce qui permet également la séparation des mots par des mesures de proximité aussi.

Figure 1.7: Extraction des mots d'un texte arabe imprimé, les sous-mots encadrés en bleu, et les mots en rouge

Le regroupement des composantes connexes en mots dans le cas de l'écriture arabe manuscrite, qui est semi-cursive, est plus difficile. En effet, l'espace inter-mots et inter-caractères irrégulier rend difficile l'estimation d'un seuil pour la séparation des mots.

1.4.3.5. Segmentation du mot en caractères

Quelques applications comme L'OCR nécessitent une segmentation des mots du texte en caractères pour faciliter leur analyse. En effet, cette séparation n'est pas toujours possible : les limites entre les caractères sont parfois difficiles à déterminer même pour un être humain. Cette difficulté a été clairement évoquée par Sayre en 1973 et peut être résumée par le dilemme suivant : « pour reconnaître les lettres, il faut segmenter le tracé et pour segmenter le tracé, il faut reconnaître les lettres ». Nous pouvons citer deux approches de segmentation des mots en caractères:

1. *Segmentation implicite* : les approches à segmentation implicite considèrent tous les points du tracé comme points de segmentation potentiels à l'aide d'une fenêtre glissante successivement décalée de 1 à quelques pixels [SEN 98] [MOR 06].

2. *Segmentation explicite*: les approches de segmentation explicite ou discrète effectuent une sélection des points de segmentation les plus probables par une analyse des composantes. Plusieurs méthodes peuvent être classées dans cette catégorie : méthodes utilisant l'espace blanc et l'hauteur [HEN 68], analyse de projections verticales [TSU 92], analyse de composantes connexes [WAN 94], et points de repère [HAR 72].

Plusieurs travaux de recherche ont étudié le problème de segmentation des mots latins imprimés ou manuscrits. Pour l'écriture arabe imprimée, nous pouvons citer le travail de Amin et al. Amin et al [AMI 86] segmentent les pseudo-mots arabes imprimés en caractères en utilisant les projections verticales. Dans l'histogramme des projections verticales, les points de connexion entre les caractères ont une somme inférieure à un certain seuil. En effet l'application de cette méthode telle quelle peut conduire au découpage des caractères en plusieurs parties (comme le cas de la lettre sin س), c'est pour ça, plusieurs chercheurs essayent de régler ce problème en appliquant un ensemble d'heuristiques. Amin et al. [AMI 89] par exemple affirment en examinant les caractères arabes que la distance entre deux pics successifs dans l'histogramme des projections n'excède pas un tiers de la largeur du caractère. Olivier et al. [OLI 96] ont proposé de segmenter les mots arabes en graphèmes[7]. L'algorithme proposé commence par extraire le contour supérieur du mot. A partir de ce dernier on extrait les minima locaux qui seront considérés comme points de segmentation primaires (PSP). Ensuite, ces points sont retenus comme points de segmentation décisifs (PSD) s'ils respectent certaines règles observées sur l'image originale du mot.

Dans le cas de l'écriture latine imprimée, les caractères sont isolés, et donc l'étape de segmentation des mots en caractères n'est pas nécessaire.

1.4.4. Extraction de caractéristiques (primitives)

Après avoir séparé les parties graphiques des parties textuelles, et la segmentation du texte en lignes, mots puis en caractères, vient le dernier processus de réduction des informations avant l'étape de décision qui est l'*extraction de caractéristiques* ou *de primitives*.

Devijver et al. [DEV 82] ont défini l'extraction de caractéristiques comme le problème *d'extraction à partir de l'image de l'information la plus pertinente pour un problème de classification donné, c'est à dire celle qui minimise la variabilité intra-classe et qui maximise la variabilité inter-classe*. Cette information pertinente prend souvent la forme d'un vecteur de valeurs numériques. Cependant pour un problème donné, les caractéristiques extraites doivent satisfaire quelques conditions : le pouvoir discriminant, la résistance au bruit, la résistance aux déformations et la compacité [PIN 01].

La difficulté de cette étape provient du fait que la qualité d'une représentation ne peut se juger que sur un problème particulier (reconnaissance de chiffres, lettres, symboles, etc.) [CHA 06].

Selon l'application visée, l'extraction de caractéristiques à partir des images de documents peut s'intéresser seulement à la partie textuelle du document, notamment pour des besoins de

[7] Les graphèmes sont considérés comme des unités graphiques stables pour un même scripteur associées à un mouvement de plume continu sans changement d'orientation ni reprise ni levé de plume.

reconnaissance de caractères, recherche…etc. comme elle peut s'intéresser aussi au codage des parties graphiques en utilisant habituellement des caractéristiques visuelles de l'image. Dans la suite de cette section nous présentons quelques caractéristiques les plus fréquemment utilisées pour le codage du texte et du graphique.

1.4.4.1. Parties textuelles

Pour la partie textuelle du document, les caractéristiques sont extraites soit sur le mot entier (approche globale), soit sur chaque caractère du mot (approche analytique) et nécessite donc une segmentation préalable du mot en caractères. Les approches globales, présentent l'inconvénient qu'elles subissent la variabilité des mots. Elles sont, de plus, peu discriminantes pour des mots différents dont la forme est proche, ce qui les limite à des applications à lexique réduit (cas des montants de chèques par exemple).

Dans le cas de caractères, la littérature propose un grand nombre de caractéristiques à extraire. Nous présentons dans la suite quelques grandes familles de caractéristiques les plus utilisées [CHA 06] [PIN 01]. Pour plus d'informations, nous vous invitons à voir [TRI 96] qui présente une bonne synthèse des caractéristiques les plus adaptées à la discrimination des caractères manuscrits.

a. Valeurs ou densités de pixels

Les valeurs de pixels sont les caractéristiques les plus simples pour représenter les caractères. L'utilisation de ces caractéristiques a l'avantage qu'elle ne nécessite aucun traitement. Notons que ces caractéristiques sont utilisées généralement dans les méthodes d'appariement de formes (template matching). Les densités de pixels de l'image sont également utilisées comme des caractéristiques simples. Cette technique consiste à découper l'image selon une grille $n \times m$ et à calculer la densité de pixels dans chaque case de la grille.

b. Histogrammes des projections

Les histogrammes des projections des images de caractères peuvent être utilisés aussi comme des caractéristiques de représentation des caractères. Ces caractéristiques peuvent être directement les valeurs des histogrammes, ou bien extraites de ces histogrammes comme les pics par exemple.

c. Caractéristiques de contour

Le contour du caractère est défini par la séquence des pixels entourant le tracé. Ce contour contient toute l'information présente dans l'image de caractère, il semble donc naturel que les caractéristiques extraites des contours soient très utilisées pour la représentation des caractères [PAL 01]. Les caractéristiques peuvent être extraites à partir du codage de Freeman ou bien en déduisant des composantes particulières (les points de rebroussement par exemple), ou encore en utilisant le contour sous forme de représentation codée (par des moments de Fourier, ou de Zernike).

d. Caractéristiques structurelles

Les caractéristiques structurelles sont extraites localement afin de capturer la structure ou la forme du caractère. Parmi ces structures nous trouvons : les boucles, les points de rebroussement, les directions principales du tracé, le nombre et la position des occlusions, des

double et triple jonctions, des extrema, d'intersections avec des sondes horizontales ou verticales...etc. Ces caractéristiques sont considérées comme des caractéristiques pertinentes pour la discrimination des caractères manuscrits. La difficulté principale de ces approches est de rendre ces caractéristiques insensibles aux différentes variations que doivent subir les caractères.

e. Caractéristiques métriques

Cette classe regroupe les caractéristiques basées sur des mesures physiques de l'image. Nous pouvons trouver dans cette catégorie les caractéristiques les plus simples, comme la hauteur, la largeur et le rapport entre ces deux grandeurs. On peut utiliser aussi des caractéristiques plus complexes comme le codage des profils. Les quatre profils (haut, bas, droite, et gauche) sont obtenus en appliquant des sondes sur le caractère à partir des bords de l'image jusqu'à rencontrer le premier pixel noir.

f. Caractéristiques morphologiques

Ces caractéristiques décrivent le caractère en termes de composantes blanches et noires, de cavités (parties blanches partiellement entourées de noir) et de boucles (parties blanches entièrement entourées de noir). L'extraction de ce type de caractéristiques s'appuie sur une étude des positions relatives des différentes composantes noires et blanches de l'image.

g. Moments invariants

Introduits la première fois par Hu [HU 61], les moments invariants sont considérés comme des caractéristiques intéressantes car ils sont invariants en translation, taille et rotation. Ce sont des mesures statistiques de la distribution des pixels autour du centre de gravité du caractère. Les moments invariants les plus utilisés actuellement sont ceux dérivés des polynômes de Zernik [GRA 03] parce que ces derniers ont des performances supérieures en termes d'invariance. Les moments de Hu ont été utilisés pour la reconnaissance de caractères arabes manuscrits dans [ELD 90], [ELL 90] et [AZI 02].

h. Descripteurs de Fourier elliptiques

La transformée de Fourier est l'une des méthodes les plus utilisées en reconnaissance de formes et de caractères [MAH 94]. Les caractéristiques extraites sont en fait les descripteurs de Fourier basés sur les coefficients complexes des séries de Fourier. Ces caractéristiques sont caractérisées par la non-variation aux rotations et aux changements d'échelle. La non-variation aux rotations peut diminuer la discrimination de ces caractéristiques, comme dans le cas de codage des deux chiffres "6" et "9". Il faut donc rajouter d'autres caractéristiques au vecteur de manière à régler ce problème.

i- Caractéristiques adaptatives

Les caractéristiques adaptatives sont obtenues directement à partir de l'image et nécessitent une étape d'apprentissage. Autrement dit, le système opère sur une représentation proche de l'image d'origine et doit lui-même construire et optimiser l'extracteur de caractéristiques.

1.4.4.2. Parties graphiques

Dans certaines applications qui s'intéressent à l'analyse des parties graphiques, comme les systèmes de recherche d'images, les caractéristiques utilisées pour la discrimination des

parties graphiques sont généralement des descripteurs visuels. Ce type de caractéristiques est choisi dans le but de s'approcher le plus possible de la compréhension qu'a l'être humain de l'image. Les caractéristiques visuelles les plus utilisées sont: la couleur, la texture et la forme.

a. Couleurs

La couleur est la propriété visuelle la plus étudiée des images. Elle correspond à un des éléments fondamentaux de la perception visuelle humaine et ne nécessite aucune transposition intellectuelle pour être interprétée. La propriété de couleur est en général définie par un triplet numérique permettant de coder l'intensité de chacune des composantes du système de couleurs. Il existe plusieurs espaces colorimétriques dont chacun possède ses propres caractéristiques.

L'espace le plus connu et le plus couramment utilisé pour le codage est le système RGB (Rouge, Vert, Bleu). Il maintient la compatibilité entre les unités qui génèrent ou affichent les images mais présente l'inconvénient de ne pas être uniforme et nécessite la manipulation de plusieurs composantes pour exprimer une seule couleur [TOL 06]. C'est pour ces raisons que d'autres systèmes préfèrent donc de considérer les couleurs dans un espace basé sur des expériences visuelles d'appréciation de la couleur. On cite par exemple l'espace HSV (Teinte, Saturation, Valeur) aussi connu sous le nom de *système de cône hexagonale*, où la teinte décrit la couleur dominante perçue (rouge, vert, …), la saturation décrit l'intensité de la couleur et la valeur décrit la luminosité de la couleur (sombre, clair, etc.). Notons que cet espace est plus intuitif à utiliser car il correspond à la façon dont nous percevons les couleurs.

b. Texture

La texture est étudiée depuis une vingtaine d'année, et malgré ça il n'existe pas de définition pertinente pour elle. Une définition possible donnée dans [TOL 06] est la suivante : *la texture est la répétition d'éléments de base construits à partir de pixels qui respectent un certain ordre*. Le sable, l'eau, l'herbe, la peau sont autant d'exemples de textures.

Selon [TOL 06] on peut distinguer deux types de textures :

- Les textures régulières, dans lesquelles la périodicité du motif est évidente : grilles, murs, tissus, alvéoles... Elles peuvent être décrites par des approches fréquentielles (spectres de Fourrier, ondelettes de Gabor) ou des approches structurelles dans lesquelles on associe un motif et des règles de placement sur un pavage régulier.
- Les textures aléatoires pour lesquelles la distribution des intensités n'est l'objet d'aucune régularité apparente : sable, nuage, herbe, foule... Elles peuvent être décrites par des lois statistiques sur les distributions.

Pour l'analyse des textures, nous pouvons utiliser les outils classiques des statistiques [TOL 06] comme par exemple la moyenne, la variance, le moment, l'énergie, l'entropie...etc. Le moment d'ordre 3 par exemple donne une indication sur la symétrie; le moment d'ordre 4 donne une indication sur l'aplatissement. Les matrices de cooccurrences sont également des outils adaptés pour l'analyse des textures.

L'avantage majeur des textures est qu'elles permettent de différencier des parties d'images dont les descripteurs de couleurs sont identiques. Par exemple, le ciel (texture unie ou nuageuse) et la mer (texture des vagues).

c. Forme

C'est l'un des descripteurs les plus importants dans les images. Elle désigne l'aspect général d'un objet, son contour…etc. Afin d'extraire les descripteurs d'une forme quelconque, nous devons définir sa fonction caractéristique. Cette fonction est représentée généralement sous la forme d'un masque dans lequel chaque pixel est représenté par le numéro de la région à laquelle il appartient. Les descripteurs de la forme vont être calculés en utilisant cette fonction soit à partir de la région entière, soit à partir des contours seulement.

Les descripteurs les plus classiques sont : [TOL 06]

- Les moments d'inertie : ayant comme avantage d'être invariants par rotation. Ils décrivent bien l'allongement de formes régulières comme des ellipses ou des distributions gaussiennes. Ils sont plus ambigus sur des formes complexes;
- La recherche des boîtes englobantes ou minimales de la région;
- Les moments invariants (nommés moments de Hilbert), invariants par translation, rotation et changement d'échelle;
- Les polygones de Cuzman. Cette approche consiste à envelopper l'objet à reconstruire dans des boites de formes de plus en plus précisément adaptées;
- Descripteurs de Fourrier;
- …

1.4.5. Décision

C'est la dernière étape dans le processus d'analyse d'images de documents. La décision constitue l'objectif des applications. Dans cette étape, on exploite les résultats des phases précédentes, notamment la phase d'extraction de caractéristiques, pour prendre une décision. Selon l'objectif du système de traitement et d'analyse de documents, l'étape de décision peut couvrir plusieurs applications : transcription, recherche, indexation, tri, classification, reconnaissance…etc.

Dans la suite de cette section, nous présentons quelques applications les plus courantes de l'étape de décision.

1.4.5.1. Reconnaissance de l'écriture

La reconnaissance de l'écriture est l'une des applications les plus populaires de l'analyse d'images de documents. Elle a connu ces dernières années de grands progrès, et les succès des travaux de recherches ont donné lieu à de nombreuses applications industrielles, dans plusieurs domaines, citons par exemple la lecture automatique de formulaires, de chèques ou d'adresses postales. La reconnaissance de l'écriture suppose une localisation préalable des entités textuelles qui peuvent êtres mots ou bien caractères. Deux modes d'écritures existent : l'écriture imprimée, et l'écriture manuscrite. En effet, la régularité de l'imprimé permet d'utiliser des techniques beaucoup plus fiables et beaucoup plus directes et rapides que celles

pour le manuscrit dont la complexité et la variabilité sont très importantes. On peut distinguer aussi la reconnaissance des caractères isolés de la reconnaissance de mots.

a. Reconnaissance des caractères isolés

Au cœur des systèmes de reconnaissance de l'écriture manuscrite, ce sont les moteurs de reconnaissance de caractères isolés qui ont le plus bénéficié des recherches. La reconnaissance de caractères isolés est considérée comme la tâche la plus basique d'un système de reconnaissance de l'écriture, dans laquelle l'effort d'analyse est concentré non pas sur tout le mot mais sur un seul élément du vocabulaire.

Chatelain [CHA 06] a exprimé le problème de la reconnaissance de caractères isolés comme suit : « étant donnée une image de caractère isolé (chiffre, lettre minuscule ou majuscule, symbole de ponctuation, symbole monétaire, etc.), la reconnaissance de caractère vise à lui attribuer sa classe d'appartenance à l'aide d'un algorithme de reconnaissance de formes ».

Le processus de reconnaissance peut donc être représenté par une classification : A partir de la description du caractère traité, le module de reconnaissance essaye d'attribuer le caractère à la classe la plus proche (en termes de similarité).

En effet, de nombreuses approches de reconnaissance existent, et dépendent du vecteur de caractéristiques extrait précédemment à partir de la forme. Selon [MEH 06], on peut distinguer cinq approches: approche structurelle, statistique, stochastique, connexionniste, et hybride.

b. Reconnaissance de mots

Il existe deux approches possibles pour la reconnaissance des mots [BEL 01] [CHA 06] :

Approche globale : elle possède une vision générale du mot. Cette approche est basée sur une description unique de l'image du mot dans son ensemble, sans chercher à identifier chacune des lettres qui le composent. L'inconvénient de cette approche réside dans sa sensibilité à la variabilité des mots. Cependant, l'aspect généraliste de cette approche la limite à des vocabulaires distincts et réduits (cas des montants numériques de chèques).

Approche analytique : vise à reconnaître les mots en identifiant les lettres qui les composent. Une étape de segmentation est donc nécessaire afin de déterminer les limites entre les lettres. Cette approche est la seule applicable dans le cas de grands vocabulaires. Son inconvénient principal demeure la nécessité de l'étape de segmentation en caractères.

1.4.5.2. Identification de la langue et de la fonte

L'identification de la langue et de la fonte dans un document est une autre application de l'analyse de documents qui vise à faciliter l'adaptation des systèmes de reconnaissance de l'écriture, surtout dans les documents présentant plusieurs langues et plusieurs fontes. Ces deux applications peuvent également être utilisées pour des besoins d'archivage, d'indexation et d'interprétation des documents [BEL 01] [WAK 98].

Spitz [SPI 97], un des pionniers du domaine a proposé pour l'identification de la langue une méthode capable de classer cinq langues différentes dans un même document en se basant sur

la position verticale des concavités par rapport à la ligne de base. Un autre système est proposé par Waked et a1. [WAK 98] pour l'identification de trois types de scripts, à savoir : l'arabe, le romain et l'idéographique. Les caractéristiques utilisées pour la distinction entre ces trois scripts sont extraites à partir de la projection horizontale et des rectangles englobantes des composantes connexes. Les fontes sont classées en fonction de la police, du style (gras, italique) et de la taille. Pour l'identification des fontes, Belaïd et Anigbogu [BEL 94] ont proposé une méthode structurelle utilisant les mêmes primitives du module de reconnaissance pour identifier la fonte majoritaire dans un bloc de texte. Zramdini [ZRA 98] a proposé le système ApOFIS capable de distinguer plus de 280 fontes différentes en combinant 10 polices, 7 tailles et 4 styles.

1.4.5.3. Reconnaissance de documents structurés

La reconnaissance de documents structurés consiste à extraire la structure logique du document [BEL 01]. Contrairement à la structure physique qui décrit l'organisation géométrique des documents dans les pages, la structure logique décompose le document en éléments d'information caractérisés par le rôle qu'ils jouent dans le document. La reconnaissance de structures logiques repose essentiellement sur l'élaboration d'une correspondance entre les entités physiques, extraites précédemment, et un ensemble d'entités logiques qui expriment généralement la sémantique du document (chapitres, rubriques, articles, titres, sous-titres, paragraphes,...). Cette mise en correspondance entre les entités physiques et logiques est communément appelée étiquetage logique (Figure 1.8).

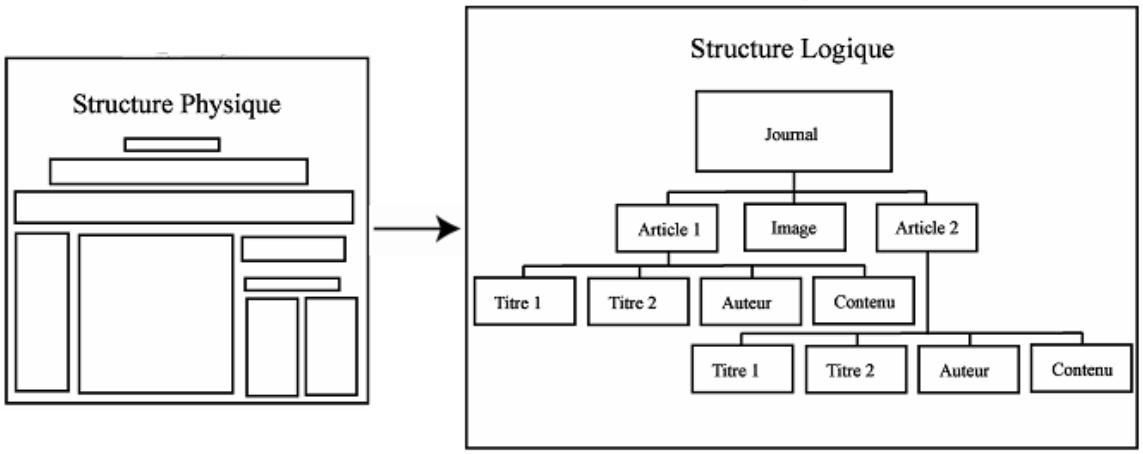


Figure 1.8: Exemple de transformation de la structure physique en structure logique d'une page de journal [HAD 06]

Dans la littérature nous trouvons plusieurs travaux qui s'intéressent à la reconnaissance de la structure logique de documents, nous pouvons citer les travaux de Tsujimoto [TSU 90], Niyogi [NIY 95], Palermo [PAL 99] et Souafi [SOU 02].

1.4.5.4. Transcription du texte assistée par ordinateur

Dans certains types de documents comme les documents anciens imprimés ou manuscrits, les systèmes de reconnaissance optique de caractères (OCR) se montrent inefficaces et ne peuvent pas donc fonctionner correctement. Ceci est justifié par le fait que la mauvaise qualité

de ces documents influence toutes les étapes d'analyse et rend donc difficile l'extraction de la structure physique des documents.

Une solution possible de ceci, est d'avoir recours à la transcription assistée par ordinateur. Le principe est simple, il consiste à identifier les formes similaires dans une image de document. Quand toutes les formes similaires sont collectées, l'ordinateur demande de les identifier manuellement, et il produit une transcription, sans erreur. Le résultat de la saisie des formes similaires peut être conservé pour une nouvelle lecture d'un document de même origine et qui utilise les mêmes formes de caractères ; cette phase s'appelle l'apprentissage [LEB 03] [EMP 03]. Les taux de redondances de formes de caractère sur les ouvrages du XVI^{ème} siècle, dépassent 98% avec un taux d'erreur presque nul, ce qui permet de faire la transcription du texte avec la saisie de moins de 2% des formes de caractères [LEB 03]. La transcription d'un ouvrage entier est réalisée en quelques heures par la saisie du texte correspondant aux différentes formes de caractères codées dans le dictionnaire.

La figure 1.9 donne la liste des 243 formes différentes de caractères identifiées dans une page de document traité dans le cadre du projet DEBORA [EMP 07]

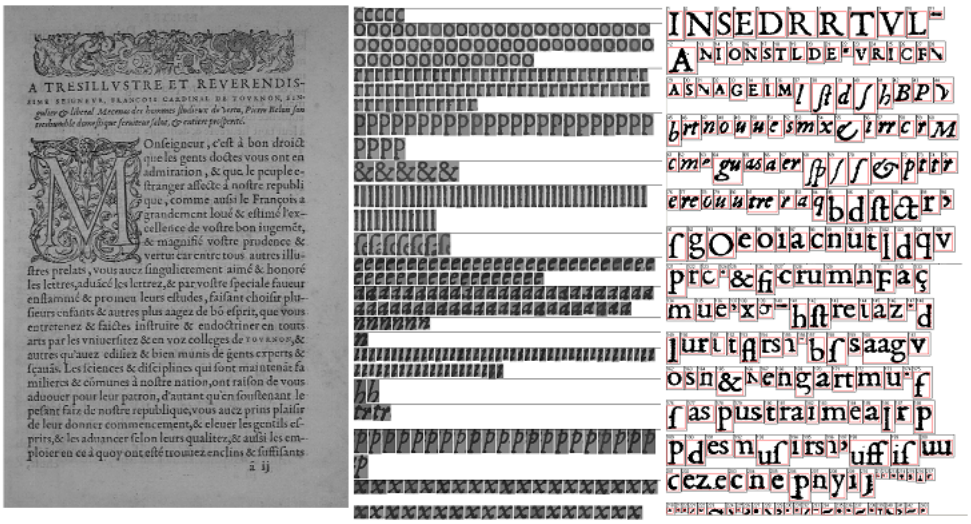


Image originale *Illustration de quelques redondances* *Liste des 243 formes différentes*

Figure 1.9: Liste des 243 formes différentes de caractères, traitées par DEBORA

1.4.5.5. Recherche de similarités

La recherche de similarités est souvent utilisée pour l'appariement de deux formes pour des besoins de recherche. La similarité entre deux formes est globalement difficile à définir. Plusieurs méthodes ont été proposées dans la littérature pour effectuer un appariement de formes. Ses méthodes se basent soit sur des techniques de correspondances de points de contours révélateurs de la silhouette des formes, soit sur des techniques basées sur des descripteurs de formes nécessitant la description d'un grand nombre de prototypes invariants pour établir des correspondances avec les éléments à retrouver [SCH 97]. Ces dernières techniques ont été largement employées pour la localisation et la reconnaissance de chiffres selon un vocabulaire restreint [LEC 98], et la recherche de visage dans des environnements

naturels [MOG 00], mais elles se montrent inefficaces pour les images textuelles présentant un vocabulaire large, surtout si le texte est manuscrit.

Ces deux techniques ne sont pas les seules, d'autres plus souples nécessitant moins de connaissances à priori sur les objets à trouver. Elles sont basées sur des informations globales de structures, de contours et de silhouettes. Nous pouvons citer dans cette famille, les travaux basés sur des descripteurs de Fourier [ZAH 72], de recherche de squelettes [SHA 98], et des points de contours [CAR 99].

1.5. Conclusion

Le savoir se transmet dans nos sociétés par l'écrit. C'est pour cette raison qu'on trouve actuellement une très grande quantité de documents de différents types et dans différents domaines. Le nombre très important de ces documents a obligé les chercheurs à développer des méthodes et des outils pour les traiter et les analyser automatiquement. En effet, plusieurs travaux ont été intéressés à l'analyse automatique de documents de différents types, et les résultats obtenus fait sortir ce domaine des laboratoires vers des applications réelles dans l'industrie. Dans ce chapitre, nous avons abordé le problème d'analyse d'images de documents comme domaine de recherche, sans se limiter à un type particulier de documents. Nous avons introduit la notion de document, défini le domaine de l'analyse de documents et cité quelques domaines d'applications. L'absence d'un objectif universel à toutes les applications de traitement et d'analyse de documents rend difficile de s'accorder sur un schéma universel pour tous les systèmes d'analyse de documents. Nous avons présenté les différentes étapes qui peuvent être incluses dans le processus d'analyse tout en présentant pour chaque étape un état de l'art de quelques méthodes les plus fréquemment citées dans la littérature.

Chapitre 2 : Appariement et recherche approximative

2.1. Introduction

Tester l'apparition d'une chaîne de caractères dans un texte (motif), compter le nombre d'occurrences d'une chaîne donnée, déterminer l'ensemble des positions de ces occurrences…etc. sont des problèmes importants qui apparaissent dans de nombreux domaines scientifiques. En informatique, on les rencontre naturellement dans l'édition de textes, en fouille de données, en analyse syntaxique ou en recherche d'information (comme dans les moteurs de recherche internet). En particulier, tous les éditeurs de textes et de nombreux langages de programmation offrent des possibilités de recherche de motifs.

Quelques problèmes peuvent apparaître et viennent généralement de la mauvaise qualité du texte (qui résulte par exemple de la reconnaissance d'une image de document), de l'hétérogénéité du texte (à cause de multilinguisme), ou bien des erreurs d'orthographe. Ce genre de problèmes rend les méthodes de recherche exacte inefficaces, ce qui nous oblige à procéder une recherche approximative des occurrences du motif dans le texte. En effet, l'ajout de l'imprécision dans la recherche permet la détection de motifs approchés mais augmente aussi beaucoup le nombre de fausses occurrences.

La recherche approximative est maintenant omniprésente dans plusieurs domaines. Une application intéressante en informatique est la correction orthographique. La recherche approximative est aussi au cœur de la bioinformatique, que ce soit pour la recherche de molécules nouvellement séquencées dans les banques de données existantes, ou pour la recherche de similitudes entre deux séquences biologiques. C'est pourquoi le développement d'algorithmes plus efficaces en performances, temps et espace de stockage pour ce problème restera toujours d'actualité.

Dans ce chapitre, nous considérons le problème général de l'appariement et de la recherche approximative de mots dans un texte. Dans un premier temps, nous donnons quelques définitions et notations nécessaires. Nous passons par la suite à la comparaison de chaînes de caractères, et à la recherche exacte d'un mot dans un texte tout en présentant brièvement quelques algorithmes des plus connus. La section 4 s'intéresse à la comparaison approximative de chaînes et les mesures de similarité. La recherche approximative est abordée dans la section 5. Dans la 6ème section, nous présentons les arbres et les automates de suffixes comme des outils avancés de recherche de motifs.

2.2. Définitions

Les définitions présentées dans cette section sont extraites principalement à partir du livre « Algorithmique et texte » de Crochemore et *al.* [CRO 01]. Nous respectons dans la suite de chapitre, les notations fournies dans cette section.

Définition 1

Un **Alphabet** Σ est un ensemble fini non vide d'éléments appelés lettres, notés *a, b, c, …*

Définition 2

Une **chaîne** ou bien une **séquence** ou encore un **mot** *s* de longueur *n* est une suite de *n* éléments de Σ, indexés de 1 à *n*.

Définition 3

La **longueur** d'une chaîne *s* notée |*s*| est le nombre d'éléments dans la chaîne.

Définition 4

La **chaîne vide** notée ε est une suite d'éléments de longueur nulle.

Définition 5

Σ^* et Σ^+ dénotent respectivement l'**ensemble des mots** et l'**ensemble des mots non-vides** constructibles à partir de l'alphabet Σ.

Définition 6

Une chaîne *z* est une **sous-chaîne** ou bien **facteur** de la chaîne *w*, s'il existe deux autres chaînes *u* et *v* tel que *w=uzv*.

Définition 7

Un **préfixe** de *w* est une sous-chaîne de *w* commençant à la position 1. Plus formellement, la chaîne *u* est un préfixe de la chaîne *w* s'il existe une chaîne *v* tel que *w=uv*.

Définition 8

Un **suffixe** de *w* est une sous-chaîne de *w* terminant à la position *n*. Plus formellement, la chaîne *u* est un suffixe de la chaîne *w* s'il existe une chaîne *v* tel que *w=vu*.

Définition 9

Un préfixe ou un suffixe d'un mot *w* est dit **propre** s'il n'est pas égal à *w*.

Définition 10

Un **bord** d'un mot non vide *w* est un facteur propre de *w* qui est à la fois un préfixe et un suffixe de *w*. Par exemple : les deux seuls bords du mot *abacaba* sont les mots *a* et *aba*.

Définition 11

Nous notons le $i^{ème}$ symbole de *s* par *s[i]* pour tout *i* tel que $1 \leq i \leq n$. Une **sous-séquence** *w* de *s* est obtenue en ôtant de *s* un certain nombre de caractères.

Plus formellement $w = s[u_1]s[u_2] \ldots s[u_k]$ est une sous-séquence de *s* si et seulement si :

- $k \leq n$;
- et *u* est une suite croissante à valeurs dans N^* tel que $1 \leq u_1$ et $u_k \leq n$.

Nous remarquons que la notion de sous-chaîne est différente de celle de sous-séquence, alors que chaîne et séquence ont le même sens.

Définition 12

L'opération de **concaténation** est notée par un **.** ; si *s* et *r* sont deux chaînes de longueurs respectives *n* et *m*, *s.r = s[1]s[2] . . . s[n]r[1] . . . r[m-1]r[m]*.

Définition 13

Un **texte** est une séquence de mots, comprenant les répétitions. L'ensemble des mots distincts d'un texte est appelé un **dictionnaire** ou **lexique** du texte. Bien sûr, un **dictionnaire** est un cas particulier de texte où les mots apparaissent uniquement une seule fois. En réalité, il n'est pas possible de rassembler tous les mots d'un langage « subsistant[8] » dans un lexique, le lexique est de nature statique, et le langage est vivant et dynamique ; de nouveaux mots sont ajoutés continuellement.

2.3. Appariement de chaînes

2.3.1. Définition

Soit une chaîne de caractères de test et une ou plusieurs chaînes de référence. Apparier (comparer) consiste à évaluer la similitude entre la chaîne de test et chaque chaîne de référence. Il existe deux approches de comparaison de chaînes. Une première approche dite comparaison exacte pour « Exact String Matching » qui consiste à dire si deux chaînes sont similaires ou non, tandis que les méthodes de la deuxième approche appelée comparaison ou appariement approximative « Approximate String Matching », donnent une mesure de ressemblance entre les chaînes.

2.3.2. Pourquoi comparer des chaînes ?

Il est souvent nécessaire de comparer deux chaînes ou plus et de mesurer jusqu'à quel point elles diffèrent. Nous pouvons trouver plusieurs domaines d'application de la comparaison de chaînes, [SAN 99] a présenté quelques domaines d'entre eux :

2.3.2.1. Biologie moléculaire:

Notamment pour les séquences biologiques (ADN, ARN ou séquences d'acides aminés). Aujourd'hui, la méthode la plus puissante pour inférer la fonction biologique d'un gène (ou de la protéine qu'il code) est la recherche de similarité de séquences dans les bases de données d'ADN et de protéines.

2.3.2.2. Informatique

L'appariement est utilisé pour la comparaison de deux fichiers, où chaque ligne est traitée comme un seul symbole. La deuxième application en informatique est le correcteur d'orthographe (recherche de mots les plus proches dans un dictionnaire), une extension prend en compte les inversions de caractères dues aux fautes de frappe.

2.3.2.3. Langage humain

La comparaison de chaînes peut être utilisée en reconnaissance de la parole ou de l'orateur, dans laquelle le flot de paroles est échantillonné en syllabes. Il s'agit de reconnaître et d'isoler des mots d'un vocabulaire limité.

[8] *C'est-à-dire un langage encore utilisé.*

2.3.2.4. Autres applications

Plusieurs autres domaines peuvent être cités : contrôles de codes et d'erreurs, strates géologiques, anneaux des troncs d'arbres et comparaison de textes.

2.3.3. Recherche exacte des chaînes de caractères

La recherche d'une chaîne de caractères (communément appelée **motif**) $P=p_1\ p_2...p_m$ de longueur m dans une autre chaîne ou plus précisément dans un **texte** $T=t_1\ t_2...\ t_n$ de largeur n est souvent connue sous le nom de **recherche de sous-chaînes**. Elle consiste à trouver toutes les occurrences du motif P dans le texte T. La recherche est implémentée donc par une comparaison de chaînes.

2.3.3.1. Quelques algorithmes de recherche exacte

Plusieurs algorithmes ont été proposés pour la recherche exacte d'un mot P de longueur m dans un texte T de longueur n. L'algorithme le plus simple c'est l'algorithme naïf ou de force brute qui consiste à réaliser une comparaison caractère après caractère du texte et de la chaîne recherchée. Le problème de cet algorithme est qu'il nécessite, dans le pire des cas, $m \times n$ comparaisons pour vérifier le texte. Le coût élevé de cet algorithme a engendré le développement de plusieurs autres algorithmes pouvant être classés en trois grandes classes [KAC 06] : les approches avec comparaison de caractères, les approches numériques, et les approches avec automates.

2.3.3.1.1. Approches avec comparaisons de caractères

Le premier algorithme qui résout le problème en temps linéaire est celui de Morris et Pratt [MOR 70] et qui a un complexité de $O(m+n)$. Le deuxième algorithme est celui de Knuth-Morris-Pratt [KNU 77] qui est considéré comme une variante de l'algorithme précédent. L'algorithme de Boyer-Moore [BOY 77], est l'un des algorithmes les plus utilisés aussi. Dans cet algorithme, la recherche est effectuée en temps moyen $O(n/ m)$.

2.3.3.1.2. Approches numériques

Ces approches se basent sur des opérations sur des bits ou sur l'arithmétique. L'algorithme de Karp-Rabin [RIC 87] constitue un exemple de ces méthodes.

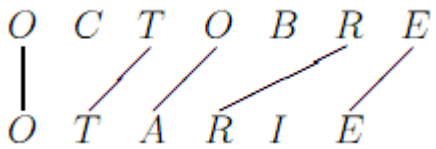
2.3.3.1.3. Approches avec automates

D'autres méthodes utilisées pour la recherche d'occurrences d'un motif dans un texte sont basées sur la construction d'un automate fini. Un exemple de ce type de méthodes est l'algorithme Aho-Corashick [AHO 75].

2.4. Comparaison approximative de chaînes

Avant de considérer le problème de la recherche des occurrences approximatives d'un mot dans un texte, considérons le problème plus élémentaire de la comparaison approximative de deux mots. Pour analyser la différence totale entre deux séquences, on peut regarder l'ensemble des différences élémentaires entre les caractères. Il existe au moins trois modes différents de représentation de cette analyse de différences entre les séquences : les traces, les alignements et les listings [SAN 99]. Etant donné deux chaînes s et r, on peut définir les trois représentations possibles comme suit :

Trace : Une trace de *s* à *r* est la séquence source *s* au dessus de la séquence cible *r*, avec des lignes joignant des caractères de la source et de la cible. Un exemple de trace des deux mots « OCTOBRE » et « OTARIE » est comme suit :



Alignement : Un alignement entre *s* et *r* est une matrice de deux lignes. La première ligne est la source *s* dans laquelle des caractères nuls ont éventuellement été insérés. Nous représentons les caractères nuls par des « - », tel que « - » n'appartient pas à l'alphabet. Par exemple l'alignement des deux mots « OCTOBRE » et « OTARIE » est donné comme suit :

$$\begin{pmatrix} O & C & T & O & B & R & - & E \\ O & - & T & A & - & R & I & E \end{pmatrix}$$

Listing : Un listing de *s* à *r* est une alternance de séquences et d'opérations élémentaires, commençant par la séquence source *s* et terminant par la séquence cible *r*, et qui satisfait la propriété suivante : deux séquences adjacentes dans le listing doivent différer seulement par l'application d'une opération élémentaire. Le listing des deux mots « OCTOBRE » et « OTARIE » est donné comme suit :

OCTOBRE
Suppression de *C*
OTOBRE
Suppression de *B*
OTORE
Substitution du 2ème *O* en *A*
OTARE
Insertion de *I*
OTARIE

Selon [ZOB 95], il existe deux grandes classes de schémas pour l'appariement approximatif de chaînes: les mesures de similarité et le codage phonétique. Les mesures de similarité calculent un degré de similarité estimé entre deux chaînes, comme par exemple le nombre de caractères communs, le nombre d'étapes nécessaires pour transformer une chaîne à l'autre…etc. Les schémas de cette classe sont généralement appropriés pour la correction de l'orthographe, d'où environ 80% des erreurs sont causées par une simple insertion, suppression ou bien changement de caractères [HAL 80]. Le codage phonétique, de l'autre côté, assigne un code phonétique à chaque chaîne ; deux chaînes sont jugées similaires si elles ont le même code. Les schémas phonétiques se montrent appropriés pour l'appariement des noms personnels [BOR 92].

2.4.1. Mesures de similarité et distances entre séquences

Une mesure de similarité associe un nombre réel *e* à une paire de chaînes de caractères *(r, s)*. En effet, plusieurs mesures ont été proposées dans la littérature pour calculer la similarité ou la différence entre deux chaînes de caractères, dont la distance d'édition est certainement la

mesure de similarité la plus connue et utilisée [HAL 80] [NAV 06]. Certaines de ces mesures ne peuvent être appliquées que lorsque les séquences sont de longueurs égales, tandis que d'autres sont capables de comparer deux chaînes de caractères de tailles différentes.

Nous présentons dans la suite de cette section quelques mesures de similarité les plus connues, tout en focalisant sur la célèbre distance d'édition.

2.4.1.1. Distance de Hamming

Cette mesure de distance [SAN 83], utilisée en informatique, traitement du signal et télécommunications, a été proposée par Richard Hamming, elle permet de quantifier la différence entre deux séquences symboliques, de même longueurs.

A deux séquences (de longueurs égales), la distance de Hamming associe le nombre de positions pour lesquelles les éléments correspondant dans chacune des séquences sont différents. Ainsi, la distance de Hamming entre deux chaînes de caractères *r* et *s* de longueur *n* se traduit par :

$$d(r,s) = nb\ \{i : r[i] \neq s[i]\}$$

Dans cette méthode, *s[i]* correspond à *r[i]* et la comparaison entre eux est exprimée par un 0 si *s[i] = r[i]* et un 1 si *s[i] ≠ r[i]* [SAN 99]. Par exemple la distance de Hamming entre la chaîne *acbba* et la chaîne *abbba* est égale à 1.

2.4.1.2. Distance d'édition

2.4.1.2.1. Définition

La distance d'édition est parfois référencée comme la distance de Levenshtein en reconnaissance de l'article [LEV 66] par Vladimir Levenshtein où la distance d'édition a été certainement discutée pour la première fois.

La distance d'édition entre deux chaînes *s* et *r* est définie comme le nombre minimal d'opérations d'édition (insertion, suppression, substitution) nécessaires pour transformer la première chaîne en la seconde.

Levenshtein a proposé d'associer le même coût à toutes les transformations pour calculer la distance d'édition entre deux chaînes. Une généralisation de la distance d'édition est d'autoriser l'association d'un poids, coût ou score arbitraire positif à chaque opération d'édition. Pour $a, b \in \Sigma$, on note :

- *Sub(a, b) :* le coût de la substitution de la lettre *a* par la lettre *b* ;
- *Del(a) :* le coût de la suppression de la lettre *a* ;
- *Ins(b) :* le coût de l'insertion de la lettre *b*.

Le calcul de la distance d'édition généralisée consiste donc à déterminer une suite d'opérations d'édition pour transformer *s* en *r* en minimisant le coût total des opérations utilisées. Nous notons $T_{s,r}$ l'ensemble des suites d'opérations d'édition permettant de transformer *s* en *r*. La distance d'édition généralisée des chaînes *s* et *r* est définie par :

$d(s,r) = \min\{co\hat{u}t\ de\ \sigma |\ \sigma\ \in\ T_{s,r}\}$

Si on prend comme exemple la transformation de la chaîne *aabcb* en *ababd*, la figure 2.1 présente quelques chemins possibles permettant la transformation. En prenant un coût de 1 pour chacune des transformations, la distance d'édition entre *aabcb* et *ababd* est de trois. Elle serait différente si le coût d'une suppression était de 0.5, celui d'une insertion de 0.75 et celui d'un changement de 0.25. Sa valeur serait alors de 1.5.

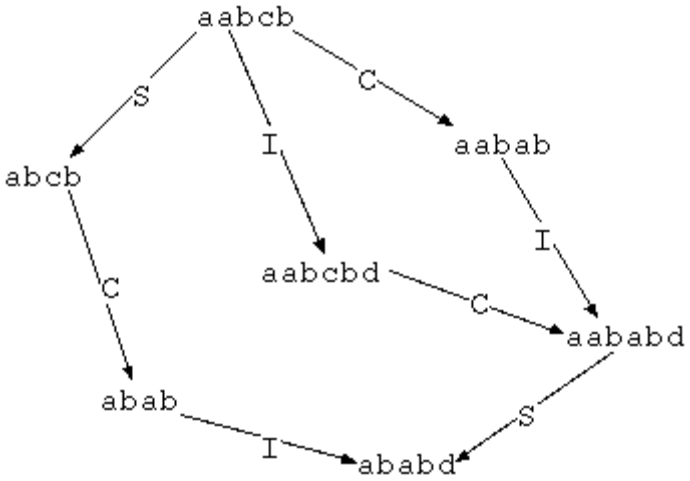


Figure 2.1 : Quelques chemins possibles pour passer de la chaîne aabcb à la chaîne ababd

2.4.1.2.2. Calcul de la distance d'édition entre deux mots

Levenshtein a introduit la définition de la distance d'édition mais il n'a jamais décrit l'algorithme permettant de trouver la distance d'édition entre deux chaînes [PEV 06]. Un algorithme de calcul de la distance d'édition entre les chaîne *X* et *Y* a été proposé par Wagner et Fischer [WAG 74]. C'est un algorithme de programmation dynamique (solution de type du bas en haut), qui utilise une matrice *D* de dimension *(n+1)×(m+1)* où *n* et *m* sont les longueurs des deux chaînes de caractères.

L'approche consiste à remplir la table *D* tel que la valeur d'une case dépend des valeurs précédemment obtenues. La première ligne de la table *D(0, i)* est initialisée à *i* et la première colonne *D(j, 0)* est initialisée à *j*.

L'algorithme peut être résumé par le pseudo-code suivant.

Pré-requis : *n* = longueur de la première chaîne *X*
Pré-requis : *m* = longueur de la deuxième chaîne *Y*
Pré-requis : *C*(*a, b*) = coût de la transformation pour passer de *a* à *b*
Pré-requis : *lambda* = vide
Assure : *D* [*m, n*] = distance d'édition de *X* à *Y*
Pour *i* = 0...*n* **Faire** *D*[0, *i*]=*i*
Pour *j* = 0...*m* **Faire** *D*[*j*, 0]=*j*

Pour *i* = 1...*m* **Faire**
  Pour *j* = 1...*n* **Faire**

D [*i* ,*j*] = Min {
D [*i*-1, *j*-1] + *C* (X_i, Y_j), // substitution (0 si les caractères X_i et Y_j sont égaux)
D [*i*-*1*, *j*] + *C* (X_i, *lambda*), // effacement
D [*i*, *j*-1] + *C* (*lambda*, Y_i)} // insertion
}
Fin Pour
Fin Pour

Par exemple, le tableau 2.1 représente la table *D* des distances entre les deux séquences *AAGCTAAG* et *AGGAGGA*.

	ε	***A***	***A***	***G***	***C***	***T***	***A***	***A***	***G***
ε	0	1	2	3	4	5	6	7	8
A	1	0	1	2	3	4	5	6	7
G	2	1	1	1	2	3	4	5	6
G	3	2	2	1	2	3	4	5	5
A	4	3	2	2	2	3	3	4	5
G	5	4	3	2	3	3	4	4	4
G	6	5	4	3	3	4	4	5	4
A	7	6	5	4	4	4	4	4	5

Tableau 2.1: Table des distances entre les séquences AAGCTAAG et AGGAGGA.

La distance d'édition correspond à la valeur contenue dans la dernière case de la matrice *D* et donc 5 dans l'exemple précédent.

La complexité de cet algorithme en temps et en espace est de *O(nm)*, où *n* et *m* représentent les longueurs des deux mots.

2.4.1.3. Distance de Jaro et Jaro-Winkler

La distance de Jaro-Winkler, a été proposée par William E. Winkler, en 1999 [WIN 99]. Son nom vient du fait qu'elle est « dérivée » de la distance de Matthew Jaro, proposée dix ans plus tôt [JAR 89]. Les deux mesures sont adaptées pour les chaînes de caractères courtes, telles que celles représentant des noms et des étiquettes, et principalement utilisées pour la détection de doublons.

La distance de Jaro-Winkler est en fait une variante de la distance de Levenshtein. Contrairement à cette dernière, le degré de similarité croît en même temps que la valeur de la distance de Jaro-Winkler. Le résultat est normalisé de façon à obtenir une mesure variant entre 0 et 1 tel que 0 représente l'absence de similarité et 1 la similarité parfaite.

2.4.1.3.1. Distance de Jaro

La distance de Jaro [JAR 89] entre deux chaînes *r* et *s* est définie par :

$$d_j = \frac{1}{3}\left(\frac{m}{|r|} + \frac{m}{|s|} + \frac{m-t}{m}\right)$$

Tel que :

- *m* est le nombre de caractères correspondants; deux caractères identiques de *r* et *s*, sont considérés comme étant correspondants si leur éloignement (c'est-à-dire la différence entre leur position dans leurs chaînes respectives) ne dépasse pas : $\left\lfloor \frac{\max(|r|,|s|)}{2} \right\rfloor - 1$
- *t* est le nombre de transpositions et est obtenu en comparant le $i^{ème}$ caractère de *r* avec le $i^{ème}$ caractère correspondant de *s*. Le nombre de transpositions est égal au nombre de fois où ces caractères sont différents, divisé par deux.

2.4.1.3.2. Distance de Jaro-Winkler

Winkler [WIN 99] a ajouté à la distance de Jaro, un coefficient de préfixe *p*, favorisant les chaînes commençant par un même préfixe (de longueur $l < 4$).

En considérant :

- d_j : la distance de Jaro entre *r* et *s*
- *l* : la longueur du préfixe commun (composé au maximum de 4 caractères)
- *p* : un coefficient favorisant les chaînes de préfixe commun (Winkler proposait $p = 0.1$)

La distance de Jaro-Winkler est donnée ainsi par:

$$d_w = d_j + (lp(1 - d_j))$$

2.4.1.4. Mesure de Jaccard :

La mesure de similarité de Jaccard entre deux chaînes de caractères *s* et *t* est obtenue tout simplement par la formule : $Jaccard(s,t) = \frac{|s \cap t|}{|s \cup t|}$

Cette mesure est le rapport entre le nombre des sous chaînes communes à *s* et *t* et le nombre total de sous chaînes de *t* et de *s*. Si les sous chaînes sont des caractères, la mesure de Jaccard correspond au nombre de caractères communs aux deux chaînes.

2.4.1.5. Mesure de Monge-Elkan

Monge et Elkan [MON 96] ont proposé une méthode récursive pour comparer deux chaînes de caractères longues. Ainsi, pour les deux chaînes de caractères $s=a_1 \ldots a_K$ et $t=b_1 \ldots b_L$, la mesure de similarité de Monge-Elkan est donnée par :

$$sim(s,t) = \frac{1}{K}\left(\sum_{i=1}^{K} \max_{j=1\ldots L}\left(sim'(a_i, b_j)\right)\right)$$

Où *sim′* est une fonction de similarité secondaire dite de base. Cette fonction peut être la fonction de Jaro, Jaro-Winkler, Jaccard, ou n'importe quelle autre fonction.

2.4.2. Méthodes phonétiques

L'autre classe de schémas pour l'appariement approximatif de chaînes est la classe des méthodes phonétiques. La méthode la plus ancienne dans cette classe est la méthode **Soundex** [HAL 80] publiée en 1918. L'idée de cette méthode est de coder les lettres selon leurs sons, afin de transformer une chaîne de caractères en une forme canonique. Le tableau 2.2 illustre les codes de chaque lettre.

Code	Caractères
0	a e h i o u w y
1	b f p v
2	c g j k q s x z
3	d t
4	l
5	m n
6	r

Tableau 2.2: Codes phonétiques de l'algorithme Soundex

L'algorithme Soundex opère en quatre étapes :

1. Remplacer toutes les lettres sauf la première par leurs codes phonétiques.
2. Eliminer toutes les répétitions consécutives des codes.
3. Eliminer toutes les occurrences du code 0 (qui correspond à l'élimination des voyelles)
4. Retourner les quatre premiers caractères de la chaîne résultante.

Par exemple, les mots *king* et *khyngge* se codent tous les deux par *k52*, mais *knight* par *k523*, alors que *night* se code par *n23*. Avec cette méthode, des chaînes différentes peuvent être transformées en le même code, comme par exemple *pulpit* et *phlebotomy*.

Les exemples précédents montrent que l'algorithme **Soundex** est assez simple, et pour cette raison que des adaptations de cet algorithme ont été proposées. Une variation ambitieuse de **Soundex** est l'algorithme **Phonix** [GAD 90].

Phonix est similaire à **Soundex** dans la transformation des lettres en un ensemble de codes, mais avant ça une transformation de groupe de lettres est nécessaire. Par exemple, *gn*, *ghn*, et *gne* se transforment en *n*, *tjV* (*V* est une voyelle) se transforme en *chV* si elle apparait en début de la chaîne, et *x* se transforme en *ecs*. Les codes de **Phonix** sont résumés par le tableau 2.3:

Code	Caractères
0	a e h i o u w y
1	b p
2	c g j k q
3	d t
4	l
5	m n
6	r
7	f v
8	s x z

Tableau 2.3: Codes phonétiques de l'algorithme Phonix

2.5. Recherche approximative d'un mot dans un texte

Dans cette section, nous passons au problème central de ce chapitre, c'est-à-dire la recherche des occurrences approximatives d'un mot dans un texte. Nous abordons dynamiquement le problème.

2.5.1. Le problème

Etant donné un texte $T = t_1 t_2 \ldots t_n$ de longueur n, une chaîne de caractères (motif) $P = p_1 p_2 \ldots p_m$ de longueur m et un nombre naturel t. Le problème de la recherche des occurrences approximatives du mot P dans le texte T consiste à trouver toutes les positions, dans T où il y a une occurrence de P, ayant au plus t erreurs [KAC 06] [HAM 02]. Dans la suite de ce chapitre nous utilisons la distance d'édition comme mesure de similarité entre les chaînes.

Notons $D(i, j)$ la distance d'édition minimale entre le préfixe $p_1 p_2 \ldots p_i$ de longueur i de P et les facteurs du texte T se terminant en position j. On dit qu'il y a une ou des occurrences approximatives de P ayant au plus t erreurs, se terminant en position j du texte si $D(m, j) \leq t$.

La solution classique pour trouver les occurrences [SEL 80], à t erreurs près, d'un mot P de longueur m, dans un texte T de longueur n est de calculer la matrice des distances $D[0..m, 0..n]$, tout en gardant en mémoire les positions j pour lesquelles $D(m, j) \leq t$.

Par exemple, on veut trouver les positions dans le texte $T = ACGTAACGAGG$ où il y a des occurrences, à une erreur près, du mot $P = AAC$. On commence par initialiser la table des distances D en posant $\forall i, D(i, 0) = i$ et $\forall j, D(0, j) = 0$. On calcule ensuite la valeur de chacune des autres cellules (i, j) en nous servant des valeurs déjà calculées:

	ε	**A**	**C**	**G**	**T**	**A**	**A**	**C**	**G**	**A**	**G**	**G**
ε	0	0	0	0	0	0	0	0	0	0	0	0
A	1	0	1	1	1	0	0	1	1	0	1	1
A	2	1	1	2	2	1	0	1	2	1	1	2
C	3	2	1	2	3	2	1	0	1	2	2	2

Tableau 2.4: Table des distances pour la recherche d'un mot dans un texte

Il y a donc des occurrences du mot $P = AAC$, à une erreur près, se terminant en chacune des positions: 2, 6, 7 et 8 du texte.

Au contraire de la recherche exacte qui consiste à balayer la chaîne à analyser de gauche à droite et s'arrêter dès que la chaîne cherchée est trouvée, dans la recherche approximative, ce n'est pas possible sauf si on trouve effectivement strictement la même chaîne.

2.5.2. Différentes approches de recherche approximative de mots

Dans cette section, nous présentons quelques approches classiques pour résoudre le problème de la recherche des occurrences approximatives d'un mot dans un texte.

2.5.2.1. Ne calculer qu'une partie de la table des distances

Comme nous avons vu, l'algorithme de calcul de la distance d'édition présenté précédemment est très coûteux en temps et en espace. Une des approches utilisées pour résoudre le problème de la recherche approximative d'un mot dans un texte consiste à analyser la table des

distances *D* afin de ne calculer qu'une partie de cette table. Rappelons que dans cette table, *D(i, j)* est définie comme étant la distance d'édition minimale entre le préfixe $p_1 \ldots p_i$, du mot *P* cherché, et les facteurs du texte *T* se terminant en position *j*. Ukkonen [UKK 85] a déduit le lemme suivant : dans la matrice *D*, on a toujours :

D(i, j) = D(i - 1, j - 1) ou D(i, j) = D(i - 1, j - 1) + 1

Ce lemme montre que plusieurs entrées de la table *D* sont inutiles pour le calcul des occurrences approximatives d'un mot, à *t* erreurs près. En effet, si on a une entrée de valeur supérieure à *t* dans la cellule *(i - 1, j - 1)* de *D* alors, il ne sert à rien de calculer les entrées des cellules *(i, j), (i + 1, j), . . . , (m, j)*, où *m* est la longueur du mot *P* cherché, puisque ces cellules contiennent des entrées de valeur supérieure à *t*.

Par exemple, étant donnée un mot *P = AACG*, et un texte *T = GCGTTGCAGGAACG*. On peut trouver les occurrences approximatives du mot *P* à une erreur près, en ne calculant qu'une partie de la table comme suit :

	ε	**G**	**C**	**G**	**T**	**T**	**G**	**C**	**A**	**G**	**G**	**A**	**A**	**C**	**G**
ε	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
A	1	1	1	1	1	1	1	1	0	1	1	0	0	1	1
A	2	2	2	2	2	2	2	2	1	1	2	1	0	1	2
C	3	×	×	×	×	×	×	×	×	2	2	×	1	0	1
G	4	×	×	×	×	×	×	×	×	×	×	×	×	1	0

Tableau 2.5: Un exemple de calcul partiel de la table des distances

Les × ici indiquent les cellules *(i, j)* pour lesquelles le calcul de *D(i, j)* n'est pas nécessaire.

Dans cet exemple, parmi 56 entrées de la table, 21 entrées ne sont pas nécessaires au calcul des occurrences approximatives de *P* dans *T*. Le lemme précédent présenté par Ukkonen nous sauve donc ici environ 40% du travail.

Notons que l'efficacité de cette méthode varie selon le mot *P* et le texte *T* choisis. Plus il y a d'occurrences approximatives de *P* dans *T*, plus le nombre de cellules inutiles dans la table *D* est petit. Dans la plupart des cas, la complexité en temps de cet algorithme est encore de *O(nm)*.

2.5.2.2. Prétraitement du mot *P* à l'aide d'un automate fini déterministe

Le prétraitement du mot *P* pour calculer les positions *j* du texte où se termine une occurrence du mot *P* cherché, a été proposé aussi par Ukkonen [UKK 85]. Ce prétraitement est effectué en construisant un automate fini déterministe M_P à partir du mot *P* cherché et du nombre d'erreurs permis *t*. Un automate peut être considéré comme un graphe orienté dont les sommets sont appelés états et les arêtes, transitions. Plus précisément, un automate fini est un quadruplet (S, Σ, T, q_0, A) où :

- Σ est un alphabet fini ;
- *S* est un ensemble fini d'états ;
- *T* est une fonction de transition ;

- $q_0 \in S$ est l'état initial ;
- $A \subseteq S$ est l'ensemble d'états finaux.

En donnant un texte T comme entrée à l'automate M_P, l'automate arrive à un état final, après la lecture d'une lettre t_j de T, si et seulement si T contient une occurrence approximative de P, ayant $e \leq t$ erreurs, se terminant en position j du texte. Le problème qui reste à résoudre est la construction de l'automate M_P. En effet, la construction de l'automate M_P nécessite la définition des ses quatre éléments. Pour un mot P fixé, chaque état de M_P correspond à une colonne pouvant apparaître dans la matrice des distances D quand le texte T varie. Un état de M_P est final si l'élément en position m dans la colonne correspondante de cet état est $\leq t$.

Exemple : Pour le mot P=*aba* et t = 1, l'automate M_P est défini comme suit : L'alphabet Σ ={*a, b*}, l'état initial de cet automate q_0 est l'état correspondant à la première colonne de la table des distances D, c'est-à-dire 0123. Le texte T peut commencer soit par un *a*, soit par un *b*. On calcule donc, pour ces deux cas, la prochaine colonne de la table D:

	ε	***a***
ε	0	0
a	1	0
b	2	1
a	3	2

	ε	***b***
ε	0	0
a	1	1
b	2	1
a	3	2

A partir de l'état initial, on peut donc arriver soit à l'état 0012 avec la transition *a* ou bien à l'état 0112 avec la transition *b*. Pour ces deux états, on calcule aussi les colonnes correspondantes avec *a* et *b* pour obtenir des nouveaux états, et ainsi de suite. On arrête l'algorithme de construction lorsque tous les nouveaux états (ou nouvelles colonnes de D) font déjà partie de l'automate. Comme le nombre d'erreurs permis est t = 1 dans cet exemple, les états finaux de l'automate sont les états se terminant par un 1 ou un 0.

L'automate M_P de cet exemple est donné par la figure 2.2 suivante :

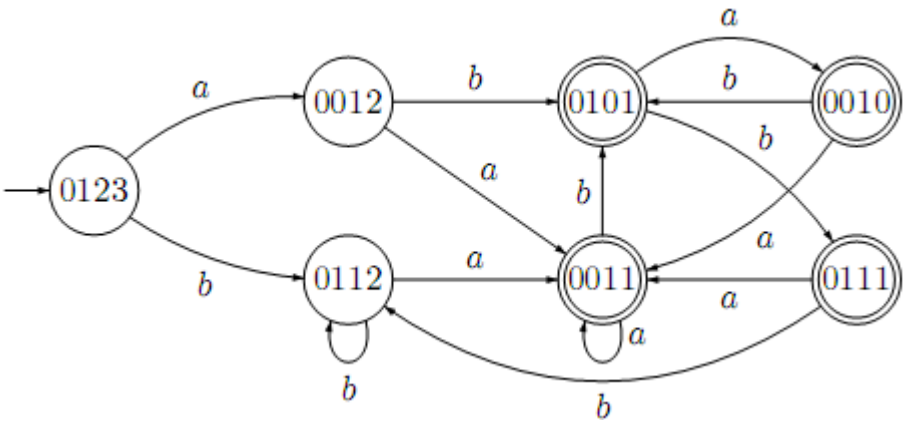


Figure 2.2: L'automate M_p du mot aba

Selon [UKK 85], cet algorithme nécessite un temps $O(m\ |\Sigma|\ k_p)$, où $k_p = min(3^m, 2^t\ |\Sigma|^t\ m^{t+1})$ pour la construction de l'automate, et $O(n)$ pour trouver toutes les positions des occurrences de P dans T, à t erreurs prés.

2.5.2.3. Prétraitement de *P* et calcul de diagonales

La diagonale *d* d'une table des distances *D* comprend l'ensemble des cellules *(i, j)* telles que $j - i = d$. Plusieurs algorithmes utilisent une idée de calcul de diagonales de la table des distances *D* pour rechercher les occurrences approximatives d'un mot dans un texte. L'idée de ces algorithmes est de représenter la table *D* en ne gardant en mémoire, pour chacune des diagonales de la table, que les positions où il y a une augmentation de la valeur de l'entrée par rapport à la valeur de l'entrée précédente sur cette même diagonale [HAM 02].

Pour une diagonale *d* et un nombre d'erreurs *e*, on définit *C(e, d)* comme étant la colonne de plus grand indice *j* telle que $D(j - d, j) = e$. Partant de cette définition, on a une occurrence approximative de *P* en position $m + d$ de *T* ayant au plus *t* erreurs, si et seulement si $C(e, d) = m + d$, pour $e \leq t$, et *m* est la longueur du mot *P*.

Le calcul des occurrences approximatives d'un mot dans un texte est possible en calculant la table des *C(e, d)*. La valeur de chaque entrée de cette table est calculée en utilisant l'algorithme suivant [HAM 02]:

1. $j \leftarrow \max(C(e - 1, d - 1) + 1, C(e - 1, d) + 1, C(e - 1, d + 1))$
2. **tant que** $t_{j+1} = p_{j-d+1}$ **faire**
3. j ← j + 1
4. **fin tant que**
5. $C(e, d) \leftarrow j$

Notons *L(e, d)* la ligne d'indice le plus grand, sur la diagonale *d* qui a comme valeur *e*. Le calcul des occurrences approximatives d'un mot dans un texte est possible aussi en calculant la table des *L(e, d)*. Avec cette méthode, si $L(e, d) = m$ alors il y a une occurrence de *P*, ayant *e* erreurs, se terminant en position $L(e, d) + d = m + d$ du texte.

Le calcul des valeurs de chaque entrée de la table des *L(e, d)* est effectué en utilisant l'algorithme suivant [HAM 02]:

1. $i \leftarrow \max(L(e - 1, d - 1) + 1, L(e - 1, d) + 1, L(e - 1, d + 1))$
2. **tant que** $p_{i+1} = t_{d+i+1}$ **faire**
3. $i \leftarrow i + 1$
4. **fin tant que**
5. $L(e, d) \leftarrow i$

Nous présentons dans cette section un algorithme basé sur le calcul de la table des *C(e, d)* et un autre basé sur le calcul de la table des *L(e, d)*.

2.5.2.3.1. Algorithme de Landau-Vishkin

Dans cet algorithme [LAN 88], les auteurs font un prétraitement du mot *P* de longueur *m* en calculant une table *Max_Length* de dimension $m \times m$ dont chaque case *(i, j)* de cette table contient la longueur du plus long préfixe commun à $p_{i+1} \ldots p_m$ et $p_{j+1} \ldots p_m$.

Pour calculer les occurrences approximatives, à *t* erreurs près, du mot *P* dans un texte *T*, l'algorithme procède en $n-m+t+1$ itérations. A chaque itération *i*, l'algorithme regarde s'il y a

une occurrence de *P*, ayant moins de *t* erreurs, commençant à la position *i+1* du texte en calculant les valeurs de *L(e, d)* pour $0 \leq e \leq t$ et $-t \leq d \leq t$, et en se servant du prétraitement du mot *P*.

En effet le prétraitement du mot *P* permet de diminuer le temps de calcul de la boucle **tant que** de l'algorithme précédent de calcul des valeurs de *L(e, d)*, en gardant certaines informations concernant des préfixes communs provenant d'une itération antérieure.

Supposons que l'on soit rendu au calcul de l'itération *i* de l'algorithme. A la fin de l'itération *i*-1, on a calculé, pour chaque position $x = 1, 2, \ldots, i$ du texte, le plus long préfixe de $t_x t_{x+1} \ldots t_n$ qui s'aligne, avec au plus *t* erreurs, avec un préfixe du mot *P*. Notons *j* la position la plus à droite dans le texte qu'on a réussi à aligner dans une itération précédente. On dénote *r* l'itération précédent *i* où cette position *j* a été atteinte.

A l'itération *r*, $r < i$, on a trouvé, un alignement entre $t_{r+1} \ldots t_j$ et un préfixe *p* de *P*, ayant au plus *t* erreurs. Cet alignement induit un alignement, ayant au plus *t* erreurs, entre $t_{i+1} \ldots t_j$ et un suffixe de *p*. Cet alignement est constitué d'au plus *t + 1* séquences d'identités séparées par les erreurs. On code l'alignement de la façon suivante:

- Si $t_{p+1} \ldots t_{p+f} = p_{c+1} \ldots p_{c+f}$ et que $t_{p+f+1} \neq p_{c+f+1}$, on dénote cette suite d'identités par le triplet *(p, c, f)*.
- Si dans l'alignement, il y a une erreur en position t_{h+1}, on la dénote par le triplet *(h, 0, 0)*.

L'alignement entre $t_{i+1} \ldots t_j$ et un suffixe de *p* peut alors être décrit par une séquence d'au plus *2t + 2* triplets. On dénote cette séquence $S_{i,j}$.

Le calcul des valeurs de *L(e, d)* pour $0 \leq e \leq t$ et $-t \leq d \leq t$ est effectué maintenant en se servant des séquences $S_{i,j}$.

Notons que la complexité en temps de cet algorithme est de $O(t^2 n)$ [HAM 02].

2.5.2.3.2. Algorithme de Galil-Park

L'idée de l'algorithme de Galil-Park [GAL 90] est semblable à celle de l'algorithme de Landau et Vishkin [LAN 88], mais cette fois-ci, on calcule la table des *C(e, d)* (et non la table des *L(e, d)*) correspondante à la table des distances.

Ici, on ne calcule plus les diagonales correspondant à une table $D^{(i)}$ pour chacune des itérations *i*, $0 \leq i \leq n - m + t$, comme Landau et Vishkin le faisaient, on calcule les valeurs de *C(e, d)* d'une seule table, la table $(m + 1) \times (n + 1)$ des distances correspondante à la recherche approximative de *P* dans *T*.

Dans leur algorithme, Galil et Park prétraitent le mot *P* de la même façon que dans Landau et Vishkin, c'est-à-dire qu'ils calculent une table de longueur de préfixes communs à tous les suffixes de *P*. Il nomme *Préfixe(i, j)* la longueur du plus long préfixe commun à $p_{i+1} \ldots p_m$ et $p_{j+1} \ldots p_m$. Ils se servent aussi de l'idée de triplets de références *(u, v, w)*, où *u* est la position de départ dans le texte, *v* la position finale dans le texte et *w* la *D*-diagonale où l'on se trouve.

Si on définit une *C*-diagonale *c* comme étant l'ensemble des *C(e, d)* telles que $e + d = c$, l'algorithme calcule la table des *C(e, d)*, *C-diagonale* par *C-diagonale*. Le calcul des *C(e, d)* se fait en utilisant la table *Préfixe* et les triplets de références en un temps *O(t)*, et comme il y a $n - m+k +1$ *C-diagonales* à calculer, la complexité en temps de cet algorithme est de *O(tn)* [HAM 02].

2.5.2.4. La méthode des 4 Russes

La méthode des 4 Russes [ARL 70] est une méthode développée en 1970 par quatre mathématiciens russes, Arlazarov, Dinic, Kronrod et Faradzev, permettant de calculer les entrées d'une matrice de dimension $n \times n$ en temps $O(n^2/\log n)$ en précalculant toutes les matrices correspondants à des problèmes plus petits (de dimension $l \times l$). Le nombre l est choisi assez petit et habituellement l divise parfaitement n. Ensuite, on subdivise la grande matrice $n \times n$ en sous-matrices de dimension $l \times l$ et on se sert des résultats pré-calculés pour trouver le résultat final.

2.5.2.4.1 Algorithme de Masek-Paterson pour l'alignement global

Masek et Paterson [MAS 80] ont l'idée de se servir de la méthode des 4 Russes pour résoudre le problème de l'alignement global de deux mots. Etant donné deux mots de longueur *n*, l'algorithme proposé commence par choisir un petit nombre *k*. Ensuite il pré-calcule toutes les sous-matrices possibles de dimension $(k + 1) \times (k + 1)$. Le calcul d'une de ces sous-matrices est déterminé par sa première ligne *L*, sa première colonne *C* et deux mots, m_1 et m_2, de longueur *k*. Pour *L*, *C*, m_1 et m_2 fixés, on garde en mémoire la dernière ligne et la dernière colonne de la sous-matrice des distances correspondante. L'algorithme termine par le calcul de la matrice des distances finale *D*, sous-matrices par sous-matrices, au lieu de cellule par cellule.

En effet, avec cette méthode on doit pré-calculer $n^2.3^{2k}.|\Sigma|^{2k}$ matrices de dimension $(k+1)\times(k+1)$. Chacune de ces matrice est calculée en temps $O(k^2)$. Cela est sans doute très long à faire. Il est donc nécessaire de réduire le nombre de sous-matrices à calculer. Pour ce faire, Masek et Paterson ont proposé de travailler avec des vecteurs de différences horizontales *(D(i, j)-D(i, j-1))* et de différences verticales *(D(i, j)-D(i-1, j))* au lieu de travailler avec des lignes et des colonnes pouvant apparaître dans la table *D*.

Etant donné un vecteur de différences horizontales, correspondant à la première ligne d'une sous-matrice de distances de dimension $(k+1)\times(k+1)$, un vecteur de différences verticales, correspondant à la première colonne de la même sous-matrice, et deux mots m_1 et m_2 sur un alphabet Σ, Masek et Paterson donnent un algorithme pour le calcul, en temps $O(k^2)$, du vecteur de différences horizontales, correspondant à la dernière ligne de la sous-matrice, et du vecteur de différences verticales, correspondant à la dernière colonne de la sous-matrice.

Avec le raffinement apporté à l'algorithme, et comme il y a 3^k vecteurs de différences possibles, le nombre de sous-matrices à pré-calculer est réduit à $3^{2k} . |\Sigma|^{2k}$ [HAM 02].

2.5.2.4.2. Algorithme de Wu, Manber et Myers pour la recherche approximative

En 1996, Wu, Manber et Myers [WU 96] ont publié un algorithme basé sur la méthode des 4 Russes, qui traite le problème de la recherche des occurrences approximatives, ayant au plus *t*

erreurs, d'un mot *P* dans un texte *T*. L'algorithme proposé se base sur une idée d'Ukkonen [UKK 85] qui consiste à faire un prétraitement du mot cherché *P* en construisant, à partir de celui-ci, un automate déterministe (voir section 2.5.2.2).

Rappelons que l'automate M_P, construit à partir de *P*, a comme états toutes les colonnes pouvant possiblement apparaître dans la table des distances *D*, lorsque le texte *T* varie. Ukkonen [UKK 85] a démontré que le nombre d'états de M_P est borné par 3^m, où *m* est la longueur du mot *P*.

On définit un vecteur de différences verticales Δv_j associé à la colonne *j* de la table des distances *D* par : $\Delta v_j[i] = \Delta v_{i,j} = D(i, j) - D(i - 1, j)$ pour $1 \leq i \leq m$. Les auteurs ont introduit une petite variation sur l'automate M_P de telle façon à le représenter en se servant des vecteurs de différences verticales Δv_j, au lieu des colonnes *j* de la table *D*, comme états de l'automate.

Si on définit le vecteur de différences horizontales, Δh_j, associé à la colonne *j* de la table des distances *D*, pour $1 \leq i \leq m$, par: $\Delta h_j[i] = \Delta h_{i,j} = D(i, j) - D(i, j - 1)$, on peut calculer Δv_j à partir de Δv_{j-1} et de Δh_j en utilisant la relation de récurrence suivante :

$$\Delta v_{i,j} = min \begin{cases} 1 \\ \Delta v_{i,j-1} - \Delta h_{i-1,j} + 1 \\ \delta(i,j) - \Delta h_{i-1,j} \end{cases}$$

Tel que $\delta(i, j) = 0$ si $x_i = y_j$ et 1 sinon et où $v_{i,0} = 1$, pour $1 \leq i \leq m$.

Et pour calculer les valeurs du vecteur Δh_j, on a :

$$\Delta h_{0,j} = 0, \ \forall j \text{ et}$$

$$\Delta h_{i,j} = \Delta_{hi-1,j} + \Delta v_{i,j} - \Delta v_{i,j-1}$$

Comme M_P peut contenir 3^m états, sa construction peut prendre un temps exponentiel en *m*. Pour remédier ce problème, les auteurs ont proposé d'utiliser la méthode des 4 Russes en simulant l'automate M_P à l'aide d'une combinaison d'automates plus petits.

Pour ce faire, chaque vecteur Δv_j est divisé en sous-vecteurs, appelés régions, de longueur *r*.

On dénote $\Delta v_j\langle q\rangle$ la $q^{ième}$ région du vecteur Δv_j. Chaque région peut être donc un des 3^r vecteurs de différences possibles.

L'idée consiste à construire un automate $M_p\langle q\rangle$, pour chaque région *q* de sorte qu'il soit possible de trouver $\Delta v_j\langle q\rangle$ à partir de $\Delta v_{j-1}\langle q\rangle$ en se déplaçant dans l'automate pendant la lecture du texte *T*. Au lieu de cela, on construit un seul automate *universel* qui servira pour chaque région. Cet automate universel est décrit par deux tables, l'une contient l'information sur les transitions de l'automate, et l'autre, l'information sur les sorties associées à ces transitions.

Maintenant, si P est de longueur m et r est la longueur des régions que l'on veut considérer alors, pour simuler l'automate M_P, on prend m/r copies de l'automate universel $M_p\langle q\rangle$, où q est n'importe quelle des régions de longueur r.

La complexité en temps de cet algorithme est de $O(nm)/\log n$ [HAM 02].

2.5.2.5 Une approche à la Boyer-Moore

L'algorithme de Boyer-Moore [BOY 77] est un algorithme de recherche exacte des mots dans un texte qui procède en deux étapes : une étape de comparaison et une étape de déplacement. L'algorithme commence par comparer le mot $P=p_1 \ldots p_m$ de droite à gauche avec le préfixe de longueur m du texte T. Dès que l'on trouve une erreur, on arrête et on se déplace le plus possible vers la droite, puis on recommence la comparaison.

En 1990, Tarhio et Ukkonen [TAR 90] ont proposé de généraliser le concept de l'algorithme de Boyer et Moore pour résoudre le problème de la recherche des occurrences approximatives d'un mot dans un texte.

Dans un premier temps, l'algorithme de Tarhio et Ukkonen trouve et marque les positions j du texte où il y a très probablement une occurrence approximative de P. Ensuite on calcule dans la table D, seulement les diagonales commençant par les entrées $D(0, j)$ marquées. Pour ce calcul, on suppose que les entrées à l'extérieur des diagonales marquées sont égales à ∞.

Afin de marquer les positions j du texte on procède comme suit : pour une diagonale d, on regarde si t_{d+i} est dans C_i, pour $i = m, m\text{-}1, \ldots, 1$. Tel que C_i est défini par $C_i = p_{i-t} \ldots p_{i+t}$, et t est le nombre d'erreurs permis. On arrête lorsque $t+1$ mauvaises colonnes ont été trouvées. Si le nombre de mauvaises colonnes est $\leq t$, alors on marque les diagonales $d\text{-}t, \ldots, d+t$, c'est-à-dire les entrées $D(0, d - t), \ldots, D(0, d + t)$ de la table D.

Pour trouver les mauvaises colonnes rapidement, on peut pré-calculer, $\forall a \in \Sigma$ et $\forall i$, $1 \leq i \leq m$,

$$Mauvaise(i,a) = \begin{cases} vrai & si\ a\ n'apparaît\ pas\ dans\ C_i \\ faux & sinon \end{cases}$$

Après l'analyse de ce qui se passe autour de la diagonale d, on doit se déplacer vers la droite et recommencer l'analyse pour une autre diagonale. Il est évident qu'on peut faire un déplacement d'au moins $t + 1$ diagonales et continuer l'analyse à la diagonale $d + t + 1$. On peut aussi permettre un décalage plus grand, en gardant en tête qu'un déplacement très grand risque d'oublier une occurrence approximative de P. Lorsqu'on arrive à la fin du texte, on a marqué toutes les diagonales pertinentes et on calcule dynamiquement ces diagonales.

La complexité en temps de l'algorithme est au pire de $O(nm/t)$, pour l'étape de comparaison et de déplacement. Pour le calcul des occurrences approximatives de P, il s'agit ensuite de calculer les valeurs de la table D pour les diagonales marquées à l'étape précédente en temps $O(Nm)$ [HAM 02], où N est le nombre de diagonales marquées.

2.5.2.6. Approches vectorielles

Un *algorithme vectoriel* est un algorithme qui trouve un vecteur de sortie en appliquant un nombre d'opérations sur le vecteur d'entrée, indépendant de la longueur du vecteur [HAM 02].

Si x et y sont des vecteurs ne comprenant que des valeurs booléennes alors ils sont appelés *vecteurs de bits* et $x \vee y$, $x \wedge y$, $\neg x$, dénotent les opérations logiques de disjonction, conjonction et négation respectivement.

Soit $x = x_1 \ldots x_m$. $\uparrow_a x = ax_1 \ldots x_{m-1}$ est appelé le *déplacement vers la droite* de x. Les valeurs du vecteur x ont été déplacées d'une position vers la droite et la première valeur du vecteur devient a.

Nous présentons dans cette section, quelques approches vectorielles à la recherche des occurrences approximatives d'un mot dans un texte.

2.5.2.6.1. Algorithme de Wu et Manber

Wu et Manber [WU 92] ont développé un algorithme vectoriel pour le problème général de recherche des occurrences approximatives en se basant sur un algorithme de recherche exacte proposé par Baeza, Yates et Gonnet [BAE 92].

L'idée de Baeza, Yates et Gonnet pour la recherche des occurrences exactes d'un mot $P=p_1 \ldots p_m$ de longueur m dans un texte $T = t_1 \ldots t_n$ est de travailler avec des vecteurs de bits de longueur m. Pour chaque position j dans le texte T, on définit un vecteur de bits R_j par :

$$R_j[i] = \begin{cases} 1 \text{ si le préfixe de longueur } i \text{ de } P \text{ est identique} \\ \qquad \text{au suffixe de longueur } i \text{ de } t_1 \ldots t_j \\ 0 \text{ sinon} \end{cases}$$

Dans cet algorithme, à chaque position j du texte, si $R_j\,[m] = 1$ alors il y a une occurrence exacte de P se terminant en position j du texte, et donc commençant en position $j - m + 1$ du texte.

Dans la modification de Wu et Manber, on calcule toujours pour chaque position j du texte, le vecteur R_j de la même façon que précédemment, mais on calcule aussi t nouveaux vecteurs, $R_j^1, R_j^2, \ldots, R_j^t$, où t est le nombre d'erreurs permis. Le vecteur R_j^d contient l'information sur les occurrences de préfixes de P, contenant au plus d erreurs, se terminant en position j du texte. Dénotons les vecteurs R_j du calcul des occurrences exactes par R_j^0. On a alors le résultat suivant:

Si $R_0^d = \overbrace{11\ldots1}^{d}\overbrace{00\ldots0}^{m-d}$, alors

$$R_{j+1}^d = \left(\uparrow_1 R_j^d \wedge t_{j+1}\right) \vee \uparrow_1 \left(R_j^{d-1} \vee R_{j+1}^{d-1}\right) \vee R_j^{d-1}$$

Le calcul de chacun des vecteurs R_j^d , 1 ≤ j ≤ n, peut être réalisé en temps *O(n)*. Comme *d* varie entre 0 et *t*, le calcul des occurrences approximatives de *P*, ayant au plus *t* erreurs, se fait en temps *O(tn)*.

2.5.2.6.2. Algorithme de Baeza-Yates-Gonnet

Cet algorithme a été publié en 1996 [BAE 96] pour résoudre le problème de recherche approximative d'un mot à *t* erreurs près dans un texte en se basant sur la simulation d'un automate fini non-déterministe, diagonale par diagonale. Simuler l'automate par diagonales permet de trouver les nouveaux états actifs en parallèle, en utilisant des vecteurs de bits.

En général, l'automate a *(m + 1)(t + 1)* états. On assigne le couple *(i, j)* à l'état situé à l'intersection de la ligne *i* et de la colonne *j*, pour $0 \leq i \leq t$ et $0 \leq j \leq m$. Au départ, les états actifs de la ligne *i* sont les états des colonnes de 0 à *i* pour indiquer le fait qu'on peut commencer par supprimer les *i* premiers caractères du mot *P* avant de continuer l'alignement.

Soit A_k la matrice booléenne, ou matrice de bits, correspondante aux états actifs de notre automate non-déterministe après la lecture du préfixe $t_1 \ldots t_k$ de *T*, définie par :

$$A_k(i,j) = \begin{cases} 1 \; si \; l'état(i,j) est \; actif, après \; la \; lecture \; de \; t_1 \ldots t_k \\ 0 \; sinon \end{cases}$$

L'idée de Baeza, Yates et Gonnet est de calculer la matrice A_k, diagonale par diagonale, à partir des valeurs de la matrice A_{k-1}. En fait, ils ne calculent pas toute la matrice A_k mais seulement les éléments correspondants aux diagonales de longueur *t + 1* de l'automate.

On peut représenter chaque diagonale *j*, $0 \leq j \leq m - t$, par un nombre $D_k[j]$ qui représente la ligne du premier état actif de la diagonale *j*, après la lecture de $t_1 \ldots t_k$. On peut donc représenter la simulation de l'automate, à tout moment de la lecture de *T*, par *m - t + 1* nombres compris entre 0 et *t + 1*. On peut calculer les valeurs de $D_k[j]$, $0 \leq j \leq m - t$, à partir des valeurs $D_{k-1}[j]$ par les équations suivantes :

$D_k[0] = 0$

$D_k[j] = min(D_{k-1}[j] + 1, D_{k-1}[j + 1] + 1, g(D_{k-1}[j - 1], t_k))$,

où $g(D_k[j], c) = min(\{t + 1\} \cup \{i \mid i \geq D_k[j] \wedge p_{i+j} = c\})$.

Avec ces $D_k[j]$, on a une occurrence de *P*, ayant au plus *t* erreurs, se terminant en position *k* du texte si $D_k[m - t] \leq t$.

Pour pouvoir calculer tous les $D_k[j]$ en parallèle, Baeza, Yates et Gonnet donnent une représentation de chaque valeur $D_k[j]$ par un vecteur de bits de longueur *t + 1*:

$$D_k[j] = \overbrace{0 \ldots 0}^{t+1-D_k[j]} \overbrace{1 \ldots 1}^{D_k[j]},$$

Ils représentent ensuite l'état de la simulation de l'automate, après la lecture de $t_1 \ldots t_k$ par un seul grand vecteur de bits, comprenant les valeurs de D_j pour $1 \leq j \leq m - t$:

$D_k = 0D_k[1]0D_k[2]0 \ldots 0D_k[m- t]$.

2.6. Les arbres et les automates de suffixes (DAWG)

2.6.1. Arbre de suffixes

2.6.1.1. Définition

Les arbres de suffixes [WEI 73] sont des structures de données largement utilisées pour le traitement de texte. L'arbre de suffixes de la chaîne *S* prise d'un alphabet Σ est une structure de données arborescente, dont les arêtes sont marquées par les éléments de Σ, de manière à ce que la concaténation des marques sur une branche soit un suffixe de *S*. Toutes les feuilles d'un arbre de suffixes représentent de manière unique les suffixes de *S*, la racine contient la chaîne vide et chaque nœud interne représente une sous-chaîne unique de *S*. Toutes les sous-chaînes de *S* peuvent être trouvées en traversant le chemin à partir de la racine. En général, on termine la chaîne *S* par un caractère spécial *$* (non présent dans le reste de *S*), pour éviter que certains suffixes se terminent sur des nœuds de l'arbre.

La figure 2.3 montre un arbre de suffixes pour le mot *BANANAS*. Deux faits sont à souligner. Premièrement, démarrant de la racine, chacun des suffixes de *BANANAS* est trouvé dans l'arbre, en commençant par *BANANAS*, *ANANAS*, *NANAS*, et finissant d'un *S* solitaire. Deuxièmement, si on a un texte de longueur *n*, et une chaîne à rechercher de longueur *m*, une recherche exhaustive fera $n \times m$ comparaisons. Les techniques optimisées, telle que l'algorithme de Boyer-Moore [BOY 77], peuvent garantir des recherches qui requièrent $m+n$ comparaisons, comme performance moyenne. Cependant, l'arbre de suffixes démolis cette performance en ne faisant que *m* comparaisons quelque soit la taille du texte.

La raison qui a fait que l'utilisation des arbres de suffixes est limitée, est que sa construction requière $O(n^2)$ temps de calcul et espace mémoire. Cette performance quadratique exclue l'emploi des arbres de suffixes en recherche dans de larges collections de texte.

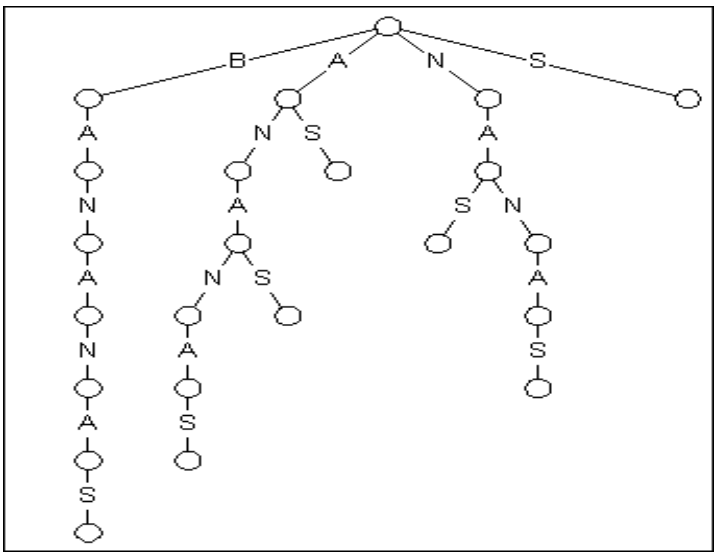


Figure 2.3: Arbre de suffixes du mot "BANANAS".

2.6.1.2. Arbre compact de suffixes

Les arbres compacts de suffixes ont été proposés par McCreight [MCC 76] pour améliorer l'espace d'utilisation. L'arbre compact de suffixes garde la même topologie qu'un arbre de suffixes, mais par contre, élimine les nœuds qui possèdent un seul descendant. Ce processus, connu sous le nom de *compression de chemins*, signifie que les bords individuels dans l'arbre représentent maintenant des séquences de textes au lieu simplement des caractères. La figure 2.4 montre un arbre compact de suffixes de la chaîne « *BANANAS* ».

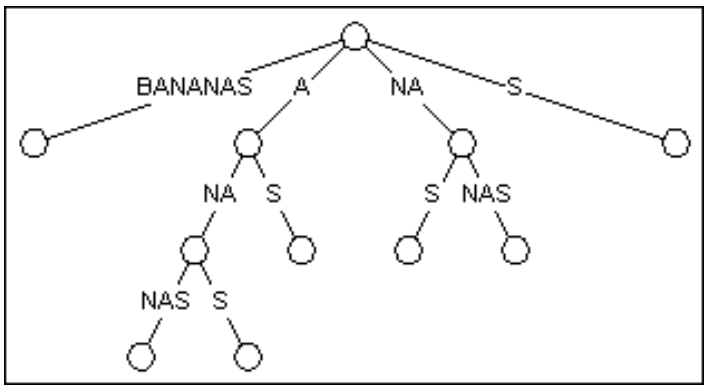


Figure 2.4: Arbre compact de suffixes de la chaîne "BANANAS"

Cette transformation réduit le temps et l'espace requis de $O(n^2)$ à $O(n)$. Dans le pire cas, un arbre compact de suffixes peut être construit avec au maximum $2n$ nœuds, où n est la longueur du texte.

L'algorithme original de McCreight construit l'arbre compact de suffixes dans l'ordre inverse, c'est-à-dire que les caractères sont ajoutés à partir de la fin du texte ce qui exclue la possibilité d'un traitement enligne et ainsi réduire les domaines d'application. Ce n'est que vingt ans plus tard qu'Ukkonen [UKK 95] proposa une version modifiée de cet algorithme qui fonctionne de gauche à droite.

2.6.1.3. Utilisation de l'arbre de suffixes pour la recherche de motifs

L'arbre de suffixes est très utilisé comme structure d'indexation et de recherche, il permet de rechercher un motif *P*, de longueur *m*, dans un texte de longueur *n* en temps $O(m)$. Ceci réduit considérablement le temps de recherche, en effet la recherche prend un temps qui dépend uniquement de la longueur du motif.

La recherche du motif *P* est relativement simple. En partant de la racine de l'arbre, il faut suivre la branche dont l'étiquette commence comme *P*. En suivant cette branche on arrive alors à un nouveau nœud et on recommence le processus sur la partie restante de *P*. Le processus est répété jusqu'à :

- Ne plus pouvoir descendre dans l'arbre par la lettre lue, il n'y a pas d'occurrence de *P* dans le texte,
- Avoir lu *P* en entier, on arrive :
  - Sur un nœud de l'arbre ou au milieu d'une branche. Le nombre de feuilles présentes dans le sous-arbre correspond au nombre d'occurrences de *P* dans le texte. De plus

chaque feuille contenant la position du début de suffixe, on peut déterminer la position de chacune des occurrences.

- Sur une feuille. Il n'y a qu'une seule occurrence de *P* dans le texte, elle se trouve à la position indiquée par le numéro de la feuille.

2.6.2. Graphe acyclique déterministe de mot (DAWG)

Le DAWG (Deterministic Acyclic Word Graph) ou bien automate de suffixes [CRO 86] construit sur une chaîne de caractères *S* de longueur *m* est un automate déterministe capable de reconnaitre toutes les sous-chaînes de *S* (voire la figure 2.5). Comme chaque nœud dans l'arbre de suffixes correspond à une sous-chaîne, le DAWG n'est rien d'autre qu'un arbre de suffixes augmenté par des *liens d'échec* pour les lettres qui n'existent pas dans l'arbre. Les DAWGs ont des applications similaires à celles des arbres de suffixes, et nécessitent aussi *O(m)* espace et temps de construction.

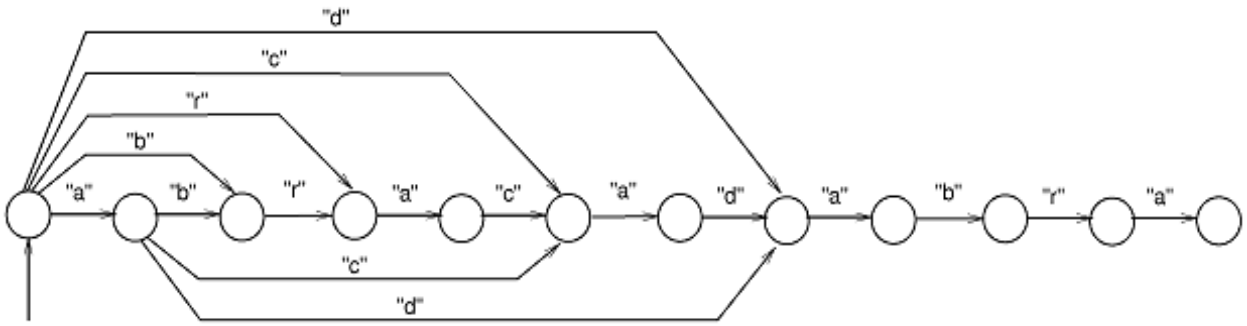


Figure 2.5: DAWG de la chaîne "abracadabra".

Notons que l'automate de suffixes d'un mot *P* de longueur *n* a au plus *2n* états. Il est montré que le nombre de ses transitions est au plus égal à *3n* [PER 89]. Plusieurs algorithmes ont été proposés pour la construction de l'automate de suffixes, citons [CRO 86] et [INE 01] et dont le calcul nécessite généralement un temps *O(n)*. Cette construction est voisine de celle de l'arbre de suffixes de [WEI 73] et [MCC 76].

Le DAWG ou bien l'automate de suffixes peut être utilisé pour la recherche du mot *P* dans un texte *T* de la façon suivante. Au fur et à mesure de la lecture du texte *T*, on tient à jour deux variables *(p, i)*. La première est l'état atteint dans l'automate et la deuxième est la longueur du plus long facteur de *P* qui est un suffixe du mot qui a été lu. L'entier *i* peut être calculé simplement en fonction des transitions de l'automate.

2.6.3. Recherche approximative avec les arbres de suffixes

Les arbres de suffixes peuvent être utilisés pour résoudre le problème de recherche des occurrences approximatives d'un mot *P* dans un texte *T*.

Landau et Vishkin [LAN 89] ont proposé un algorithme basé sur les arbres de suffixes permettant de trouver les occurrences approximatives d'un mot dans un texte. Cet algorithme se sert encore du calcul des *L-diagonales* (présenté dans la section 2.5.2.3) mais, cette fois-ci, l'algorithme se sert d'un arbre de suffixes pour calculer les *L(e, d)* en temps constant.

L'algorithme procède en deux étapes comme suit:

1. Dans la première étape, on construit l'arbre de suffixes de la séquence *S* obtenue de la concaténation du texte *T*, du mot *P* cherché et du caractère \$. La construction de l'arbre se fait en temps *O(n +m)*.
   Par exemple, si le mot *P = AACG* et le texte *T = GCGTTGCAGGAACG* alors, l'arbre de suffixes de la séquence *S = GCGTTGCAGGAACGAACG\$* est donnée par la figure 2.6:

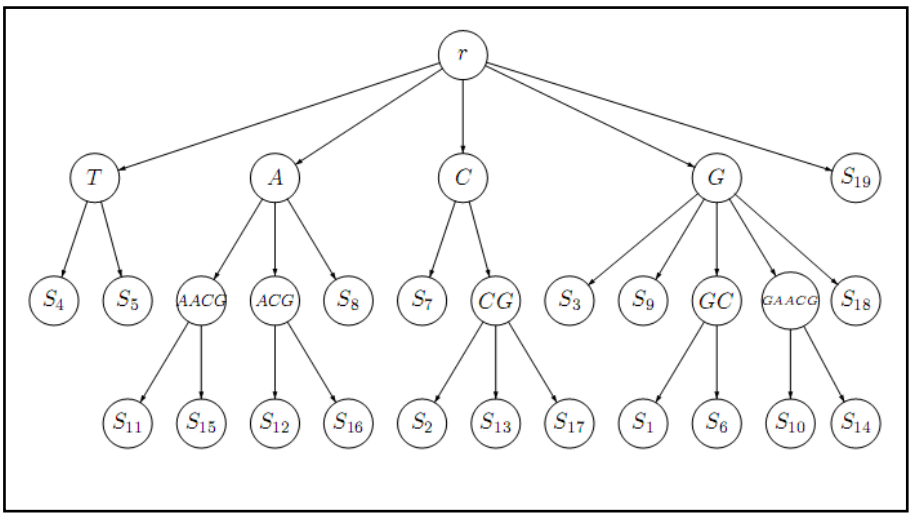


Figure 2.6: Un arbre de suffixes

2. Dans la deuxième étape, on trouve les occurrences approximatives de *P* dans *T*, ayant au plus *t* erreurs, en calculant les *L-diagonales* correspondant à la table des distances *D*.

Si on dénote *LCA(i, d)* le plus petit ancêtre commun des feuilles représentant le suffixe $p_{i+1}...p_m$ de *P* et le suffixe $t_{i+d+1}...t_n$ de *T*, l'algorithme de calcul des *L(e, d)* devient:

1. $i \leftarrow \max(L(e-1, d-1) + 1, L(e-1, d) + 1, L(e-1, d+1))$
2. $q \leftarrow$ **longueur** $(LCA(i, d))$
3. $L(e, d) \leftarrow i + q$

La complexité en temps de cet algorithme est de *O(tn)*, étant donné que l'on doit calculer *O(n) L-diagonales* et que chacune de ces *L-diagonales* comprend *t* valeurs *L(e, d)*.

2.7. Conclusion

La recherche des occurrences de mots dans un texte est un problème important dans plusieurs domaines, en informatique comme dans d'autres branches. Lorsque le texte ou le mot à chercher ne soit pas exacte, les méthodes de recherche classiques se montrent inefficaces, et on a donc recours à une recherche approximative. Nous pouvons trouver dans la littérature plusieurs mesures de similarité et distances entre les chaînes de caractères dont certainement la distance la plus connue est la distance d'édition introduite la première fois par Levenshtein [LEV 66]. Malheureusement le calcul de cette distance est coûteux en temps et en espace. C'est pour cette raison que plusieurs approches ont été proposées pour résoudre ce problème.

Dans ce chapitre nous avons considéré le problème général de l'appariement de chaînes de caractères et la recherche des occurrences approximatives d'un mot dans un texte. Nous avons donné tout d'abord quelques définitions et notations nécessaires. Ensuite, nous avons passé à l'appariement de chaînes et à la recherche exacte de motifs. Nous avons abordé par la suite le problème de comparaison approximative de chaînes ainsi que les mesures de similarité entre les mots tout en présentant quelques mesures les plus connues. La recherche approximative est étudiée dans la section 5, dans laquelle nous avons présenté plusieurs approches de recherche approximative. La dernière section a été consacrée aux arbres et aux automates de suffixes comme des outils rapides et efficaces de recherche de mots dans un texte.

Chapitre 3 : Conception

3.1. Introduction

D'importants fonds documentaires existent actuellement dans les bibliothèques, musées et autres institutions à caractères pédagogiques ou sociopolitiques. Les documents historiques des civilisations anciennes et les archives nationales sont l'exemple typique de telles richesses qui représentent le patrimoine, l'histoire et la dignité des nations. En effet, les documents anciens sont fragilisés par le temps et lorsqu'ils ne se détériorent pas naturellement, souffrent d'être trop souvent manipulés pour consultation. Afin de surmonter ce problème, la numérisation a été mise en place. Cependant, le processus de numérisation génère souvent des images de mauvaise qualité, à fond hétérogène, avec effet de transparence, etc. et ce malgré les avancées spectaculaires de la technologie des outils de capture numérique. En effet, la numérisation permet de préserver les documents originaux, mais elle ne facilite pas leur accès pour consultation et recherche.

Pour accéder au contenu des documents, deux approches sont possibles. La première approche dite analytique concerne le développement d'algorithmes et méthodes permettant de reconnaître le contenu des documents et de procurer une transcription textuelle ou une annotation du contenu. La deuxième approche dite holistique quant-à-elle permettra à l'utilisateur de rechercher, visualiser, et naviguer des bases d'images de documents sans recourir à une identification complète du contenu. Dans telles approches le mot peut être représenté par des caractéristiques de bas niveau, de niveau intermédiaire ou de haut niveau. Parmi les caractéristiques de bas niveau on trouve les profils de mots, les directions des traits, …etc. les caractéristiques intermédiaires sont : les bords, concavités, convexités,…etc. tandis que les caractéristiques de haut niveau incluent les hampes, jambages, …etc.

En effet, l'OCR ne peut être appliquée que sur des documents imprimés ou des documents manuscrits avec un lexique limité. Dés que les documents soient dégradés, complexes du point de vue structure et disposition spatiale du contenu textuel et graphique et avec un lexique plus large, l'OCR devient inefficace. Dans le cas des anciens documents arabes qui sont le cœur de notre travail, la difficulté augmente à cause des caractéristiques morphologiques de l'écriture arabe qui compliquent de plus la tâche de l'OCR à différents niveaux de traitement.

Nous devons indiquer ici que le but ultime des systèmes de recherche d'images de documents n'est pas de reconnaître les motifs, textuels ou graphiques dans les documents, mais de trouver les documents originaux correspondant à un besoin de l'utilisateur. Afin de s'occuper de l'incapacité des techniques de reconnaissance à comprendre le contenu des documents anciens et en gardant en tête le but des systèmes de recherche d'images de documents, nous proposons une approche permettant la recherche d'images de documents arabes anciens sans recourir à une reconnaissance du contenu.

Le présent chapitre est consacré à la présentation de l'approche proposée. La suite de ce chapitre est organisée de la manière suivante : tout d'abord, nous présentons les caractéristiques des anciens documents arabes pour montrer la complexité de traitement de ce genre de documents. Par la suite, nous donnons un petit état de l'art sur la recherche d'images

de documents. Dans le reste du chapitre, nous décrivons la méthodologie développée pour la conception du système proposé, Avant de conclure.

3.2. Caractéristiques des documents arabes anciens

Puisque dans notre travail, nous nous intéressons aux anciens documents arabes, il nous semble intéressant de présenter même brièvement les caractéristiques de ce genre de documents, afin de montrer la difficulté et la complexité de leur traitement. Nous divisons cette section en deux parties: une présente les caractéristiques des documents anciens en général, et l'autre présente les caractéristiques de l'écriture arabe.

3.2.1. Caractéristiques des documents anciens

Les documents anciens possèdent plusieurs caractéristiques qui viennent de leurs natures, leurs états de stockage…etc. et qui rendent les algorithmes classiques de traitement d'images non fiables. Les caractéristiques des documents anciens peuvent être classées en trois groupes comme suit :

1- Des caractéristiques de dégradation (tâches, trous, traces d'humidité, dégradation de l'encre et traits fins, effet de transparence, présence de plis et de déchirures, etc.), voir la figure 3.1 :

Figure 3.1: Exemples de dégradations des documents anciens

2- Des caractéristiques de défauts de numérisation : défauts humains ou bien techniques, on peut citer : défauts de courbure, de lumière…

3- Des caractéristiques liées à la nature des documents (la page peut contenir des zones graphiques de différentes tailles et des lettrines, faible séparation entre les blocs de texte, grande variabilité entre les styles d'écriture, espace inter-mots irrégulier, variation de la couleur du papier…etc.). La figure 3.2 montre quelques exemples de ce type de caractéristiques.

Figure 3.2: Exemples de caractéristiques liées à la nature des documents

3.2.2. Caractéristiques de l'écriture arabe

La langue arabe a vu ses racines naître dans la péninsule Arabique et les premières traces de l'écriture arabe, telle qu'on la connaît ne remontent qu'au VI^{ème} siècle. L'arabe appartient à la famille des langues sémitiques comme l'hébreu et l'araméen [HAO 98].

L'écriture arabe imprimée ou manuscrite possède des caractéristiques différentes d'autres langues en structure et en mode de liaison entre les caractères formant un mot. Parmi ces caractéristiques on peut citer :

- La présence d'une ligne de base horizontale dite ligne de référence ou d'écriture. C'est le lieu de ligatures horizontales des caractères d'une même chaîne ;
- Les caractères arabes s'écrivent cursivement de la droite vers la gauche, aussi bien dans le cas de l'imprimé que du manuscrit ;
- L'arabe contient 28 caractères de base tels qu'ils sont illustrés dans la figure 3.3, dont 16 incluent dans leurs formes des points diacritiques qui peuvent être au nombre de 1, 2 ou 3. Ces points font la différence entre les caractères ayant un corps identique, ils peuvent être situés au dessus ou au dessous du corps du caractère de base.

أ ب ت ث ج ح
خ د ذ ر ز س
ش ص ض ط ظ ع
غ ف ق ك ل م
ن ه و ي

Figure 3.3: Les 28 lettres de l'alphabet arabe

- La forme d'une lettre écrite dépend de son contexte, elle diffère selon que le caractère apparaît en position initiale, médiane ou isolée dans une chaîne de caractères.
- L'écriture arabe est par nature semi-cursive, ce qui rend la phase de segmentation une opération cruciale.

- Les caractères arabes peuvent être voyellés. Les voyelles peuvent se placer au dessus ou au dessous du caractère.
- Les caractères arabes ne possèdent pas une taille fixe (hauteur et largeur). Leur taille varie d'un caractère à un autre et d'une forme à une autre pour un même caractère.
- Dans certaines fontes, plusieurs caractères peuvent être écrits de façon combinée. Les combinaisons ou ligatures verticales sont utilisées pour des raisons d'esthétique.
- L'écriture arabe est une écriture calligraphique, elle varie selon les milieux et les régions, d'une extrême simplicité formelle à la complexité exhaustive de l'arabesque. La figure 3.4 montre un exemple de différents styles graphiques de l'écriture arabe.

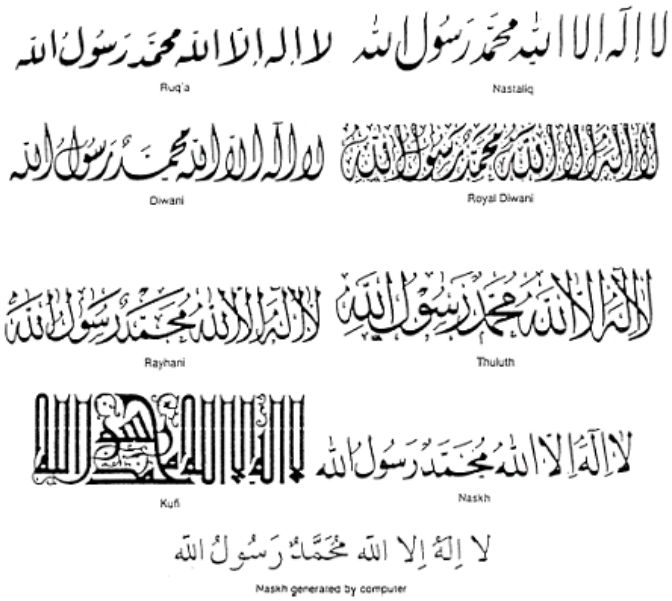


Figure 3.4: Différents styles et fontes pour l'écriture arabe

3.3. Recherche d'images de documents : Etat de l'art

La recherche de mots dans les documents latins a suscité récemment une attention considérable. De nombreux travaux ont été effectués sur la recherche de mots par *word spotting* ainsi que par reconnaissance des caractères dans les images de documents, mais malgré ce nombre important de travaux, les résultats obtenus jusqu'à maintenant ne sont pas suffisants pour traiter des volumes de données importants [KHU 08].

Le travail de Spitz semble être l'un des plus proches de notre investigation. Spitz dans [SPI 95] a proposé de coder les caractères du texte imprimé selon leurs formes. La méthode commence par l'extraction des différents mots du texte. Pour chaque mot détecté, elle extrait les caractéristiques de ses caractères en se basant sur le nombre de composantes connexes dans chaque caractère et sur leur position par rapport aux deux lignes de base. Les caractères sont ensuite codés selon leurs caractéristiques. Par exemple, les lettres contenant des hampes sont codées par un 'A', celles ayant des jambages par un 'g', etc. La séquence de codes obtenue compose un *jeton de forme de mot* (WST pour *word shape token*). Ainsi, le mot « Goods » peut être codé par le WST *AxxAx*. Les requêtes cherchées sont codées de la même façon en WSTs et la recherche peut être effectuée comme d'habitude, mais en utilisant les

WSTs à la place des mots. Notons que cette méthode n'identifie pas les caractères, elle détermine seulement leurs formes générales. L'utilisation de cette méthode pour la recherche des documents de la base de données anglaise de l'université de Washington [PHI 93] donne un rappel de 72% et une précision de 82%. Après, Smeaton et Spitz [SME 97] ont montré que cette technique n'est utile que dans le cas où les images sont de mauvaise qualité ou de faible résolution impliquant un échec de l'OCR. Il est noté que la recherche basée sur les WSTs est parfois moins performante que la recherche traditionnelle basée sur les mots même pour une mauvaise qualité de la sortie de l'OCR.

Chen et al. [CHE 98c] ont proposé une approche sans segmentation basée sur l'information des formes des mots au lieu des caractères. Tout d'abord, ils identifient les contours supérieur et inférieur de chaque mot en utilisant la morphologie mathématique. A partir de ces contours, ils extraient l'information de la forme en se basant sur l'emplacement des pixels. Ensuite, le décodage de Viterbi de la forme de mot codée est utilisé pour apparier l'image de mot avec le mot-clé cherché.

La recherche de mots est souvent plus difficile pour les textes manuscrits. Un des premiers travaux sur la recherche des documents manuscrits est celui de Manmatha et al. [MAN 96] dans lequel les auteurs ont proposé une technique semi-automatique pour indexer les documents manuscrits. L'approche commence par la réduction de l'image de document par un filtrage gaussien, et l'image résultante est binarisée puis segmentée en mots. Ensuite les images de mots similaires sont regroupées en classes d'équivalences. Pour ce faire, chaque image de mot est appariée avec les autres images qui possèdent presque la même taille en utilisant deux algorithmes de comparaison. Les classes d'équivalence contenant un grand nombre de mots sont éliminées puisqu'elles représentent des mots de jonction comme par exemple *and*, *or*, *the*, etc. et les classes restantes sont codées manuellement en ASCII et utilisées comme des index.

Plusieurs travaux ont été effectués aussi sur la recherche de documents historiques. [CAM 04] par exemple propose une approche pour la recherche dans les fiches d'incorporation militaire du XIX^{ème} siècle. En se basant sur la connaissance à priori de la structure de ces documents, l'idée de cette approche est d'indexer les formulaires anciens automatiquement par une chaîne ordonnée de graphèmes associée à la case du patronyme manuscrit, et de transformer également la requête alphabétique de l'utilisateur en une chaîne de graphèmes. La première étape de cette approche consiste à localiser la structure du document en utilisant la méthode DMOS (voir [COU 02]). Ensuite, elle procède une binarisation, extraction du squelette, et traitement des intersections pour extraire les graphèmes. Pour rechercher les images qui correspondent à un certain patronyme, l'utilisateur formule sa requête sous forme d'une chaîne de caractères. La requête est transformée ensuite en une chaîne de graphèmes et comparée avec les index extraits précédemment à partir des documents en utilisant la distance d'édition [WAG 74].

Dans [PUJ 02], des manuscrits en langue Telugu ont été caractérisés avec des représentations par ondelettes des mots. La représentation par ondelettes fournit des informations sur le contenu de l'image à différentes échelles. Elles exploitent les caractéristiques inhérentes des

caractères du Telugu. L'application de cette représentation par ondelettes sur les caractères latins ne donne pas de bons résultats [PUJ 02].

Rath et Manmatha [RAT 07] [RAT 03] ont présenté une approche holistique de recherche de mots appliquée sur des documents manuscrits historiques. Cette approche implique de grouper les images de mots dans des groupes semblables en utilisant l'appariement d'images de mots. Ensuite elle cherche l'ensemble de groupes les plus représentatifs et les étiqueter. Chaque groupe étiqueté sert d'index de recherche. Dans [RAT 07], les auteurs proposent quatre caractéristiques de profil pour les images de mots qui sont alors appariées en utilisant différentes méthodes. [RAT 03] a employé les correspondances entre les points anguleux pour classer les images de mots par similitude dans des manuscrits historiques. Le détecteur de points anguleux de Harris est employé dans les images de mots. Des correspondances entre ces points sont établies en comparant des fenêtres locales. La similarité entre les mots est donnée par la distance euclidienne entre les points mis en correspondance.

Motivés par le travail de Rath et Manmatha [RAT 07], Adamek et al. [ADA 07] ont proposé une méthode qui consiste à comparer les contours de mots au lieu des profils entiers, pour la recherche dans des manuscrits historiques. La méthode proposée commence par la binarisation à seuillage dynamique de l'image de document et ensuite elle opère une segmentation multi-échelles afin de détecter les contours des mots. La recherche d'un mot dans un document se traduit donc par une comparaison entre les contours des mots. L'application de cette méthode pour la recherche dans la collection de George Washington donne une précision de 83% [ADA 07].

Un autre travail récent sur l'indexation et la recherche des anciens documents est celui de Ramel et al. [RAM 07]. Dans ce travail, les auteurs ont proposé une approche de recherche dans des documents du Centre d'Etude Supérieur de la Renaissance (du 14ème au 17^{éme} siècle) sans reconnaissance préalable du modèle de documents. Ils ont fait une étude des caractéristiques structurelles de ces documents. Ensuite, ils ont appliqué une analyse hybride qui profite simultanément des avantages des méthodes d'analyse ascendantes et descendantes. Bien que cette méthode donne un taux de reconnaissance élevé, elle est lente et sensible aux inclinaisons [BEN 09].

Dans [KHU 08], les auteurs s'intéressent à la recherche de mots dans des images de documents imprimés anciens dont la requête correspond à une image d'un mot. Contrairement au [RAT 07], les auteurs ont proposé ici de travailler avec les caractères et non pas avec les mots. La méthode commence par la binarisation de l'image de document en utilisant l'algorithme NICK [KHU 09]. Ensuite, elle sépare le texte des parties graphiques de l'image et segmente le texte en mots en utilisant l'algorithme *RLSA*[9] [WAN 82]. Les mots correspondent aux composantes connexes de l'image obtenue après traitement par *RLSA*. Pour chaque mot détecté, les caractères qui le composent sont trouvés en revenant aux composantes connexes de l'image binarisée. On obtient alors les caractères sur lesquels un ensemble de caractéristiques seront extraites. Les auteurs utilisent six vecteurs de caractéristiques pour

[9] *Run length smoothing algorithm*

représenter les images de caractères. Les caractéristiques extraites à partir d'une image de document sont ensuite stockées dans un fichier d'index correspondant. Lors de la recherche, la requête est traitée de manière analogue aux documents et les caractéristiques des caractères de la requête sont appariées avec les caractéristiques des caractères des mots déjà stockées dans les fichiers d'index en utilisant un algorithme de type DTW. Selon les auteurs, cette méthode atteint une précision de 95% et un rappel de 89%.

Bai et al. [BAI 09] ont proposés une approche rapide pour la recherche de mots dans les images de documents. La première étape de cette approche est le prétraitement de l'image de document, qui sera suivi par une segmentation afin d'extraire les mots. Pour chaque mot détecté, on extrait un ensemble de 7 caractéristiques. En se basant sur la position et le type des caractéristiques, chaque mot est codé par une séquence de codes. Dans la recherche, la requête formulée est codée de la même manière en une séquence de codes. Ensuite, on calcule un degré de similarité entre le code de la requête et le code de chaque mot dans chaque document, en utilisant l'algorithme DTW.

Pour la recherche des anciens documents arabes, nous pouvons citer le travail de Benmohamed et al. [BEN 09] dans lequel les auteurs ont proposé une approche semi-automatique d'indexation et de recherche de documents. L'aspect manuel dans cette méthode réside dans le choix des index, tandis que l'aspect automatique est présent dans leur paramétrage et stockage. Après le choix des index manuellement, la méthode détecte leurs contours puis paramètre ces contours en les représentant sous forme de « chaîne de codes ». Cette dernière sera affinée afin de diminuer sa taille, et enfin rangée dans une table d'index. Lors de la recherche, le mot cherché est codé de la même manière procédée lors de l'indexation, et le code optimal obtenu est comparé par la suite avec les codes optimisés des index rangés dans la table à l'aide de l'algorithme DTW. L'application de cette méthode sur un échantillon de 1200 enveloppes, avec le choix du nom de ville comme l'information la plus pertinente, donne un taux de reconnaissance de 90% [BEN 09].

3.4. Architecture du système proposé

L'objectif que nous nous sommes assignés s'articule autour du développement d'un système de recherche de mots dans des anciens documents arabes manuscrits sans recourir à une reconnaissance du contenu.

Notre idée de base consiste à transposer le problème d'indexation et de recherche d'images de documents du domaine de l'analyse de documents au domaine de la recherche d'information. Ainsi, on peut allier représentation symbolique et représentation sémique et exploiter les techniques issues des deux domaines de recherche.

Chaque document de notre collection est tout d'abord représenté par une signature qui est constituée de l'ensemble d'informations pertinentes qu'on peut extraire de l'image de document. Cette description (signature) contient les caractéristiques de l'écriture et elle est codée en ASCII ce qui nous permettra d'employer facilement les techniques les plus

aboutissantes de recherche d'information à savoir les techniques de recherche approximative et les arbres de suffixes.

Afin de réaliser notre système nous procédons en deux phases : une phase de traitement et d'analyse d'images de documents et une phase de recherche dont laquelle on exploite les résultats de la phase précédente pour répondre à une requête de l'utilisateur.

3.4.1. La première phase : Analyse de documents

Cette phase a pour objectif d'analyser les images composant notre base afin d'en extraire leurs caractéristiques. Elle commence par l'acquisition des documents (images d'anciens manuscrits arabes), les prétraiter, les segmenter en lignes et en sous-mots, et enfin l'extraction de leurs caractéristiques. Ces dernières seront codées par la suite et enregistrées pour former une base de fichiers de codes. La recherche se fera dans cette base de codes et non pas dans les documents eux-mêmes. Les différentes étapes de cette phase sont illustrées par la figure 3.5. Notons que le temps de traitement n'est pas important ici parce que les traitements sont effectués hors ligne.

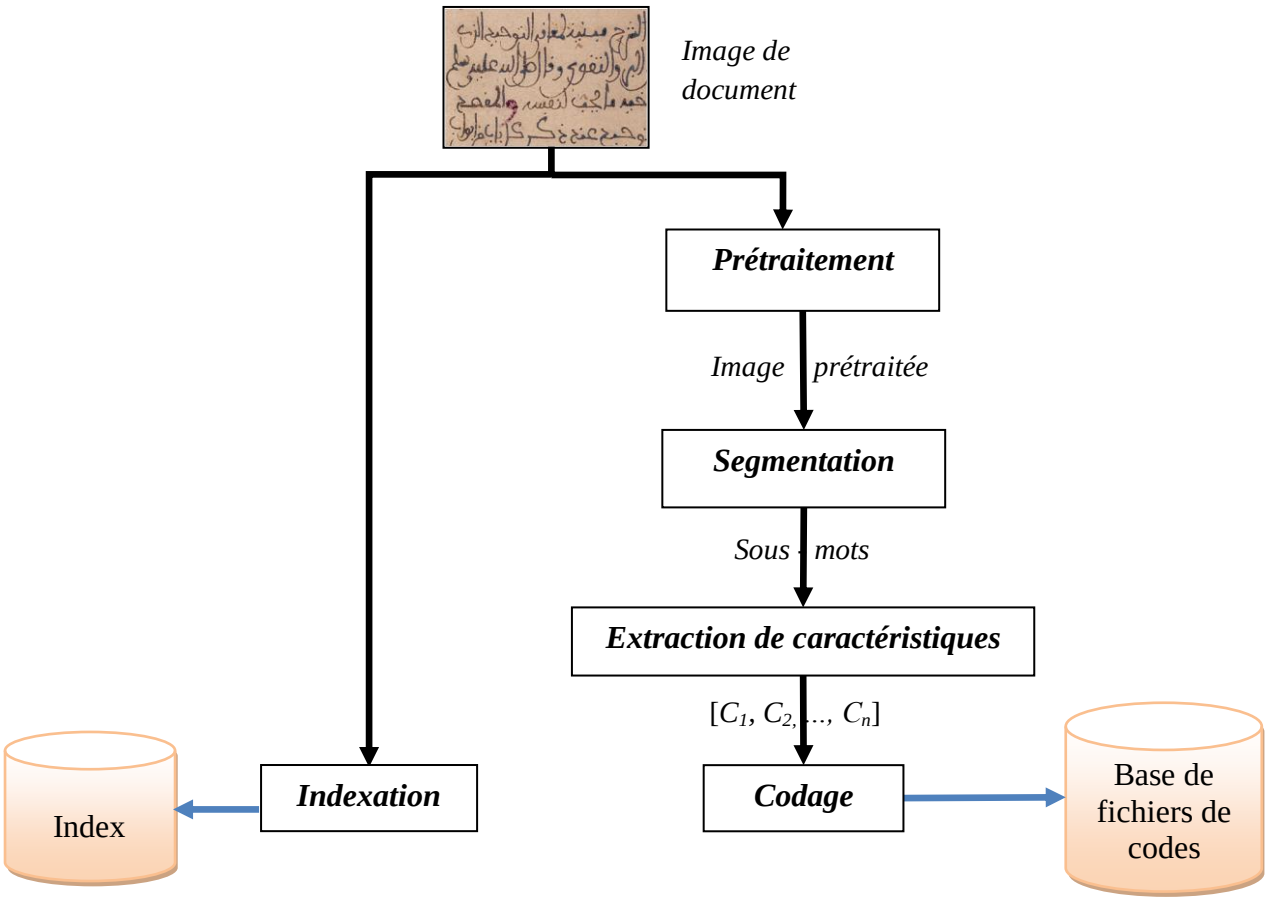


Figure 3.5: Schéma général de la phase d'analyse

3.4.1.1. Prétraitement

Le rôle du prétraitement est de préparer l'image de document aux traitements suivants. Il s'agit essentiellement de réduire le bruit superposé aux données et ne garder que l'information significative de la forme représentée. Dans notre système, le prétraitement regroupe : la transformation en niveaux de gris, la binarisation, le lissage, et la correction de l'inclinaison.

3.4.1.1.1. Transformation de l'image en niveaux de gris

Plusieurs images de notre base sont en couleurs. Ces images doivent être transformées en niveaux de gris (256 niveaux de gris).

La transformation en niveaux de gris peut être effectuée tout simplement en utilisant le pseudo-code suivant :

Algorithme de transformation en niveaux de gris

Entrée : image en couleurs

Sortie : image en niveaux de gris

Début

Pour chaque pixel *p* de l'image faire :

r ← la quantité de la couleur rouge du pixel *p*

v ← la quantité de la couleur verte du pixel *p*

b ← la quantité de la couleur bleue du pixel *p*

p ← (*r*+*v*+*b*)/3

Fin Pour

Fin.

3.4.1.1.2. Binarisation

La binarisation (seuillage) est un traitement irréversible qui permet de transformer une image en niveaux de gris ou en couleurs en une image bimodale (contenant seulement du noir et du blanc) en fonction d'un seuil à définir.

Comme nous avons vu dans le premier chapitre, plusieurs techniques ont été proposées dans la littérature pour la binarisation d'images en niveaux de gris. Dans [KEF 09] Nous avons effectué une étude comparative de douze méthodes de seuillage à savoir : le seuillage global fixe, la méthode d'Otsu [OTS 79], ISODATA [VEL 80], Kapur [KAP 85], Cheng et Chen [CHE 98], Li-Lee [LI 93], Bernsen [BER 86], Niblack [NIB 86], Sauvola [SAU 97], Wolf [WOL 02], Nick [KHU 09] et la segmentation hiérarchique floue [GAD 00]. Cette étude comparative, nous montre qu'une méthode peut être efficace pour quelques images mais inefficace pour d'autres, c'est pour cela que nous avons proposé d'intégrer dans notre système plusieurs algorithmes de seuillage et nous laissons à l'utilisateur le choix de la méthode qu'il

veut appliquer, afin de garantir une meilleure qualité de l'image résultante. Les algorithmes choisis sont ceux étudiés dans l'étude comparative et mentionnés plus-haut.

La figure 3.6 suivante illustre le résultat de cette étape sur une image en niveaux de gris.

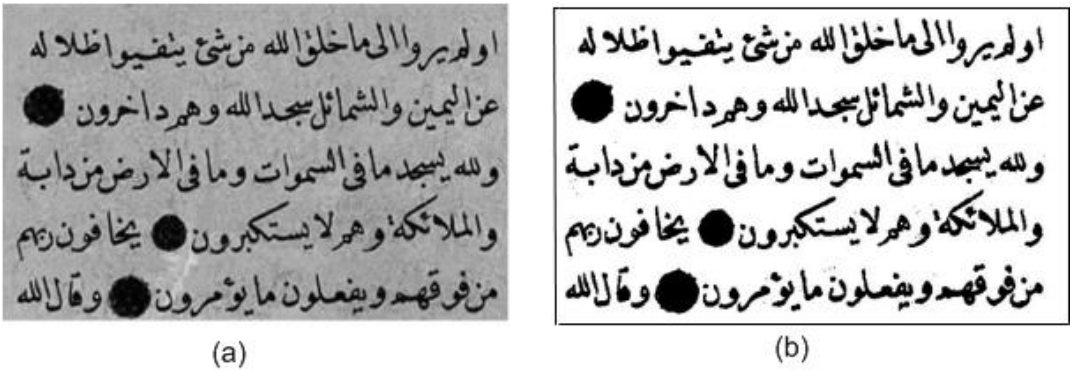


Figure 3.6: Binarisation d'une image de document, (a) image en niveaux de gris, (b) image binaire

3.4.1.1.2. Lissage

Dans certains cas, les processus d'acquisition ou de binarisation peuvent introduire du bruit dans l'image, qui se traduit en particulier par la présence d'irrégularités le long des tracés des caractères, et qui peut donc dégrader les performances de notre système. Pour pallier ce problème, nous appliquons un lissage en utilisant l'algorithme de [MAH 94] qui réduit le bruit d'une image binaire en éliminant les pixels isolés d'une part et en bouchant les trous vides d'autre part. Cette technique simple et efficace est basée sur une décision statistique. En effet, la nouvelle valeur de chaque pixel dans l'image binarisée est calculée en fonction de sa valeur initiale et celles des pixels voisins (8-voisinages).

P_1	P_2	P_3
P_4	$\mathbf{P_0}$	P_5
P_6	P_7	P_8

Figure 3.7: Le pixel courant P_0 et ses voisinages

L'algorithme de lissage utilisé peut être donné par le pseudo-code suivant :

Algorithme de Lissage

Entrée : Image binaire *I*

Sortie : Image lissée *I'*

Début

Pour tout pixel *P* de l'image *I* faire:

Si *P* =blanc alors

n ← le nombre de 8-voisins noirs du pixel *P*

Si $n > T$ Alors $P \leftarrow$ noir

Fin Si

Sinon

$n \leftarrow$ le nombre de 8-voisins blancs du pixel P

Si $n > T$ Alors la $P \leftarrow$ blanc

Fin Sinon

Fin Pour

Fin.

Tel que T est un seuil fixe. Des expérimentations ont montré que la valeur 5 pour le seuil T donne les meilleurs résultats. La figure 3.8 montre le résultat d'application de cet algorithme sur une image de la lettre Tad.

Figure 3.8: Lissage d'une image de la lettre Tad, (a) image binarisée, (b) image lissée

3.4.1.1.3. Correction de l'inclinaison

Les anciens documents composant notre base sont parfois inclinés. La présence des inclinaisons influe considérablement sur les étapes de segmentation et d'extraction de caractéristiques et donc le résultat final de notre système. Le système développé doit donc disposer d'un module de correction de l'inclinaison. Dans notre système, nous avons utilisé une méthode de correction de l'inclinaison basée sur les projections partielles. La méthode procède en cinq étapes comme suit :

1) Division de l'image en colonnes

Le choix de largeur des colonnes est critique car elle influence l'extraction des lignes de base. Dans nos expérimentations, une largeur de colonnes égale à cent pixels a donné des bons résultats.

2) Calcul de l'histogramme des projections horizontales pour chaque colonne

L'histogramme des projections horizontales de chaque colonne de l'étape précédente est obtenu en calculant la somme de pixels noirs sur chaque ligne. Comme la largeur des colonnes est assez petite par rapport à la largeur de l'image, l'inclinaison à l'intérieur d'une colonne est moins importante que l'inclinaison globale, et donc l'histogramme des projections horizontales est plus exploitable et comporte plusieurs pics et vallées. La figure 3.9 montre le résultat de cette étape pour une image inclinée subdivisée en cinq colonnes.

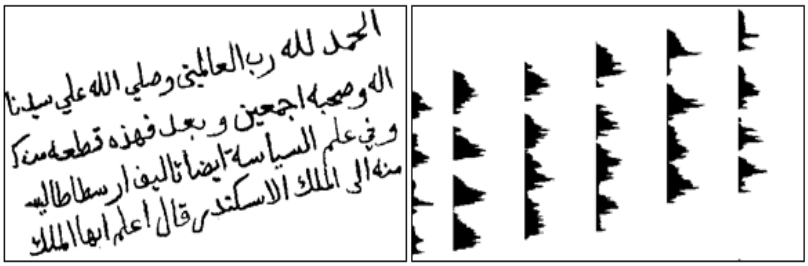


Figure 3.9: (a) Image inclinée, (b) histogrammes des projections de chaque colonne

3) Extraction des lignes de base

Les lignes de base correspondent aux pics des histogrammes calculés auparavant. Si on considère l'histogramme des projections comme une fonction discrète *f(x)*, au niveau d'un pic, la dérivée seconde de *f* s'annule, c'est-à-dire que la dérivée première change de signe. En d'autres termes si *f(k-1) < f(k)* et *f(k+1) < f(k)* alors *k* est considéré comme un pic. En réalité, la formule précédente permet de trouver les maxima locaux, et donc elle peut détecter des faux pics (figure 3.10.b). Une étape de filtrage est alors nécessaire.

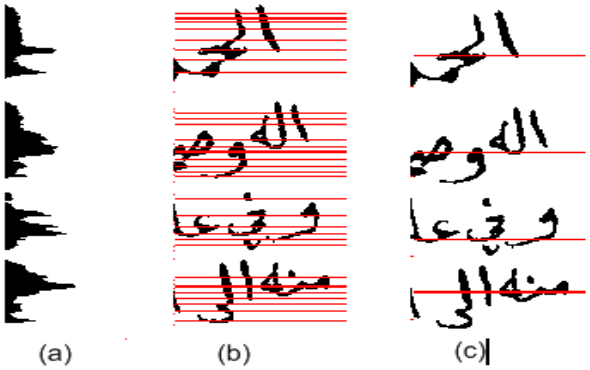


Figure 3.10: (a) Histogramme des projections horizontales d'une colonne, (b) détection des pics, (c) filtrage des pics

Le filtrage des maxima locaux est effectué en deux passes. Dans la première, nous trouvons le plus long pic, mettons *largMax* sa largeur, on enlève tous les pics dont la largeur < *largMax/2*, car les lignes de base dans un document possèdent approximativement le même nombre de pixels. Dans la deuxième on enlève l'un des deux pics très proches, car la distance entre deux lignes de base successives est presque la même dans tout le document. Pour ce faire, on calcule d'abord la distance moyenne *distanceMoyen* entre deux pics successifs. Si la distance entre deux pics successifs est < *2*(distanceMoyen)/3*, le plus court d'entre eux sera enlevé. Le choix de *2*(distanceMoyen)/3* est effectué par expériences. La deuxième passe sera répétée jusqu'à la convergence (jusqu'aucun pic n'est enlevé). Les deux passes sont exécutées sur chaque colonne à part (voir figure 3.10.c).

4) Calcul de l'angle d'inclinaison

Considérons deux lignes de base appartenant à deux colonnes successives, ces deux lignes forment un angle θ (figure 3.11).

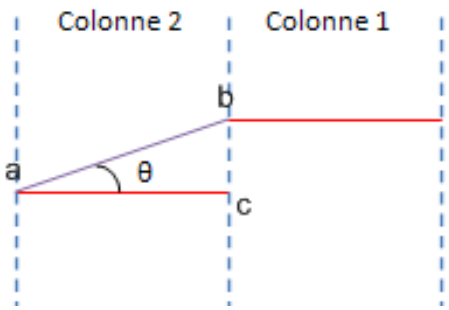


Figure 3.11: L'angle d'inclinaison entre deux colonnes successives

A partir de cette figure, on peut calculer *tangente (θ) = cb/ ac*. L'angle *θ* peut être trouvé tout simplement en utilisant la fonction *arc Tangente*.

Le problème est que l'angle d'inclinaison dans un document n'est pas fixe. En plus, les colonnes peuvent ne pas comporter le même nombre de lignes de base à cause d'une mauvaise détection des pics. Pour remédier à ce problème, nous avons considéré l'angle d'inclinaison du document comme la moyenne de tous les angles formés par deux lignes de base appartenant à deux colonnes successives.

5) Correction de l'inclinaison

La dernière étape dans cette méthode consiste à corriger l'inclinaison de l'image en faisant une rotation d'angle *θ*. Dans la nouvelle image, les coordonnées des pixels sont calculées à partir des anciennes en utilisant la formule suivante :

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x\cos\theta + y\sin\theta \\ -x\sin\theta + y\cos\theta \end{pmatrix}$$

Le résultat final de la correction de l'inclinaison est donné par la figure 3.12 suivante :

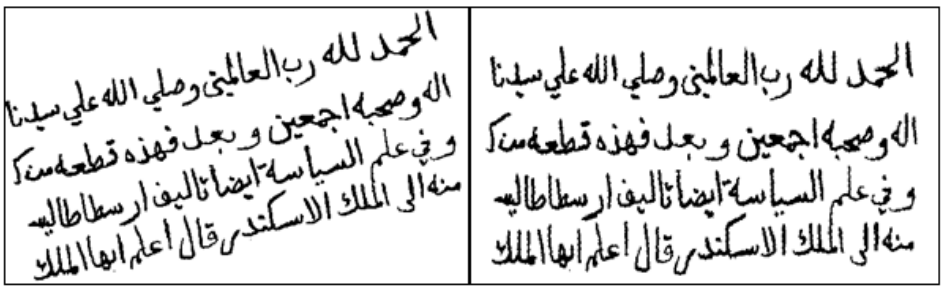


Figure 3.12: Correction de l'inclinaison (a) image inclinée, (b) image bien alignée

3.4.1.2. Segmentation

Après le prétraitement, nous devons extraire les sous-mots du texte. Le choix de travailler sur les sous-mots est justifié par qu'ils nous semblent offrir un meilleur compromis entre la complexité de la segmentation des mots arabes manuscrits en lettres et l'approche globale qui emploie des dictionnaires de petites tailles. En effet, avec un petit nombre de sous-mots on peut construire un grand nombre de mots et leur extraction est plus facile.

Dans ce projet, nous supposons que les documents ne contiennent pas de parties graphiques. Pour extraire les sous-mots, nous devons tout d'abord segmenter le texte en lignes, et par la suite segmenter chaque ligne en sous-mots.

3.4.1.2.1. Segmentation en lignes

Pour extraire les lignes du texte, nous appliquons une méthode basée sur la technique des projections horizontales. Elle consiste à séparer les lignes du texte en utilisant des mesures de densités des plages blanches à partir des projections horizontales, les vallées de l'histogramme correspondent aux zones de séparation entre les lignes. La méthode procède en quatre étapes comme suit :

1) Calcul de l'histogramme des projections horizontales

Comme nous avons dit, l'histogramme des projections horizontales est obtenu en calculant le nombre de pixels noirs dans chaque ligne de l'image. Comme le texte a été bien aligné, l'histogramme des projections horizontales correspondant sera constitué de pics et de vallées, représentant les lignes du texte et les espaces entre elles respectivement (figure 3.13.b).

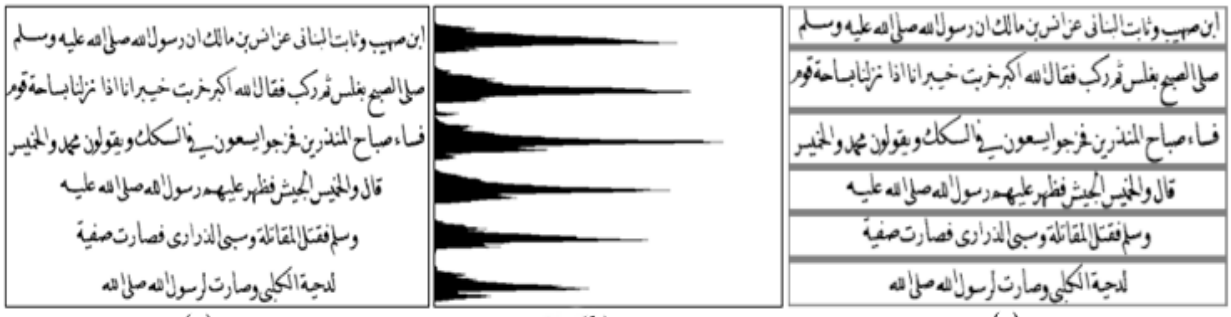


Figure 3.13: Segmentation en lignes, (a) image bien alignée, (b) histogramme des projections correspondant, (c) image segmentée en lignes

2) Extraction des minima locaux

Si on considère l'histogramme des projections comme une fonction discrète *f(x)*, pour *k* allant de 1 jusqu'à la hauteur de l'image-1, *k* est considéré comme un minimum local si $f(k-1) > f(k)$ et $f(k+1) > f(k)$.

3) Filtrage des minima locaux

Le filtrage des minima locaux obtenus auparavant est nécessaire afin d'éliminer les fausses vallées qui peuvent être détectées par la formule précédente. Le filtrage des minima locaux est effectué en deux passes. Dans la première passe, on élimine les minima locaux ayant une largeur plus grande qu'un seuil donné. Le seuil est choisi comme la largeur du plus long minimum local / 2. L'espace entre deux minima successifs correspond à la hauteur d'une ligne du texte. Dans la deuxième passe, on enlève l'un des deux minima très proches, car la hauteur de lignes du texte est presque la même dans tout le document. Pour ce faire, on calcule d'abord la distance moyenne *distanceMoyen* entre deux minima successifs. Si la

distance entre deux minima successifs est < *2*(distanceMoyen)/3*, le plus long d'entre eux sera enlevé.

Les minima restants correspondent aux zones de séparation entre les lignes du texte.

4) Résolution de conflits

Il nous reste maintenant d'attribuer les pixels noirs existant dans les zones séparatrices à la ligne du texte la plus proche par analyse de proximité. A la fin de cette étape, chaque ligne du texte sera représentée par une image à part (figure 3.13.c).

3.4.1.2.2. Segmentation en sous-mots

Comme nous avons dit précédemment, nous avons choisi de travailler avec les sous-mots et non pas avec les caractères, car ces derniers ne sont pas toujours aisément séparables. Cette difficulté a été clairement évoquée par Sayre en 1973 et peut être résumée par le dilemme suivant : « pour reconnaître les lettres, il faut segmenter le tracé et pour segmenter le tracé, il faut reconnaître les lettres », voir également [SAR 07].

L'extraction des sous-mots consiste à étiqueter les différentes composantes connexes de l'image en regroupant les pixels noirs voisins dans une unité distincte, et on utilise pour cela la méthode d'agrégation des pixels (croissance de régions). A la fin de cette étape, chaque composante connexe sera délimitée par un rectangle englobant (figure 3.14). Dans ce sens et à ce niveau, on considère chaque point diacritique comme une composante connexe à part.

Figure 3.14: Ligne de texte segmentée en composantes connexes

Notons que cette segmentation est effectuée pour chaque ligne de texte.

Dans cette étape nous enregistrons les coordonnées des composantes connexes étiquetées, qui vont être utilisées dans la deuxième phase pour localiser les mots cherchés dans le document.

3.4.1.3. Extraction de caractéristiques

L'extraction de caractéristiques a comme but d'identifier les propriétés les plus importantes pour la discrimination de classes de formes. Un des problèmes fondamentaux de l'analyse d'images est de déterminer quelles caractéristiques à employer pour avoir un bon résultat. D'après l'étude bibliographique que nous avons effectuée sur les caractéristiques de l'écriture arabe, les primitives structurelles issues de la perception humaine qui sont reliées à la forme de l'écriture [AMI 89] [CHE 98b] sont considérées comme des caractéristiques pertinentes pour la discrimination des caractères manuscrits.

Dans notre système, nous avons choisi d'extraire à partir de chaque sous mot, quatre caractéristiques structurelles les plus utilisées pour l'écriture arabe, à savoir :

- **Les hampes :** correspondent aux extensions hautes pouvant exister dans les caractères ;
- **Les jambages :** correspondent aux extensions basses pouvant exister dans les caractères ;
- **Les points diacritiques :** correspondent aux parties secondaires des caractères;
- **Les boucles :** correspondent aux occlusions pouvant exister dans les caractères.

Notons que les caractéristiques structurelles précédentes sont extraites à partir des contours des sous-mots, et en s'aidant de la connaissance à priori que les extensions hautes et basses des caractères sont toujours en dehors d'une zone médiane. Afin d'extraire donc ces caractéristiques, nous procédons comme suit :

3.4.1.3.1. Détection de la ligne de base

La ligne de base est la ligne sur laquelle repose les caractères qui ne possèdent pas des descendants. Dans les textes arabes, la ligne de base porte une information assez importante sur l'orientation des textes et la position des points diacritiques. La méthode la plus répondue pour détecter les lignes de base est la projection horizontale de l'image. La ligne de base correspond à la ligne dont la projection contient le plus grand nombre de pixels noirs (la ligne rouge dans la figure 3.15).

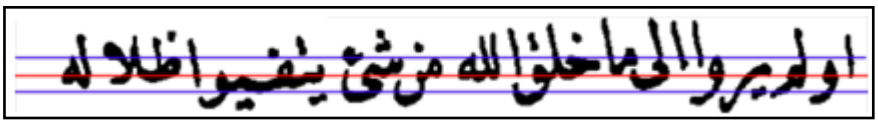


Figure 3.15: Ligne de base et zone médiane détectées sur une ligne de texte

3.4.1.3.2. Localisation de la zone médiane

Le corps des mots arabes, apparaît généralement dans une zone appelée la zone médiane, c'est pour cette raison que la localisation de cette zone est importante pour l'extraction des caractéristiques. Dans notre application, nous détectons la zone médiane en traçons deux frontières, haute et basse par rapport à la ligne de base. La zone médiane sera donc l'espace compris entre ces deux frontières.

Pour tracer la frontière haute (respectivement basse), on calcule d'abord le nombre moyen de pixels dans chaque ligne existante au dessus (respectivement au dessous) de la ligne de base. On parcourt par la suite ces lignes de bas en haut (respectivement de haut en bas), jusqu'à trouver la première ligne contenant un nombre de pixels inférieur à la moyenne. Cette ligne correspond donc à la frontière haute (respectivement basse). Dans la figure 3.15, les deux frontières délimitant la zone médiane sont tracées en bleu.

3.4.1.3.3. Suivi de contour

Le suivi de contour est communément utilisé pour l'extraction des caractéristiques structurelles des caractères. La chaîne de Freeman est la méthode la plus utilisée pour la description des contours dans les images. C'est une technique de représentation des directions du contour (on code la direction le long du contour dans un repère absolu lors du parcours du contour à partir d'une origine donnée). Les directions peuvent se présenter en 4 connexités ou en 8 connexités. Les codes des contours sont donnés par la figure 3.16 suivante.

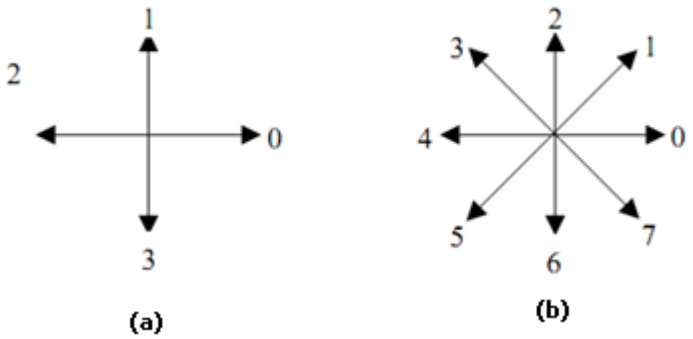


Figure 3.16: Le code de Freeman (a) en 4-connexités, (b) en 8-connexités

Chaque contour est codé en spécifiant un point de départ suivi par une chaîne ou une séquence de codes de la chaîne de Freeman. Suivant ce codage, chaque image est représentée sous forme d'une liste de contours. L'analyse de ces contours, leurs types (interne ou externe) et leurs positions nous permet de distinguer la présence des boucles, des points diacritiques, et des extensions hautes et basses. Dans notre système, l'étiquetage des contours consiste à attribuer à chaque contour de l'image un numéro unique. Comme on s'intéresse au traitement des textes arabes, on commence le parcours de haut en bas et de droite à gauche. Un pixel est considéré comme un point du contour si c'est un pixel noir et a au moins un voisin blanc. Le résultat de cette étape pour une ligne de texte arabe, est montré par la figure 3.17:

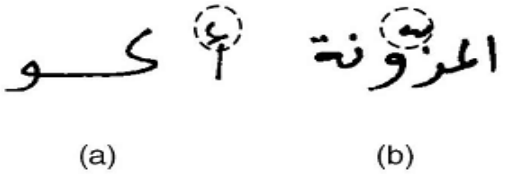
Figure 3.17: Contours d'une ligne de texte

3.4.1.3.4. Extraction des diacritiques

Comme nous avons vu dans la section 3.2.2, l'écriture arabe est riche en points diacritiques qui constituent les parties secondaires des caractères. En effet, la taille des points diacritiques est généralement petite ce qui les rend sensibles aux bruits d'acquisition ou aux prétraitements de l'image. Les points diacritiques peuvent être simples ou multiples.

En plus des points diacritiques on peut trouver les voyelles qui sont des signes diacritiques associés aux lettres sur lesquelles ils s'appliquent (figure 3.18). Comme on peut trouver d'autres signes : la hamza, la chadda (figure 3.19).

Figure 3.18: Voyelles en arabe : (a) A, (b) OU, (c) I, (d) - , (e) AN , (f) OUN, (g) IN



(a) (b)

Figure 3.19: Autres signes diacritiques : (a) hamza, (b) chadda

Dans notre système, nous considérons que ces signes de voyellation ne sont pas présents dans les textes traités, et que les textes ne contiennent que des points diacritiques simples. En d'autres termes, la séparation des points multiples en deux points ou trois points, sort de notre intérêt.

Pour l'extraction des diacritiques, nous avons utilisé une heuristique basée sur la taille des composantes connexes et leurs positions. Selon nos expérimentations, une composante connexe est considérée comme un point diacritique si sa taille est inférieure à un certain seuil. Nous avons choisi d'utiliser un seuil égal à la moyenne des tailles des composantes connexes divisée par 4. Pour distinguer les points diacritiques hauts et bas, on compare leurs positions par rapport à la ligne de base.

3.4.1.3.5. Extraction des boucles

Dans l'écriture arabe, les boucles ou occlusions se trouvent généralement dans la zone médiane et à proximité de la ligne de base. Elles correspondent aux contours internes dans les tracés des caractères.

Les boucles sont utiles pour l'identification de certains caractères. La présence ou non d'une boucle, le nombre de boucles sont des caractéristiques à déterminer. Par exemple, le caractère Haa (هـ) est le seul caractère arabe qui contient deux boucles.

Pour détecter les boucles, il faut parcourir la liste de tous les contours et comparer chaque contour avec les autres. Si toutes les cordonnées d'un contour englobent toutes les coordonnées correspondantes d'un autre, alors il existe une boucle.

3.4.1.3.6. Extraction des hampes et des jambages

Une hampe ou bien un ascendant correspond à une montée qui dépasse la zone médiane (au dessus de la frontière haute), de même un jambage correspond à une descente qui se termine en dehors de la zone médiane c'est-à-dire située dans la zone inférieure.

Pour extraire les hampes, on parcourt la zone supérieure de l'image, toutes les composantes qui se trouvent dans cette zone et qui ne sont pas des points diacritiques sont considérées comme des hampes. Pour les jambages, on procède de la même manière, mais cette fois on parcourt la zone inférieure de l'image.

A la fin de l'étape d'extraction de caractéristiques, chaque type de caractéristiques sera représenté d'une couleur : les boucles en rouge, les points diacritiques en bleu, les hampes en violet, et les jambages en vert (voir figure 3.20).

Figure 3.20: Caractéristiques extraites à partir d'une ligne de texte arabe

3.4.1.4. Codage

Après avoir extrait les caractéristiques de l'image du document, vient la dernière étape d'analyse qui est le codage de l'image. L'image du document sera codée donc selon les caractéristiques extraites précédemment en faisant correspondre à chaque caractéristique un code en ASCII (les hampes en *h*, les jambages en *j*, les boucles en *b*, les points diacritiques hauts en *p*, et les points diacritiques bas en *q*). On ajoute un autre caractère « # », qui représente l'espace inter sous-mots.

Pour coder une image d'une ligne de texte, on commence par une chaîne vide, et on parcourt l'image obtenue après extraction des caractéristiques de droite à gauche. Une fois on tombe sur une caractéristique, on ajoute le code correspondant à la chaîne de codes. A la fin de chaque sous-mot, on ajoute un « # », pour séparer les caractéristiques de deux sous-mots voisins.

Par exemple l'image de la figure 3.20 est codée par la séquence:

h#j#hbj#qj#bj#h#h#h#h#phjp#h#hhb#bpj#pjp#qpbpqj#h#pbhh#hb

Comme chaque ligne de texte est représentée dans une image à part, et que les traitements précédents se font sur chaque image de ligne, on obtient pour chaque ligne de texte une chaîne de codes. Ces codes seront stockés dans un ficher de codes associé au document traité.

3.4.1.5. Indexation

L'indexation est une technique largement utilisée dans les systèmes de recherche. Elle a pour but d'extraire et de représenter le sens d'un document de manière à ce qu'il puisse être retrouvé par l'utilisateur. L'indexation textuelle est la méthode la plus utilisée pour l'indexation des images de documents. Nous pouvons citer deux types d'indexation : indexation manuelle, et indexation automatique.

L'indexation textuelle manuelle d'images de documents est réalisée par un humaniste. Son rôle est de classer et indexer les images en les associant à des catégories et à des groupes de mots, souvent extraits d'un thésaurus, permettant de retrouver facilement les images. Notons que ce type d'indexation devient très fastidieux, voir impossible dans le cas d'un grand volume d'images à indexer.

Lorsque l'on parle d'indexation automatique du contenu de documents, plusieurs orientations sont généralement proposées. La première qui semble la plus naturelle consiste à proposer une analyse du texte lui-même. Il faut dès lors employer des moteurs de reconnaissance de caractères et de mots qui fonctionnent de façon performante sur les textes de documents contemporains. Malheureusement, les tests effectués sur les textes anciens donnent des performances décevantes [EMP 07], nécessitant des corrections nombreuses et coûteuses. La

seconde approche consiste à analyser la structure du document ou de l'ouvrage. Il s'agit plus ici de retrouver la structure physique et logique donnant une autre forme d'information que le contenu sémantique. Cette structure est riche et permet de naviguer rapidement dans l'ouvrage facilitant ainsi la recherche d'information (sommaires, titres, …).

Dans notre système nous avons choisi d'indexer les images de notre base, comme plusieurs autres travaux d'indexation des documents anciens, manuellement. Pour chaque document de la base, on analyse son contenu, et on lui assigne trois index (chaînes de caractères). Comme les documents de notre base et la requête formulée par l'utilisateur sont tous les deux en langue arabe, la première idée qui a venu à la tête est d'indexer les documents par des mots arabes. La recherche sera par la suite effectuée en comparant la requête avec les index des documents. Mais comme la requête sera codée, la comparaison s'effectue donc entre le code de la requête et les codes des index. Pour éliminer le temps de codage des index lors de la recherche, nous choisissons d'indexer les images par des codes des mots arabes au lieu de les indexer par des mots arabes.

Par exemple, nous indexons le document présenté dans la figure 3.21 par les codes des mots encerclés, qui représentent le titre, l'auteur et le thème du document.

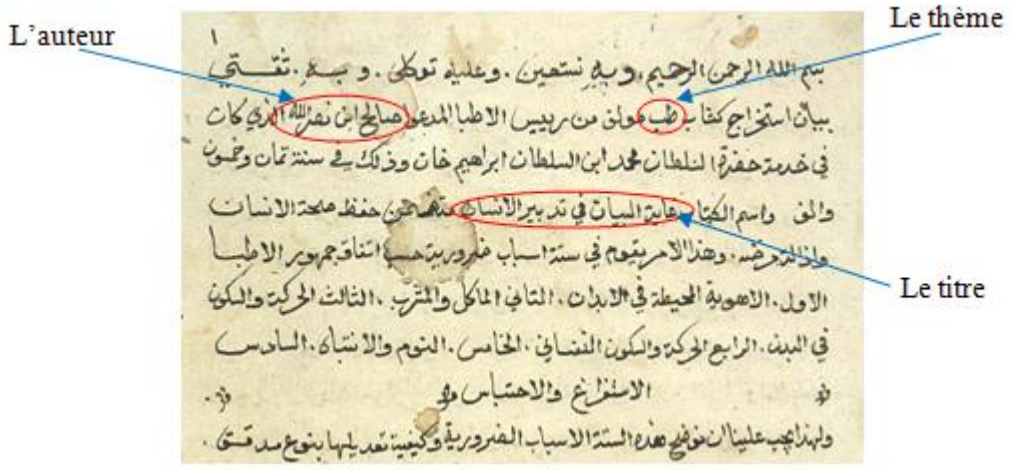


Figure 3.21: Mots choisis comme des index

Les index seront donc :

- « bh#hj#qjp#pbj#h#hhb » correspond au code de la chaîne **صالح بن نصر الله**
- « h#hbhq » correspond au code de la chaîne **الطب**
- « ph#qbp#h#hqqh#jp#bpjq#p#qqj#h#hbh#ph#jp » correspond au code de la chaîne **غاية البيان في تدبير الإنسان**

3.4.2. La deuxième phase : Recherche de mots

Dans cette phase, nous exploitons les résultats obtenus par la première phase notamment les fichiers de codes et les index associés aux documents pour répondre à une requête de l'utilisateur. Le système répond à la requête en retournant un ensemble d'images jugées pertinentes pour l'utilisateur. Un document est jugé pertinent pour l'utilisateur s'il contient la requête cherchée (recherche exacte), ou bien s'il contient des mots proches de la requête

(recherche approximative). Notons que dans cette phase, nous combinons la technique d'arbres de suffixes avec celle de la recherche approximative.

L'utilisateur commence par la formulation de sa requête qui sera codée par la suite, et le système procède une recherche dans les index des documents. Ensuite, le système effectue une recherche dans les fichiers de codes associés aux documents jugés non pertinents dans la première étape de recherche.

L'architecture générale de la deuxième phase est donnée par le schéma de la figure 3.22 suivante:

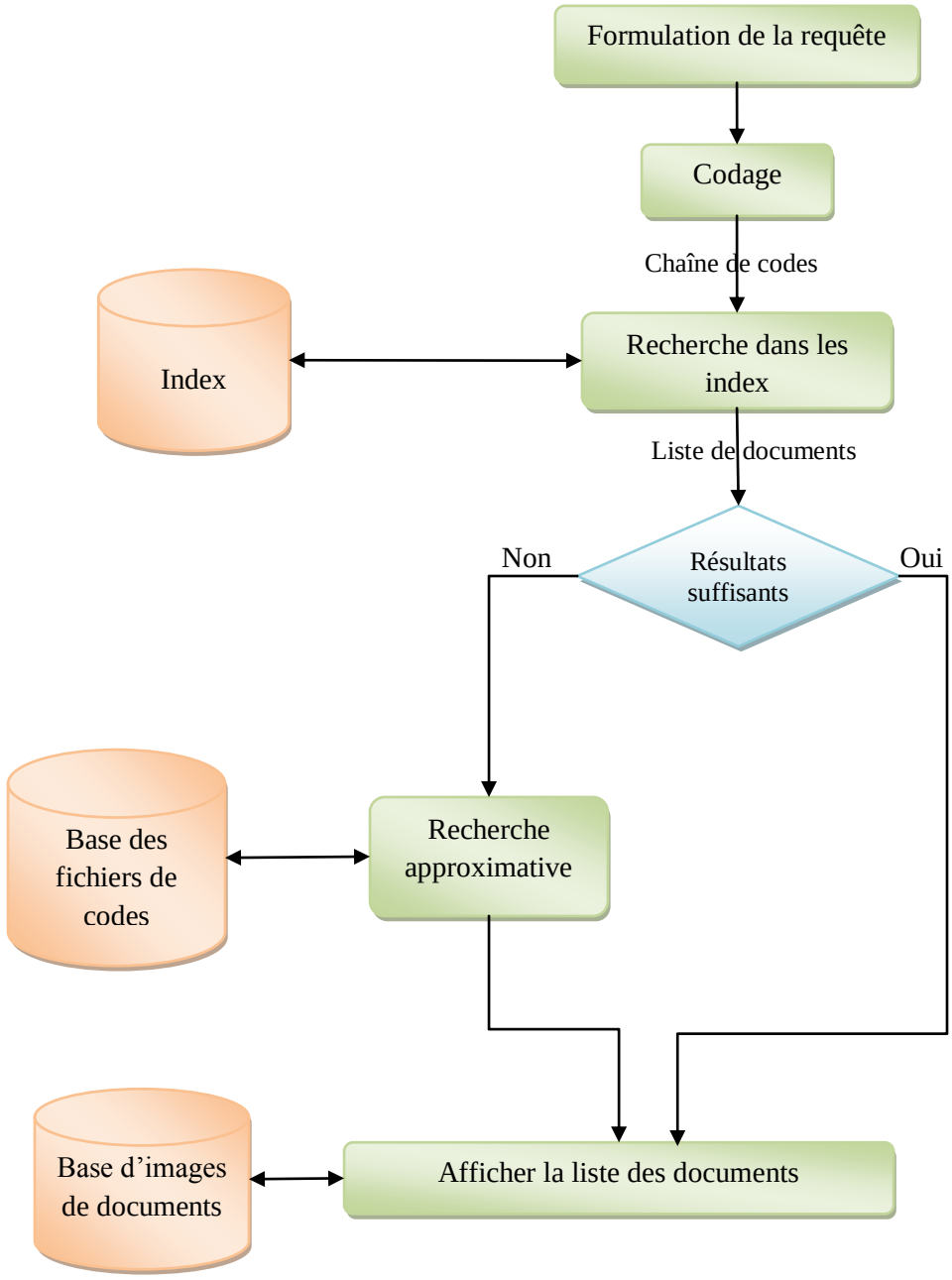


Figure 3.22: Schéma général de la phase de recherche

3.4.2.1. Formulation de la requête

L'objectif de l'utilisateur est de retrouver tous les documents dans la base comportant certains mots. Dans notre système, l'utilisateur formule sa requête en langue arabe sous forme d'une chaîne de caractères. La requête peut être :

- Un mot simple : الزمان
- Un mot composé : علم السياسة

3.4.2.2. Codage de la requête

Dans la première phase, nous avons codé les documents de notre base selon les caractéristiques structurelles des sous-mots composant le texte. Comme la recherche se fait dans les fichiers de codes correspondant aux images de documents, la requête de l'utilisateur doit être codée de la même manière que le codage des images de documents. C'est-à-dire que la requête sera codée selon les caractéristiques structurelles qu'elle contient.

La requête de l'utilisateur est codée en faisant correspondre à chaque lettre de la requête un code précis, ce qui nous donne une chaîne de codes. Nous avons établi un tableau de correspondance qui résume les codes des différentes lettres arabes (tableau 3.1).

Caractère	Code	Désignation
ا - ـا - ـك - ـل	h	Hampe
إ	hq	Hampe+point bas
أ - ـأ	ph	Point haut+hampe
ل	hj	Hampe+jambage
ط	bh	Boucle+hampe
ظ	bph	Boucle+point haut+hampe
لا	hbh	Hampe+boucle+hampe
ك	hp	Hampe+point haut
ي	jq	Jambage+point bas
غ - خ - ذ- ت- ئ - ن	p	Point haut
غ	jp	Jambage+point haut
شـ - ث	pp	Point haut+point haut
ن- ز- خ - ئ	jp	Jambage+point haut
ش	ppj	Point haut +point haut+jambage
ض	bpj	Boucle +point haut+jambage
ضـ - فـ- ةـ غـ - قـ	bp	Boucle +point haut
ق	pbj	Point haut+boucle+jambage
يـ - جـ-ب	q	Point bas
ح-ع- س- ر- ى	j	Jambage
ج	jq	Jambage+point bas
ـعـ - مـ - صـ ـهـ - ه	b	Boucle
هـ	bb	Boucle+boucle
ـع - و- ص- م	bj	Boucle+jambage
ـلا	hh	Hampe+hampe

ـة	pb	Point haut+boucle
ـغ	pbj	Point haut+boucle+jambage
ؤ	bjp	Boucle+jambage+point haut
لأ	hbhp	Hampe+boucle+hampe+point haut
لإ	hbqh	Hampe+boucle+point bas+hampe

Tableau 3.1 : Les codes des lettres arabes

Le mot العلم par exemple est codé par la chaîne: **h#hbhbj.**

En analysant le tableau précédent, nous pouvons tirer les observations suivantes :

- La description structurelle d'une lettre varie suivant sa position dans le mot ;
- Pour certaines lettres, des caractéristiques visuelles sont éliminées, d'autres sont ajoutées et d'autres sont totalement modifiées en passant d'une position à une autre ;
- Différentes lettres peuvent avoir une description structurelle identique ;
- Certaines combinaisons de caractéristiques ne sont jamais possibles ;

La recherche sera donc effectuée par la chaîne de codes et non pas par la requête originale de l'utilisateur. A partir de maintenant, nous désignons par requête la chaîne de codes associée.

3.4.2.3. Recherche dans les index

Comme première étape, nous effectuons une recherche dans les index des documents. Notons que ce type de recherche est une recherche exacte, parce que les documents sont indexés manuellement, ce qui nous encourage à supposer que les index et la requête ne comportent pas d'erreurs.

Dans cette étape, la requête sera comparée avec les index de tous les documents composant la base. Un document est jugé pertinent pour l'utilisateur si au moins un de ses index comporte la requête cherchée. A la fin de cette étape, l'utilisateur décide si les résultats sont suffisants ou non. Si les résultats obtenus ne sont pas suffisants pour l'utilisateur, nous procédons une recherche dans les fichiers de codes, en appliquant une recherche approximative.

Pour comparer la requête avec les index, nous employons la technique d'arbres de suffixes. Comme nous avons vu dans le deuxième chapitre, les arbres de suffixes sont des techniques très efficaces et rapides pour la recherche d'un motif P, de longueur m, dans un texte de longueur n en temps $O(m)$, ce qui permet de réduire considérablement le temps de recherche.

La recherche peut être effectuée en utilisant le pseudo-code suivant.

Algorithme de recherche dans les index

Début

Pour chaque document *d* faire

  Pour chaque index *i* de *d* faire

    Construire l'arbre de suffixes de *i*

    Rechercher la requête dans l'arbre de suffixes

Si le code est trouvé alors

Ajouter le document *d* à la liste des documents pertinents

Sortir de la boucle

Fin Si

Fin Pour

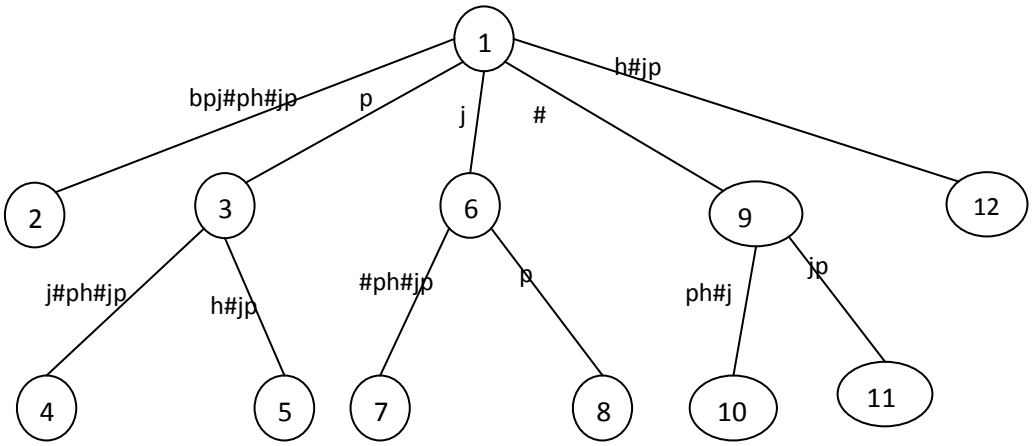
Fin Pour

Fin.

Pour la construction de l'arbre de suffixes, nous utilisons l'algorithme de McCreight [MCC 76].

Notons *P* la requête, la recherche dans l'arbre est très facile, partant de la racine de l'arbre, il faut suivre la branche dont l'étiquette commence comme *P*. En suivant cette branche on arrive alors à un nouveau nœud et on recommence le processus sur la partie restante de *P*.

Par exemple l'arbre de suffixes de la chaîne « bpj#ph#jp » correspondante au mot قرآن est donné comme suit :



3.4.2.4. Recherche approximative dans les fichiers de codes

Le système doit répondre à la requête de l'utilisateur. Lorsqu'un document n'a pas été retourné dans la première étape de recherche, ça n'implique pas que surement la requête n'existe pas dans le document. Il est possible que la requête n'apparaisse pas dans les index du document mais existe dans son fichier de codes, parce que comme il est connu les index d'un document ne peuvent pas contenir tous ses mots. C'est pour ça que nous avons proposé d'effectuer une recherche dans les fichiers de codes.

Mais comme nous l'avons dit précédemment, la base de fichiers de codes n'est pas précise et dans certains cas elle est incomplète. Cela vient généralement de la mauvaise qualité des documents qui rend les algorithmes de traitement classiques inefficaces, ce qui influe considérablement sur l'étape d'extraction de caractéristiques. En plus, s'ajoutent d'autres problèmes comme l'inefficacité de l'algorithme d'extraction de caractéristiques ce qui en

résulte la non détection de certaines caractéristiques, la non discrimination des caractéristiques employées (plusieurs caractères possèdent le même code)…etc. Tous ces problèmes risquent de faire ignorer des documents pertinents si on procède une recherche exacte dans les fichiers de codes.

Pour pallier à ce problème, nous avons proposé donc de faire une recherche approximative de la requête dans les fichiers de codes afin de tenir compte d'éventuelles erreurs. En effet, l'ajout de l'approximation dans la recherche permet la détection de motifs approchés mais augmente également le nombre de fausses occurrences. A mon avis personnel, retourner des fausses occurrences c'est mieux qu'ignorer des occurrences pertinentes.

La recherche approximative d'un mot *P* dans un texte *T* consiste à trouver toutes les occurrences dans *T* proches du mot *P*. La décision que deux mots sont proches nécessite l'introduction d'un seuil *k* qui détermine le nombre d'erreurs permis. Dans notre système nous avons proposé d'adapter le seuil *k* en fonction de la longueur de la requête.

Dans cette étape, nous utilisons deux mesures de distance entre chaînes : la distance d'édition définie comme le nombre minimal d'opérations d'édition nécessaires pour transformer la première chaîne en la seconde, et la distance de Jaro-Winkler [WIN 99] qui permet de mesurer la similarité entre deux chaînes de caractères, et qui est normalisée de façon à avoir une mesure entre 0 et 1, où le zéro représente l'absence de similarité. Pour la distance d'édition, nous utilisons deux algorithmes : l'algorithme de Wagner et Fischer [WAG 74] qui consiste à calculer la matrice des distances entre deux chaînes de caractères données, et celui proposé par Ukkonen [UKK 85] qui consiste à ne calculer qu'une partie de la table des distances.

La recherche en utilisant l'algorithme de Wagner et Fischer peut être donné par le pseudo code suivant :

Algorithme de recherche approximative par l'algorithme de Wagner et Fischer

Entré : le mot *P* et le texte *T*

Pré-requis : *n* = longueur du mot *P*

Pré-requis : *m* = longueur du texte *T*

Pré-requis : *coût* *// coût de la transformation*

Assure : matrice *D*[0..*n*, 0..*m*]

Début

1. **Pour** *i* de 0 à *n* **faire** *D*[*i*, 0]← *i ;*
2. **Pour** *j* de 0 à *m* **faire** *D*[0, *j*]← *0 ;*

3. **Pour** *i* de 1 à *n* faire

4. **Pour** *j* de 1 à *m* faire

5. **Si** *P*[*i-1*]=*T*[*j-1*] **alors** *coût*←0

6. **Sinon** *coût* ←1 ;

7. *D*[*i*][*j*]←min(*D*[*i*-1, *j*] + 1, // effacement
D[*i*, *j*-1] + 1, // insertion
D[*i*-1, *j*-1] + *coût*) // substitution

8. **Fin Pour**

9. **Fin Pour**

Fin.

La recherche en utilisant l'algorithme d'Ukkonen peut être obtenue en remplaçant les instructions de 5 à 7 de l'algorithme précédent par :

Si(*D*[*i-1*, *j-1*]<=*k*) **alors début**
Si *P*[*i-1*]=*T*[*j-1*] **alors** *coût*←0
Sinon *coût* ←1 ;
D[*i*][*j*]←min(*D*[*i*-1, *j*] + 1, // effacement
D[*i*, *j*-1] + 1, // insertion
D[*i*-1, *j*-1] + *coût*) // substitution
Finsi.

Le calcul des distances est effectué entre la requête et tous les fichiers de codes de notre base. Un document est jugé pertinent si le fichier de codes correspondant contient au moins une occurrence de la requête ayant au plus *k* erreurs. Cela est garanti si la dernière ligne de la matrice des distances contient une valeur inférieure ou égale au nombre d'erreurs permis *k*.

L'algorithme de Jaro-Winkler peut être donné par le pseudo-code suivant :

Algorithme de recherche approximative par l'algorithme de Jaro-Winkler

---

Entré : Deux chaînes de caractères *X* et *Y* ;

Pré-requis : *n* = longueur de la chaîne *X* ;

Pré-requis : *m* = longueur de la chaîne *Y ;*

Début

nb ←le nombre de caractères correspondants ;

t ← le nombre de transpositions ;

jaro←(*nb*/*m*+*nb*/*n*+(*nb*-*t*)/*nb*)/3 ; //distance de Jaro

l ← min (longueur du préfixe commun, 4) ;

p←0.1;

Jaro_winkler←*jaro*+*l***p**(1-*jaro*); //distance de Jaro-Winkler

Fin.

La même chose avec cette mesure de distance, un document est jugé pertinent si le fichier de codes correspondant contient une occurrence de la requête ayant au plus *k* erreurs, mais cette fois-ci, le seuil *k* est compris entre 0 et 1.

Enfin nous combinons les résultats obtenus en utilisant les deux mesures de distance.

3.4.2.4. Affichage des résultats

Après la recherche, le système affiche la liste de documents trouvés, ordonnés selon leur pertinence pour l'utilisateur. La pertinence pour l'utilisateur est exprimée par la distance minimale avec la requête.

3.5. Conclusion

Dans ce chapitre, nous avons présenté un système permettant l'analyse et la recherche des occurrences d'un mot dans des images d'anciens documents arabes sans recourir à une reconnaissance du contenu afin d'éviter le coût élevé et l'effort ardu de l'OCR. Les anciens documents arabes se caractérisent par une mauvaise qualité due à plusieurs facteurs, en plus des caractéristiques de l'écriture arabe. Ces caractéristiques compliquent le traitement de ces documents à plusieurs niveaux. L'idée de base de notre système consiste à transposer le problème de recherche d'images de documents du domaine de l'analyse de documents au domaine de la recherche d'information. Le système proposé opère en deux phases : une phase de traitement et d'analyse, dans laquelle chaque document de notre collection est représenté par une signature constituée de l'ensemble d'informations pertinentes qu'on peut extraire de l'image du document. Cette signature contient les caractéristiques structurelles de l'écriture et elle est codée en ASCII. Dans la deuxième phase on combine les techniques d'arbres de suffixes avec celle de recherche approximative pour répondre à une requête textuelle de l'utilisateur. L'évaluation du système proposé, les résultats obtenus, et les discussions accompagnées seront l'objet du prochain chapitre.

Chapitre 4 : Expérimentations et Résultats

4.1. Introduction

Le but de ce chapitre est d'évaluer notre système de recherche d'images d'anciens documents arabes par le contenu. L'évaluation est effectuée à travers un ensemble d'expérimentations et tests qui concernent à la fois la partie analyse de documents et la partie recherche. Dans la partie analyse de documents, l'évaluation a pour objectif de valider les méthodes utilisées dans chaque étape de traitement, ou de comparer plusieurs méthodes pour pouvoir sélectionner une à utiliser dans notre système. La validation dans cette partie est effectuée visuellement dans la plupart du temps à cause de l'absence des techniques automatiques standards d'évaluation dans le cas des documents arabes anciens (voir [KEF 09] et [KEF 10]). L'estimation du temps de traitement n'entre pas ici comme un critère d'évaluation parce que toutes les étapes de traitement de documents sont effectuées hors-ligne. Pour la partie recherche, nous évaluons chaque étape à part et l'évaluation sera en termes de rappel, précision et temps de recherche.

Toutes les expérimentations sont effectuées sur deux bases de données : une des anciens documents et l'autre d'enveloppes postales algériennes pour tester les performances de notre système sur un autre type d'images pour lequel l'application n'était pas préalablement conçue.

La suite de ce chapitre est organisée de la manière suivante : dans la section 2 nous illustrons l'environnement expérimental de notre système et le contexte des expérimentations, ensuite dans la section 3 nous présentons les deux bases de données utilisées pour évaluer toutes les étapes de traitement. Dans la section 4, nous décrivons les différentes expérimentations, les résultats obtenus des expérimentations et nous discutons ces résultats, avant de conclure.

4.2. Environnement expérimental

Notre système a été développé en langage de programmation Java, avec l'environnement de développement **Borland JBuilder 9 Entreprise** (voir figure 4.1), qui est un outil RAD (Rapid Application Development) basé sur le concept de programmation orientée objet. C'est un environnement qui permet aux développeurs, même non expérimentés, de créer facilement des interfaces homme/machine d'aspect **Windows, Web…etc.**

Le choix du langage Java est motivé principalement par sa portabilité et son ouverture sur Internet, en plus de l'existence de plusieurs bibliothèques en Java permettant d'effectuer plusieurs traitements dont on a besoin.

Tous les tests effectués sont exécutés sur un PC disposant d'un processeur Intel Céléron cadencé à 2.4 GHz, de 384 Mo de mémoire centrale et d'un disque local de 200 Go.

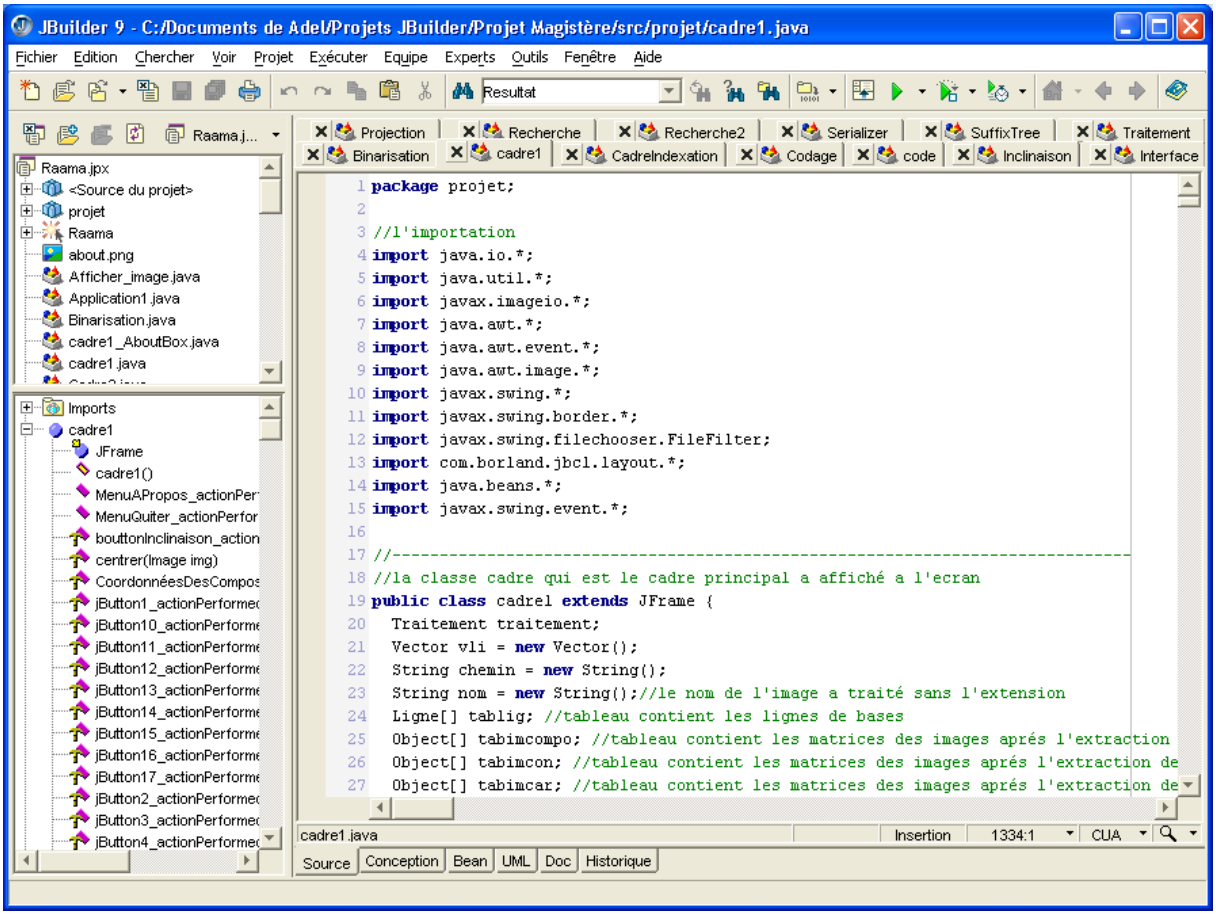


Figure 4.1: Interface de l'environnement de développement JBuilder9 Entreprise

4.3. Bases de données

Pour estimer les performances de notre système, nous avons besoin d'une base de données. Vu l'absence d'une base de données de benchmark des anciens documents arabes pour pouvoir l'utiliser dans la validation, une base locale de 100 images a été construite. Les images composant cette base ont été collectées à partir du web, et donc elles ne possèdent pas les mêmes caractéristiques. Elles sont de natures très différentes, provenant de plusieurs sources, couvrant plusieurs domaines, et présentant différents types de dégradations liées aux documents anciens (effet de transparence (figure 4.2.a), tâches d'encre et humidité (figure 4.2.b), faible contraste (figure 4.2.c), etc.).

Figure 4.2 : Quelques images d'anciens documents de notre base

Nous avons également utilisé dans nos tests une autre base de données composée de 298 images d'enveloppes postales algériennes. La base de données utilisée est disponible dans notre laboratoire de recherche, et les enveloppes composant cette base sont écrites par différents scripteurs. Nous distinguons des enveloppes réelles envoyées à l'université (figure

4.3.a), des enveloppes artificielles bien écrites par nous (figure 4.3.b), et des enveloppes artificielles avec quelques défauts (figure 4.3.c).

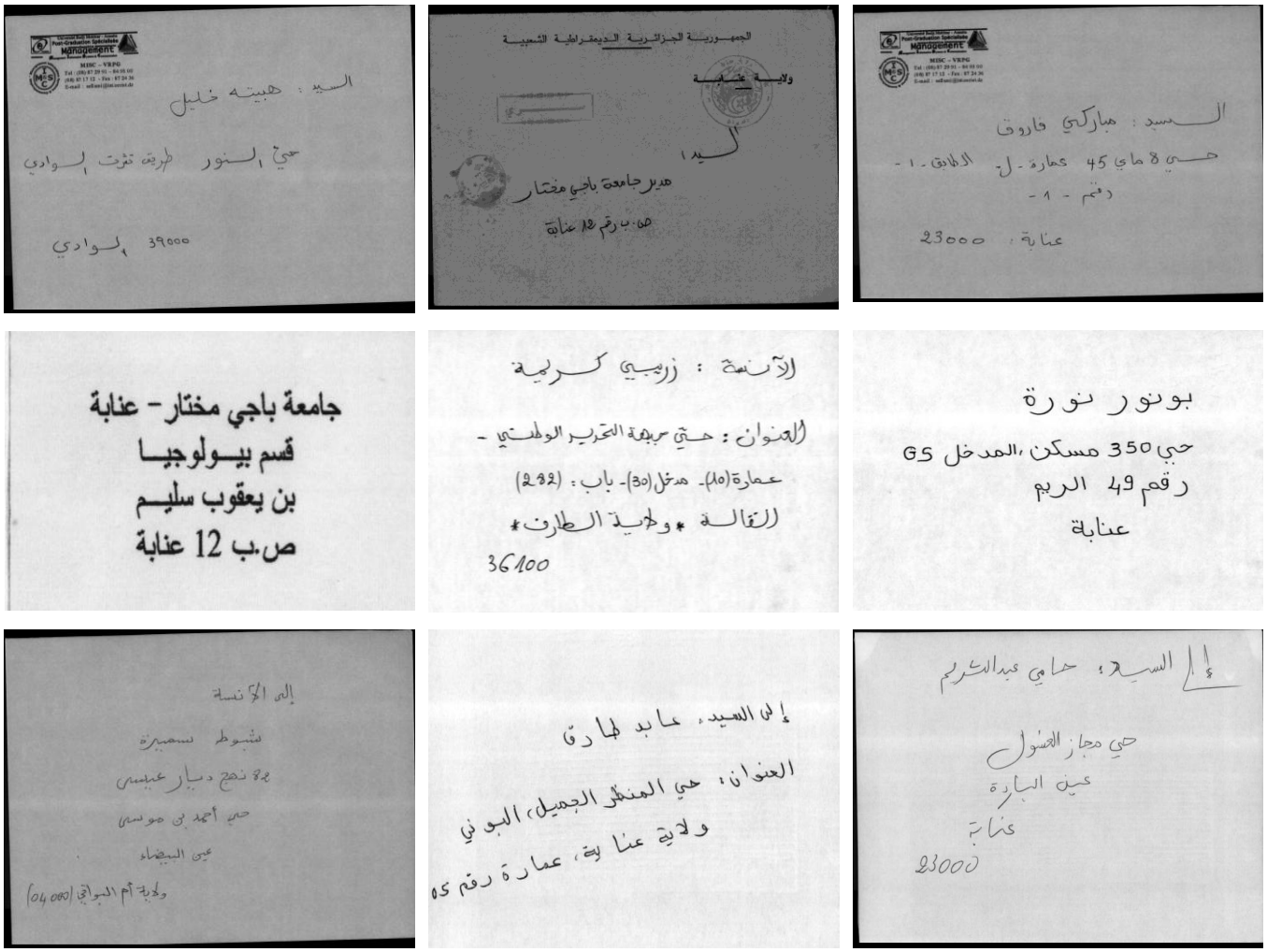


Figure 4.3 : Quelques images d'enveloppes composant notre deuxième base

Les images de la première base sont en couleur tandis que celles de la deuxième base sont en niveaux de gris, et elles sont sous les formats JPEG, PNG, GIF et BMP.

4.4. Tests, résultats et discussions

Dans cette section, nous exposons les différentes expérimentations effectuées, les résultats obtenus de ces expérimentations, et nous discutons ces résultats.

Les expérimentations adressées dans cette section concernent à la fois la partie analyse de documents ainsi que la partie recherche de mots sur les deux bases de données présentées précédemment.

4.4.1. Partie analyse de documents

Pour la partie analyse de documents, nous avons effectué plusieurs expérimentations afin de tester les performances de chaque étape de cette partie, et pour choisir la meilleure méthode à

appliquer dans chaque étape, en se basant principalement sur des critères visuels, à cause de l'absence de critères standards d'évaluation.

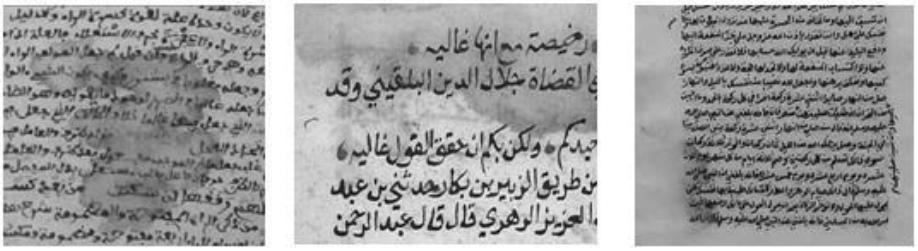
4.4.1.1. Binarisation

Pour choisir la meilleure méthode de binarisation à utiliser dans notre système, nous avons effectué une étude comparative de plusieurs méthodes de binarisation les plus connues dans la littérature. Pour les méthodes globales, nous avons choisi le seuillage global fixe, la méthode d'Otsu [OTS 79], ISODATA [VEL 80], Kapur [KAP 85], Cheng et Chen [CHE 98], Li-Lee [LI 93], avec une version globale de la méthode Nick [KHU 09]. Pour les méthodes locales nous avons choisi la méthode de Bernsen [BER 86], Niblack [NIB 86], Sauvola [SAU 97], Wolf [WOL 02], Nick [KHU 09] et la segmentation hiérarchique floue [GAD 00] de Gadi et Benslimane.

Pour les méthodes locales, nous avons testé plusieurs valeurs pour les paramètres *k*, *R* et la taille de fenêtre glissante *r*, et nous avons choisi celles qui donnent les meilleurs résultats. Ainsi, pour la méthode de Bernsen nous avons fixé *r*=15×15, pour la méthode de Niblack, *k*= -0.2 et *r*=25×25, pour la méthode de Sauvola, *k*= 0.2, *R*= 128 et *r*=15×15. Pour la méthode de Wolf les valeurs *k*= 0.2 et *r*= 7×7 et pour la méthode Nick, *k*= -0.2 et *r*=19×19.

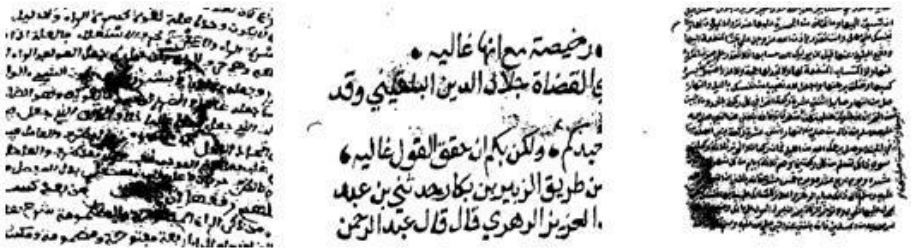
Les résultats d'application de ces algorithmes sur trois images d'anciens documents arabes sont illustrés par la figure 4.4 suivante :

Images originales

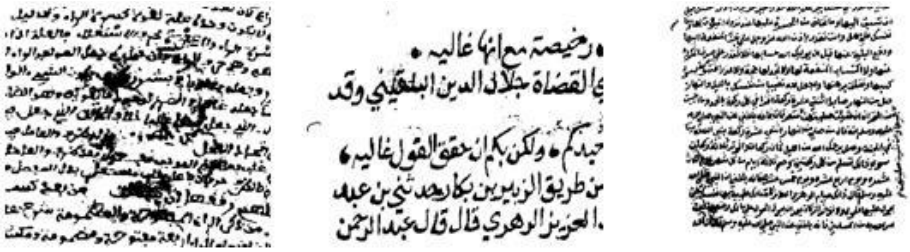


Méthodes globales

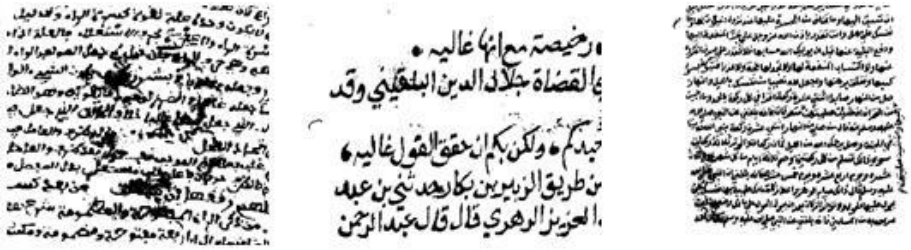
Absolue S=127



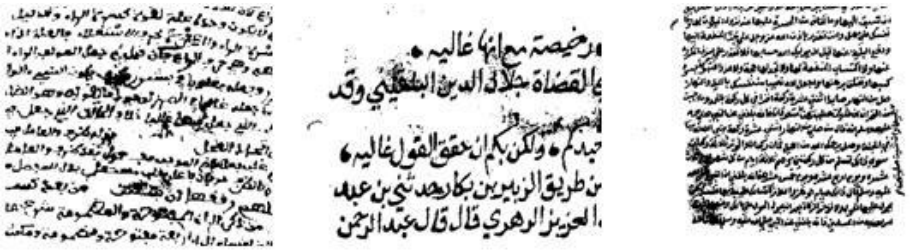
Otsu



ISODATA

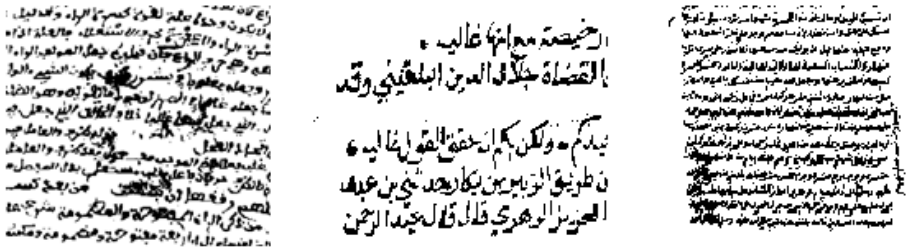


Kapur



Cheng et Chen

Li et Lee



Nick
k= -0.1

Méthodes locales (r désigne la taille de fenêtre)

Bernsen
r=15×15

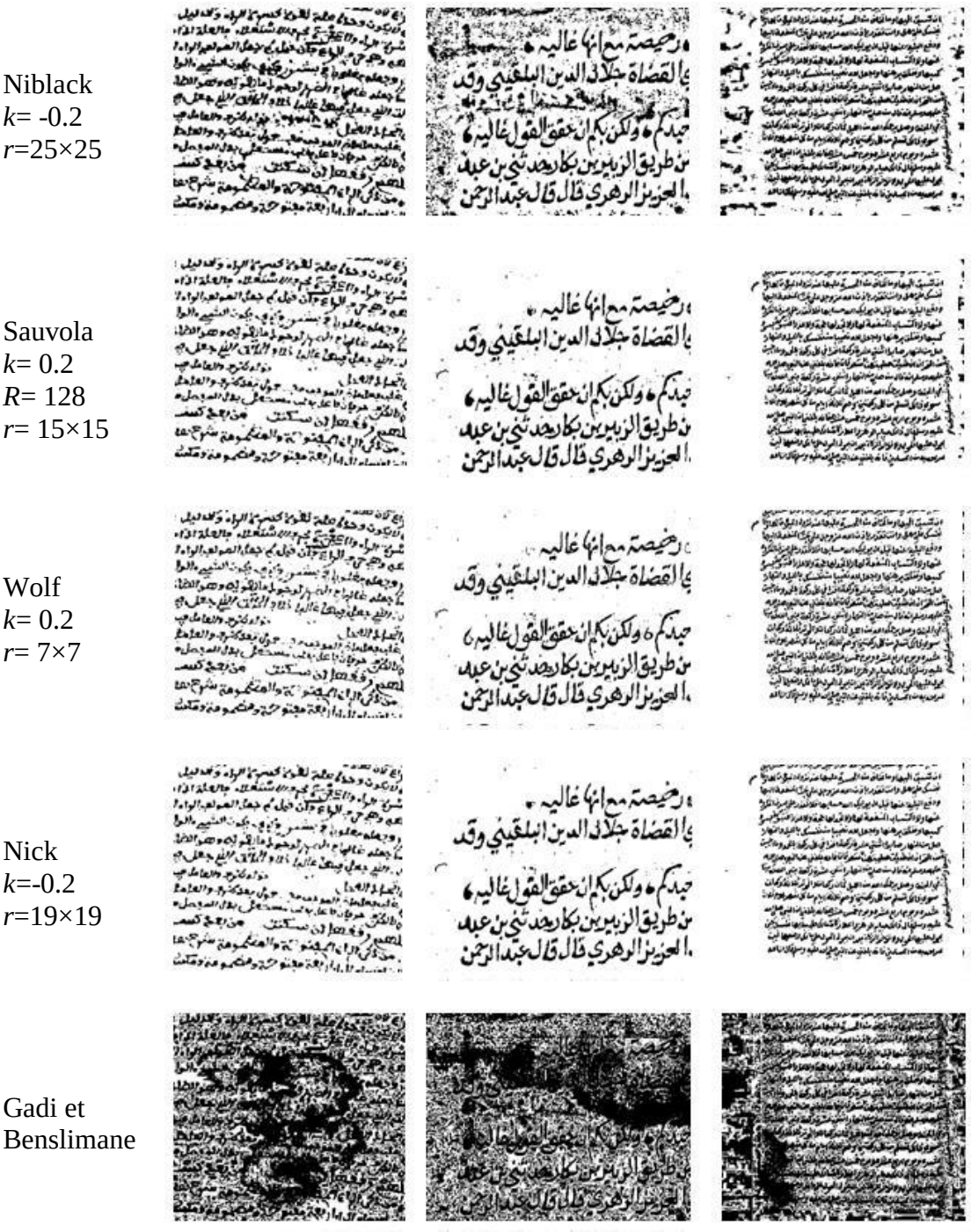


Figure 4.4 : Résultats d'application des algorithmes de seuillage sur 3 images de documents arabes anciens

Discussion

Dans plusieurs travaux de comparaison des méthodes de binarisation d'images de documents, l'évaluation est basée sur le calcul du taux de reconnaissance des caractères présents dans les documents en utilisant un logiciel commercial de reconnaissance de l'écriture comme par exemple ABBYY FineReader. Dans d'autres travaux, l'évaluation est effectuée en calculant une distance (distance d'édition par exemple) entre les images résultantes de la binarisation et des images de vérité terrain (*Ground truth images*).

L'application de l'OCR comme critère d'évaluation, pour notre cas, n'est pas possible à cause de l'absence de logiciels fiables de reconnaissance de l'écriture arabe manuscrite pour l'utiliser dans l'évaluation. Comme les images de vérité terrain ne sont pas disponibles pour notre collection, l'utilisation de la distance avec les images résultantes de la binarisation est impossible aussi.

Comme plusieurs travaux de comparaison des méthodes de binarisation (lorsque les images de vérité terrain ne sont pas disponibles), nous nous limitons dans notre étude comparative par une évaluation visuelle. Nous avons présenté les images résultantes de la binarisation de 20 images en utilisant les différentes méthodes à un groupe de 50 visionneurs. Ils ont mis leurs observations indépendamment pour chaque image de test, en choisissant la méthode avec laquelle le résultat de binarisation est le meilleur. Finalement, nous calculons pour chaque méthode combien de fois elle est choisie comme la meilleure. Le tableau 4.1 résume les résultats obtenus pour chaque méthode.

Méthode	***Nb de fois où la méthode est choisie comme la meilleure***
Nick (locale)	315
Otsu	180
ISODATA	145
Niblack	92
Nick (globale)	83
Cheng et Chen	58
Sauvola	41
Wolf	39
Li-Lee	20
Bernsen	19
Seuillage globale fixe	8
Kapur	0
segmentation hiérarchique floue	0

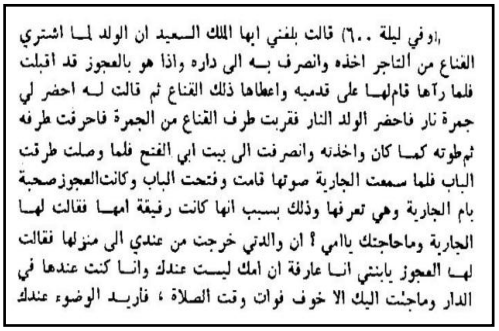
Tableau 4.1 : Scores attribués à chaque méthode de seuillage

Les observations notées par les différents visionneurs montrent que la méthode Nick est jugée comme la meilleure dans la plupart des cas.

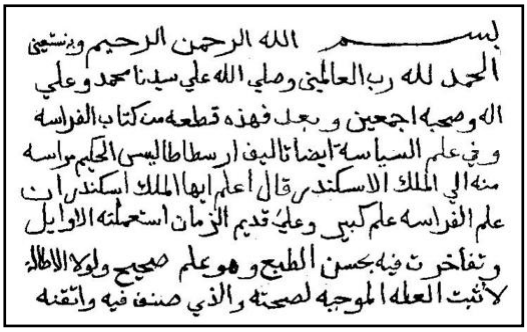
4.4.1.2. Correction de l'inclinaison

Afin de valider la méthode qu'on a utilisé pour la correction de l'inclinaison, nous avons pris un ensemble d'images de documents arabes bien alignées, et les fait roter des angles : -10, -5, 0, 5 et 10 degrés. Sur les images obtenues, on applique l'algorithme de détection de l'inclinaison (détaillé dans le chapitre précédent) qui renvoie l'angle d'inclinaison estimé. La figure 4.5 montre quelques images utilisées dans le test.

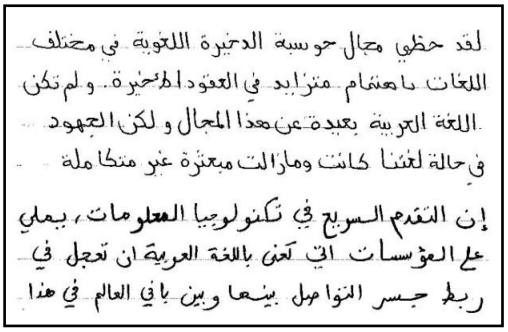
(graphique.jpg)



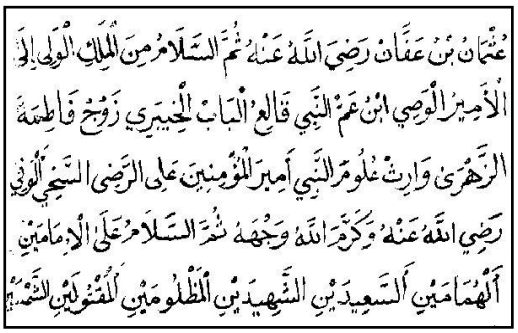
(imprimé.jpg)



(man_1.jpg)



(man_2.jpg)



(man_3.jpg)

(man_4.jpg)

Figure 4.5 : Quelques images utilisées pour l'évaluation de la méthode de correction de l'inclinaison utilisée

L'utilisation des images pivotées manuellement à des angles connus auparavant, nous permet de comparer les angles détectés avec les angles réels. Les résultats de détection de l'angle d'inclinaison des images de la figure précédente, pivotées à différents angles, sont résumés par le tableau 4.2 suivant.

Angle réel / ***Image***	***-10°***	***-5°***	***0°***	***+5°***	***+10°***
graphique	-18.94	-7.33	4.18	9.20	6.305
imprimé	-10.02	-5.01	-0.01	4.995	10.04
man_1	-9.73	-5.26	0.24	5.63	9.82
man_2	-11.09	-5.14	-0.22	4.74	9.91
man_3	-9.85	-4.84	0.14	4.98	10.37
man_4	-8.23	-3.96	0.36	6.19	10.81
Angle moyen	**-11.31**	**-5.25**	**0.78**	**5.95**	**9.54**

Tableau 4.2 : Résultats de détection des angles d'inclinaison des images de test

Discussion

A partir du tableau précédent, on trouve que la méthode donne les meilleurs résultats pour l'image de document imprimé. Ceci s'explique par la régularité de la structure des documents imprimés par rapport aux documents manuscrits (même espace interlignes, absence de chevauchements, même angle d'inclinaison dans tout le document…etc.).

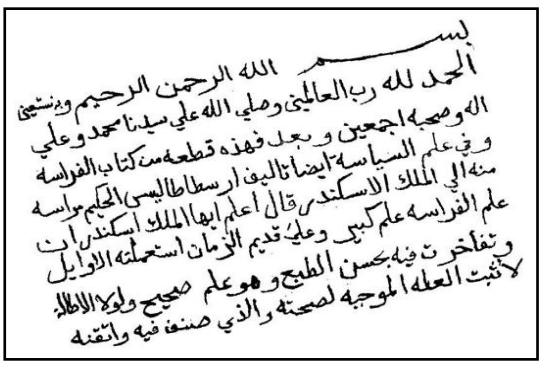
La présence d'une partie graphique dans l'image (*graphique.jpg*), complique la détection des lignes de base à partir des histogrammes des projections horizontales, ce qui influence donc sur l'estimation de l'angle d'inclinaison.

Pour les autres documents manuscrits, les angles d'inclinaison estimés sont généralement proches des angles réels. En général, l'approche utilisée se montre efficace (la différence moyenne entre les angles réels et les angles estimés est égale à 0.28°) si on considère que la plupart de documents de notre collection ne comportent pas des parties graphiques. Les défauts de calcul sont dus généralement à la non-régularité de la structure des documents manuscrits (présence de chevauchements entre les lignes du texte, espace interlignes non régulier,…etc.), en plus de la présence de plusieurs angles d'inclinaison à l'intérieur d'un même document.

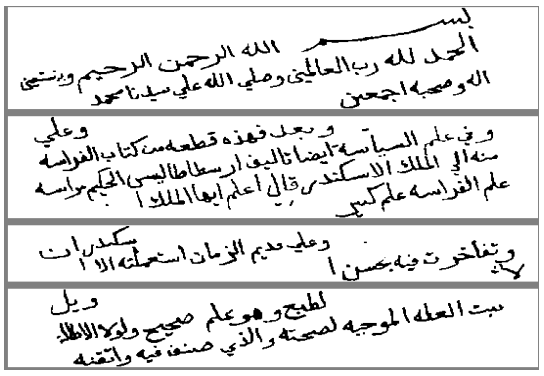
4.4.1.3. Segmentation en lignes

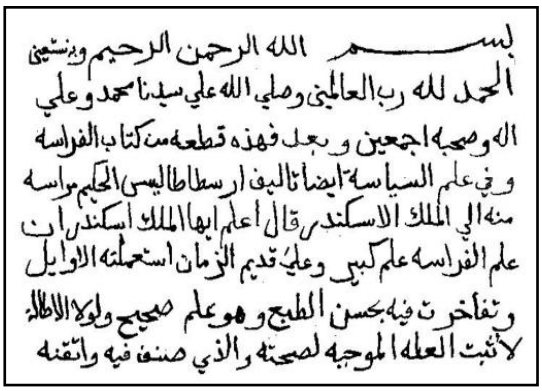
La méthode de segmentation en lignes utilisée a été testée sur plusieurs images de documents arabes imprimés et manuscrits, contenant des parties graphiques et textuelles ou bien des parties textuelles seulement. Nous avons utilisés des images de documents inclinés et d'autres alignés correctement. La figure 4.6 illustre quelques images de documents utilisées pour le test ainsi que les résultats de segmentation.

Image **Résultat de segmentation**

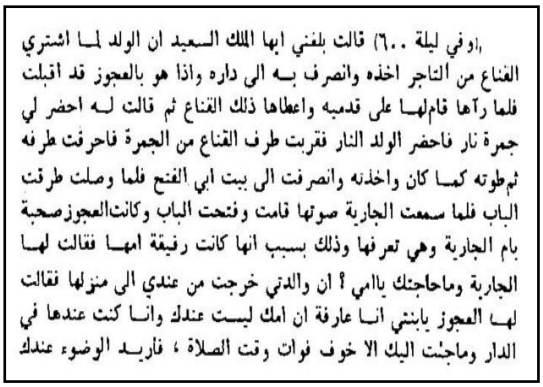


(man_1_incliné.jpg)





(man_1.jpg)



(imprimé.jpg)

(graphique.jpg)

Figure 4.6: Quelques images utilisées pour l'évaluation de la méthode de segmentation en lignes utilisée

Discussion

Comme il est remarquable, l'image de document inclinée (*man_1_incliné.jpg*) n'est pas segmentée correctement. Ceci est expliqué par le fait que la méthode employée qui se base sur la projection horizontale est sensible à l'inclinaison, ce qui engendre des faux minimas. Une telle image nécessite une étape préalable de correction de l'inclinaison. Le résultat de cette étape est donné par l'image (*man_1.jpg*). On remarque que les lignes de cette image sont bien séparées. Même chose pour l'image imprimée (*imprimé.jpg*) qui présente la meilleure régularité de structure. Pour l'image (*graphique.jpg*) la détection des lignes a échoué à cause de la présence des parties graphiques dans l'image, compliquant la détection des minimas lors de la projection horizontale.

En général, la méthode marche très bien pour les images de documents imprimés ou manuscrits bien alignés qui ne comporte pas des parties graphiques (ce qui est garanti dans notre collection), la seule contrainte imposée est que les caractères de deux lignes voisines ne doivent pas se toucher à plusieurs endroits. Quelques chevauchements sont tolérés.

4.4.1.4. Segmentation en sous-mots

La segmentation en sous-mots ne pose pas des grands problèmes, tous les pixels noirs connectés sont considérés comme une unité distincte (composante connexe). En suite, les points diacritiques sont attribués au sous-mot le plus proche par analyse de proximité.

A cette étape de traitement, nous enregistrons les coordonnées des sous-mots détectés qui vont être utilisées dans l'étape de recherche afin de localiser le mot trouvé. Pour réaliser ceci, nous avons effectué deux expérimentations :

4.4.1.4.1. Première expérimentation

Comme première tentative, et en profitant d'une caractéristique intéressante du langage Java qui est la sérialisation des objets, nous avons choisi d'enregistrer les coordonnées des sous-mots sous forme d'objets sérialisés.

La sérialisation d'un objet en Java est le processus de stockage d'un objet complet sur disque ou sur tout autre système de stockage, d'où il pourra être restauré à tout moment. La sérialisation d'un objet consiste à le transformer en une séquence d'octets et à l'enregistrer sous forme de fichier sur un disque. Lorsque survient une requête de restauration (désérialisation) d'un objet à partir d'un fichier, la séquence d'octets est désérialisée et retrouve sa structure d'origine. La prise en charge par Java de la sérialisation se compose de l'interface **Serializable**.

Dans cette expérience, les coordonnées de chaque sous-mot sont représentées par des objets sérialisés puis enregistrées tels quelles sur disque.

Les tests effectués montrent qu'avec cette méthode le temps moyen nécessaire au chargement des fichiers et à leur désérialisation lors de l'étape de recherche est 138.6 *ms*.

4.4.1.4.2. Deuxième expérimentation

Notre objectif est de diminuer au maximum le temps nécessaire à la recherche d'une requête. Cette deuxième expérimentation est effectuée afin de diminuer le temps nécessaire au chargement des coordonnées des sous-mots et donc pour accélérer la recherche. Dans cette expérimentation, nous avons proposé d'enregistrer les coordonnées des sous-mots d'un document dans un fichier texte. Le fichier texte prend le format suivant :

- Une ligne dans le fichier texte contient les coordonnées des sous-mots d'une et une seule ligne du document

- Les coordonnées d'un sous-mot sont représentées par le point haut gauche (i_{min}, j_{min}) et le point bas droit (i_{max}, j_{max}) de son rectangle englobant, séparés par un point virgule : (i_{min}, j_{min}) ; (i_{max}, j_{max})
- Les coordonnés de deux sous-mots voisins sont séparées par un #

Un exemple d'une ligne d'un tel fichier est le suivant:

(3,428);(52,688)#(17,390);(46,401)#(17,344);(46,392)#(18,387);(44,325)#(15,334);(60,280)#(28,323);(57,219)#

Lors de la recherche, et une fois une occurrence du mot cherché est trouvée dans un document, on charge le fichier de coordonnées correspondant. A partir de ce fichier, on accède à la ligne contenant l'occurrence trouvée. Les coordonnées du mot trouvé sont retournées sous forme d'une chaîne de caractères. Cette dernière doit être analysée, segmentée puis convertie en nombres entiers.

Malgré toutes ces étapes, les tests montrent que le temps nécessaire au chargement de ces fichiers, leur analyse et utilisation est beaucoup plus petit que celui nécessaire dans la première proposition (15.6 *ms* en moyenne).

4.4.1.5. Extraction de caractéristiques

Nous avons représenté chaque caractéristique par une couleur sur l'image de document (les boucles en rouges, les points diacritiques en bleu, les hampes en violet et les jambages en vert). La figure 4.7 illustre un exemple.

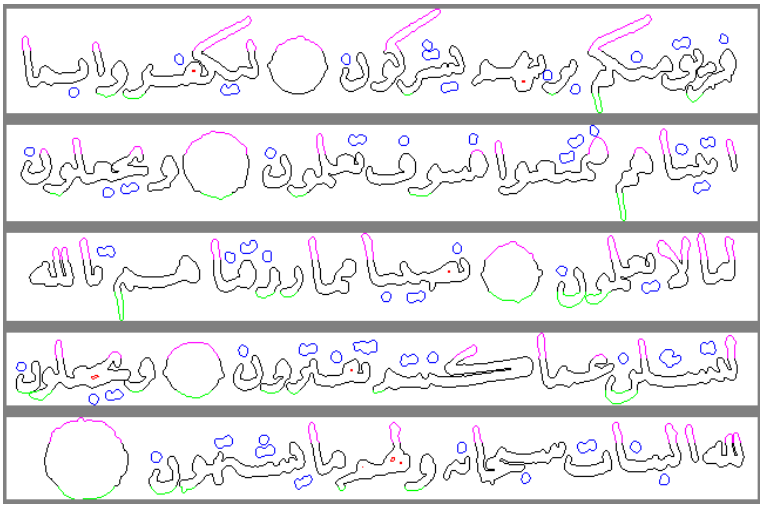


Figure 4.7: Caractéristiques extraites sur une image de document arabe ancien

Sur cette figure, on remarque que quelques caractéristiques ne sont pas bien détectées (la boucle dans la lettre ق du mot فريق par exemple) comme on remarque la détection de fausses caractéristiques (comme la hampe du lettre ف dans le mot فسوف de la deuxième ligne). Ceci est dû principalement à la non régularité de l'écriture manuscrite en général et ancienne plus particulièrement, en plus de l'étape de prétraitement qui peut enlever des caractéristiques ou ajouter d'autres (générer une boucle, effacer un point diacritique, etc.). Une mauvaise localisation de la zone médiane peut aussi entraîner une mauvaise détection des

caractéristiques. Pour toutes ces raisons, on doit utiliser des algorithmes de recherche efficaces.

4.4.2. Partie recherche

Dans cette section, nous présentons les différentes expérimentations effectuées pour évaluer les performances de la deuxième partie de notre système qui est la partie recherche de mots sur les deux bases de données présentées précédemment et sur lesquelles ont été effectués les différents traitements de la première partie du système.

Afin d'évaluer les performances de la partie recherche, nous avons utilisé une centaine de requêtes textuelles en langue arabe de longueur varie de 4 à 20 caractères (40 requêtes pour la base des anciens documents et 60 requêtes pour la base des enveloppes), pour lesquelles nous avons établi manuellement la liste de documents pertinents. Les tableaux 4.3 et 4.4 récapitulent les requêtes utilisées dans les différentes expérimentations pour interroger chacune des deux bases ainsi que le nombre de documents attendus correspondants :

Requête	Nombre de documents attendus	Requête	Nombre de documents attendus	Requête	Nombre de documents attendus
الله	16	الزمان	9	الأعمال	10
الرحمن	36	الحكيم	8	التراب	9
كتاب	20	منصور	11	مبعثرة	1
الملك	12	الحرص	2	اللغات	7
ارسطا طاليس	5	التدبير	2	الإنسان	21
رسول	3	الشيخ	19	الأبيات	9
العالم	8	رسالة	14	الأرض	6
السياسة	2	لطيفة	14	إبراهيم	5
الفلك	2	مختصرة	6	اللغة العربية	3
الدائرة	4	الصلاة	21	متكاملة	2
القوس	5	الارتفاع	3	التواصل	2
المشرق	11	المغرب	11	المقالة	3
الحافظ	9	الأنصاري	1	يعلمون	4
السماوات	6				

Tableau 4.3 : Les requêtes utilisées pour interroger la première base ainsi que leurs nombres d'occurrences respectifs.

Requête	Nombre de documents attendus	Requête	Nombre de documents attendus	Requête	Nombre de documents attendus
عنابة	128	إلى السيد	38	الذرعان	19
الطارف	42	إلى الآنسة	26	أول نوفمبر	6
فالمة	4	السيد	60	عبد القادر	9
سوق أهراس	10	الآنسة	38	بالرحال	11

5	عين الباردة	10	حي البرتقال	8	تبسة
8	حجر الديس	18	سيدي عمار	2	خنشلة
4	عين اعلام	6	جبهة التحرير الوطني	4	بجاية
5	سيدي قاسي	4	حي الأبطال	2	تيبازة
14	دائرة	10	حي المنظر الجميل	6	لبنى
38	ولاية	4	حي وادي الذهب	8	فارس
3	بالفودار	6	حي الزهور	4	فاطمة الزهراء
2	سيدي حرب	6	حي المقاومة	8	فاطمة
2	بوساحة	4	حي الشهداء	4	سامية
4	السهل الغربي	4	حي سيدي علي	4	توفيق
2	بن ناجي	4	حي جندي علي	4	زينب
7	سونلغاز	6	حي الصفصاف	4	منى
5	حي القدس	4	حي الياسمين	4	أمال
5	باجي مختار	6	بن مهيدي	2	بحيرة الطيور
5	الحي العمراني	8	بوحجار	4	عين العسل
16	بلدية	4	القالة	70	عمارة

Tableau 4.4 : Les requêtes utilisées pour interroger la deuxième base ainsi que leurs nombres d'occurrences respectifs.

Les expériences présentées dans cette section sont destinées à évaluer chaque étape de recherche séparément (recherche dans les index et recherche approximative dans les fichiers de codes), pour mieux connaître l'apport de chaque étape. Nous avons testé aussi une recherche exacte dans les fichiers de codes pour la comparer avec la recherche approximative et pour voir si elle porte d'intérêt pour nous.

L'évaluation des résultats obtenus est effectuée en termes de rappel, précision et temps de réponse. Le rappel et la précision sont les deux principaux facteurs permettant d'évaluer un système de recherche d'information (SRI) [KOS 03] [BEN 09b]: Le rappel mesure la capacité du SRI à sélectionner tous les documents pertinents. Il donne une indication sur le nombre de documents pertinents trouvés par rapport au nombre total de documents pertinents pour la requête, tandis que la précision mesure la capacité du système à rejeter tous les documents non pertinents. Elle donne une indication sur la proportion des documents pertinents renvoyés par le SRI. Comme la recherche est effectuée en-ligne, le temps de réponse évalue la capacité de notre système à répondre aux requêtes des utilisateurs au bout d'un temps raisonnable.

4.4.2.1. Expérience 1 : Recherche dans les index

Le but de cette expérience est d'évaluer la première étape de recherche qui est la recherche dans les index des documents en termes de rappel, précision et surtout de temps de réponse pour valider notre proposition de procéder cette étape de recherche.

Comme nous avons déjà dit, la recherche dans les index des documents est effectuée en utilisant la technique d'arbres de suffixes. Dans cette expérience, nous avons testé plusieurs algorithmes de recherche exacte, à savoir l'algorithme brute force, Boyer Moore [BOY 77] et Knuth Morris et Pratt [KNU 77] pour pouvoir les comparer avec la technique d'arbres de suffixes. Les tableaux 4.5 et 4.6 récapitulent le rappel, la précision, et le temps de réponse

moyens obtenus dans ce mode de recherche en utilisant les différents algorithmes et en interrogeant les deux bases par les différentes requêtes :

Critères / *Méthodes*	*Rappel*	*Précision*	*Temps de réponse (ms)*
Brute Force	0.147	1	190.180
Boyer Moore	0.147	1	211.415
Knuth Morris Pratt	0.147	1	190.390
Arbre de suffixes	0.147	1	153.225

Tableau 4.5 : Résultats de recherche dans les index pour la base des anciens documents

Critères / *Méthodes*	*Rappel*	*Précision*	*Temps de réponse (ms)*
Brute Force	0.194	1	220.250
Boyer Moore	0.194	1	311.625
Knuth Morris Pratt	0.194	1	220.375
Arbre de suffixes	0.194	1	182.125

Tableau 4.6 : Résultats de recherche dans les index pour la base des enveloppes

Les deux tableaux précédents montrent une précision moyenne parfaite (100%) des résultats de la recherche et ça sur les deux bases et avec les différents algorithmes employés. Ceci vient du fait que tous les documents retournés comme résultats contiennent surement des vraies occurrences de la requête cherchée, parce que l'indexation des documents a été effectuée manuellement.

De l'autre coté, le rappel moyen obtenu avec ce mode de recherche sur les deux bases n'est pas suffisant (presque 15% pour la base d'anciens documents et 20% pour la base des enveloppes). En effet, plusieurs documents pertinents ont été ignorés lors de la recherche, parce que la requête cherchée n'existe pas dans les index de ces documents (le temps où les documents contiennent cette requête). On remarque aussi que le rappel obtenu sur la base des enveloppes est plus grand que celui obtenu sur la base des anciens documents. Ceci est justifié par le fait que les enveloppes contiennent beaucoup moins de mots que les anciens documents ce qui augmente la probabilité que les requêtes cherchées existent dans les index des documents, et augmente donc le rappel.

En ce qui concerne le temps de réponse, il nous montre (comme il est attendu) que la technique d'arbres de suffixes est plus rapide que les autres algorithmes employés, ce qui soutient le choix d'utilisation de cette technique d'arbres de suffixes dans cette étape de recherche.

4.4.2.2. Expérience2 : Recherche exacte dans les fichiers de codes

Comme deuxième étape de recherche, nous avons expérimenté et évalué une recherche exacte dans les fichiers de codes. Dans cette expérience, nous avons testé trois algorithmes de recherche exacte de la requête dans les fichiers de codes à savoir l'algorithme brute force, Boyer Moore [BOY 77] et Knuth Morris et Pratt [KNU 77]. Le rappel, la précision et le temps de réponse moyens obtenus en utilisant les différents algorithmes et en interrogeant les deux bases par les différentes requêtes sont résumés par les deux tableaux 4.7 et 4.8:

Critères / ***Méthodes***	***Rappel***	***Précision***	***Temps de réponse (ms)***
Brute Force	0.297857143	0.9824	752.75
Boyer Moore	0.297857143	0.9824	416
Knuth Morris Pratt	0.297857143	0.9824	404.125

Tableau 4.7 : Résultats de recherche exacte dans les fichiers de codes pour la base des anciens documents

Critères / ***Méthodes***	***Rappel***	***Précision***	***Temps de réponse (ms)***
Brute Force	0.2436944	0.978571429	812.36
Boyer Moore	0.2436944	0.978571429	521.85
Knuth Morris Pratt	0.2436944	0.978571429	502.03

Tableau 4.8 : Résultats de recherche exacte dans les fichiers de codes pour la base des enveloppes

L'analyse des deux tableaux précédents nous montre que la précision de la recherche sur les deux bases est diminuée par rapport au mode de recherche précédent. De l'autre coté, le rappel augmente pour les deux bases et ça parce que la recherche dans les fichiers de codes retourne plusieurs occurrences qui ont été ignorées avec le mode de recherche précédent (la requête existe dans le fichier de codes d'un document mais pas dans ses index). Malgré cette augmentation de rappel avec ce mode de recherche (de presque le double dans le cas de la première base), il reste insuffisant pour répondre aux besoins des utilisateurs.

En plus de ces algorithmes, nous avons expérimenté aussi une recherche en utilisant la technique d'arbres de suffixes. Pour la construction des arbres de suffixes, nous avons implémenté l'algorithme de McCreight [MCC 76]. La construction des arbres de suffixes peut être effectuée en-ligne lors de la recherche ou bien hors-ligne dans la première phase d'analyse. Nous avons expérimenté les deux modes afin de comparer leurs performances.

4.4.2.2.1. Construction en ligne

Dans ce mode, lorsque l'utilisateur saisit une requête, le système procède une recherche comme suit : il charge en mémoire les fichiers de codes un par un, pour chaque fichier chargé il construit l'arbre de suffixes correspondant puis cherche la requête dans cet arbre.

Malheureusement, les différents tests effectués montrent que la recherche dans ce mode est très lente (le temps de réponse moyen à une requête est 18033 *ms*). En effet, la construction en ligne des arbres de suffixes nécessite un temps $O(n^2)$, pour un fichier contenant n caractères, ce qui augmente par conséquence le temps de réponse. Par contre le temps de recherche dans les arbres est négligeable devant le temps de construction.

4.4.2.2.2. Construction hors-ligne

Dans ce mode, les arbres de suffixes sont construits dans la première phase d'analyse, et enregistrés sur disque. La première idée qui est venue en tête, est de construire un arbre de suffixes pour chaque fichier de codes. Pour enregistrer les arbres de suffixes sur disque, nous avons testé deux méthodes :

1- En considérant les arbres de suffixes comme des graphes orientés, la première méthode consiste à implémenter les arbres par des matrices de transitions, dont les lignes correspondent aux nœuds et les colonnes désignent les étiquettes des branches des arbres, et de les enregistrer dans des fichiers Excel. Lors de la recherche d'une requête, le système charge chaque fichier Excel correspondant à un arbre de suffixes en mémoire, le transforme en une matrice de transitions, puis procède une recherche dans cette matrice.
2- Dans la deuxième méthode nous avons enregistré les arbres de suffixes tels quels sur disque sous forme d'objets sérialisés, en profitant de la caractéristique de sérialisation des objets du langage Java. Dans cette méthode, lorsqu'on veut chercher une requête, le système charge les arbres de suffixes tels quels en mémoire un par un, puis cherche la requête dans ces arbres.

Le problème avec ces méthodes est le temps de chargement très grand des arbres de suffixes en mémoire, ce qui alourdi considérablement la recherche (en moyenne 15891 *ms* pour la première méthode et 12247 *ms* pour la deuxième). Ceci est dû principalement aux tailles très grandes des arbres de suffixes construits pour chaque fichier (80000 nœuds en moyenne).

La deuxième idée que nous avons implémenté, consiste à construire un arbre de suffixes pour chaque ligne de chaque fichier de codes à part. Ceci permet de diminuer la taille des arbres de suffixes construits et par conséquent le temps nécessaire à leur chargement, mais augmente le nombre d'arbres construits. Pour la sauvegarde des arbres de suffixes nous avons testé les deux méthodes précédentes.

Les résultats expérimentaux montrent qu'avec cette idée, le temps de recherche est diminué considérablement, mais il reste très grand (en moyenne 7966 *ms* pour la première méthode et 5090 *ms* pour la deuxième).

Comme nous avons vu, les différents tests effectués nous montrent que le temps de réponse moyen à une requête en utilisant la technique d'arbres de suffixes est très grand par rapport au

temps de réponse nécessaire avec les trois algorithmes précédents. Le rappel et la précision obtenus avec cette technique sont ceux obtenus avec les trois algorithmes de recherche exacte précédents.

4.4.2.3. Expérience 3 : Recherche approximative

Dans cette expérience, nous avons utilisé deux mesures de distance (distance d'édition et distance de Jaro-Winkler) pour la recherche des occurrences approximatives de la requête dans les fichiers de codes correspondant aux documents.

Comme la recherche approximative permet de retourner des occurrences proches de la requête à *k* erreurs près (distance maximale pour considérer une occurrence comme pertinente), nous avons testé dans nos expérimentations pour chacune des deux mesures de distance (distance d'édition et distance de Jaro-Winkler), plusieurs valeurs du seuil *k* afin de choisir celle qui maximise le rappel et la précision de la recherche.

Nous présentons dans la suite les résultats obtenus pour les deux mesures de distance.

4.4.2.3.1. Distance d'édition

Pour le calcul de la distance d'édition, nous avons utilisé deux algorithmes : le premier est l'algorithme proposé par Wagner et Fischer [WAG 74] qui consiste à calculer la matrice des distances entre deux chaînes de caractères données et le deuxième algorithme est celui proposé par Ukkonen [UKK 85] qui consiste à ne calculer qu'une partie de la table des distances.

Les tableaux 4.9 et 4.10 suivants récapitulent le rappel, la précision, et le temps de réponse moyens obtenus en utilisant les deux méthodes de calcul de la distance d'édition avec plusieurs valeurs de *k*, et en interrogeant les deux bases par les différentes requêtes.

Valeur de k	***Méthode 1* [*WAG 74*]**			***Méthode 2* [*UKK 85*]**		
	Rappel	***Précision***	***Temps de réponse (ms)***	***Rappel***	***Précision***	***Temps de réponse (ms)***
0	0.297857143	0.98240000	619.125	0.297857143	0.98240000	339.00
1	0.529166667	0.72723214	541.125	0.529166667	0.72723214	338.00
2	0.839456202	0.68648825	536.875	0.839456202	0.68648825	341.25
3	1	0.62312500	556.750	1	0.62312500	353.75
4	1	0.39253000	563.110	1	0.39253000	346.89
5	1	0.11250100	578.650	1	0.11250100	339.20

Tableau 4.9 : Rappel, précision et temps de réponse moyens pour la base des anciens documents et avec les deux méthodes de calcul de la distance d'édition

Valeur de k	Méthode 1 [WAG 74]			Méthode 2 [UKK 85]		
	Rappel	Précision	Temps de réponse (ms)	Rappel	Précision	Temps de réponse (ms)
0	0.243694400	0.978571429	721.15	0.243694400	0.978571429	405.36
1	0.383612920	0.768045522	748.32	0.383612920	0.768045522	411.89
2	0.672036681	0.671100339	701.99	0.672036681	0.671100339	413.23
3	0.871713272	0.567266443	803.11	0.871713272	0.567266443	402.00
4	0.935992844	0.547570119	736.00	0.935992844	0.547570119	389.79
5	0.985658915	0.531372848	755.69	0.985658915	0.531372848	392.75
6	1	0.416013226	787.43	1	0.416013226	412.12
7	1	0.329079179	790.30	1	0.329079179	405.03

Tableau 4.10: Rappel, précision et temps de réponse moyens pour la base des enveloppes et avec les deux méthodes de calcul de la distance d'édition

4.4.2.3.2. Distance de Jaro-Winkler

Le rappel, la précision et le temps de recherche moyens obtenus en utilisant la distance de Jaro-Winkler [WIN 99] avec plusieurs valeurs de *k* et en interrogeant les deux bases par les différentes requêtes sont résumés par les deux tableaux 4.11 et 4.12 suivants:

Valeur de k	Rappel	Précision	Temps de réponse (ms)
0.00	0.297857143	0.98240000	572
0.01	0.312541283	0.93750000	613
0.02	0.372023810	0.83333333	537
0.03	0.791666667	0.64096791	572
0.04	0.833333333	0.53647186	589
0.05	0.937501500	0.50343615	602
0.06	1	0.47010430	590
0.07	1	0.38850630	598

Tableau 4.11: Rappel, précision et temps de réponse moyens pour la base des anciens documents en utilisant la distance de Jaro-Winkler

Valeur de k	Rappel	Précision	Temps de réponse (ms)
0.00	0.243694400	0.978571429	622.50
0.01	0.272959834	0.898571429	678.23
0.02	0.376226722	0.877551020	596.00
0.03	0.552354033	0.836904762	599.32

0.04	0.651388866	0.772164240	612.00
0.05	0.749364000	0.711803300	603.47
0.06	0.879509317	0.660912430	622.00
0.07	0.927804683	0.601744935	665.00
0.08	0.961240310	0.585478978	644.96
0.09	1	0.463174777	638.71
0.10	1	0.367143626	603.18

Tableau 4.12 : Rappel, précision et temps de réponse moyens pour la base des enveloppes en utilisant la distance de Jaro-Winkler

4.4.2.3.3. Discussion

A partir des tableaux 4.9 et 4.10, les deux méthodes de calcul de la distance d'édition testées sur les deux bases présentent le même rappel et la même précision, pour toutes les valeurs de *k*, mais le temps de réponse de la deuxième méthode est plus petit que celui de la première méthode par presque la moitié. Ce gain important de temps de recherche est justifié par le fait que la deuxième méthode de recherche ne calcule qu'une partie de la table des distances, ce qui permet de sauver jusqu'à 50% de travail et accélère donc la recherche.

Les résultats expérimentaux montrent aussi que le rappel et la précision de la recherche dépendent de la valeur du seuil *k*, et varient d'une manière inverse (lorsque le rappel augmente, la précision diminue et inversement), et ça pour le deux mesures de distances. En effet la meilleure précision est obtenue avec la valeur 0 de *k*. par contre avec cette valeur le rappel est le plus mauvais. Ceci est justifié par le fait que *k*=0 signifie que le système ne retourne que les occurrences exactes, ce qui augmente la précision de la recherche jusqu'à atteindre 100% dans plusieurs requêtes. Mais comme les fichiers de codes ne sont pas précis, la recherche avec *k*=0, ignore plusieurs documents pertinents, contenant des occurrences proches de la requête, et diminue donc le rappel.

Lorsque le seuil *k* (le nombre d'erreurs permis) augmente, le nombre de documents retournés augmente, et le rappel de la recherche augmente aussi jusqu'à atteindre 100%. Pour la base des anciens documents, le rappel maximal est obtenu à partir de *k*=3 dans le cas de la distance d'édition, et *k*=0.06 dans le cas de la distance de Jaro-Winkler, et pour la base des enveloppes le rappel maximal est obtenu avec *k*=6 dans le cas de la distance d'édition, et *k*=0.09 dans le cas de la distance de Jaro-Winkler. Avec ces valeurs, la précision est la plus mauvaise.

Comme nous avons dit, le rappel et la précision varient dans l'ordre inverse, la meilleure valeur de *k* est donc celle qui présente le meilleur compromis entre le rappel et la précision. Les différents tests effectués, montrent que le meilleur compromis entre le rappel et la précision n'est pas constant pour toutes les requêtes, mais il diffère selon la taille des requêtes ou plus précisément des codes des requêtes. Le tableau 4.13 illustre quelques exemples.

Requête		Code de la requête	Taille	Meilleure valeur de k	
				distance d'édition	Distance de Jaro-Winkler
Base des anciens documents	الملك	h#hbhhp	7	1	0.03
	ارسطا طاليس	h#j#bhh#bhh#hqj	15	3	0.05
	كتاب	hph#q	5	1	0.02
	صلى الله	bhj#h#hhb	9	2	0.04
Base des enveloppes	عنابة	ph#qbp	6	2	0.05
	سوق أهراس	bj#pbj#ph#bbj#h#j	17	5	0.09
	فاطمة	bph#bhbbp	9	3	0.07
	سيدي عمار	q#jq#bh#j	9	3	0.07

Tableau 4.13 : Meilleures valeurs de k pour quelques requêtes en utilisant les deux mesures de distance

Pour tenir compte de cette remarque, nous avons proposé d'adapter la valeur de *k* en fonction des tailles des requêtes (l'attribution des valeurs de *k* est effectuée par expériences). La valeur de *k* est attribuée comme suit :

a. Distance d'édition :

Si la taille de la requête <= 7 alors *k*=1 ;

Sinon, si 7 < taille de la requête < 12 alors *k*=2 ;

Sinon *k*=3 ;

b. Distance de Jaro-Winkler :

Si la taille de la requête <= 5 alors *k*=0.02 ;

Sinon, si 5< taille de la requête <=8 alors *k*=0.03

Sinon, si 8< taille de la requête <=12 alors *k*=0.04

Sinon *k*=0.05

4.4.2.3.4. Combinaison des deux mesures

Après avoir évalué les deux mesures de distance et choisir les meilleurs valeurs du seuil, nous avons combiné les deux mesures : distance d'édition et distance de Jaro-Winkler afin d'obtenir les meilleurs résultats. Donc finalement, les documents retournés sont ceux retournés par la distance d'édition ou de Jaro-Winkler, et bien sûr les documents retournés par les deux mesures sont en tête de liste.

Considérons des valeurs de seuils optimales pour différentes longueurs de mots, notre système atteint une précision de 72% tout en obtenant un rappel de 89% pour la base des anciens documents, et 82% de rappel et 66% de précision pour la base des enveloppes.

4.5. Conclusion

Dans ce chapitre, nous avons présenté les différentes expérimentations effectuées afin d'évaluer les performances de notre système de recherche d'images de documents arabes anciens détaillé dans le chapitre 3. L'évaluation de notre système concerne l'évaluation de ces deux parties : partie analyse et partie recherche séparément.

Le système a été testé sur deux bases : une contenant 100 images d'anciens documents arabes, et l'autre composée de 298 images d'enveloppes postales algériennes. La première base a été construite par nous en l'absence d'une base standard pour les anciens documents arabes manuscrits, et la deuxième base afin de tester les performances de notre système sur un autre type d'images pour lequel l'application n'était pas conçue.

Les résultats expérimentaux obtenus (en termes de rappel et précision) sur la base des anciens documents, sont à notre avis très satisfaisants, si on considère les problèmes de traitement de l'écriture arabe et des documents anciens. Pour l'autre base, il nous montre aussi que les résultats obtenus sont très bons si on considère que l'application n'était pas destinée à ce type de documents. Pour le temps de réponse, la recherche dans les index de documents comme première étape de recherche nous permet de diminuer le temps de recherche dans le cas où la requête est trouvée dans ces index. Si ce n'est pas le cas, le système procède une recherche approximative en combinant deux mesures de distance et le temps de réponse reste acceptable.

Les résultats obtenus (du point de vue rappel et précision), montrent la faisabilité et robustesse de la démarche employée, et sa généralité pour qu'elle puisse être employée sur d'autres types de documents.

Conclusion générale et perspectives

1. Conclusion

Le travail adressé dans ce mémoire s'inscrit dans la démarche de sauvegarde et de valorisation de données patrimoniales dont la communauté internationale a pris conscience de l'intérêt. Nous s'intéressons dans ce mémoire aux images d'anciens documents arabes qui portent une richesse scientifique et culturelle inestimable, et qui encourent une dégradation progressive à cause de plusieurs facteurs, et en plus ils ne sont accessibles que par un petit nombre de personnes. Notre contribution concerne essentiellement le développement d'un système de traitement et de recherche d'images d'anciens manuscrits arabes sans recourir à une reconnaissance du contenu afin d'éviter le coût élevé et l'effort ardu de la reconnaissance.

En effet, la recherche d'images de documents par le contenu nécessite une analyse détaillée de la structure physique et logique conjointement avec une description des objets figurant sur le document. Nous nous sommes focalisés dans cette première ébauche sur les parties textuelles des documents arabes manuscrits. Le texte dans les documents arabes anciens constitue la grande partie des informations sur le contenu. Certes une panoplie de méthodes de reconnaissance de l'écriture arabe manuscrite existe dans les laboratoires de recherche, mais aucune ne peut prétendre la prise en compte des anciennes écritures sur les documents historiques. Une variabilité et diversité des formes des mots et caractères sont observées rendant l'automatisation de la tâche très ardue. Une recherche et indexation par le contenu s'avèrent également un défi majeur pour les recherches actuelles dans le domaine de la recherche d'images de documents.

Le système développé est composé de deux parties essentielles, la première partie regroupe plusieurs techniques tirées principalement du domaine de l'analyse de documents : prétraitements, segmentation, extraction de caractéristiques et codage, ayant comme but de représenter chaque document de notre collection par une signature constituée de l'ensemble d'information pertinente qu'on peut extraire de l'image du document. Cette signature contient les caractéristiques structurelles extraites à partir des sous-mots notamment les hampes, jambage, boucles et points diacritiques, et elle est codée en ASCII ce qui nous a permis d'employer facilement les techniques les plus aboutissantes de recherche d'information à savoir les techniques de recherche approximative et les arbres de suffixes.

La deuxième partie de notre système qui est la partie recherche est représentée par une interface, permettant à l'utilisateur de chercher des documents, en formulant une requête textuelle en langue Arabe. Le système répond à cette requête en retournant un ensemble de documents jugés pertinents pour l'utilisateur. Pour ce faire, nous procédons en deux étapes de recherche. Dans la première étape de recherche, nous effectuons une recherche exacte dans les index de documents crées dans la première partie d'analyse, et nous utilisons pour cela la technique d'arbres de suffixes qui se montre une solution rapide et efficace. Procéder cette étape de recherche nous permet de diminuer considérablement le temps de recherche lorsque la requête cherchée est présente dans les index des documents. Ensuite, nous effectuons une recherche dans les fichiers de codes (signatures) correspondants aux documents non retournés par la première étape de recherche. Dans cette étape nous avons proposé d'employer une recherche approximative, car les fichiers de codes ne sont pas précis et dans certains cas ils

sont incomplètes, à cause de plusieurs facteurs : non discrimination des caractéristiques employées, inefficacité des algorithmes de traitements utilisés sur des images de mauvaise qualité…etc. Nous avons utilisé dans cette étape deux mesures de distances entre les chaînes : la célèbre distance d'édition, et celle de Jaro-Winkler. Pour chacune des deux mesures, nous avons proposé d'adapter le nombre d'erreurs permis selon la taille de la requête afin d'obtenir le meilleur compromis entre le rappel et la précision. Le résultat final de cette étape est donc la combinaison des résultats obtenus en utilisant les deux mesures distances.

Afin d'évaluer les performances de notre système, nous avons effectué plusieurs expérimentations et tests visant à évaluer séparément chaque étape des deux parties du système. Les tests sont réalisés sur deux bases de données : la première base est crée à cause de l'absence d'une base de données de benchmark des anciens documents arabes, et elle est composée de 100 images d'anciens documents collectées à partir du Web, et possèdent plusieurs types de dégradations. La deuxième base est composée de 298 images d'enveloppes postales algériennes, et elle est utilisée afin de tester les performances de notre système sur un autre type de documents.

Les différents tests effectués montrent qu'en considérant des valeurs optimales du seuil (nombre d'erreurs permis), le système atteint un rappel moyen de recherche égal à 89% et une précision moyenne égale à 72%, et le temps de recherche moyen à une requête se montre acceptable. Ces résultats sont à notre avis très encourageants si on considère les problèmes de traitement de documents anciens et de l'écriture arabe. Les résultats obtenus montrent donc la faisabilité et la robustesse de la démarche employée et sa généralité pour qu'elle puisse être appliquée sur d'autres types de documents.

2. Perspectives

Comme tous les travaux de recherche, plusieurs extensions sont envisageables pour améliorer le système proposé dans ce mémoire :

- Tester l'approche sur une base de données réelle et très large d'images de documents arabes anciens contenants différents types de dégradations, complexité de structure et variations d'écriture.
- Elargir le système pour qu'il regroupe d'autres étapes de traitements, tel que la restauration, et utiliser d'autres techniques plus complexes et plus performants notamment pour la correction de l'inclinaison et la segmentation.
- Ajouter un autre module permettant la séparation entre les parties textuelles et les parties graphiques, et permettre une recherche mixte texte-graphique.
- Utiliser d'autres caractéristiques de l'écriture permettant une meilleure discrimination des caractères comme par exemple les demi-boucles, les dents, les profils de sous-mots,…etc. et séparer les points diacritiques simples et multiples.
- Tester d'autres algorithmes de recherche qui peuvent apparaître plus performants et les combiner afin d'obtenir des meilleurs résultats.

- Penser à la possibilité d'ajouter un module de reconnaissance de sous-mots (en utilisant les HMM par exemple). Ce module va concurrencer avec le module de recherche pour valider les résultats de ce dernier.
- Développer une ontologie de recherche d'images de documents arabes anciens afin d'introduire la sémantique dans la recherche.

Références bibliographiques

[ADA 07] T. ADAMEK, N.E. O'CONNOR, A.F. SMEATON, « Word matching using single closed contours for indexing handwritten historical documents », International Journal on Document Analysis and Recognition, vol. 9, No. 2-4, pp. 153-165, avril 2007.

[AHM 02] M. AHMED, R. WARD, « Rotation Invariant Rule-Based Thinning Algorithm for Character Recognition », IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 24, No. 12, pp. 1672-1678, Décembre 2002.

[AHO 75] A.V. AHO, M.J. CORASICK, « Efficient String Matching: An Aid to Bibliographic Search », Communications of the ACM, vol. 18, No. 6, pp. 333-340, Juin 1975.

[ALL 03] B. ALLIER, « Contribution à la numérisation des collections : apports des contours actifs », Thèse de doctorat, Institut National des Sciences Appliquées de Lyon, 2003.

[AMI 86] A. AMIN, G. MASINI, « Machine Recognition of Multifont Printed Arabic Texts », 8th International Conference on Pattern Recognition, vol. 1, pp. 392-395, Octobre 1986.

[AMI 89] A. AMIN, J. F. MARI, « Machine Recognition and Correction of Printed Arabic Text », IEEE Transactions on Systems, Man and Cybernetics, vol. 19, No. 5, pp.1300-1306, 1989.

[AMI 00] A. AMIN, S. FISHER, « A document skew detection method using the Hough transform », Pattern Analysis and Applications, vol. 3, No. 3, pp. 243-253, 2000.

[AND 03] T. ANDERSEN, W. ZHANG, « Features for neural net based region identification of newspaper documents », International Conference on Document Analysis and Recognition, vol. 1, pp. 403-407, 2003.

[ARA 06] V.N. M. ARADHYA, G.H. KUMAR, P. SHIVAKUMARA, « Skew Detection Technique for Binary Document Images based on Hough Transform », International Journal of Information Technology, vol. 3, No. 3, pp. 194-200, 2006.

[ARI 01] N. ARICA, F. T. YARMAN-VURAL, « An overview of character recognition focused on off-line handwriting », IEEE transactions on systems, Man and Cybernetics - part C: Applications and reviews, vol. 31, No. 2, pp. 216-233, 2001.

[ARL 70] V.L. ARLAZAROV, E.A. DINIC, M.A. KRONROD, I.A. FARADZEV, « On economic construction of the transitive closure of a directed graph », Tranduction anglaise dans Soviet Math. Dokl, vol. 11, pp. 1209-1210, 1970.

[AVI 05] B.T. AVILA, R.D. LINS, « A fast orientation and skew detection algorithm for monochromatic document images », ACM symposium on Document engineering, pp. 118-126, 2005.

[AZI 02] N. AZIZI, « Combinaison de Classifieurs neuronaux basée sur la logique floue: application à la reconnaissance des mots arabes manuscrits », Mémoire de Magistère, Université de Annaba, 2002.

[BAC 98] B. BACHIMONT « Bibliothèques numériques audiovisuelles. Des enjeux scientifiques et techniques », Revue Document numérique, vol. 2, No. 3-4, pp. 219-242, Hermès, 1998.

[BAD 06] E. BADEKAS, N. NIKOLAOU, N. PAPAMARKOS, « Text binarization in color documents », International Journal of Imaging Systems and Technology, vol. 16, No. 6, pp. 262-274, 2006.

[BAE 92] R. BAEZA-YATES, G.H. GONNET, « A new approach to text searching », Communications of the ACM, vol. 35, No. 10, pp. 74-82, October 1992.

[BAE 96] R. BAEZA-YATES, G. NAVARRO, « A faster algorithm for approximate string matching », 7th Annual Symposium on Combinatorial Pattern Matching, pp. 1-23, 1996.

[BAI 92] H.S. BAIRD, H. BUNKE, K. YAMAMOTO, « Structured document image analysis », Springer-Verlag New York, 1992.

[BAI 93] H. BAIRD, « Document Image defect Models and Their Uses », 2nd International Conference on Document Analysis and recognition, pp. 62-67, 1993.

[BAI 09] S. BAI, L. LI, C.L. TAN, « Keyword Spotting in Document Images through Word Shape Coding », 10th International Conference on Document Analysis and Recognition, pp. 331-335, 2009.

[BAP 98] F. BAPST, « Reconnaissance de documents assistée: architecture logicielle et intégration de savoir-faire », Thèse de doctorat, Université de Fribourg (Suisse), 1998.

[BAR 02] N. BARBEY, J. GUILLEMAIN, G. PEOCH, P. RACT, « La renaissance du livre ancien : bilan du projet DEBORA et perspectives d'avenir », DCB, mémoire de recherche, Enssib, juin 2002.

[BEL 92b] A. BELAID, Y. BELAID, « Reconnaissance des formes : méthodes et application », InterEdition Paris, 1992.

[BEL 94] A. BELAÏD, J.C. ANIGBOGU, « Mise à contribution de plusieurs classifieurs pour la reconnaissance de textes multifontes », Revue Traitement du signal, vol. 11, No. 1, pp. 57-75, 1994.

[BEL 01] A. BELAÏD, « Reconnaissance automatique de l'écriture et du document », Pour la science (Article dans une revue de vulgarisation), 2001.

[BEN 99] A.BENNASRI, A.ZAHOUR, B. TACONET, « Extraction des lignes d'un texte manuscrit arabe », Vision Interface, pp.42-48, 1999.

[BEN 06] M. BEN HALIMA, W. BOUSSELLAA, M. CHARFI, M.A. ALIMI, « Restauration des images couleurs de documents arabes anciens basée sur les EDPs », Colloque international Francophone sur l'Ecrit et le Document, pp. 103-108, 2006.

[BEN 09] A. BENMOHAMED, T. SARI, M. SELLAMI, « Une approche semi automatique pour la recherche de documents anciens », Journées Gestion Electronique de Documents & Réseaux de Recherche en sciences et Technologies d'information, pp. 156-163, Annaba-Algérie, Mai 2009.

[BEN 09b] M. BEN AOUICHA, « Une approche algébrique pour la recherche d'information structure », Thèse de doctorat, Université Paul Sabatier de Toulouse, 08 Janvier 2009.

[BER 86] J. BERNSEN, « Dynamic thresholding of grey-level images », 8th International Conference on Pattern Recognition, pp. 1251-1255, France, 1986.

[BOR 92] C.L. BORGMAN, S.L. SIEGFRIED, « Getty's Synoname and its cousins: A survey of applications of personal name-matching algorithms », Journal of the American Society for Information Science, vol. 43, No. 7, pp. 459-476, 1992.

[BOU 06] W. BOUSSELLAA, A. ZAHOUR, B. TACONET, A. BENABDELHAFID, A. ALIMI, «

Segmentation texte /graphique : Application au manuscrits Arabes Anciens », 9ème Colloque International Francophone sur l'Ecrit et le Document, Suisse, Septembre 2006.

[BOY 77] R.S. BOYER, J.S. MOORE, « A fast string searching algorithm », Communications of the ACM, vol. 20, No. 10, pp. 762-772, October 1977.

[BOZ 97] A. BOZZI, S. CALABRETTO, « Digital Library and Computational Philology: the BAMBI (LIB -3114) project », 1st European Conference on Research and Advanced Technology for Digital Libraries, pp. 269-285, Italie, September 1997.

[CAM 04] J. CAMILLERAPP, L. PASQUER, B. COÜASNON, « Indexation automatique de formulaires anciens par reconnaissance du patronyme manuscrit », Reconnaissance des Formes et Intelligence Artificielle, pp. 1493-1502, France, Janvier 2004.

[CAR 99] S. CARLSSON, « Order structure, correspondence and shape based categories », Shape Contour and Grouping in Computer Vision. Springer LNCS 1681, pp. 58-71, 1999.

[CHA 06] C. CHATELAIN, « Extraction de séquences numériques dans des documents manuscrits quelconques », Thèse de doctorat, Université de Rouen, 5 décembre 2006.

[CHE 95] S. CHEN, R.M. HARALICK, and I.T. PHILLIPS, « Automatic text skew estimation in document images », 3rd International Conference on Document Analysis and Recognition, pp. 1153-1156, 1995.

[CHE 95b] S. CHEN, R.M. HARALICK, « Recursive Erosion, Dilation, Opening and Closin Transforms », IEEE, Bans on Image Processing, No. 3, March 1995.

[CHE 98] H.D. CHENG, J. R. CHEN, J. LI, « Threshold selection based on fuzzy c-partition entropy approach », International Conference on Pattern Recognition, vol. 31, No. 7, pp. 857-870, 1998.

[CHE 98b] M. CHERIET, H. MILED, C. OLIVIER, Y. LECOURTIER, « Visual Aspect of Cursive Arabic Handwriting Recognition », Vision Interface, pp. 262-270, 1998.

[CHE 98c] F. CHEN, D. BLOOMBERG, « Summarization of imaged documents without OCR », Computer Vision and Image Understanding**,** vol. 70, No.3, 1998.

[COU 02] B. COÜASNON, J. CAMILLERAPP, « Une méthode générique de rétroconversion de documents pour la constitution de dossiers numériques », Document Numérique, vol. 6, No. 1-2, pp. 129-144, 2002.

[CRO 86] M. CROCHEMORE, « Transducers and repetitions », Theoretical Computer Science, vol. 45, No. 1, pp. 63-86, 1986.

[CRO 01] M. CROCHEMORE, C. HANCART, T. LECROQ, « Algorithmique du texte », Edition Vuibert, 2001.

[DEF 95] O. DEFORGES, P. PIQUIN, C. VIARD-GAUDIN, D. BARBA, « Segmentation d'images de documents par une approche multi-résolution. Extraction précise des lignes de texte », Traitement du Signal, vol. 12, No. 6, 1995.

[DEV 82] P.A. DEVIJVER, J. KITTLER, « Pattern recognition, a statistical approach », Englewood Cliffs, London, 1982.

[DRI 07] F. DRIRA, « Contribution à la Restauration des Images de Documents Anciens », Thèse de doctorat, Institut National des Sciences Appliquées de Lyon, 2007.

[DUB 01] E. DUBOIS, A. PATHAK, « Reduction of bleed-through in scanned manuscripts documents », Proceeding IS&T Image processing, Image Quality, Image capture Systems, pp. 177-180, Canada, 2001.

[ELD 90] S.S. EL-DABI, R. RAMSIS, A. KAMEL, « Arabic Character System: A System Approach for Recognizing cursive Typewritten Text », International Conference on Pattern Recognition, vol. 23, No. 5, pp. 485-495, 1990.

[EIK 96] A.K. EIKVIL, « Text Page Recognition Using Grey-Level Features and Hidden Markov Models », International Conference on Pattern Recognition, vol. 29, No. 6, pp. 977-995, 1996.

[ELL 90] D.G. ELLIMAN, « A review of segmentation and contextual analysis techniques for text recognition », International Conference on Pattern Recognition, vol. 23, No. 3, pp. 316-337, 1990.

[EMP 03] H. EMPTOZ, F. LEBOURGEOIS, V. EGLIN, Y. LEYDIER, « La reconnaissance dans les images numérisées : OCR et transcription, reconnaissance des structures fonctionnelles et des méta-données », La numérisation des textes et des images : techniques et réalisations, éditions Presse de Lille 3 France, pp. 105-129, Janvier 2003.

[EMP 07] H. EMPTOZ, F. LEBOURGEOIS, « DEBORA -Digital accEs to Books of RenaissAnce », International Journal on Document Analysis and Recognition, Vol. 9, No .2-4, pp. 193-221, 2007.

[FEL 01] M. FELDBACH, K.D. TÖNNIES, « Line detection and segmentation in Historical Church registers », International Conference on Document Analysis and Recognition, pp. 743-747, Septembre 2001.

[GAD 90] T. N. GADD, « PHONIX: The algorithm », Program: Automated Library and Information Systems, vol. 24, No. 4, pp. 363-366, 1990.

[GAD 00] : T. GADI, R. BENSLIMANE, « Segmentation hiérarchique floue », Traitement du signal, vol.7, No. 1, 2000.

[GAL 90] Z. GALIL, K. PARK, « An Improved Algorithm for Approximate String Matching », SIAM Journal Computer, vol. 19, No. 6, pp. 989-999, 1990.

[GAR 06] U. GARAIN, T. PAQUET, L. HEUTTE, « On foreground-background separation in low quality document images », International Journal on Document Analysis and Recognition, vol. 8, No.1, pp. 47-63, 2006.

[GRA 00] I. GRANADO, M. MENGUCCI, F. MUGE, « Extraction de textes et de figures dans les livres anciens à l'aide de la morphologie mathématique », Colloque International Francophone sur l'Ecrit et le Document, pp. 81-90, Lyon, juillet 2000.

[GRA 03] F. GRANDIDIER, « Un Nouvel Algorithme de Sélection de Caractéristiques : Application à la Lecture Manuscrite », Thèse de Doctorat, Montréal, 2003.

[GUP 07] M.R. GUPTA, N.P. JACOBSON, E.K. GARCIA, « OCR binarization and image pre-processing for searching historical documents », International Conference on Pattern Recognition, vol. 40, pp. 389-397, 2007.

[GUS 99] Gusnard de Ventadert (nom collectif), « Les documents anciens », Document numérique, Hermès (Edition), vol. 3, No. 1-2, pp. 57-73, Juin 1999.

[HAD 06] K. HADJAR, « Une étude de l'évolutivité des modèles pour la reconnaissance de documents arabes dans un contexte interactif », Thèse de doctorat, Université de Fribourg (Suisse), 2006.

[HAL 80] P.A.V. Hall, G.R. Dowling, « Approximate string matching », Computing Surveys, vol. 12, No. 4, pp. 381-402, 1980.

[HAM 02] S. HAMEL, « Algorithmes vectoriels et bioinformatique », Thèse Présentée comme exigence partielle du doctorat en mathématiques, Université Du Québec à Montréal, Avril 2002.

[HAO 98]: H. HAOUALA, M.C. FEHRI, « Arabic document analysis and recognition a case study: official journal of the Tunisian republic », Computational Engineering in Systems Applications (CESA), pp. 9-12, Tunisia, April 1998.

[HAR 72] L.D. HARMON, « Automatic recognition of print and scripts », Proc. IEEE, vol. 60, pp. 1165-1177, 1972.

[HAR 92] R.M. HARALICK, S. CHEN, T. KANUNGO, « Recursive Opening Transform », CVPR, Champaign, June 1992.

[HEN 68] R.B. HENNIS, « The IBM 1975 optical page reader, system design », IBM Journal of Research and Development, pp. 346-353, 1968.

[HOB 97] J.D. HOBBY, T.K. HO, « Enhancing Degraded Document Images via Bitmap Clustering and Averaging », 4th International Conference on Document Analysis and Recognition, pp. 394-400, Germany, 1997.

[HU 61] M.K. HU, « Pattern recognition by moment invariants », In Proc. IRE, pp. 1428, September 1961.

[INE 01] S. INENAGA, H. HOSHINO, A. SHINOHARA, M. TAKEDA, S. ARIKAWA, G. MAURI, G. PAVESI, « On-Line Construction of Compact Directed Acyclic Word Graphs », 12th Annual Symposium on Combinatorial Pattern Matching, vol. 146, No. 2, pp. 156 - 179, 2001.

[JAR 89] M.A. JARO, « Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida », Journal of the American Statistical Association, 84, pp. 414-420, 1989.

[JIA 99] X. JIANG, H. BUNKE, D. WIDMER-KLJAJO, « Skew Detection of Document Images by Focused Nearest-Neighbor Clustering », International Conference on Document Analysis and Recognition, 1999.

[JOU 04] D. JOURDY, « Culture & Recherche n°100 février 04 », Journal du ministère de la culture et de la communication, 2004.

[JOU 06] N. JOURNET, « Analyse d'images de documents anciens : une approche texture », Thèse de doctorat, Université de La Rochelle - UFR Sciences, 1 décembre 2006.

[KAC 06] O. KACZOR, « Application d'algorithmes de bio-informatique à la recherche de patrons de conception », Mémoire présenté en vue de l'obtention du grade de Maître ès sciences (M.Sc.) en informatique, Université de Montréal, Avril 2006.

[KAN 00] T. KANUNGO, R.M. HARALICK, H.S. BAIRD, W. STUEZLE, D. MADIGAN, « A Statistical, Nonparametric Methodology for Document Degradation Model Validation », IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, No. 11, pp. 1209-1223, 2000.

[KAP 85] J.N. KAPUR, P.K. SAHOO, A.K.C. WONG, « A New method for gray-level picture threshold using the entropy of the histogram », Graphical Models and Image Processing, vol. 29, 1985.

[KEF 09] A. KEFALI, T. SARI, M. SELLAMI, « Evaluation de plusieurs techniques de seuillage d'images de documents arabes anciens », 5éme symposium international Images Multimédias Applications Graphiques et Environnements, pp. 123-134. Biskra, Algérie, Novembre 2009.

[KEF 10] A. KEFALI, T. SARI, M. SELLAMI, «Evaluation of several binarization techniques for old Arabic documents images», 1st international symposium on modelling and implementation of complex systems, pp. 88-98, Constantine, Algérie, Mai 2010.

[KHU 08] K. KHURSHID, C. FAURE, N. VINCENT, « Recherche de mots dans des images de documents par appariement de caractères », 10ème Colloque International Francophone sur l'Écrit et le Document, France, pp. 91-96, 2008.

[KHU 09] K. Khurshid, I. Siddiqi, C. Faure, N. Vincent, « Comparison of Niblack inspired Binarization methods for ancient documents », 16th Document Recognition and Retrieval Conference, USA, Jan 2009.

[KNU 77] D.E. KNUTH, J.H. MORRIS, V.R. PRATT, « Fast pattern matching in strings », SIAM Journal on Computing, vol. 6, No. 2, pp. 323-350, 1977.

[KOS 03] M. KOSKELA, « Interactive Image Retrieval Using Self-Organizing Maps », Thèse de doctorat, Helsinki University of Technology, 12 Novembre 2003.

[LAN 88] G. LANDAU, U. VISHKIN, « Fast string matching with k differences », Journal of Computer and Systems Science, vol. 37, pp. 63-78, 1988.

[LAN 89] G.M. Landau, U. Vishkin, « Fast parallel and serial approximate string matching », Journal of Algorithms, vol. 10, pp. 157-169, 1989.

[LAR 86] Librairie Larousse, editor, « Petit Larousse Illustré ». Larousse Diffusion, Paris (France), 1986.

[LAR 89] Librairie Larousse, editor, « Larousse de Poche », Larousse Diffusion, Paris (France), 1989.

[LEB 03] F. LEBOURGEOIS, H. EMPTOZ, E. TRINH, « Compression et accessibilité aux images de documents numérisés : Application au projet DEBORA », Document Numérique, vol. 7, No. 3-4, pp. 103-127. 2003.

[LEC 98] Y. LECUN, L. BOTTOU, Y. BENGIO, P. HAFFNER, « Gradient-based learning applied to document recognition », Proceedings of the IEEE, vol. 86, No. 11, pp. 2278-2324, November 1998.

[LEE 01] S.W. Lee , D.S. Ryu, « Parameter-Free Geometric Document Layout Analysis », IEEE Transactions on Pattern Analysis and Machine Intelligent, vol. 23, No. 11, pp. 1240-1256, 2001.

[LEV 66] A. LEVENSTEIN, « Binary Codes Capable of Correcting Deletions, Insertions and Reversals », Soviet Physics Doklady, vol. 10, No. 8, pp. 707-710, 1966.

[LEY 04] Y. LEYDIER, F. LEBOURGEOIS, H. EMPTOZ, « Serialized k-means for adaptative color image segmentation: Application to document images and others », Document Analysis Systems VI, 6th International workshop, pp. 252-263, ITALY, 2004.

[LI 93] C.H. LI, C.K. LEE, « Minimum Cross Entropy Thresholding », International Conference on Pattern Recognition, vol. 26, No. 4, pp. 616-626, 1993.

[LIK 94] L. LIKFORMAN-SULEM, C. FAURE, « Extracting lines on handwritten documents by perceptual grouping », In Advances in Handwiting and drawing : a multidisciplinary approach, pp. 21-38, Europia, Paris, 1994.

[LIK 95] L. LIKFORMAN-SULEM, A. Hanimyan, C. Faure, « A Hough Based Algorithm for Extracting Text Lines in Handwritten Documents », International Conference On Document Analysis and Recognition, pp. 774-777, Montréal, 1995.

[LIK 08] L. LIKFORMAN-SULEM, « Analyse d'images de documents non contraints: de la structuration a la reconnaissance », Habilitation à Diriger des Recherches, Université Pierre et Marie Curie (Paris VI), 16 Juillet 2008.

[MAH 94] A.S. MAHMOUD, « Arabic Character Recognition Using Fourier Descriptors and Character Contour Encoding », International Conference on Pattern Recognition, vol. 27, No. 6, pp. 815-824, 1994.

[MAN 96] R. MANMATHA, C. HAN, E. RISEMEN, « word spotting: a new approach to indexing handwriting », IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 631-637, 1996.

[MAR 94] Y. MARCOUX, « Les formats normalisés de documents électroniques », Intelligence artificielle et sciences cognitives, vol.6, No.1 - 2, pp. 56-65, Québec, 1994.

[MAS 80] W.J. Masek, M.S. Paterson, « A faster algorithm for computing string edit distances », Journal of Computer and System Sciences, vol. 20, pp. 18-31, 1980.

[MCC 76] E.M. MCCREIGHT, « A Space-Economical Suffix Tree Construction Algorithm », Journal of ACM, vol. 23, No. 2, pp. 262-272, 1976.

[MEH 06] Z. MEHENNAOUI, « Reconnaissance de l'écriture arabe manuscrite a base des machines à vecteurs de supports », Mémoire de Magistère, Université Badji Mokhtar- Annaba, 2006.

[MOG 00] B. MOGHADDAM, T. JEBARA, A. PENTLAND, « Bayesian face recognition », International Conference on Pattern Recognition, vol. 33, No. 11, pp. 1771-1782, November 2000.

[MON 96] A. MONGE, C. ELKAN, « The field-matching problem: algorithm and applications », 2nd International Conference on Knowledge Discovery and Data Mining, 1996.

[MOR 70] J.H. MORRIS, V.R. PRATT, « A Linear Pattern Matching Algorithm », Technical Report 40, Computing Center, University of California, Berkeley, 1970.

[MOR 06] M. MORITA, R. SABOURIN, F. BORTOLOZZI, C.Y. SUEN, « Segmentation and Recognition of Handwritten Dates: An HMM-MLP hybrid approach », International Journal on Document Analysis and Recognition, 2006.

[NAV 06] G. NAVARRO, « A guided tour to approximate string matching », Association of Computing Machinery Computing Surveys, vol. 33, pp. 31-88, 2001.

[NEW 99] W. NEWMAN, C. DANCE, A. TAYLOR, S. TAYLOR, M. TAYLOR, T. ALDHOUS, « CamWorks: A Video-based Tool for Efficient Capture from Paper Source Documents », IEEE International Conference on Multimedia Computing and Systems, vol. 2, June 1999.

[NIB 86] W. NIBLACK, « An Introduction to Digital Image Processing », Prentice Hall, pp. 115-116, 1986.

[NIY 95] D. NIYOGI, S. SRIHARI, « Knowledge-Based Derivation of Document Logical Structure », 3rd International Conference on Document Analysis and Recognition, pp. 472-475, Canada, 1995.

[OFF 06] Office québécois de la langue française, « Le grand dictionnaire terminologique », (en ligne), septembre 2006. Disponible sur : http://www.granddictionnaire.com.

[OKU 99] O. OKUN, M. PIETIKAINEN, J. SAUVOLA, « Robust Skew Estimation on Low-Resolution Document Images », International Conference on Document Analysis and Recognition, 1999.

[OLI 96] C. OLIVER, H. MILED, K. ROMEO, Y. LECOURTIER, « Segmentation and Coding of Arabic Handwritten Words », 13th International Conference on Pattern Recognition, vol. 3, Track C, pp. 264-268, October 1996.

[OTS 79] N. OTSU, « A Threshold Selection Method from Gray-Level Histograms », IEEE transactions on Systems, Man and Cybernetics, vol. 9, No. 1, pp. 62-66, 1979.

[PAL 99] G.I. PALERMO, Y.A. DIMITRIADIS, « Structured document labeling and rule extraction using a new recurrent fuzzy-neural system », 5th International Conference on Document Analysis and Recognition, pp. 181-184, India, September 1999.

[PAL 01] U. PAL, A. BELAID, C. CHOISY, « Water Reservoir Based Approach for Touching Numeral Segmentation », International Conference on Document Analysis and Recognition, pp. 892-897, 2001.

[PER 89] D. PERRIN, « Automates et algorithmes sur les mots », Ann. Telecommun, vol. 44, No. 1-2, pp. 20-33, 1989.

[PEV 06] P.A. PEVZNER, « Bio-informatique moléculaire Une approche algorithmique », collection IRIS, dirigé par Nicolas Puech, Traduction : Delphine Hachez, Springer Editions, Octobre 2006.

[PHI 93] I.T. PHILLIPS, S. CHEN, R.M. HARALICK, « CD-ROM Document Database Standard », International Conference on Document Analysis and Recognition, pp. 478-483, Japan, 1993.

[PIN 01] J.R. PINALES, « Reconnaissance hors-ligne de l'écriture cursive par l'utilisation de modèles perceptifs et neuronaux », Thèse de doctorat, Ecole Nationale Supérieure des Télécommunications, 21 septembre 2001.

[PIT 90] I. PITAS, A.N. VENETSANOPULOS, « Digital nonlinear filters », Kluwer Academic Press, 1990.

[PUJ 02] A.K. PUJARI, C.D. NAIDU, B.C. JINAGA, « An adaptive character recogniser for telugu scripts using multiresolution analysis and associative memory », ICVGIP, 2002.

[PUN 80] T. PUN, « A New method for gray-level picture threshold using the entropy of the histogram », Signal processing, vol. 2, No. 3, 1980.

[RAM 07] J.Y. RAMEL, « User driven page layout analysis of historical printed books », International Journal on Document Analysis and Recognition, 2007.

[RAT 03] T.M. RATH, R. MANMATHA, « Features for Word Spotting in Historical Manuscripts », 7th International Conference on Document Analysis and Recognition, 2003.

[RAT 07] T.M. RATH, R. MANMATHA, « Word Spotting for historical documents », International Journal on Document Analysis and Recognition, vol. 9, No. pp. 139-152, 2007.

[RIC 87] K. RICHARD et R. MICHAEL, « Efficient Randomized Pattern-Matching Algorithms », IBM Journal of Research and Development, vol. 31, No. 2, pp. 249 - 260, 1987.

[ROB 02] J. PICOCHE « Dictionnaire étymologique du français », Paris : Le Robert, Edition Gilles Firmin, collection « les usuels », 2002.

[SAN 83] D. SANKOFF, J. KRUSKAL, « TimeWarps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison », Addison-Wesley Publishing Company, 1983.

[SAN 99] D. SANKOFF, J.B. KRUSKAL, « Time Warps, String Edits and Macromolecules: the Theory and Practice of Sequence Comparison », Center for the Study of Language and Inf, 2em edition. page(s) 36, 37, 38, 41, 82, 87, 1999.

[SAR 07] T. SARI, M. SELLAMI, « State of the art of line Arabic handwriting segmentation », International Journal of Computer Processing of Oriental Languages, vol. 20, No. 1, pp.53-73, 2007.

[SAU 97] J. SAUVOLA, T. SEPPANEN, S. HAAPAKOSKI, M. PIETIKAINEN, « Adaptive document binarization », 4th International Conference on Document Analysis and Recognition, pp. 147-152, 1997.

[SAU 00] J. SAUVOLA, M. PIETIKAINEN, « Adaptive document image binarization », International Conference on Pattern Recognition, vol. 33, No. 2, pp. 225-236, 2000.

[SCH 97] C. SCHMID, R. MOHR, « Local grayvalue invariants for image retrieval », In Pattern Analysis Machine Intelligence, vol. 19, No. 5, pp. 530-534, 1997.

[SEH 05] A. SEHAD, L. MEZAI, L. SEKKAI, M. CHERIET, « Méthode de détection de l'inclinaison des documents arabes imprimés basée sur la réduction des points et a transformée de Hough », In the 3rd International Conference: Sciences of Electronic, Technologies of Information and Telecommunications, TUNISIA, March 2005.

[SEL 80] P.H. SELLERS, «The Theory and Computation of Evolutionary Distances: Pattern Recognition », Journal of Algorithms, vol. 1, pp. 359-373, 1980.

[SEN 98] A.W. SENIOR, A.J. ROBINSON, « An Off-Line Cursive Handwriting Recognition System », IEEE Transaction on Pattern Analysis and Machine Intelligent, vol. 20, No. 3, pp. 309-321, 1998.

[SHA 98] D. SHARVIT, J. CHAN, H. TEK, B. KIMIA, « Symmetry-based indexing of image databases », Journal of Visual Communication and Image Representation, 1998.

[SHA 00] G. SHARMA, « Cancellation of show-through in duplex scanning », IEEE International Conference on Image processing, vol. 2, pp. 609-612, 2000.

[SME 97] SMEATON, A. SPITZ, « Using character shape coding for information retrieval », 4th International Conference on Document Analysis and Recognition, pp.974-978, 1997.

[SMI 04] E. SMIGIEL, A. BELAID, H. HAMZA, « Self-organizing Maps and Ancient Documents, Document Analysis Systems VI », 6th international workshop, pp.125-134, ITALY, 2004.

[SOB 00] K. SOBOTTKA, H. KRONENBERG, T. PERROUD, H. BUNKE, « Text extraction from colored book and journal covers », International Journal on Document Analysis and Recognition, vol. 2, pp. 163-176, June 2000.

[SOU 02] S. SOUAFI, « Contribution à la reconnaissance des structures de documents écrits : Approche probabiliste », Thèse de doctorat, Institut National des Sciences Appliquées de Lyon, 21 septembre 2002.

[SPI 95] A. SPITZ, « Using character shape codes for word spotting in document images », Dori D. and Bruckstein A. (Eds.), Shape, Structure and Pattern Recognition, World Scientific, pp.382-389, Singapore, 1995.

[SPI 97] A.L. SPITZ, « Determination of the script anf language content of document images », IEEE Transaction On Pattern Analysis and Machine Intelligence, vol. 19, No. 3, pp. 235-245, March 1997.

[TAB 98] M. TABIZA, « Filtres Lp : Etude des propriétés et Application en Traitement d'images », Thèse de doctorat, université de Savoie, 16 Mars 1998.

[TAR 90] J. TARHIO, E.UKKONEN, « Boyer-Moore approach to approximate string matching », Proc. SWAT, LNCS 447, pp. 348-359, 1990.

[TOL 06] S. TOLLARI, « Indexation et recherche d'image par fusion d'informations textuelles et visuelles », Thèse de doctorat, Université du Sud Toulon-Var, 24 Octobre 2006.

[TRI 96] O.D. TRIER, A.K. JAIN, T. TAXT, « Feature extraction methods for character recognition: A Survey », International Conference on Pattern Recognition, vol. 29, No. 4, pp. 641-662, 1996.

[TRI 03] E. TRINH, « De la numérisation à la consultation des documents anciens », Thèse de doctorat dans le cadre d'un projet européen DEBORA et d'un projet industriel, INSA de Lyon, France, 2003.

[TSA 02] C.M. TSAI, H.J. LEE, « Binarization of color document images via luminance and saturation color features », IEEE Transaction on Image Processing, vol. 11, pp. 434 - 451, April 2002.

[TSU 90] S. TSUJIMOTO, H. ASADA, « Understanding multi-articled documents », 10th International Conference on Pattern Recognition, Atlantic City, pp. 551-556, USA, June 1990.

[TSU 92] S. TSUJIMOTO, H. ASADA, « Major components of a complete text reading system », Proc. IEEE, vol. 80, pp. 1133, July 1992.

[TUK 71] J.W. TUKEY, « Nonlinear (non-superposable) method for smoothing data », Conf. Rec. EASCON' 74, pp. 673.

[UKK 85] E. UKKONEN, « Finding approximate patterns in strings », Journal of Algorithms, vol. 6, pp. 132-137, 1985.

[UKK 95] E. UKKONEN, « On-Line Construction of Suffix Trees », Algorithmica, vol. 14, pp. 249-260, 1995.

[VEL 80] F.R.D. VELASCO, « Thresholding using the isodata clustering algorithm », IEEE Transaction on system, Man and Cybernitics (SMC), pp. 771-774, 1980.

[WAG 74] R.A. WAGNER, M.J. FISHER, « The string to string correction problem », Communication of the ACM, vol. 20, No. 10, May 1974.

[WAK 98] B. WAKED, S. BERGLER, C.Y. SUEN, S. KHOURY, « Skew Detection, Page Segmentation, and Script Classification of Printed Document Images », IEEE Systems, Man, and Cybernetics, 1998.

[WAN 82] K.Y. WANG, R.G. CASEY, F.M, WAHL, « Document analysis system », IBM Journal of Research and Development, vol. 26, pp. 647-656, 1982.

[WAN 94] J. WANG, J. JEAN, « Segmentation of merged characters by neural networks and shortest path », International Conference on Pattern Recognition, vol. 27, No. 5, pp. 649, 1994.

[WAN 03] Q. WANG, T. XIA, C.L. TAN, L. LI, « Directional Wavelet Approach to Remove Document Image Interference », International Conference on Document Analysis and Recognition, pp. 736-740, Edinburgh, 2003.

[WEI 73] P. WEINER, « Linear pattern matching algorithm », 14th IEEE Symposium on Switching and Automata Theory, pp. 1-11, 1973.

[WHI 96] A. WHICHELLO, H. YAN, « Linking Broken Character Borders with Variable Sized Masks to Improve Recognition », Pattern Recognition, vol. 29, No. 8, pp. 1429-1435, 1996.

[WIN 99] W.E. WINKLER, « The state of record linkage and current research problems », Technical report, Statistics of Income Division, Internal Revenue Service Publication R99/04, 1999.

[WOL 06] C. WOLF, « Document Ink bleed-through removal with two hidden Markov random fields and a single observation field », Technical Report RRLIRIS2006-019, Lyon: LIRIS, INSA Lyon, 2006.

[WOL 02] C. WOLF, J.M. JOLION, F. CHASSAING, « Extraction de texte dans des vidéos : le cas de la binarisation », 13ème congrès francophone de reconnaissance des formes et intelligence artificielle, pp. 145-152, 2002.

[WON 82] K. WONG, R. CASEY, F. WAHL, « Document analysis system », I.B.M. Journal of Research and Development, vol. 26, No 6, 1982.

[WU 92] S. WU, U. MANBER, « Fast text searching allowing errors », Communications of the ACM, vol. 35, No. 10, pp. 83-91, October 1992.

[WU 96] S. Wu, U. Manber, G. Myers, « A Subquadratic Algorithm for Approximate Limited Expression Matching », Algorithmica, vol. 15, pp. 50-67, 1996.

[YU 92] B. YU, X. LIN, Y. WU, B. YUAN, « Isothetic polygon representations for contours », Image Understanding, vol. 56, pp. 264-268, 1992.

[YU 96] B. YU, A.K. JAIN, « A robust and fast skew detection algorithm for generic documents », International Conference on Pattern Recognition, vol. 29, No. 10, 1599-1630, 1996.

[YU 01] D.YU, H.YAN, « Reconstruction of Broken Handwritten digits based on structural morphological features », International Conference on Pattern recognition, No. 34, pp. 235-254, 2001.

[ZAH 72] C. ZAHN, R. ROSKIES, « Fourier descriptors for plane closed curves », IEEE Transactions on Computers, vol. 21, No. 3, pp. 269-281, March 1972.

[ZAH 04] A. ZAHOUR, B. TACONET, S. RAMDANE, « Contribution à la segmentation de textes manuscrits anciens », Colloque International Francophone sur l'Ecrit et le Document, 2004.

[ZHA 01] Z. ZHANG, C.L. TAN, « Recovery of Distorted Document Images from Bound Volumes », 6th International Conference on Document Analysis and Recognition, pp. 429-433, 2001.

[ZHE 01] Q. ZHENG, T. KANUNGO, « Morphological Degradation Models and Their Use in Document Image Restoration », Rapport LAMP-TR-065/CS-TR- 4218/CAR-TR-962, University of Maryland, College Park, USA, 2001.

[ZOB 95] J. ZOBEL, « Finding Approximate Matches in Large Lexicons », Software-Practice and Experience, vol. 25, No. 3, pp. 331-345, March 1995.

[ZRA 98] A. ZRAMDINI, R. INGOLD, « Optical font identification using typographic features », International Conference on Pattern Analysis and Machine Intelligence, vol. 20, No. 8, pp. 877-882, August 1998.