REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider – BISKRA
Faculté des Sciences Exactes et des Sciences de la Nature et de la Vie
Département d'informatique

N° d'ordre :.....
Série :.....

Mémoire

Présenté en vue de l'obtention du diplôme de magister en Informatique

Option : **Intelligence Artificielle et Systèmes Distribués**

Les Supports Vecteurs Machines (SVM) pour la reconnaissance des caractères manuscrits arabes

Par :

M. ZAIZ Faouzi

Soutenu le : 15/07/2010

Devant le jury :

DJEDI Nour Eddine	Professeur	Université de Biskra	Président
CHERIF Foudil	Maître de conférences A	Université de Biskra	Examinateur
MOUSSAOUI Abdelouahab	Maître de conférences A	Université de Sétif	Examinateur
BABAHENINI Med Chaouki	Maître de conférences A	Université de Biskra	Rapporteur

Dédicace

Tout d'abord, je veux rendre grâce à Dieu, le Clément et le Très Miséricordieux pour son amour éternel. C'est ainsi que je dédie ce mémoire à :

ma mère pour sa tendresse et mon père pour sa patience et encouragement
mes très chers frères et ma chère sœur pour leurs conseils,
mes cousins et cousines,
tous ceux que j'aime,
tous mes amies.

Remerciements

A Dieu, le tout puissant, nous rendons grâce pour nous avoir donné santé, patience, volonté et surtout raison.

En premier lieu, je tiens à remercier mon encadreur Mr. Babahenini M.C qui m'a aidé et conseillé durant ce travail.

Je remercie également tous les enseignants du département de l'informatique de l'université de BISKRA pour leur aide et encouragement.

Enfin, je remercie tous ceux qui en soutenu, encouragé et donné l'envie de mener à terme ce travail.

ملخص

التعرف على الكتابـــة (OCR: Optic Character Recognition) يلعـــب دور هام جدا في وقتنا الحالي، فهو قادر على حل مشاكل معقدة كما يسهل العديد مـــن الأنشطة التي يقوم بها الإنسان. التعرف على الكتابة العربية يعود إلى السبعينات، أين العديد من الحلول اقترحت وهي متعددة كتعدد الحلول بالنسبة للغات اللاتينية.

يمثل هذا العمل دراسة حول مجال التعرف على خط اليد العربي حيث تم طـــرح دراسة عامة حول أنظمة التعرف على الكتابة، وبعد ذلك تم التطرق لمرحلة مهمة من مراحل التعرف على الكتابة وهي مرحلة التجزئة(حيث يتم تقسيم صورة الكلمة إلى أحرف).

قمنا بتقديم لمحة عن مختلف طرق التجزئة، يليها تقديم مجال التعرف على الكتابة، اللغة العربية، و بعض مشاكل الضبط في الكتابة العربية بإيجاز.

بعد عرض مقارنة لمختلف طرق التجزئة لخط اليد العربي، قدمنا مقترح مشاركة في هذا المجال من خلال خوارزمية تجزئة.

كلمات مفتاحية: التعرف على الكتابة (OCR)، التجزئة، الحروف العربية، جـــزء كلمة، المعالجة، المصنف SVM.

Abstract

Optic Character Recognition (OCR) has a main role in the present time. It's capable to resolve complex problems and simplify human activities. The OCR yields to 70's, since many solutions had proposed, and they are varied as Latin methods.

This work presents a survey concerning Arabic Optic Character Recognition (AOCR). A general study of handwritten character recognition systems is developed, then it takes apart the segmentation phase because it is a crucial phase of the recognition process.

We had presented a state of the art of the character segmentation methods, after that a view of the OCR area, Arabic language, and some of Arabic writing normalization problems are presented.

After a comparison between Arabic handwritten characters segmentation methods, we had proposed a contribution through a segmentation algorithm.

Key words: OCR, segmentation, Arabic characters, PAW, post-processing, SVM.

Résumé

La reconnaissance de caractère joue un rôle très important dans le monde actuel. Elle est capable de résoudre des problèmes complexes et rendre les activités de l'homme plus simple. La reconnaissance de l'écriture arabe remonte aux années 70, depuis plusieurs solutions ont été proposées. Elles sont aussi variées que celles utilisées pour le latin.

Le présent travail porte sur une étude concernant le domaine de reconnaissance optique de caractères arabes manuscrits. Une étude générale sur les systèmes de reconnaissance de l'écriture a été développée, puis elle a été affinée par un intérêt particulier à une phase considérée comme cruciale dans le procédé de reconnaissance: la phase de segmentation.

Nous avons présenté un état de l'art des méthodes de segmentation des caractères, ensuite nous avons présenté la langue arabe et le domaine de l'OCR, nous avons soulevé certains problèmes de normalisation dans l'écriture arabe.

Après que nous ayons une comparaison de méthodes de segmentation de caractères arabes manuscrits, nous avons proposé une contribution par un algorithme de segmentation.

Mots clés: OCR, segmentation, caractères arabes, pseudo mot, post-traitement, SVM.

Table de matières

Liste des figures

Liste des tableaux

Abréviations

AOCR : Arabic Optical Character Recognition.
ASCII : American Standard Code Information Interchange.
ASMO : Arabic Standard Metrology Organization.
CCW : Counter Clock Wise.
CXX : Composantes Connexes.
C-LAG : Compressed Line Adjacency Graph
DSP : Decisive Segmentation point.
HMM : Hidden Markovian Model.
KNN : K Nearest Neighbor.
LAG : Line Adjacency Graph.
OCR : Optical Character Recognition.
PAO : Publication Assistée par Ordinateur.
PAW : Piece of Arabic Word.
PS : Pen Size.
PSP : Primary Segmentation Points.

Reconnaissance des formes

Allographes : tracés résultat de la représentation d'un même symbole.

Ascendant : hampe dans la partie supérieure d'une lettre.

Descendant : hampe dans la partie inférieure d'une lettre.

Bigramme : deux lettres consécutives d'un mot.

Graphèmes : groupe ou ensemble de primitives résultant de la pré-segmentation et qui correspondent à une partie de lettre.

Ligature : trait qui relie deux lettres cursives.

Segmentation : opération par laquelle il est possible de restituer l'ordonnancement gauche droite d'un tracé.

Trigramme : trois lettres consécutives d'un mot.

Introduction

L'un des objectifs de la recherche en informatique est de repousser les limites de ce qui est automatisable. Les tâches répétitives, fastidieuses, portant sur de gros volumes de données, constituent de bons candidats. Citons par exemple le traitement des chèques bancaires, le tri du courrier postal, l'indexation d'archives nationales (archives militaires, formulaires de recensements, fonds bibliothécaires, ...), indexation d'archives privées, traitement du courrier entrant des entreprises, etc . . .

Bien qu'une machine soit capable de réaliser des calculs complexes et dépasse souvent les capacités humaines, elle n'en demeure pas moins limitée. Pour communiquer avec elle, nous devons nous imposer la discipline du clavier, tâche pénible pour certains ou à tout le moins peu naturelle.

Pour l'instant, et bien que la recherche dans ce domaine se poursuive depuis plus de trente ans, la solution générale au problème de la lecture automatique d'écriture cursive n'est toujours pas connue. Il semble cependant que la reconnaissance d'écriture cursive ait un rôle important à jouer dans les systèmes futurs de reconnaissance et donc, que ce domaine de recherche soit toujours d'actualité.

L'automatisation de cette tâche nécessite de transmettre à la machine la capacité de lire l'écriture manuscrite. Or les types d'écriture et les styles peuvent varier de façon considérable selon les scripteurs. Si le fait de lire est un acte relativement banal pour un humain, cette activité met néanmoins en oeuvre des processus complexes. Reconnaître correctement des symboles isolés ne suffit pas.

Etudier la manière dont nous réussissons à accomplir cette tâche complexe pourrait s'avérer utile pour apprendre aux machines la lecture de textes manuscrits. Quelles primitives sont détectées pendant la lecture? Comment accédons-nous à l'information nous permettant de comprendre la signification d'un mot? Est-ce que la perception d'un mot se construit à partir de la perception de ses lettres ou à partir de la perception de sa forme générale?

Depuis plusieurs années, des chercheurs dans les domaines de la biologie, de la neurophysiologie, de la psychologie cognitive et de la linguistique ont étudié ces questions, et des modèles de lecture ont résulté de ces investigations. Même si ces modèles sont susceptibles d'améliorations et bien que plusieurs de ces théories défendant différents point de vue ne fassent pas l'unanimité, nous croyons pouvoir profiter de leurs observations afin de développer un système robuste de reconnaissance d'image de mots cursifs isolés.

En fait, les primitives détectées au cours de la lecture jouent un rôle déterminant pour faire un choix éclairé de primitives à rechercher prioritairement lors du processus de reconnaissance. Le problème de l'accès lexical peut également influencer le choix de l'architecture retenue pour la méthode développée. Toutefois, la plupart de ces modèles de lecture ont été élaborés à partir de textes imprimés.

Peu d'études ont été faites sur les mécanismes impliqués dans la lecture de l'écriture cursive. Les auteurs arrivent à la conclusion que même si la lecture de mots cursifs diffère à première vue de la lecture de mots imprimés, une fois achevée ce qu'ils

appellent la normalisation du cursif, les mots imprimés et manuscrits semblent sujet à des processus de traitement similaires.

La reconnaissance est dite en-ligne lorsque les données dynamiques sont acquises pendant l'écriture. On pense ici à une tablette graphique ou un papier électronique où l'utilisateur écrit avec un stylo. Par contre, on dit qu'elle est hors-ligne lorsqu'il s'agit de reconnaître l'image d'un mot obtenue avec un scanner.

L’objectif de cette mémoire est de proposer un système de reconnaissance de l’écriture manuscrite arabe hors-ligne. Ce système s’appuie sur une méthode de segmentation structurelle et utilise les Machines à Vecteurs de Support (SVM) dans la phase de classification.

Le premier chapitre est un rappel de certaines notions générales d'OCR. ainsi que les étapes nécessaires pour la réalisation d'un système de reconnaissance de l'écrit, ainsi une étude de l'OCR et la langue arabe, où nous trouvons un rappel sur certaines données de la calligraphie arabe, suivie de notions d'OCR sur l'écriture arabe.

Le deuxième chapitre est spécifique à l'état de l'art de la segmentation des textes dans le cas général. Pour celui nous décrivons le processus de l'étape de la détection des objets dans une page, à la segmentation des blocs de texte en lignes puis en mot puis en caractères. Nous mettons l'accent sur les méthodes utilisées dans ce type de segmentation.

Le troisième chapitre va mettre l'accent sur la méthode de classification choisie, par l'étude de la méthode de Machines à Vecteurs de Support (SVM). Le quatrième chapitre constitue notre contribution, il s'agit d'un algorithme permettant segmentation des textes arabes manuscrits, suivie des les tests et résultats obtenus.

Nous terminons le travail par une conclusion sur les résultats obtenus par la méthode utilisée, et des perspectives de ce travail.

Chapitre 1

Reconnaissance de l'écriture

- **Introduction**
- **Situation du problème**
- **Différents aspects de l'OCR**
- **Problèmes liées à l'OCR**
- **Organisation générale d'un système de reconnaissance**
- **Calligraphie et typographie arabe**
- **Avancées en OCR arabe**
- **Conclusion**

Introduction

Pour accéder au sens d'un document écrit, l'humain passe par trois niveaux de perception: la vue, la reconnaissance et la compréhension. De manière immédiate le document est perçu grâce aux organes de la vue, toute information écrite peut être reprise dans une chaîne de traitement informatisée à différentes fins : la rédaction et l'édition de rapports, la diffusion de documents dans un système de messagerie … conduisent à exploiter des informations disponibles seulement sur papier.

La reconnaissance optique de caractères (OCR) est une opération informatique rapide permettant de réaliser la transformation d'un texte écrit sur papier en un texte sous forme d'un fichier informatique en représentation symbolique (par exemple pour les écritures latines, le codage opéré est le code ASCII*(American Standard code for information interchange)*, tandis que pour l'arabe on utilise généralement le code ASMO (Arabic Standard Metrology Organization).

L'objectif de ce chapitre consiste d'une part à introduire et de présenter un état de l'art du domaine de la reconnaissance de document, ce qui nous permettra de situer le problème d'OCR, et d'autre part à exposer les différentes approches, méthodes et techniques réalisées depuis plusieurs années.

1. Situation du problème

Dans cette section nous allons situer la reconnaissance par rapport à la production de documents puis nous parlerons des étapes de la reconnaissance et des structures de documents. Ainsi de situer la reconnaissance de caractères par rapport à la reconnaissance de documents.

1.1 Production et reconnaissance

La reconnaissance de documents est le processus inverse de la production. De la forme papier, elle essaie de remonter à la forme logique. La figure 1.1 illustre les deux processus.

La figure 1.2 illustre de manière plus détaillée les étapes de la reconnaissance de documents [59].

Le document papier est saisi à l'aide d'un scanner de manière à obtenir une image sous la forme électronique (couleur, résolution). Une image bruitée et biaisée appelée image brute. Cette image est ensuite prétraitée pour fournir une image épurée avec une représentation plus claire [59].

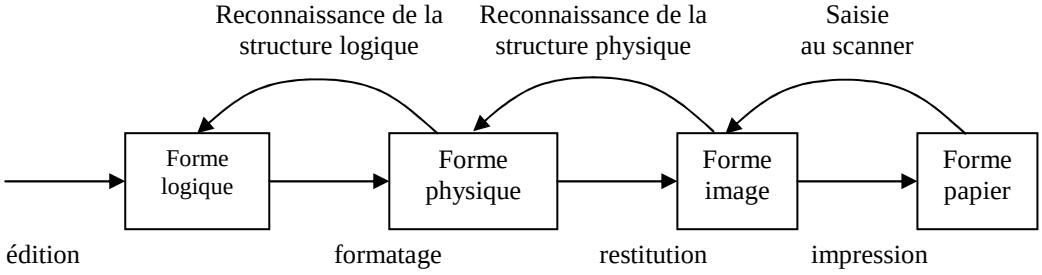


Figure 1.1 : Processus de production et de reconnaissance de documents [59].

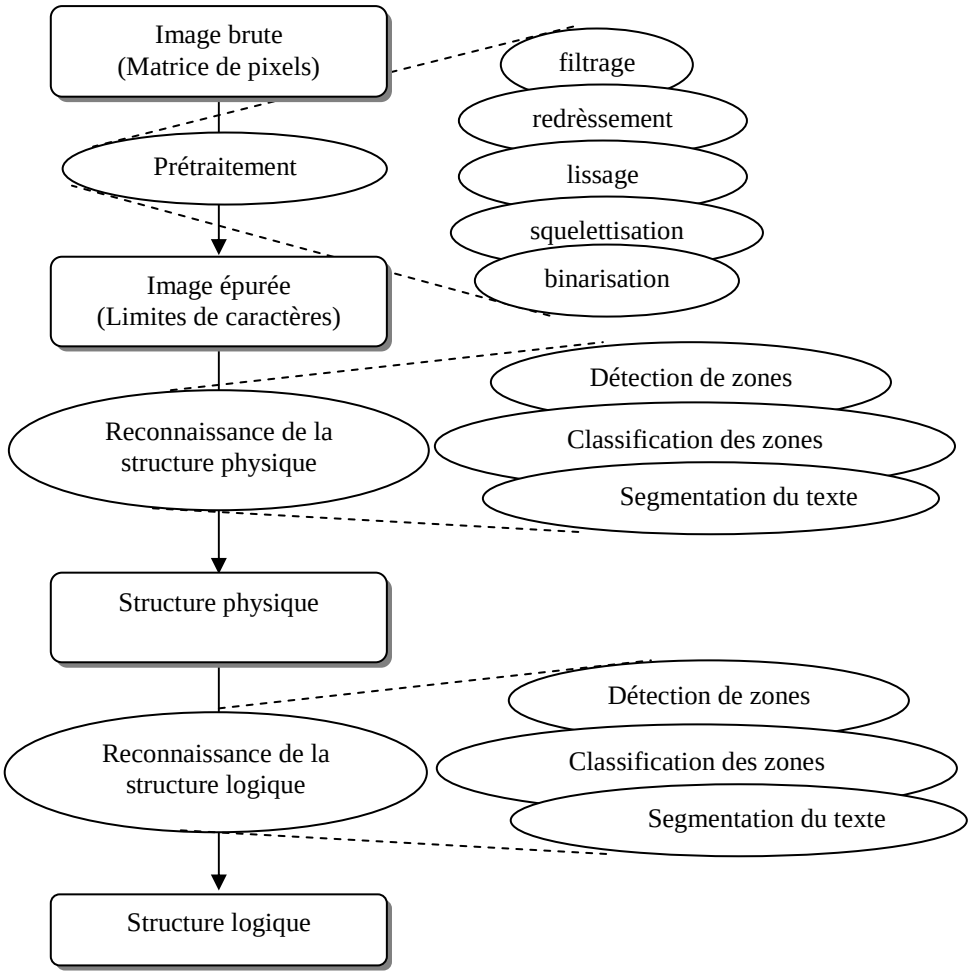


Figure 1.2 : Etapes de la reconnaissance de documents [59].

La reconnaissance de la structure physique (ou forme physique) consiste d'une part à la détection et la classification des différentes zones de l'image en texte, graphique, table, formule, dessin ou photo et d'autre part à la découpe du texte en colonnes, paragraphes, lignes, mots et signes [59].

Finalement, la reconnaissance de la structure logique (ou forme logique) consiste à faire un étiquetage logique aux différents objets de la structure physique et à réorganiser ces objets conformément au flux de lecture [59].

1.2 Structures de documents

La connaissance de la structure du document à traiter est une nécessité puisque nous allons l'exploiter durant la phase de reconnaissance. Elle permet de définir une stratégie de lecture, de segmentation et d'identification des entités de base [67].

Dans un document écrit nous pouvons distinguer deux niveaux de structuration [67]:

- La structure physique, qui résulte de la mise en page.
- La structure logique, qui précède la mise en page et se rapporte plutôt au continu.

Une autre variante de classification de structure de documents, en les classant par la notion niveau de complexité [59]:

- Structure simple, document contenant des objets simple (par exemple: texte, …etc.).
- Structure complexe, document contenant des objets complexe (par exemple: texte, images, schémas…etc.).

2. Différents aspects de l'OCR

Il est très difficile de créer un système OCR capable de reconnaître n'importe quel écriture ou fonte. Tout dépend de l'application vissée ou voulu [90] et des données à traiter [1, 76]. Généralement, les systèmes OCR sont classés en se basant sur trois critères:

- Outil d'acquisition: Les systèmes qualifiés de « en-ligne » ou « hors-ligne »,
- Approches de reconnaissance: approches globales ou analytiques selon que l'analyse s'opère sur la totalité du mot, ou par segmentation en caractères.
- Nature des traits caractéristiques: approches statistiques, structurelles ou stochastiques.

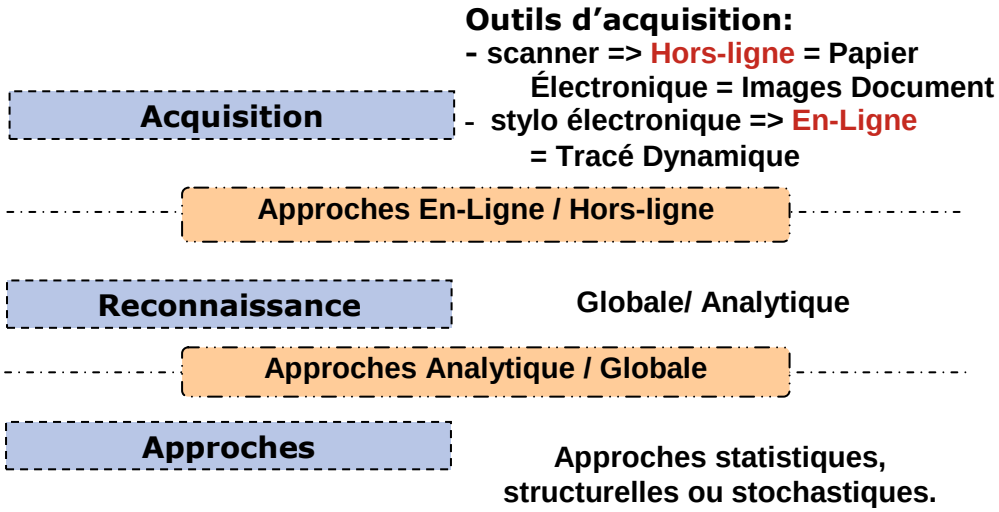


Figure 1.3 : Différents aspects de l'OCR.

2.1 Type d'acquisition

Les méthodes de reconnaissance se distinguent tout d'abord par le type d'acquisition des données. La reconnaissance est dite en-ligne lorsque les données dynamiques sont acquises pendant l'écriture. On pense ici à une tablette graphique ou un papier électronique où l'utilisateur écrit avec un stylo. La reconnaissance est dite hors-ligne lorsqu'il s'agit de reconnaître l'image d'un mot obtenue avec un scanner.

On s'accorde généralement à dire que la reconnaissance hors-ligne est plus complexe que la reconnaissance en-ligne pour plusieurs raisons.

2.1.1 La reconnaissance en-ligne (on-line) :

Dans le cas de la reconnaissance *en-ligne* (dynamique [74]) le système reçois les images d'entée de données en temps réel, ce qui permet d'intégrer les mouvements du stylo électronique (Tracé Dynamique [14]) et pressure information [2], et calcule la

relation entre les points pour extraire les caractéristiques afin de reconnaître les symboles au fur et à mesure qu'ils sont écrits à la main [90]. Ce mode est réservé généralement à l'écriture manuscrite. La reconnaissance en-ligne présente un avantage majeur c'est la possibilité de correction et de modification de l'écriture de manière interactive vu la réponse en continu du système [76].

2.1.2 La reconnaissance hors-ligne (off-line) :

La reconnaissance *hors-ligne* (ou en différé, ou encore statique) est obtenue par la saisie d'un texte déjà existant, obtenue par un scanner ou une caméra. Dans ce cas, on dispose d'une image binaire ou en niveaux de gris, ayant perdu toute information temporelle sur l'ordre des points. De plus, ce mode introduit une difficulté supplémentaire relative à la variabilité du tracé en épaisseur et en connectivité, nécessitant l'application de techniques de prétraitement [74].

La difficulté d'un système de reconnaissance de caractères hors-ligne s'accroît quand le système traite des scripts d'écriture manuscrite cursive tel qu'un script d'écriture manuscrite arabe [91].

Le tableau suivant présente une brève comparaison des deux approches.

Critère de comparaison	En-ligne	Hors-ligne
Outils d'acquisition	- Stylo électronique plus tablette graphique.	- Scanner ou caméra.
Bruit d'image	- faible	- existence d'un bruit important.
Informations disponibles	- la position, - la direction du mouvement, - les points de fin, - les points de début, - ordre des traits.	- absence d'informations contextuelles.

Tableau 1.1: Comparaison brève entre les approches en-ligne et hors-ligne [90].

2.2 Approches de reconnaissance

Durant les dernières décennies, beaucoup de méthodes de segmentation ont été développées dans le but d'avoir un système de reconnaissance de caractères plus robuste. Malgré tous les efforts, la situation reste loin d'atteindre les ambitions. En se basant sur le processus de segmentation, deux approches ont été appliquées [91]:

- Approche globale.
- Approche analytique.

2.2.1 Approche globale

L'approche globale essaye de reconnaître la représentation intégrale des mots de l'image d'entée [91] et de le décrire indépendamment des caractères qui le constituent. Cette approche présente l'avantage de garder le caractère dans son contexte avoisinant, ce qui permet une modélisation plus efficace des variations de l'écriture et des dégradations qu'elle peut subir [1].

Cependant cette méthode est pénalisante par la taille mémoire, le temps de calcul et la complexité du traitement qui croit linéairement avec la taille du lexique considéré, d'où une limitation du vocabulaire [2].

2.2.2 Approche analytique

L'approche analytique au contraire de celle présentée précédemment, isole les différents caractères des mots [91]. L'idée de base de l'approche analytique est de segmenter l'image du mot en entrée en caractères ou en fragments morphologiques significatifs inférieurs au caractère appelés graphèmes. La reconnaissance du mot consiste à reconnaître les entités segmentées puis tendres vers une reconnaissance du mot, ce qui constitue une tâche délicate pouvant générer différents types d'erreurs. Un processus de reconnaissance selon cette approche est basé sur une alternance entre deux phases : la phase de segmentation et la phase d'identification des segments [1,2].

Deux solutions sont alors possibles : la segmentation explicite (externe) ou la segmentation implicite (interne). Par ailleurs, les méthodes analytiques par opposition aux méthodes globales, présentent l'avantage de pouvoir se généraliser à la reconnaissance d'un vocabulaire sans limite à priori, car le nombre de caractères est naturellement fini. De plus l'extraction des primitives est plus aisée sur un caractère que sur une chaîne de caractères [90, 91].

2.3 Nature des traits caractéristiques

La nature des caractéristiques varie d'une approche à une autre. Généralement, les caractéristiques peuvent être classés en cinq groupes principaux [1, 6, 70] :

- caractéristiques topologiques,
- caractéristiques structurelles,
- caractéristiques statistiques,
- globales ou locales, et
- superposition des modèles et corrélation.

2.3.1 Caractéristiques topologiques ou métrique

Ce type de primitives est basé sur des densités de pixels. On peut par exemple projeter des images de tailles différentes (les graphèmes) dans une matrice de taille fixe. Les caractéristiques extraites sont les valeurs des cellules de cette matrice. Dans ce type de primitives, on compte également les profils et histogrammes. Pour maintenir un vecteur de taille fixe, on divise l'image en un nombre fixe de bandes horizontales et verticales. Les caractéristiques sont les moyennes des valeurs sur ces bandes [70].

Elles consistent à compter dans une forme le nombre de trous, évaluer les concavités, mesures des pentes et autres paramètres de courbures et évaluer des orientations,

mesurer la longueur et l'épaisseur des traits, détecter les croisements et les jonctions des traits, mesures les surfaces et les périmètres,…[6]

2.3.2 Caractéristiques structurelles

Elles ressemblent beaucoup aux primitives topologiques. La différence est qu'elles sont généralement extraites non pas de l'image brute mais à partir du squelette ou du contour de la forme en donnant ses propriétés globales et locales. Ainsi, on ne parle plus de trous, mais de boucles ou de cycles dans une représentation filiforme du caractère. Parmi ces caractéristiques on peut citer [6]:

- Les traits et les anses *dans* les différentes directions ainsi que leurs tailles.
- Les points terminaux.
- Les points d'intersections.
- Les boucles.
- Le nombre de points diacritiques et leur position par rapport à la ligne de base.
- Les symboles diacritiques et les zigzags (hamza).
- La hauteur et la largeur du caractère.
- La catégorie de la forme (partie primaire ou point diacritique, etc.).

Plusieurs autres caractéristiques peuvent être tirés, suivant qu'ils soient extraits d'une courbe, un trait ou un segment de contour.

2.3.3 Caractéristiques statistiques

Les caractéristiques statistiques décrivent une forme en terme d'un ensemble de mesures extraites à partir de cette forme. Les caractéristiques utilisées pour la reconnaissance de textes arabes sont [1]:

- Le zonage (zoning), consiste à superposer une grille n×m sur l'image du caractère et pour chacune des régions résultantes, calculer la moyenne ou le pourcentage de points en niveaux de gris, donnant ainsi un vecteur de taille n×m de caractéristiques.
- caractéristiques de lieu géométrique: en utilisant la méthode Loci qui est basée sur le calcul du nombre de segments blancs et de segments noirs le long d'une ligne verticale traversant la forme, ainsi que leurs longueurs.
- La méthode des moments : les moments d'une forme par rapport à son centre de gravité sont invariants par rapport à la translation et peuvent être invariants par rapport à la rotation. Ils sont aussi indépendants de l'échelle. Ces caractéristiques peuvent être facilement et rapidement extraites d'une image de texte, ils peuvent tolérer modérément les bruits et les variations.

2.3.4 Caractéristiques globales ou locales

Les primitives globales cherchent à représenter au mieux la forme générale d'un caractère et sont donc calculées sur des images relativement grandes (ex : transformée de Fourier et transformée de Hough). Les primitives locales sont calculées lors d'un parcours des pixels de l'image avec un pas d'analyse qui dépend de la modélisation, du type de primitive et de la taille de l'image [1,6].

2.3.5 Superposition des modèles et corrélation

La méthode de 'template matching' appliquée à une image binaire (en niveaux de gris ou squelettes), consiste à utiliser l'image de la forme comme vecteur de caractéristiques pour être comparé à un modèle (template) pixel par pixel dans la phase de reconnaissance, et une mesure de similarité est calculée [1].

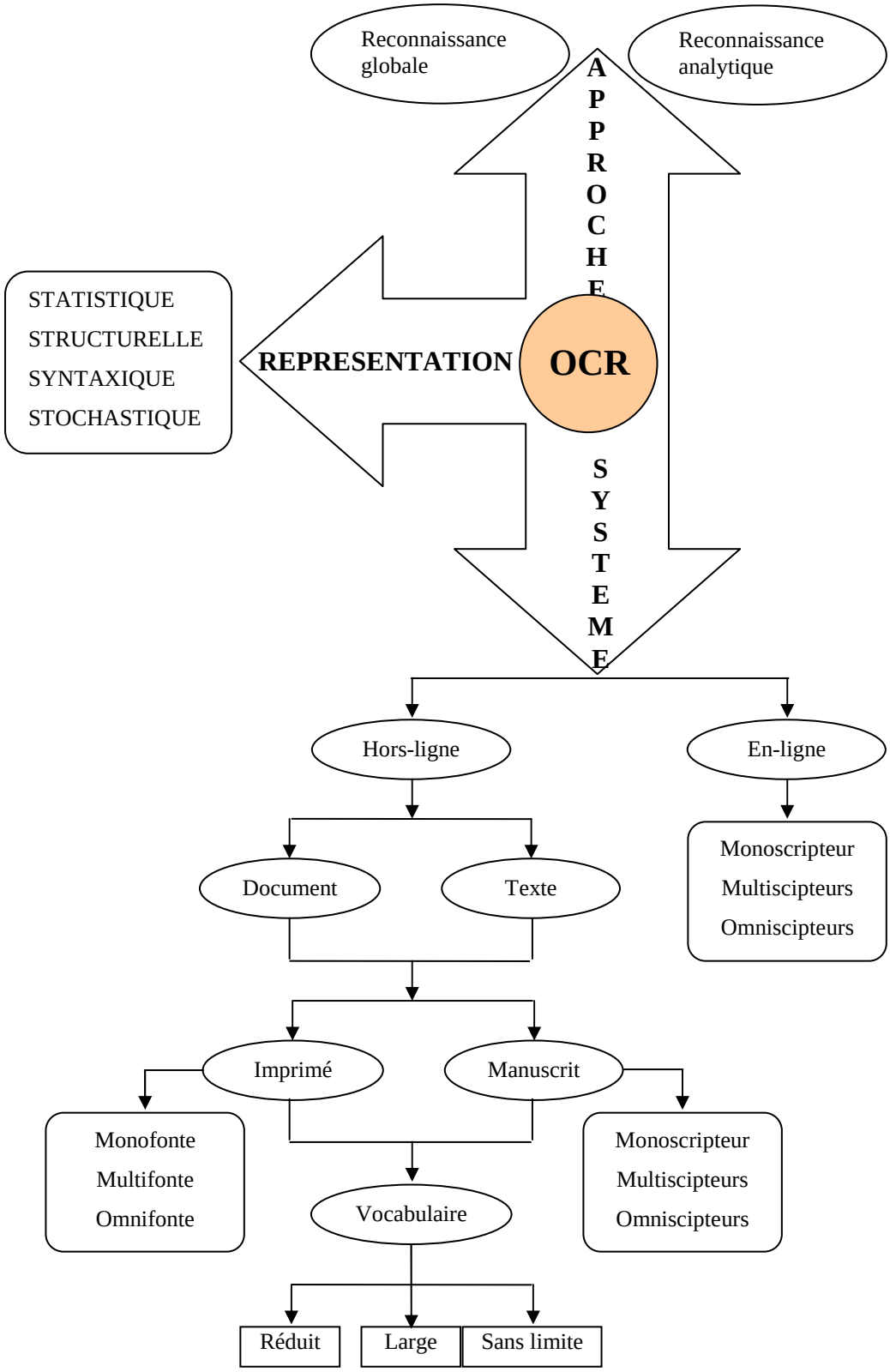


Figure 1.4 : Différents systèmes, représentations et approches de reconnaissance [1].

3. Problèmes liés à l'OCR

Divers problèmes compliquent le processus de reconnaissance, ce qui rend la tâche de l'OCR très délicates, parmi ces problèmes nous pouvons citer [74]:

- Disposition spatiale du texte,
- Nombre de scripteurs,
- Taille du vocabulaire.

3.1 Disposition spatiale du texte

La classification de Tappert [74] indique que la présentation du texte peut subir deux types de contraintes : *externes* conduisant à une écriture *pré casée*, *zonée*, *guidée* ou *générale*; et *internes* provenant des habitudes propres à chaque scripteur et conduisant à une écriture *détachée*, *groupée*, *script* (bâton), purement *cursive* ou *mixte*. Il est évident que l'écriture détachée reste la plus facile à réaliser du fait de la séparation quasi immédiate des lettres ; Au contraire, l'écriture cursive nécessite plus d'efforts du fait de l'ambiguïté des limites entre les lettres [1, 74].

3.2 Nombre de scripteurs

La difficulté de reconnaissance croît avec ce nombre, divisant l'échelle en trois : mono, multi et omni scripteurs. En multi scripteur, le système doit s'adapter à l'écriture de plusieurs scripteurs, tandis qu'en omni scripteur, le système doit être capable de généraliser son apprentissage à n'importe quel type d'écriture [74].

3.3 Taille du vocabulaire

On fait la différence entre les applications à vocabulaire limité (< 100 mots) et celles à vocabulaire très étendu (> 10 000 mots). Il est évident que dans le premier cas, la complexité est moindre, car la réduction du nombre limite l'encombrement mémoire et favorise l'utilisation de méthodes de reconnaissance directes et donc rapides, par balayage systématique de l'ensemble des mots du lexique [74]. La figure 1.5 montre la complexité des systèmes de reconnaissance de l'écriture.

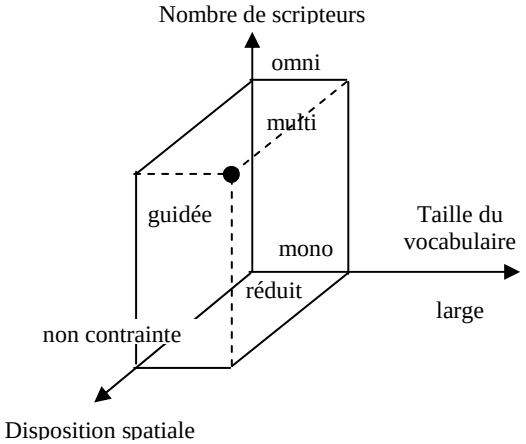


Figure 1.5: Graphe de complexité des systèmes de reconnaissance [74].

D'autres types de critères peuvent influencer la complexité des systèmes de reconnaissance. Ils sont relatifs aux variations intrinsèques de l'écriture dans un contexte d'écriture cursive.

Parmi ces variations nous pouvons noter celles [74]:

- propres au scripteur,
- propres à l'écriture manuscrite,

3.4 Variations propres au scripteur

Les variations propres au scripteur traduisent le style personnel en termes de rapidité, de continuité et de régularité. Tous ces éléments influent sur la forme des lettres (écriture penchée, bouclée, arrondie, linéaire, etc.) et bien sûr sur la forme des ligatures, compromettant parfois le repérage des limites entre lettres [74].

3.5 Variations propres à l'écriture manuscrite

La forme d'une lettre dépend de sa position dans le mot (début, milieu, fin) ainsi que des lettres voisines.

Toutes ces variations vont conduire à des formes morphologiques (dessins) différentes d'une même lettre, appelées *allographes* [74]. La figure 1.6 donne des exemples d'allographes.

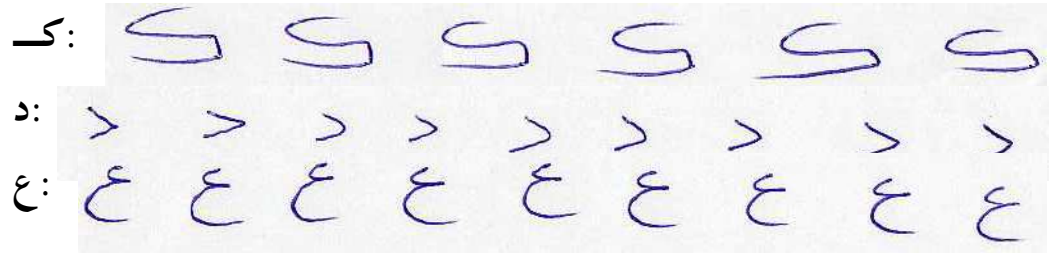


Figure 1.6: Exemples d'allographes des caractères arabes.

4. Organisation générale d'un système de reconnaissance

La reconnaissance de l'écrite manuscrite s'intéresse à identifier correctement l'entrée d'une image du texte écrit sur papier scannée ou photographié [90], en la convertissant en un texte sous forme d'un fichier informatique en format d'édition telle HTML ou Latex [59]. Typiquement, quelque soit le système de reconnaissance du manuscrit, il fait appel des phases suivantes [1, 90] :

- Acquisition (scanning, Numérisation),
- Prétraitement,
- Segmentation à des caractères séparés ou segments reliés à un caractère,
- Extraction des caractéristiques,
- Classification, suivis éventuellement d'une phase de post-traitement.

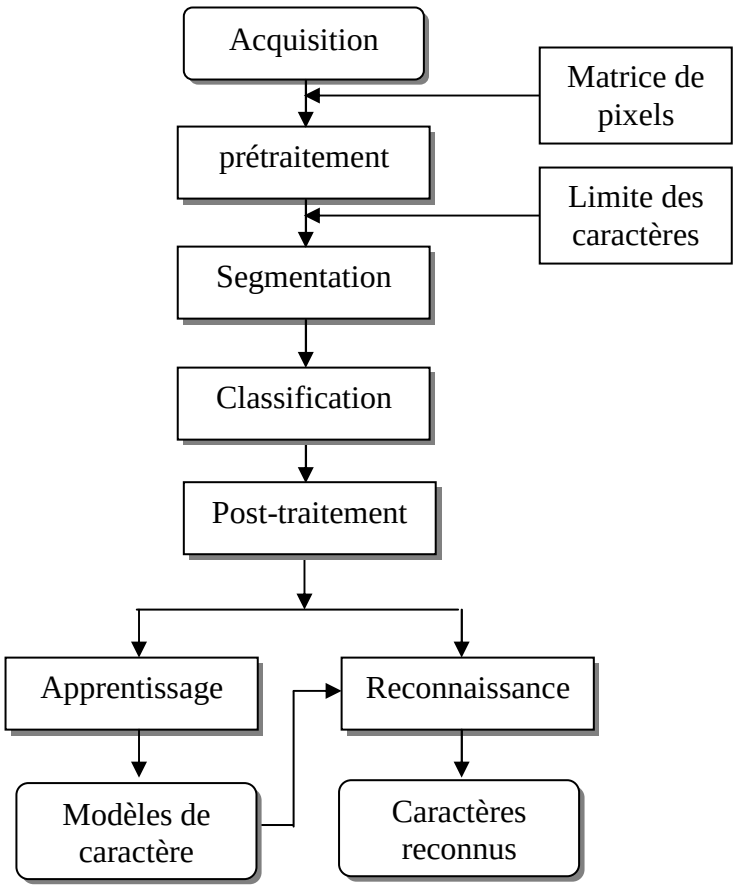


Figure 1.7 : Schéma général d'un système de reconnaissance de caractères.

4.1 Phase d'acquisition

La phase d'acquisition consiste à capter l'image d'un texte au moyen des capteurs physiques (scanner, caméra,…) et de la convertir en grandeurs numériques adaptés au système de traitement, avec un minimum de dégradation possible [1,12]. La numérisation consiste à [90]:

- Eliminer les bruits.
- Bouchage des trous (gap filling).
- Translation d'échelle.
- Normalisation.
- Binairisation.

4.2 Phase de prétraitements

Le prétraitement est le processus permettant qui couvre de la forme originale de l'image d'entrée à une forme squelettique ce qui permet de simplifier la phase d'extraction de caractéristiques [90]. Il s'agit essentiellement de réduire le bruit superposé aux données et essayer de ne garder que l'information significative de la forme représentée. Le bruit peut être dû aux conditions d'acquisition (éclairage, mise incorrecte du document, …) ou encore à la qualité du document d'origine [7, 12].

Parmi les opérations de prétraitement généralement utilisées nous pouvons citer :

- l'extraction des composantes connexes,
- le redressement de l'écriture,
- le lissage,
- la normalisation, et
- la squelettisation.

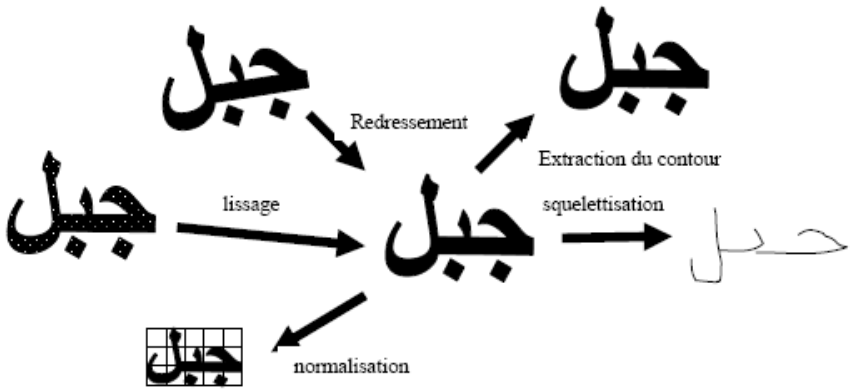


Figure 1.8 : Effet de certaines opérations de prétraitement [1].

4.2.1 Extraction de composantes connexes

L'écriture arabe est cursive et les mots sont séparés par des espaces. Toutefois, un mot peut contenir plusieurs pseudo mots (PAW) qui sont une portion du mot incluant une ou plusieurs lettres liées. Une segmentation typique est basée sur une analyse d'histogramme de projection, et le regroupement des composantes connexes [74].

Une composante connexe (CXX) est un ensemble de points dans le plan. Elle peut correspondre à un point diacritique, un accent, au corps d'un caractère ou d'une chaîne de caractères… Une fois localisés les CXX sont regroupées pour former les mots. Cette technique est utilisée pour le repérage des points diacritiques dans les images de textes arabes [1].

4.2.2 Redressement de l'écriture

L'idée est de rendre horizontaux les mots à l'aide d'une transformation géométrique de type rotation isométrique (angle – α si α est l'angle d'inclinaison) des points de l'image du mot [74]. L'inclinaison peut provenir de la saisie, si le document a été placé en biais, ou être intrinsèque au texte. [1] :

$$\begin{cases} x' = x\cos\alpha + y\sin\alpha \\ y' = y\cos\alpha + x\sin\alpha \end{cases}$$

4.2.3 Lissage de l'écriture

L'image des caractères peut être entachées de bruits dû aux artefacts de l'acquisition et à la qualité du document, conduisant soit à une absence de points ou à une surcharge de points [1, 7]. L'opération de lissage est destinée à rendre plus homogène les différentes parties de l'image, et préparer ainsi la détection des contours par

l'élimination des fortes variations d'intensité lumineuse ponctuelles non significatives. Généralement, un filtre passe bas est utilisé, mais il a comme effet secondaire la diminution du contraste (rendre l'image plus floue) [63].

4.2.4 Normalisation

La normalisation peut être indispensable pour certains types de systèmes (comme les réseaux de neurones), Elle permet de ramener les images de mots à des tailles standard. La différentielle pousse le principe de normalisation à un degré plus fin en essayant de normaliser localement différentes parties du mot, de manière à augmenter la similarité d'une image à une autre [7, 74].

Les parasites, les hampes et les jambages provoquent des décalages verticaux des mots qui désynchronisent la présence des informations, par exemple: la hampe des caractères [ل، ظ، ا] peuvent être éliminées [1].

4.2.5 Squelettisation

Le processus de squelettisation permet de réduire et de compacter la taille de l'image, et trouver un axe médian, définit comme l'ensemble de pixels S qui possèdent une égalité de distance par rapport aux pixels de frontière qui les entourent. La sortie de ce processus est un squelette d'un mot manuscrit [91]. Le processus s'effectue par passes successives pour déterminer si un tel ou tel pixel est essentiel pour le garder ou non dans le tracé. La squelettisation des caractères arabes peut induire en erreur : deux points diacritiques sont souvent confondus avec un seul [1].

La représentation squelettique possède les avantages suivants [91]:

- une bonne manière pour représenter les relations structurelles entre les composants de modèle (échantillon).
- Très utilisée dans les systèmes de reconnaissance de caractères, mot, signature et empreinte.

Remarque :

Selon la qualité du document à traiter, le type de l'écriture et la méthode d'analyse adoptée, une ou plusieurs techniques de prétraitement sont utilisées. Mais pas forcément toutes.

4.3 Phase de segmentation

Après que l'image soit acquise et prétraitée, nous allons maintenant extraire les différentes parties logiques de l'image en procédant d'abord par la séparation des blocs de texte et des blocs graphiques [8], ensuite on extrait les lignes de chaque bloc de texte (par une projection partielle), nous procédons à un suivi de contour partiel de chaque ligne [9]), finalement de chaque ligne nous pouvons extraire les mots depuis lesquels on peut avoir les caractères (ou parties du caractère) [92].

4.4 Phase d'analyse ou d'extraction des caractéristiques

L'extraction des caractéristiques consiste à extraire les mesures des entrées pour distinguer entre les classes, Devijver et Kittler dans [90], suggèrent que le problème d'extraction des caractéristiques des données d'entée soit fait par la sélection des

informations les plus significatives et appropriées pour les buts de classification et capable de discriminer entre classes, dans le sens de [90, 94]:

1. *minimiser* la variabilité entre les exemples de la même classe, et
2. *maximiser* la variabilité entre les exemples des classes.

Les types de caractéristiques peuvent être classés en quatre groupes principaux :

- caractéristiques structurelles,
- caractéristiques statistiques,
- globales ou locales, et
- superposition des modèles et corrélation

4.5 Phase de classification

Le processus de classification consiste à déterminer à quelle classe appartient une entrée de données (voir figure 1.7) [90]. La classification dans un système OCR regroupe deux tâches : l'apprentissage, la reconnaissance et décision. A cette étape les caractéristiques de l'étape précédente sont utilisées pour identifier un segment de texte et l'attribuer à un modèle de référence [67].

4.5.1 Etape d'apprentissage

Avant de pouvoir prendre une décision, il faut acquérir des connaissances t les organiser en modèles de référence ou *classes*. C'est durant la phase d'apprentissage qu'est réalisé ce travail. Lorsque le concepteur (ou professeur) indique le nom de la forme d'entrée, l'apprentissage est dit supervisé. Si la construction des classes est automatique, on parle d'apprentissage non supervisé [67].

4.5.2 Etape de reconnaissance et décision

Le module de décision cherche parmi les modèles de référence une description des paramètres plus proche de celle du caractère. La reconnaissance peut conduire à un *succès* si la réponse est unique (un seul modèle répond à la description de la forme du caractère). Elle peut conduire à une *confusion* si la réponse est multiple. Enfin elle peut conduire à un *rejet* de la forme si aucun modèle ne correspond à sa description. Dans les deux premiers cas, la décision peut être accompagnée d'une *mesure de vraisemblance*, appelée aussi *score* ou *taux de reconnaissance.* Les approches de reconnaissance peuvent être regroupées en quatre groupes principaux [1, 67]:

- approche statistique,
- approche structurelle,
- approche stochastique,
- et l'approche hybride.

A. Approche statistique

Dans l'approche statistique, la forme est représentée par un vecteur de n composant correspondant à la mesure de n caractéristiques observé sur elles. La reconnaissance consiste à trouver la classe à laquelle le symbole possède la plus grande probabilité d'appartenir et à évaluer le risque lié à la prise de la décision [67].

A.1 Méthode bayésienne

L'approche bayésienne consiste à choisir parmi un ensemble de caractères, celui pour lequel la suite de primitives extraites a la plus forte probabilité à posteriori par rapport aux caractères préalablement appris [38, 44].

A.2 Méthode du plus proche voisin (KNN)

L'algorithme KNN (K Nearest Neighbors) affecte une forme inconnue à la classe de son plus proche voisin en la comparant aux formes stockées dans une classe de références nommée prototypes [35].

Il renvoie les K formes les plus proches de la forme à reconnaître suivant un critère de similarité. Une stratégie de décision permet d'affecter des valeurs de confiance à chacune des classes en compétition et d'attribuer la classe la plus vraisemblable (au sens de la métrique choisie) à la forme inconnue [38].

Cette méthode présente l'avantage d'être facile à mettre en oeuvre et fournit de bons résultats. Son principal inconvénient est lié à la faible vitesse de classification due au nombre important de distances à calculer [1, 38].

A.3 Réseaux de neurones

Un réseau de neurones est un graphe orienté pondéré. Les noeuds de ce graphe sont des automates simples appelés neurones formels. Les neurones sont dotés d'un état interne, l'activation, par lequel ils influencent les autres neurones du réseau. Cette activité se propage dans le graphe le long d'arcs pondérés appelés liens synaptiques [11, 35].

En OCR, les primitives extraites sur une image d'un caractère (ou de l'entité choisie) constituent les entrées du réseau. La sortie activée du réseau correspond au caractère reconnu. Le choix de l'architecture du réseau est un compromis entre la complexité des calculs et le taux de reconnaissance [44, 86].

Par ailleurs, le point fort des réseaux de neurones réside dans leur capacité de générer une région de décision de forme quelconque, requise par un algorithme de classification, au prix de l'intégration de couches de cellules supplémentaires dans le réseau [88].

A.4 Machines à Vecteurs de Support (SVM)

L'algorithme des machines à vecteurs de support a été développé dans les années 90 par Vapnik. Il a initialement été développé comme un algorithme de classification binaire supervisée. Il s'avère particulièrement efficace de par le fait qu'il peut traiter des problèmes mettant en jeu de grands nombres de descripteurs, qu'il assure une solution unique (pas de problèmes de minimum local comme pour les réseaux de neurones) et il a fourni de bons résultats sur des problèmes réels [13, 16].

L'algorithme sous sa forme initiale revient à chercher une frontière de décision linéaire entre deux classes, mais ce modèle peut considérablement être enrichi en se projetant dans un autre espace permettant d'augmenter la séparabilité des données. On

peut alors appliquer le même algorithme dans ce nouvel espace, ce qui se traduit par une frontière de décision non linéaire dans l'espace initial.

Nous reviendrons aux détails de cette méthode dans le chapitre 3 puisque cette approche fait précisément l'objet du sujet [34, 89].

B. Approche structurelle

En représentation structurelle, chaque caractère est représenté par une phrase dans un langue où le vocabulaire est constitué de primitives. Les caractères d'une même famille sont représentés par une grammaire. La reconnaissance consiste à déterminer si la phrase de description du caractère peut être générée par la grammaire. L'inconvénient de cette méthode est l'absence d'algorithme efficace pour l'inférence grammaticale directe [67].

B.1 Méthodes de tests

Elles consistent à appliquer sur chaque caractère traité des tests de plus en plus fins sur la présence ou l'absence de primitives, de manière à répartir les échantillons en classes. Le processus le plus habituel consiste à diviser à chaque test l'ensemble des choix en deux jusqu'à n'obtenir qu'une seule forme correspondant au caractère entré. Ce choix dichotomique est très rapide et très simple à mettre en oeuvre, mais il est très sensible aux variations du tracé [1, 67].

B.2 Comparaison de chaînes

Les caractères sont représentés par des chaînes de primitives. La comparaison du caractère traité avec le modèle de référence, consiste à mesurer la ressemblance entre les deux chaînes et à se prononcer sur celui-ci. La mesure de ressemblance peut se faire par calcul de distance ou par examen de l'inclusion de toute ou une partie d'une chaîne dans l'autre [1, 67].

B.3 Approche syntaxique :

En représentation syntaxique, chaque caractère est représenté par une phrase dans un langue où le vocabulaire est constitué de primitives. Les caractères d'une même famille sont représentés par une grammaire. La reconnaissance consiste à déterminer si la phrase de description du caractère peut être générée par la grammaire. L'inconvénient de cette méthode est l'absence d'algorithme efficace pour l'inférence grammaticale directe [1].

C. Approche stochastique

L'approche stochastique considère la forme comme un signal continu dans le temps, observable à différents endroits constituant les "états d'observation" (elle n'est donc utilisable que sur des objets comportant une information temporelle). Le modèle décrit ces états à l'aide de probabilités, de transition d'état à état, et d'observation d'états [67, 93].

D. Approche hybride

Pour améliorer les performances de reconnaissance, la tendance aujourd'hui est de construire des systèmes hybrides qui utilisent différents types de caractéristiques, et qui combinent plusieurs classifieurs en couches [67, 70].

4.6 Phase de post-traitement

Le post-traitement est effectué quand le processus de reconnaissance aboutit à la génération d'une liste de lettres ou de mots possibles, éventuellement classés par ordre décroissant de vraisemblance. Le but principal est d'améliorer le taux de reconnaissance en faisant des corrections orthographiques ou morphologiques à l'aide de dictionnaires de digrammes, tri-grammes ou n-grammes. Quand il s'agit de la reconnaissance de phrases [74].

Le processus consiste à faire une sélection de la solution en utilisant des niveaux d'information plus élevés (syntaxique, lexicale, sémantiques…). Le post-traitement se charge également de vérifier si la réponse est correcte (même si elle est unique) en se basant sur d'autres informations non disponibles au classifieur [70].

5. Calligraphie et Typographie arabes

Dans cette section, nous présentons les caractéristiques morphologiques de l'écriture arabe. Nous exposons ensuite les problèmes majeurs rencontrés dans ce domaine.

5.1 Caractéristiques de l'écriture arabe

L'arabe est écrit par plus de 250 millions de personnes [92]. Bien que l'arabe parlée est quelque peut différente d'un pays à un autre, le système d'écriture est une version standard utilisée par tous le peuple arabe pour leur communication. L'automatisation de la reconnaissance des caractères arabes est considérée par plus de 27 nations qui utilisent les caractères arabes dans leur écriture [91, 96].

L'écriture arabe a été développée à partir d'un type d'Araméen. La langue araméenne comporte moins de consonants que l'arabe, alors de nouvelles lettres ont été créée en ajoutant des points aux lettres déjà existantes. D'autres petites marques appelées diacritiques sont utilisées pour indiquer de courtes voyelles, mais elles ne sont généralement pas utilisées [1,5].

Différemment de plusieurs autre langues tel que les scripts du Latin, du Chinois, et du Japonais qui sont largement examinés, la reconnaissance de texte arabe manuscrit reste un défis comme il y'a des travaux limités là-dessus [92]. L'écriture arabe possède les caractéristiques suivantes [91, 92, 96]:

- par nature, le script arabe est cursif,
- le texte arabe est écrit de droite à gauche,
- avec 28 lettres de base,
- plus de la moitié des lettres arabes sont composées d'un corps principal et des composantes secondaire: par exemples la lettre Beh (ب) et possède un point au-dessus de sa corps principal, Teh (ت) possède deux points et Kef (ك) possède un zigzag ci-joint avec le corps principal [96],

- le type et la position de la composante secondaire sont des caractéristiques très importante des lettres arabes. Par exemple, la reconnaissance de deux points au-dessous du corps principal est suffisant pour reconnaître la lettre Yeh (ي) parce que c'est la seule lettre possédant deux points au-dessous de sa corps principal. En outre, certaines lettres ne peuvent être distinguée que par les composantes secondaires. Par exemple: Teh (ت) et Theh (ث) se diffèrent seulement par le nombre de points au-dessus du corps principal, et Teh médiane (تـ) Yeh médiane (يـ) se diffèrent seulement par la position des deux points[96],
- il y'a une variation importante dans le dessin des composantes secondaire (voir tableau 1.2) [96],

	1	2	3
A			
B			
C			
D			
E			
F			

Tableau 1.2: Exemples de variations en dessin des composantes secondaire [96].

- parmi les lettres arabes 16 possèdent des points,
- Les points peuvent être un, deux ou trois, comme ils peuvent être au-dessus ou au-dessous de la ligne de base,
- D'un autre côté le largueur des lettres arabes diffère d'une lettre à une autre; dans les scripts manuscrits. En plus, habituellement il y'a des différences en formes des lettres d'une personne à une autre,
- Il y'a un nombre petit de lettre qui possèdent la même forme quelque soit la position comme: " و " et " ر ",
- La plupart des lettres des mots arabes sont liés entre eux, dépendant de leurs position (Début, Milieu, Fin) (voir tableau 1.3),

caractère	Position			
	Initiale	médiane	finale	Isolé
Alif			ـا	ا
Beh	بـ	ـبـ	ـب	ب
Teh	تـ	ـتـ	ـت	ت
Theh	ثـ	ـثـ	ـث	ث
Jim	جـ	ـجـ	ـج	ج
Ha	حـ	ـحـ	ـح	ح
Kha	خـ	ـخـ	ـخ	خ
Del			ـد	د
Thel			ـذ	ذ
Ra			ـر	ر
Zey			ـز	ز
Sin	سـ	ـسـ	ـس	س
Chin	شـ	ـشـ	ـش	ش
Sad	صـ	ـصـ	ـص	ص
Dhad	ضـ	ـضـ	ـض	ض
Tad	طـ	ـطـ	ـط	ط
Dha	ظـ	ـظـ	ـظ	ظ
Ayn	عـ	ـعـ	ـع	ع
Ghayn	غـ	ـغـ	ـغ	غ
Fa	فـ	ـفـ	ـف	ف
Qaf	قـ	ـقـ	ـق	ق
Kaf	كـ	ـكـ	ـك	ك
Lam	لـ	ـلـ	ـل	ل
Mim	مـ	ـمـ	ـم	م
Noun	نـ	ـنـ	ـن	ن
He	هـ	ـهـ	ـه	ه
Waw			ـو	و
Ya	يـ	ـيـ	ـي	ي

Tableau 1.3: Les différents formes de caractères selon la position dans le mot [1].

- Ces liaisons sont basées sur une ligne que nous l'appelons ligne de base (voir figue 1.9),

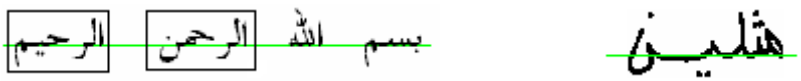
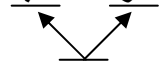


Figure 1.9: Exemple d'écriture arabe montrant la ligne de base [91].

- les lettres arabes peuvent nécessité 1, 2 ou 3 directions roulées par le stylo au cours de l'écriture (voir figure 1.10),

Figure 1.10: Directions roulées au cours de l'écriture [91].

- il existe trois types de mots en arabe:
  1. mots avec des caractères isolés: ودود
  2. mots avec des caractères liés: هبة
  3. mots avec des pseudo mots: حبا مر



02 pseudo mots

- la distance entre les pseudo mots peut être plus grande que celle entre les mots,
- certains lettres possèdent un Hamza (forme zigzag) et med (voir les tableaux suivants),

(a)

caractère	Position			
	Initiale	médiane	finale	Isolé
Alif+med			ـآ	آ
Alif+hamza			ـأ	أ
Teh			ـإ	إ
Waw+hamza			ـؤ	ؤ
Ya+hamza	ـئـ	ـئـ	ـئ	ئ

(b)

caractère	Position			
	Initiale	médiane	finale	Isolé
Ta			ـة	ة
Lamalif			ـلا	لا

(c)

caractère	Position			
	Initiale	médiane	finale	Isolé
Lamalif+med			ـلآ	لآ
Lamalif+hamza			ـلأ	لأ
			ـلإ	لإ

Tableau 1.4 : (a) Les caractères additionnels, (b) et (c) Hamza et Med et les positions qu'elles occupent avec Alif, Waw et Ya.

- certains lettres sont combinés ou ligaturés [90].

لـ، مـ	مـ، جـ	فـ، جـ	قـ، جـ
لـ، حـ	مـ، حـ	فـ، حـ	قـ، حـ
لـ، جـ	مـ، خـ	فـ، خـ	قـ، خـ
لـ، خـ			

Tableau 1.5 : Caractères susceptibles d'être ligaturés verticalement [1].

5.2 Alphabet arabe : Données graphiques

L'alphabet arabe n'a qu'un système d'écriture dans lequel les lettres sont liées ou ne sont pas liées entre elles selon des règles précises. Il existe différents styles d'écriture, mais dans aucun d'eux il est possible de juxtaposer des lettres totalement isolées les unes des autres. Il n'y a pas de lettres d'imprimerie en arabe, il n'y a que des caractères typographiques copiés de l'écriture manuscrite. Le caractère arabe est en effet dessiné non pas en fonction des contraintes géométriques des procédés de composition pour imprimerie, mais en fonction de la main et d'une esthétique visuelle

héritée de la calligraphie. La fonctionnalité et la lisibilité sont sacrifiées à l'esthétique calligraphique qui substitue l'élégance à la clarté [1,5].

5.3 Définition de la notion de fonte

Une *police* (fonte) est un ensemble de caractères d'une même famille, d'une même graisse et pour un corps donné. Ces caractéristiques typographiques sont normalisées dans l'imprimerie, tant au niveau du symbole (dimensions et dessin qui représente la forme et l'épaisseur du caractère), qu'à celui de la chaîne (mot : suite de symboles appartenant à la même fonte ou des fontes compatibles) dans chaque ligne du texte [1].

5.3.1 Notion de la chasse

La chasse est l'espace qu'occupe un caractère d'imprimerie. Elle dépend du dessin, du style et de la grosseur du caractère, et comprend en plus de la largeur, l'espace entre caractères [1].

5.3.2 Notion du corps

Le corps désigne la hauteur d'un caractère typographique comprenant le blanc de séparation horizontale avec la ligne au dessus. Le corps varie en fonction de l'usage prévu pour le *caractère :* texte courant, titrage ou affiche. La dimension du corps s'exprime en points. Le point est l'unité de mesure typographique, équivalent 0.376 mm [1].

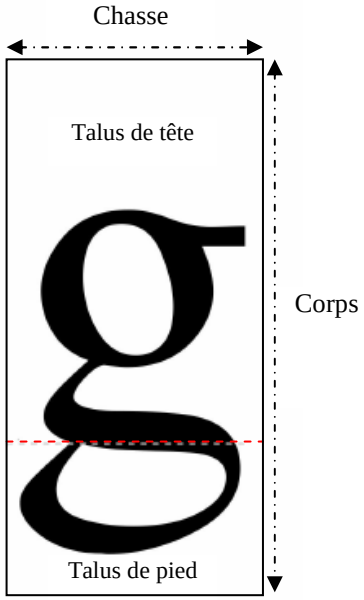


Figure 1.11: Chasse et corps d'un caractère [97].

6. Avancées en OCR arabe

L'OCR est utilisé dans beaucoup de domaines et des systèmes réels sont utilisés dans la vie quotidienne pour d'autres langues tel que l'anglais. Autant que les recherches restent ouvertes pour d'autres langues tel que l'arabe. En plus il est difficile d'appliquer les algorithmes utilisés pour les langues latins sur le langue arabe [90].

La reconnaissance de l'écriture arabe (AOCR : Arabic OCR) remonte aux années 70, depuis, plusieurs solutions ont été proposées. Elle sont aussi variées que celles utilisées dans le latin [5]. Dés les premiers travaux de reconnaissance de l'écriture arabe, les deux modes de reconnaissance, statique et dynamique ont été considérés. L'intérêt a été d'autant porté sur les travaux dans le domaine de l'écriture manuscrite que l'écriture imprimée [91].

6.1 Prétraitements

La littérature montre que les opérations de prétraitements connues en traitement d'images, ne sont pas toutes appropriées à l'Arabe Ce qui nécessite de proposer d'autres prétraitements qui prennent en compte les caractéristiques particulières de l'écriture arabe [10].

Le problème lié est que des boucles risquent d'être bouchées ou ouvertes. En plus, les points diacritiques peuvent être éliminés à la suite de certaines opérations de prétraitements ou encore confondus avec du bruit. En effet, les prétraitements peuvent altérer surtout la forme des points diacritiques de manière à les confondre avec du bruit s'ils sont trop amincis. Ils peuvent également être accolés au corps du caractère associé à cause d'une dégradation ou d'une normalisation de taille [70].

Une mauvaise squelettisation, peut aussi posée des problèmes particulièrement dans le cas du manuscrit, par exemple deux points peuvent être considérer comme un seul. Très souvent, dans les deux cas nous obtenant un segment de droite. Pour ces raisons, dans la plupart des travaux, les points sont éliminés au début du traitement [70].

Les étapes suivantes du traitement sont donc effectuées sur le corps du caractère (ou du PAW), ainsi le nombre de formes considérées est réduit sensiblement, la phase de classification devient moins complexe et plus rapide. Pour retrouver l'identité exacte du caractère une fois son corps identifié, un algorithme d'assemblage corps/points est utilisé [7,12].

6.2 La segmentation

La reconnaissance de caractères améliore l'interaction homme machine. Pour cette raison, un système de reconnaissance de caractères arabes réussi est extrêmement bénéfique, et son succès ne peut être accomplit sans qu'il ait surmonté la difficulté de la phase de segmentation [91].

Pour reconnaître un PAW il faut d'abord l'extraire de la page, donc, nous supposons qu'une décomposition de la page est préalablement faite, ce qui consiste à retrouver la structure physique du document en délimitant les différentes parties homogènes (texte, graphe, photographie …) [59].

6.2.1 Segmentation de texte en lignes

La segmentation de texte en lignes utilise souvent une projection horizontale afin d'extraire les lignes. Cependant la présence des points/diacritiques complique cette extraction et conduit parfois à la fusion des paragraphes [92].

Ce problème a lieu quand l'interligne est pris comme un seuil fixe calculé par une simple moyenne des différents interlignes (figure 1.11). Pour remédier à ce problème, l'utilisation d'un seuillage adaptatif est la solution. [59].

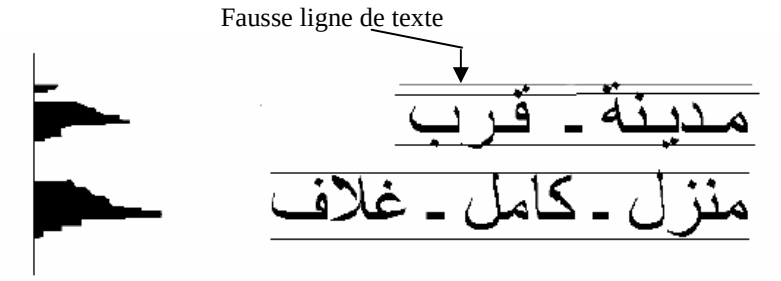


Figure 1.12: Exemple d'histogrammes horizontaux et d'une fausse ligne de texte qui en résulte [1].

6.2.2 Segmentation de ligne en mots et PAWs

La segmentation de ligne en mots et PAWs est réalisée en déterminant les histogrammes de projection verticale des différentes lignes de texte. Cependant, cette méthode pose des problèmes quand les PAWs se chevauchent verticalement (figure 1.12) [1]. Dans ce cas, d'autres techniques sont utilisées telles que la détermination du contour, du squelette, ou encore des composantes connexes. Le choix de la technique est souvent guidé par la méthode d'analyse [92].

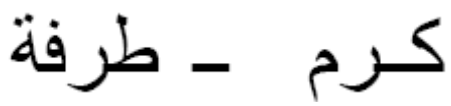


Figure 1.13: Exemple de chevauchement de PAWs respectivement de droite à gauche entre : « م، ر» et « ف، ر» [1].

6.2.3 Segmentation de PAWs en caractères

La segmentation en caractères constitue la tâche la plus délicate de la reconnaissance de l'écriture arabe. Les difficultés rencontrées à ce niveau sont du même type que celles affrontées lors de la reconnaissance du latin manuscrit, mais souvent plus complexes à cause de la diversité des formes du caractère arabe, de la courte liaison qui existe entre les caractères successifs, de l'allongement des ligatures horizontales et de la présence des ligatures verticales [1].

6.3 Extraction des primitives, classification

La synthèses des travaux étudiés, montre que les différents types de primitives (structurelles, géométriques, statistiques, transformations globales, corrélations…) et les différentes méthodes de classification (statistiques, structurelles, syntaxique…) qui existent dans la littérature, ont été pratiquement toutes utilisées dans la description de l'écriture arabe. Souvent, quatre arbres de décision sont élaborés, afin de déterminer l'identité du caractère selon sa position dans le PAW [12].

Les classifieurs connexionnistes constituent un nouveau paradigme en reconnaissance de formes, les travaux utilisant cette approche en AOCR, sont relativement récents.

Les modèles utilisés par la majorité des travaux, appartiennent à la famille des réseaux à couches. Le principe des réseaux à couches est de transmettre l'information recueillie sur une couche d'entrée vers une couche de sortie qui exprime la réponse du réseau [11, 12].

Par ailleurs, peu de travaux ont utilisés des méthodes de classification hybrides. Les études récentes en OCR recommandent cette approche, toutefois le choix ainsi que nombre de classifieurs, qui devraient être complémentaires, dépend de l'application considérée [59].

6.4 Post-traitement

Des vérifications contextuels classiques tels que la recherche dans un dictionnaire, les probabilités d'occurrence de bigramme et de trigramme…, sont appliquées dans les différents travaux qui prévoient un post-traitement. La méthode du dictionnaire est traditionnellement simplifiée pour accélérer la recherche et réduire la complexité du calcul : le dictionnaire est construit à partir de mots réduit à leurs racines, les suffixes et les préfixes sont éliminés. Cependant des modèles sont élaborés afin de spécifier la relation racine suffixe préfixe [70].

Par ailleurs, le post-traitement, malgré l'amélioration des scores qu'il peut apporter, n'est pas très utilisé en AOCR, ce qui peut s'expliquer par le manque de dictionnaires de validation et de statistiques élaborées par rapport au vocabulaire de référence. Or les statistiques sont relatives à l'application considérée et au vocabulaire de test [70].

Conclusion

Dans ce chapitre, nous avons vus quelques concepts de bases de l'OCR, les principales méthodes de reconnaissance en général. En plus, les principaux problèmes rencontrés dans ce domaine. Ensuite nous avons abordé les différentes étapes intervenant dans la conception d'un système de reconnaissance de caractères.

Par la suite, nous avons vus les principales propriétés morphologiques et typographiques de l'écriture arabe. Le manque de normalisation des typographies a montré la complexité de l'adaptation de l'écriture arabe aux exigences technologiques modernes.

De plus, nous avons vus les problèmes majeurs dans ce domaine qui se ramènent à la cursivité de l 'écriture et à la sensibilité de certaines caractéristiques topologiques de l'arabe à la dégradation, en l'occurrence les points diacritiques et les boucles. Donc, la reconnaissance optique de caractères arabe reste une tâche encore non résolue.

Chapitre 2

Panorama sur la segmentation

- **Introduction**
- **Problématique posée par l'écriture du cursif**
- **Structure physique et structure logique**
- **Principe général**
- **Stratégies de segmentation**
- **Segmentation de l'écriture cursive**
- **Composition du mot**
- **Conclusion**

Introduction

Bien reconnaître les mots d'un texte écrit revient à bien reconnaître les différents caractères constituant ces mots ce qui rend la phase de segmentation une phase très importante et cruciale dans le processus de reconnaissance. Elle permet d'avoir une séquence d'images représentant les différents caractères à partir d'une image source contenant le mot à reconnaître.

Dans ce chapitre nous allons exposer les différentes techniques de segmentation, et en particulier la segmentation du mot en caractères. Pour les différentes écritures (à savoir cursive ou non cursive).

1. Problématique posée par l'écriture du cursif

L'image d'un mot cursif est représentée par un signal bidimensionnel dans lequel, aucune information d'ordonnancement n'est présente, bien que par convention, il existe dans ce mot une séquence de lettres dont l'ordre logique d'interprétation va de la droite vers la gauche [76].

En regardant la figure 2.1 on constate qu'il est pas possible de déterminer a priori si le point B se trouve ou non dans une lettre antérieure au point A.

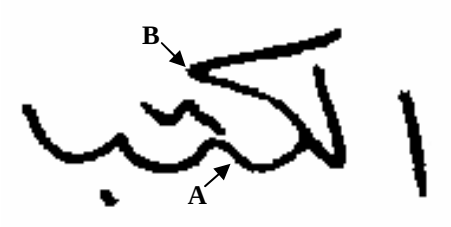


Figure 2.1: Mot cursif " **الكتب** " [76].

La seule façon de restituer l'ordre cohérent du tracé est de segmenter le mot en lettres. Or dans ce type d'écriture, la localisation précise du début et de la fin d'une lettre s'avère très difficile à réaliser, voire impossible. C'est pourquoi, le problème de la segmentation se trouve au cours de la reconnaissance hors-ligne de mots cursifs [76].

2. Structure physique et structure logique

Dans le domaine de l'analyse de documents, nous pouvons identifier deux types de recherches :

1. l'analyse de composition,
2. et l'interprétation du document.

Ces deux systèmes de traitement permettent de faire la distinction entre une information physique (correspondant aux objets physiques présents dans le document) et une information logique (liée à l'interprétation de l'organisation des objets du document) [59].

Le premier niveau de données accessibles au système d'analyse est la structure physique du document. Il concerne la répartition spatiale de l'information du document. La structure logique se rapporte au sens de cette organisation. La

connaissance de la structure physique permet de déduire la structure logique si les règles de présentation et de composition sont claires et connues [67].

3. Principe général

L'extraction des caractères est une étape importante dans un processus de reconnaissance, car elle est au moins assez importante que la reconnaissance des caractères elle-même, du moment qu'une bonne segmentation permet de présenter au système les caractères à reconnaître dans de bonnes conditions, une mauvaise segmentation quant à elle va entraîner une chute du taux de reconnaissance [64].

Généralement, nous pouvons distingués quatre niveaux de segmentation, comme suit:

- segmentation de la page,
- segmentation de texte en lignes,
- segmentation de lignes en mots,
- segmentation de mots en caractères.

La figure suivante montre le passage entre ces étapes.

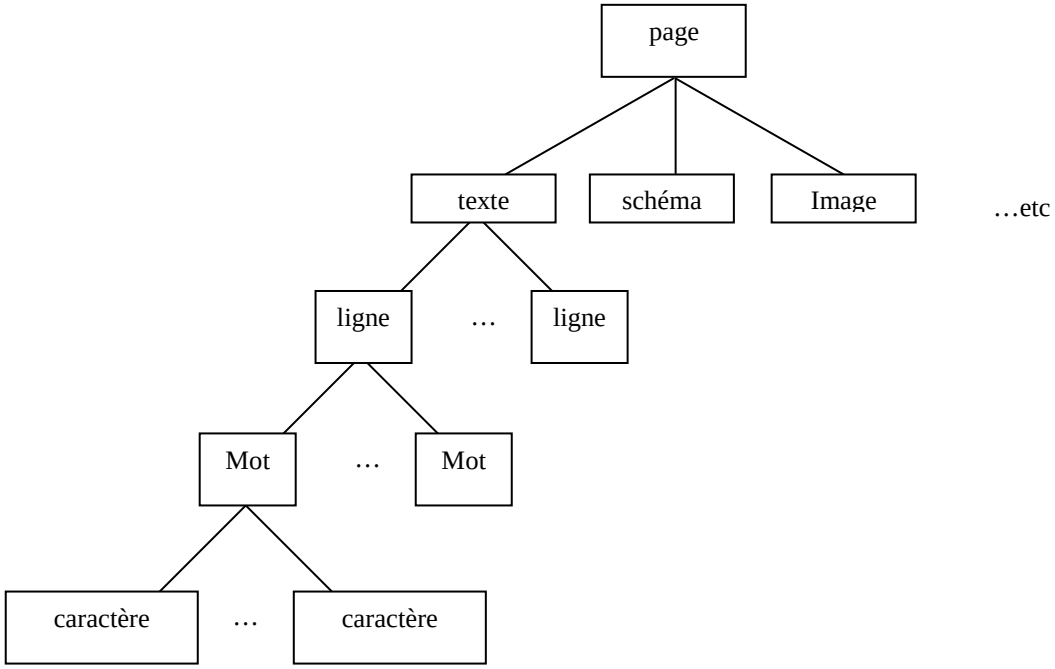


Figure 2.2 : Illustration du processus de segmentation.

Le premier permet de tirer les différentes zones d'information conformément à leur apparence physique, le second de séparer les lignes du texte, le troisième d'isoler les mots dans la phrase à mesure que le scripteur écrit, et le quatrième de localiser à leur tour les caractères dans chacun des mots isolés [55].

Chaque caractère segmenté est immédiatement validé par la reconnaissance. Les parties de mot non reconnues sont alors considérées comme étant formées de lettres cursives et envoyées au module spécialisé dans la reconnaissance de ce type de tracé [25].

3.1 Segmentation de la page

Cette étape permet de localiser dans chaque page, les zones d'information conformément à leur apparence physique. Elle est associée généralement à l'étiquetage logique qui consiste à déterminer la nature du media représenté dans chaque zone (texte, graphique, photographie etc.) [59]. Cette classification permet ensuite d'orienter la reconnaissance vers des systèmes spécialisés dans l'analyse de chaque type de media [60]. La figure suivante illustre cette étape.

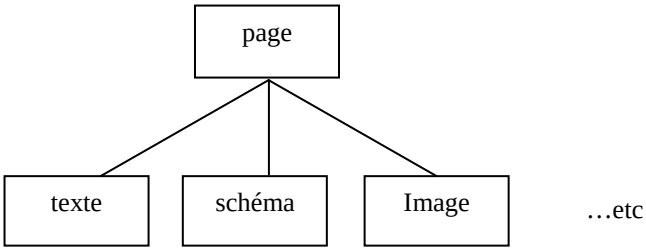


Figure 2.3 : Détection des différentes zones d'une page de document.

3.2 Segmentation d'un bloc de texte en lignes

Cette étape consiste à séparer les différentes lignes du texte pour en extraire les mots puis les caractères composants les mots. La plupart des études proposées dans ce domaine s'appuient sur une décomposition de l'image en composantes connexes [12].

D'autres par contre utilisent des techniques s'appuyant en grande partie sur les histogrammes des projections horizontale [9], et certains auteurs optent pour des méthodes spécialisées pour la segmentation en lignes de l'écriture arabe manuscrite [60]. La figure 2.4 montre un exemple d'extraction des lignes d'un texte.

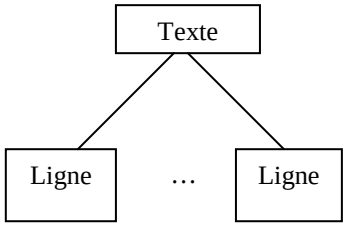


Figure 2.4 : Segmentation de texte en lignes.

3.3 Segmentation de lignes en mots

La segmentation en mots est réalisée en déterminant l'histogramme des projections verticales des lignes pour détecter les espaces entre les mots et pouvoir les séparer. Cependant cette technique peut ne pas être efficace dans certains cas où les mots se chevauchent (cas par exemple de l'écriture arabe) [55].

Dans ce cas d'autres techniques sont utilisées telles que : le suivi du contour, détermination du squelettes (voir section 5) ou la détermination des composantes connexes … [60]. La figure 2.5 montre un exemple de segmentation de ligne en mots.

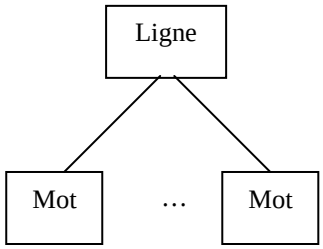


Figure 2.5 : Segmentation de Ligne en Mots.

3.4 Segmentation de mots en caractères

La segmentation des caractères est une opération qui tente de décomposer une image de séquence de caractères (mot) en sous-images de symboles individuels [64]. C'est l'un des processus de décision dans un système de reconnaissance optique de caractères. Son but est de décider si un motif isolé d'une image (caractère ou autre entité identifiable du mot) est correct ou non [60]. La figure suivante illustre cette étape.

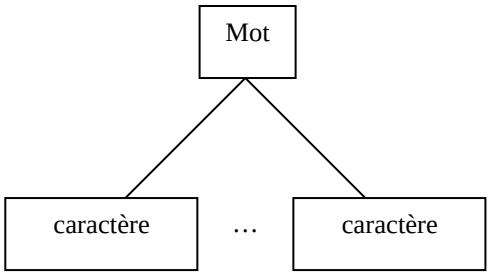


Figure 2.6 : Segmentation de Mot en Caractères.

4. Stratégies de segmentation

Certains auteurs parlent de segmentation *interne* et *externe*, dépendant de si la segmentation se fait séparément ou simultanément avec la reconnaissance. D'autres auteurs utilisent les termes *straight segmentation* et *segmentation recognition*, pour exprimer le même sens que précédemment [1].

Selon le point de vue de Casey et Lecolinet, la classification des méthodes suivant l'utilisation ou non de la reconnaissance durant la phase de segmentation n'est pas une bonne classification [64]. Parce que nous pouvons par exemple utiliser un correcteur d'orthographe comme post-processeur et dans ce cas il peut suggérer de substituer une lettre sortie par le classifieur par deux lettres, et cela est en fait une utilisation d'une segmentation de la sous image [74].

Selon lui la distinction entre les méthodes est basée sur comment la segmentation et la classification interagissent dans tout le processus. Dans l'exemple précédent par exemple la segmentation intervient en deux temps. Une fois avant la classification et une seconde fois après la classification.

Après examen des méthodes, il les classifie en trois stratégies de segmentation, plus d'autres méthodes hybrides à base des trois stratégies de base [12,64]:

- *Approche analytique explicite*, dans laquelle les segments sont identifiés à base de propriétés de ressemblance de caractères. Elle utilise une technique de découpage de l'image en composants significatifs elle est appelée *dissection*.
- *Approche analytique implicite*, dans laquelle le système cherche des composants qui correspondent à son alphabet dans l'image.
- *Approche globale*, dans lesquelles le système essaye de reconnaître le mot comme un tout. Evitant ainsi le besoin de segmentation en caractères.

A cela, s'ajoute les approches hybrides combinant dans des proportions différentes de ces trois approches élémentaires.

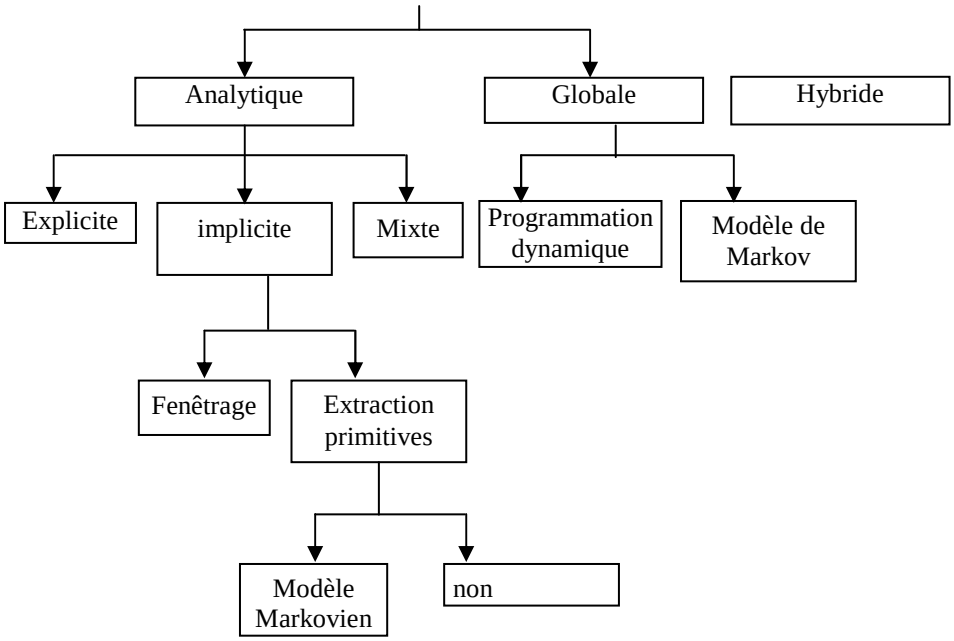


Figure 2.7: Hiérarchie des méthodes de segmentation selon R.G.Casey [76].

4.1 Approche analytique explicite

Cette approche classique, appelée aussi dissection par Casey et Lecolinet, repose sur la segmentation et la reconnaissance d'éléments caractéristiques proches des lettres (que nous appelons graphèmes) [64]. Il y'a dans ce cas une pré segmentation explicite qui s'appuie souvent sur des critères perceptifs tels que l'emplacement de ligatures entre les lettres. Puis, les graphèmes sont reconnus individuellement. L'identification du mot est achevée grâce à l'apport d'information contextuelle [76].

4.2 Approche analytique implicite

L'approche analytique implicite évite certains écueils de la segmentation explicite. Cette fois-ci, il n'y a pas de pré-segmentation du mot. La segmentation s'effectue pendant la reconnaissance. Le système recherche dans l'image, des composants ou des groupements de graphèmes qui correspondent à ses classes de lettres [65].

Classiquement, il peut le faire de deux manières [76]:

1. Fenêtrage, Le principe est d'utiliser une fenêtre mobile de largeur variable pour trouver des séquences de points de segmentations potentiels qui seront confirmés ou non par la reconnaissance de caractères (voir section 5.5).
2. Recherche de primitives, consiste à détecter les combinaisons de primitives qui donnent la meilleure reconnaissance possible. Pour faire la reconnaissance, différentes techniques sont employées dont les modèles de Markov cachés, réseaux de neurones…etc.

4.3 Approche analytique mixte

La plupart des méthodes actuelles pratiquent un mélange des deux stratégies précédentes [59]. D'abord, une phase de pré segmentation est appliquée pour sur segmenter l'image. Puis la meilleure combinaison de reconnaissance est choisie, en considérant les combinaisons possibles de 1, 2,…n graphèmes [76].

4.4 Approche globale

Dans l'approche globale, le mot est reconnu par sa forme générale. Il n'y a pas de segmentation (ni explicite, ni implicite) [12]. C'est pourquoi cette approche est censée être robuste au bruit ou aux imperfections du signal. Habituellement, la reconnaissance dépend d'un lexique (une liste de modèles de mots) [56].

Puisque cette méthode nécessite un modèle pour chaque mot du lexique et que chaque modèle doit être appris, elle est plutôt utilisée pour des applications où le lexique est restreint comme c'est le cas dans le traitement automatique des chèques. Ici aussi, différentes techniques sont employées dont la programmation dynamique et les modèles de Markov cachés [76].

4.5 Approches hybrides

Les approches hybrides combinent plusieurs stratégies de reconnaissance afin d'exploiter les forces et les faiblesses d'approches complémentaires (stratégies ascendantes et descendantes). Il y a une pré segmentation d'où sont tirées des hypothèses qui seront validées par la suite [76]. L'ajout d'informations supplémentaires sur le style du scripteur tel que proposé par Crettez, peut contribuer également é améliorer les performances en choisissant un extracteur de primitives spécifique à chacun des styles d'écriture [76].

5. Segmentation de l’écriture cursive

Les graphèmes sont des images extraites de l’image à segmenter. Passer d’une seule image à une séquence de graphèmes pose le problème de la taille de ces éléments. Ils ne doivent pas être trop petits afin d’être statistiquement significatifs, et pas trop gros afin de ne pas dépasser la taille d’une lettre. Il est en effet important qu’un graphème donné soit une sous partie d’une seule lettre : cette condition est nécessaire pour construire un modèle de mot comme étant la concaténation de modèles de lettres [70].

5.1 Segmentation à partir du squelette

A partir du squelette, on cherche à repérer certains motifs, pour en déduire les candidats de points de coupures. La détection de ces motifs introduit des calculs de

courbures et d'angles, qui sont comparées à des seuils ajustés de manière à obtenir le résultat désiré [90].

X.Dupré [76] souligne que cette approche est erronée dans environ 10% des cas. Les configurations difficiles à segmenter sont celles pour lesquelles les lettres sont souvent enchevêtrées, comme les "tt", ou les lettres `a liaison haute ('b', 'o', 'v', 'w') avec leur successeur [70].

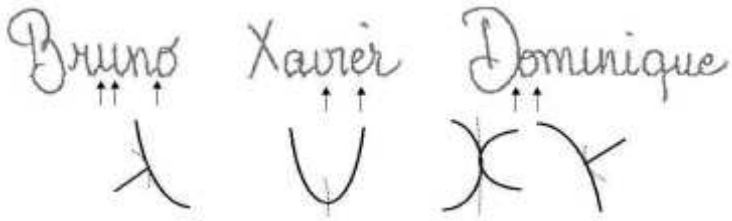


Figure 2.8: Segmentation à base du squelette [70].

5.2 Segmentation à partir du contour

La segmentation à partir du contour consiste à déterminer les meilleurs points candidats de coupure entre graphèmes, en s'appuyant sur les extrema locaux du contour, qui sont associées selon un critère de proximité (voir figure 2.9) [70].

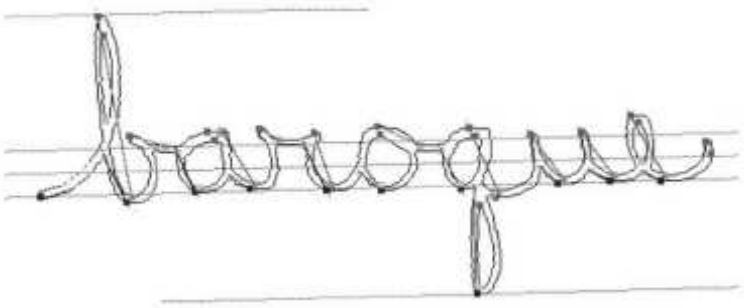


Figure 2.9: Extrema du contour supérieur et inférieur sont associés, et reliés par une corde [70].

Comme la segmentation en graphèmes à partir du contour nécessite de nombreux ajustements avant de trouver les critères de décision. Cette mise au point par tâtonnements est le point commun de nombreux traitements d'images liés à la reconnaissance de l'écriture manuscrite. Faciles à ajuster lorsque la qualité de l'écriture est bonne, ces prétraitements peuvent avoir des comportements tout `a fait erratiques lorsque l'écriture est de mauvaise qualité [70].

5.3 Segmentation à partir des histogrammes

La segmentation en utilisant des histogrammes est méthode proposée par B. Yanikoglu et P. Sandon [76]. Elle consiste à calculer des histogrammes de projection dans plusieurs directions proches de la verticale [12]. Les droites choisies sont celles qui interceptent le moins de pixels noirs, avec une contrainte d'espacement régulier dans l'image (voir figure 2.10) [9].

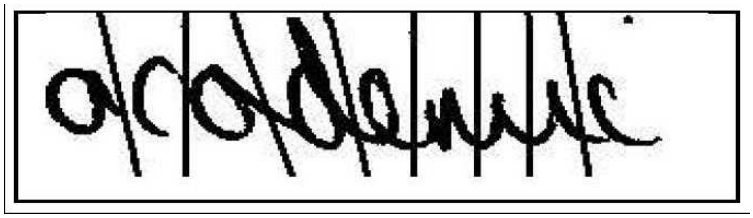


Figure 2.10: Segmentation à partir d'histogrammes de projection selon plusieurs directions [70].

Néanmoins, Cette méthode montre des limites lorsque les lettres sont très proches ou enchevêtrées.

5.4 Segmentation basée sur des réservoirs

X. Dupré étend à l'écriture cursive la technique à base de réservoirs initialement appliquée à la segmentation de chiffres liés. Il souligne que les règles de décision sont plus difficiles à mettre en place dans le cas des lettres, car ces dernières sont de tailles variables [70].

5.5 Segmentation basée sur les fenêtres glissantes

Le principe est d'utiliser une fenêtre mobile de largeur variable en découpant l'image en bandes verticales. Ce découpage peut être régulier ou non, éventuellement avec recouvrement partiel des bandes successives (voir figure 2.11). Ce qui permet de trouver des séquences de points de segmentations potentiels qui seront confirmés ou non par la reconnaissance de caractères [12].

En variant la taille de la fenêtre et sa position, on obtient plusieurs séquences de points de segmentation qui seront analysées par le système de reconnaissance. L'analyse du contenu de la fenêtre peut se faire directement sur les pixels de l'image ou peut s'opérer sur le regroupement de primitives de bas niveau. Cette méthode nécessite deux étapes [76]:

1. génération d'hypothèse de segmentation (séquences de points de segmentation obtenus par le fenêtrage),
2. choix de la meilleure hypothèse de l'étape de la reconnaissance (validation).

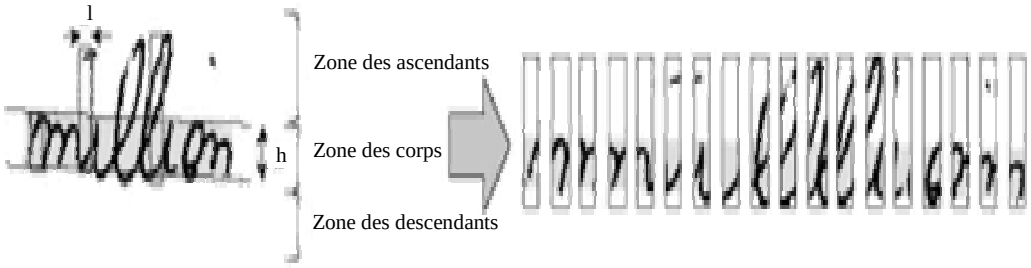


Figure 2.11: Segmentation à base de fenêtre glissante : découpage du mot en bandes verticales [70].

Cette technique présente l'avantage d'être simple, robuste au bruit, et est indépendante de la connexité. Néanmoins, la largeur de la fenêtre d'observation n'est pas facile à déterminer a priori et il faut gérer les conflits entre les différentes

hypothèses envisagées [6]. De plus, la séquence générée d'images contient beaucoup de bruit (recouvrement de deux lettres successives). C'est également vrai dans le cas des lettres superposées verticalement, mais qui ne se touchent pas nécessairement : une barre de 't' avec la lettre suivante, ou les descendants comme ' ر ' ou ' و ' en arabe [70].

6. Composition du mot

La composition est le processus inverse de la segmentation, durant lequel le mot est construit à l'aide des différentes étiquettes (labels) de caractères obtenus après la phase de classification plus un dictionnaire contenant les modèles des mots [56].

Chaque étiquette est comparée avec l'étiquette du mot du dictionnaire tant que la séquence de ces étiquettes nous répétons l'opération sur l'étiquette suivante sinon le mot considéré comme connu et l'étiquette en cours de traitement est pris pour la construction d'un nouvel mot (voir figure 2.12) [52].

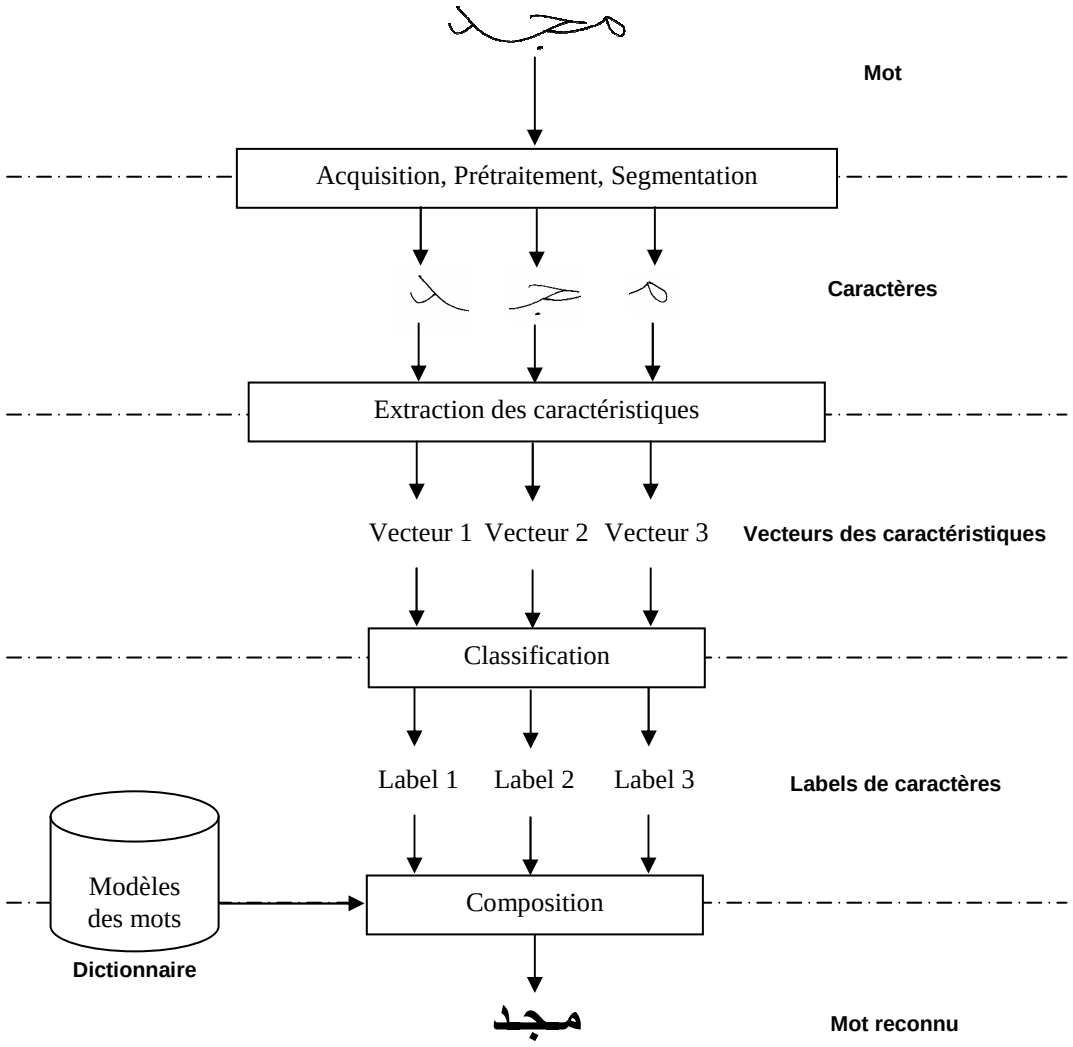


Figure 2.12: Processus de composition.

Conclusion

Dans ce chapitre nous avons essayé d'exposer les différentes méthodes utilisées dans la segmentation du mot. Ces méthodes ont connu beaucoup de progrès ces dernières années. Des techniques variées influencées par l'évolution dans les domaines tels que la reconnaissance de la parole et la reconnaissance en ligne des caractères ont émergés.

Une segmentation performante dépend généralement de plusieurs facteurs :

- la nature et la qualité du document,
- l'outil d'acquisition (scanner), et
- méthodes de prétraitement et algorithmes de segmentation choisis.

Le taux de mauvaise segmentation croit progressivement à partir de l'écriture imprimée à l'écriture manuscrite jusqu'à l'écriture manuscrite cursive où la difficulté devient plus importante. La performance d'un système de reconnaissance de l'écriture ne dépend pas seulement des résultats de la phase de segmentation, mais aussi du type de classifieur utilisé pour la reconnaissance de ces segments.

Chapitre 3

Support Vector Machines

- **Introduction**
- **Pourquoi les machines à vecteur de support**
- **Apprentissage statistique et SVM**
- **SVM principe de fonctionnement général**
- **Fondements mathématiques**
- **Les domaines d'applications**
- **Conclusion**

Introduction

Parmi les méthodes à noyaux, inspirées de la théorie statistique de l'apprentissage de Vladimir Vapnik, les Machines à Vecteurs de Support (SVM) constituent la forme la plus connue. SVM est une méthode de classification binaire par apprentissage supervisé, elle fut introduite par Vapnik en 1995. Cette méthode est donc une alternative récente pour la classification.

Elle repose sur l'existence d'un classificateur linéaire dans un espace approprié. Puisque c'est un problème de classification à deux classes, cette méthode fait appel à un jeu de données d'apprentissage pour apprendre les paramètres du modèle. Elle est basée sur l'utilisation de fonctions dites noyau (kernel) qui permettent une séparation optimale des données. Dans la présentation des principes de fonctionnements, nous schématiserons les données par des « points » dans un plan.

1. Pourquoi les Machines à Vecteurs de Support (SVM) ?

L'algorithme des machines à vecteurs de support a été développé dans les années 90 par le russe Vladimir Vapnik. Initialement, les SVM ont été développé comme un algorithme de classification binaire supervisée. Il s'avère particulièrement efficace de par le fait qu'il peut traiter des problèmes mettant en jeu de grands nombres de descripteurs, qu'il assure une solution unique (pas de problèmes de minimum local comme pour les réseaux de neurones) et il a fourni de bons résultats sur des problèmes réels [34].

L'algorithme sous sa forme initiale revient à chercher une frontière de décision linéaire entre deux classes, mais ce modèle peut considérablement être enrichi en se projetant dans un autre espace permettant d'augmenter la séparabilité des données. On peut alors appliquer le même algorithme dans ce nouvel espace, ce qui se traduit par une frontière de décision non linéaire dans l'espace initial [34].

2. Apprentissage statistique et SVM

La notion d'apprentissage étant importante, nous allons commencer par effectuer un rappel. L'apprentissage par induction permet d'arriver à des conclusions par l'examen d'exemples particuliers. Il se divise en apprentissage supervisé et non supervisé. Le cas qui concerne les SVM est l'apprentissage supervisé. Les exemples particuliers sont représentés par un ensemble de couples d'entrée/sortie. Le but est d'apprendre une fonction qui correspond aux exemples vus et qui prédit les sorties pour les entrées qui n'ont pas encore été vues. Les entrées peuvent être des descriptions d'objets et les sorties la classe des objets donnés en entrée [27].

2.1 Objectif de l'apprentissage statistique

Effectuer une classification consiste à déterminer une règle de décision capable, à partir d'observations externes, d'assigner un objet à une classe parmi plusieurs. Le cas le plus simple consiste à discriminer deux classes. D'une manière plus formelle, la classification bi-classe revient à estimer une fonction $f : x \rightarrow \{+1,-1\}$ à partir d'un ensemble d'apprentissage constitué de couples (xi, yi), qu'on suppose i.i.d. suivant une distribution de probabilité $P(x, y)$ inconnue, tels que

$$(xi, yi) \in X \times Y \text{ où } i = 1,\ldots, Nx \text{ et } Y = \{+1,-1\},$$

de sorte à ce que f classifie correctement des exemples inconnus(xt, yt). Par exemple, on peut assigner xt à la classe (+1) si f (xt) ≥ 0, et à la classe (-1) sinon. Les exemples inconnus sont supposés suivre la même distribution de probabilité P(x, y) que ceux de l'ensemble d'apprentissage. La meilleure fonction f est celle obtenue en minimisant le risque :

$$R[f] = \int L\,[f(x), y]\, d\,P(x, y). \qquad (4.1)$$

Où L désigne une fonction de coût, comme par exemple :

$$L\,[f(x),y] = (f(x)-y)^2$$

Malheureusement, le risque (4.1) ne peut être directement minimisé dans la mesure où la distribution de probabilité sous-jacente P(x, y) est inconnue. Aussi, on va chercher une fonction de décision proche de celle optimale à partir de dont on dispose, c'est-à-dire l'ensemble d'apprentissage et la classe de fonctions F est à laquelle la solution f appartient. Pour ce faire, on approxime le minimum du risque théorique par le minimum du risque empirique qui s'écrit :

$$R_{emp}[f] = \frac{1}{Nx} \sum_{i=1}^{Nx} L\,[f(x_i),y_i]\ . \qquad (4.2)$$

Il est possible de donner des conditions au classifieur pour qu'asymptotiquement (si $Nx \rightarrow \infty$), le risque empirique (4.2) converge vers le risque (4.1). Cependant, si on dispose de peu d'exemples pour faire l'apprentissage (i.e Nx petit), on s'expose au risque de sur-apprentissage (Figure 4.1). Pour éviter le sur-apprentissage, on peut restreindre la complexité de la classe F à laquelle appartient *f*. Intuitivement, une fonction de décision simple (la classe la plus simple se constituant des fonctions linéaires) capable de discriminer correctement les données est préférable à une fonction complexe. Pour cela, on introduit un terme de régularisation pour limiter la complexité des fonctions de F.

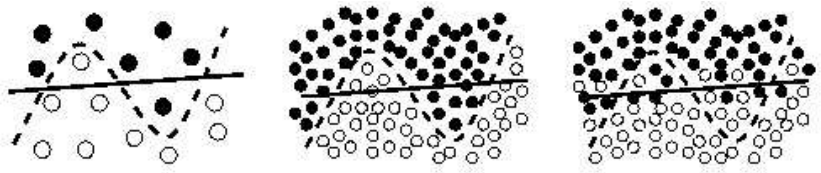


Figure 3.1 : Exemple du problème de sur apprentissage.

Etant donné un petit ensemble d'apprentissage (schéma de gauche), deux frontières de discrimination (représentées par les lignes continue et discontinue) sont possibles. La ligne discontinue est plus complexe mais minimise davantage le risque empirique. Seul un ensemble d'exemples plus grand permet de déterminer la meilleure des deux frontières de décision. S'il s'agit de la ligne discontinue, alors la ligne continue n'est pas suffisamment discriminante (schéma du milieu) ; s'il s'agit de la ligne continue, alors la ligne discontinue ne convient pas et caractérise un sur apprentissage (schéma

de droite) [34].

2.2 Théorie de Vapnik-Chervonenkis

Une manière de contrôler la complexité d'une classe de fonctions est donnée par la théorie de Vapnik-Chervonenkis (VC) et le principe de minimisation du risque structurel. Ici, le concept de complexité de la fonction de décision f s'exprime par la dimension de VC (notée h) de la classe de fonctions F à laquelle appartient f. Grossièrement, la dimension de VC mesure combien d'échantillons de l'ensemble d'apprentissage peuvent être séparés par toutes les classifications possibles issues des fonctions de la classe [34].

Considérons une famille imbriquée de classes de fonctions:

$$F_1 \subset F_2 \subset \ldots \subset F_k ;$$

Avec une dimension de VC non-décroissante, et $f_1 \ldots f_k$ les fonctions minimisant le risque empirique dans chacune de ces classes.

La minimisation du risque structurel consiste à choisir la classe F_i (et la fonction f_i) de sorte à ce qu'une borne supérieure de l'erreur de généralisation puisse être minimisée (grâce, par exemple, au théorème suivant) [34].

Théorème 1 : Soient h la dimension de VC de la classe de fonctions F, Remp[f] le risque empirique défini par (4.2) avec la fonction perte 0/1 (i.e. L [f(xi),yi] =H(-yf(x)) Où H désigne la fonction de Heaviside). Pour tout $\delta>0$ et $f \in F$, l'inégalité bornant le risque

$$R[f]= R_{emp}[f]+\sqrt{\frac{h(\ln\frac{2Nx}{h}+1)-\ln(\frac{\delta}{4})}{Nx}} \qquad (4,3)$$

est vraie avec une probabilité de moins $(1- \delta)$pour $Nx>h$ [12].

Cette borne n'est qu'un exemple et des formulations du même type ont été démontrées pour d'autres fonctions perte et d'autre mesures de complexité. Le but recherché ici est de minimiser l'erreur de généralisation R[f] en obtenant un faible risque empirique Remp [f] tout en gardant la plus petite classe de fonctions possible.

L'inégalité (4.3) fait apparaître deux cas extrêmes:

- une très petite classe de fonctions (par exemple F_1) fait décroître rapidement le terme de complexité (celui en racine carrée), mais le risque empirique demeure grand,

- une très grande classe de fonctions (par exemple F_k) implique un risque empirique petit, mais le terme de complexité explose.

La meilleure classe de fonctions est généralement intermédiaire entre la plus petite et la plus grande, puisque nous cherchons d'avoir une fonction qui explique au mieux

les données tout en préservant un faible risque empirique (Figure 4.2) [34].

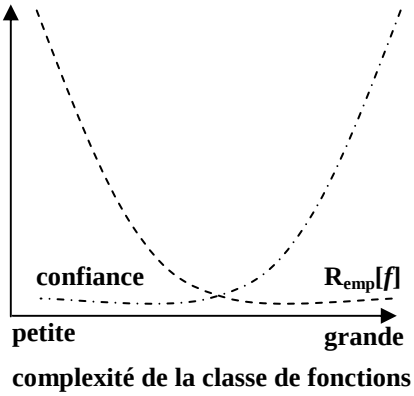


Figure 3.2 : Illustration de l'inégalité (4.3) [34].

La courbe croissante, appelée confiance, correspond à la borne supérieure du terme de complexité. Les comportements du terme de complexité et de l'erreur empirique sont clairement opposés. On recherche donc le meilleur compromis entre complexité et erreur empirique [34].

2.3 Marge et dimension de VC

Supposons pour l'instant que les échantillons de l'ensemble d'apprentissage sont séparables par un hyperplan (Figure 4.3), i.e on choisit des fonctions de décision de la forme :

$$f(x)=\langle w, x\rangle +b. \qquad (4.4)$$

La marge est la distance minimale entre les échantillons de l'ensemble d'apprentissage et la frontière de décision.

Il a été montré que pour la classe des hyperplans, la dimension de VC peut être bornée en fonction de la marge. La marge peut à son tour être mesurée grâce au vecteur poids *w*: puisque nous supposons que les échantillons sont séparables, on peut redéfinir *w* et *b* de sorte à ce que les échantillons *x* les plus proches de l'hyperplan satisfassent $|\langle w, x\rangle +b|=1$.

Considérons maintenant deux échantillons x_1 et x_2 de classes différentes telles qu'on ait $\langle w, x_1\rangle +b = +1$et $\langle w, x_2\rangle +b = -1$. La marge γ correspond alors à la distance entre x_1 et x_2 mesurée perpendiculairement à l'hyperplan :

$$\gamma=\langle w / \|w\|, x_1 - x_2\rangle =2/\|w\| ;$$

Les résultats liant la dimension de VC de la classe des hyperplans de séparation à la marge et à la longueur du vecteur poids *w* sont respectivement donnés par les inégalités suivantes :

Où R est le rayon de la plus petite boule englobant les données. Ainsi, en bornant la marge de la classe de fonction, on peut contrôler sa dimension de VC [30,34].

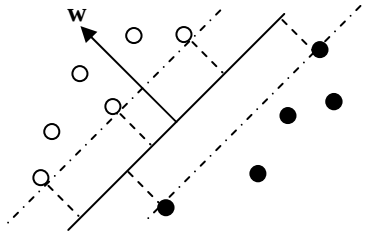


Figure 3.3 : Classifieur linéaire et marge [34].

Un classifieur linéaire est défini par un vecteur normal à l'hyperplan w et un biais b:la frontière de décision est {x| $\langle w, x \rangle$ +b=0}(ligne continue).Chacun des deux sous-espaces séparés par l'hyperplan correspond à une classe, i.e. f(x)=signe($\langle w, x \rangle$ +b). La marge du classifieur linéaire est la distance minimale entre les échantillons de l'ensemble d'apprentissage et la frontière de décision. Sur le schéma, il s'agit de la distance entre la ligne continue et les lignes discontinues [34].

3. SVM principe de fonctionnement général

3.1 Notions de base: Hyperplan, marge et support vecteur

Pour deux classes d'exemples donnés, le but de SVM est de trouver un classificateur qui va séparer les données et maximiser la distance entre ces deux classes. Avec SVM, ce classificateur est un classificateur linéaire appelé hyperplan.

Dans le schéma qui suit, on détermine un hyperplan qui sépare les deux ensembles de points [27].

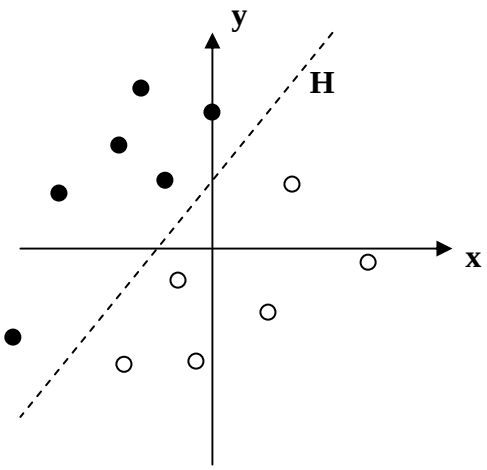


Figure 3.4 : Exemple d'un hyperplan séparateur [27].

Les points les plus proches, qui seuls sont utilisés pour la détermination de l'hyperplan, sont appelés vecteurs de support.

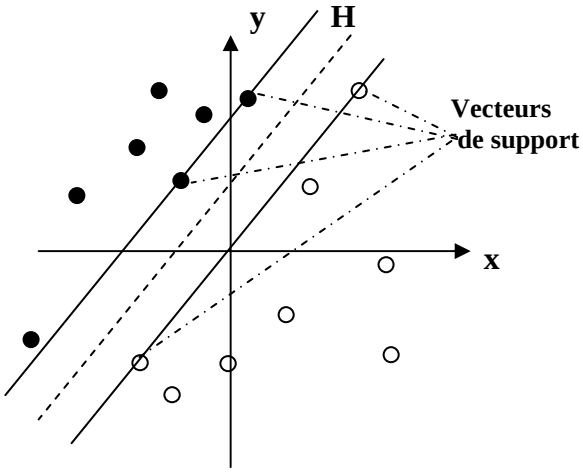


Figure 3.5 : Exemple de vecteurs de support [27].

Il est évident qu'il existe une multitude d'hyperplan valide mais la propriété remarquable des SVM est que cet hyperplan doit être optimal. Nous allons donc en plus chercher parmi les hyperplans valides, celui qui passe « au milieu » des points des deux classes d'exemples. Intuitivement, cela revient à chercher l'hyperplan le « plus sûr » [49].

En effet, supposons qu'un exemple n'ait pas été décrit parfaitement, une petite variation ne modifiera pas sa classification si sa distance à l'hyperplan est grande. Formellement, cela revient à chercher un hyperplan dont la distance minimale aux exemples d'apprentissage est maximale [49].

On appelle cette distance « marge » entre l'hyperplan et les exemples. L'hyperplan séparateur optimal est celui qui maximise la marge. Comme on cherche à maximiser cette marge, on parlera de séparateurs à vaste marge [49].

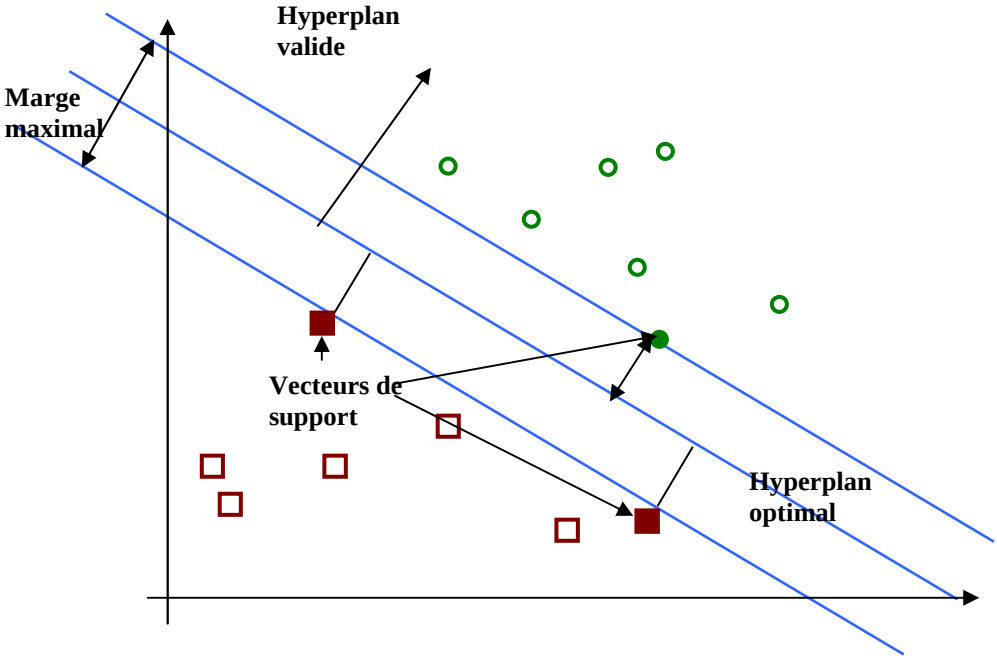


Figure 3.6 : Exemple de marge maximal (hyperplan valide).

3.2 Pourquoi maximiser la marge ?

Intuitivement, le fait d'avoir une marge plus large procure plus de sécurité lorsque l'on classe un nouvel exemple. De plus, si l'on trouve le classificateur qui se comporte le mieux vis-à-vis des données d'apprentissage, il est clair qu'il sera aussi celui qui permettra au mieux de classer les nouveaux exemples. Dans le schéma qui suit, la partie droite nous montre qu'avec un hyperplan optimal, un nouvel exemple reste bien classé alors qu'il tombe dans la marge. On constate sur la partie gauche qu'avec une plus petite marge, l'exemple se voit mal classé [27].

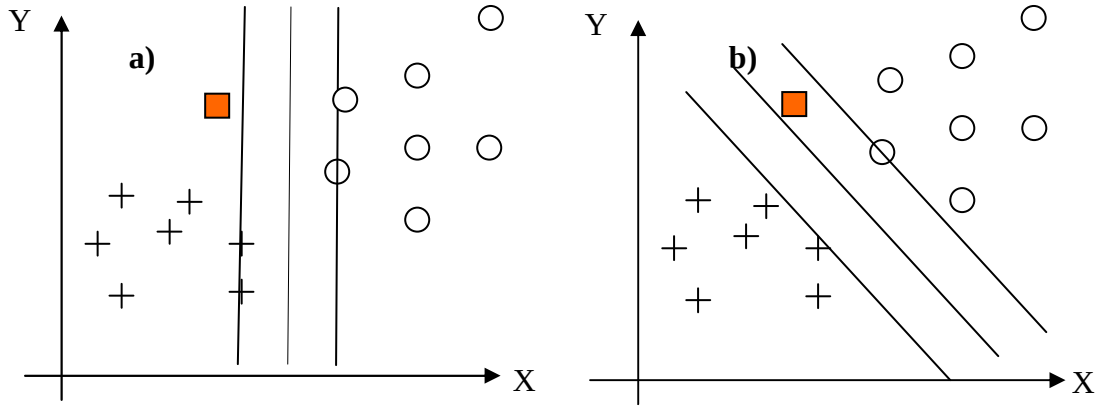


Figure 3.7 :a) Hyperplan avec faible marge, **b)** Meilleur hyperplan séparateur [27].

En général, la classification d'un nouvel exemple inconnu est donnée par sa position par rapport à l'hyperplan optimal. Dans le schéma suivant, le nouvel élément sera classé dans la catégorie des « + ».

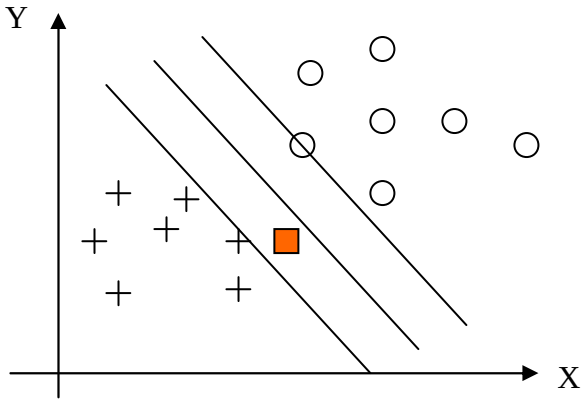


Figure 3.8 : Exemple de classification d'un nouvel élément.

3.3 Linéarité et non-linéarité

Parmi les modèles des SVM, on constate les cas linéairement séparable et les cas non linéairement séparable. Les premiers sont les plus simples de SVM car ils permettent de trouver facilement le classificateur linéaire. Dans la plupart des problèmes réels il n'y a pas de séparation linéaire possible entre les données, le classificateur de marge maximale ne peut pas être utilisé car il fonctionne seulement si les classes de données d'apprentissage sont linéairement séparables [27].

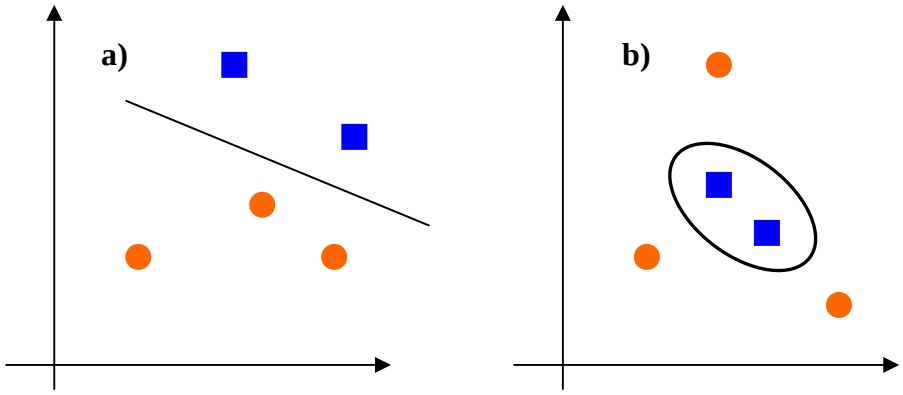


Figure 3.9 : a) Cas linéairement séparable, **b)** Cas non linéairement séparable [27].

3.4 Cas non linéaire

Pour surmonter les inconvénients des cas non linéairement séparable, l'idée des SVM est de changer l'espace des données. La transformation non linéaire des données peut permettre une séparation linéaire des exemples dans un nouvel espace. On va donc avoir un changement de dimension. Cette nouvelle dimension est appelé « espace de re-description ».En effet, intuitivement, plus la dimension de l'espace de re-description est grande, plus la probabilité de pouvoir trouver un hyperplan séparateur entre les exemples est élevée. Ceci est illustré par le schéma suivant [27]:

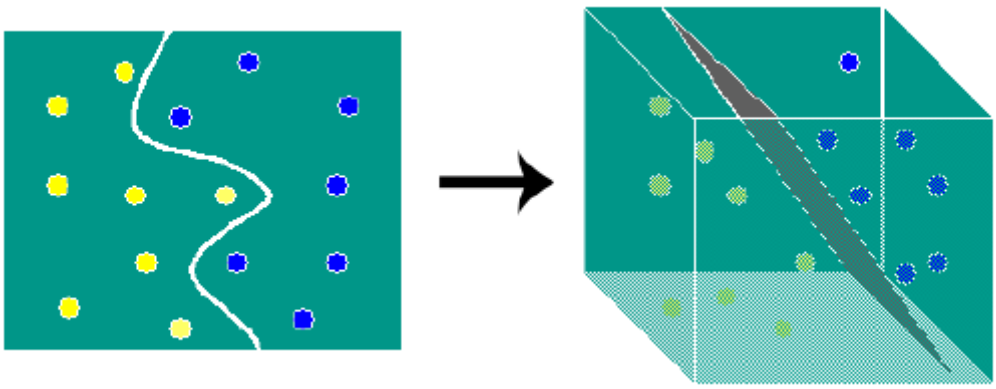


Figure 3.10 : Exemple de changement de l'espace de données.

On a donc une transformation d'un problème de séparation non linéaire dans l'espace de représentation en un problème de séparation linéaire dans un espace de re-description de plus grande dimension.Cette transformation non linéaire est réalisée via une fonction noyau [27].

En pratique, quelques familles de fonctions noyau paramétrables sont connues et il revient à l'utilisateur de SVM d'effectuer des tests pour déterminer celle qui convient le mieux pour son application. On peut citer les exemples de noyaux suivants : polynomiale, gaussien, sigmoïde et laplacien [27].

3.5 Illustration de transformation de cas non linéaire : le cas XOR

Le cas de XOR n'est pas linéairement séparable, si on place les points dans un plan à deux dimensions, on obtient la figure suivante :

Coordonnées des points : (0,0) ; (0,1) ; (1,0) ; (1,1)

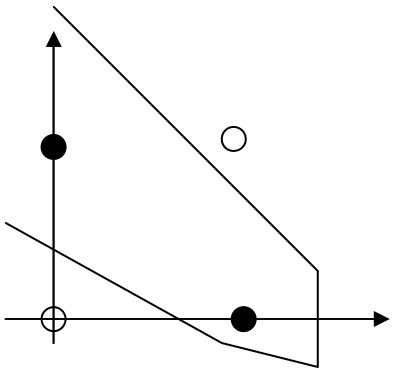


Figure 3.11 : Illustration de cas non linéairement séparable (le cas XOR) [27].

Si on prend une fonction polynomiale (x,y) → (x,y,x.y) qui fait passer d'un espace de dimension 2 à un espace de dimension 3, on obtient un problème en trois dimensions linéairement séparable :

(0,0) → (0,0,0)
(0,1) → (0,1,0)
(1,0) → (1,0,0)
(1,1) → (1,1,1)

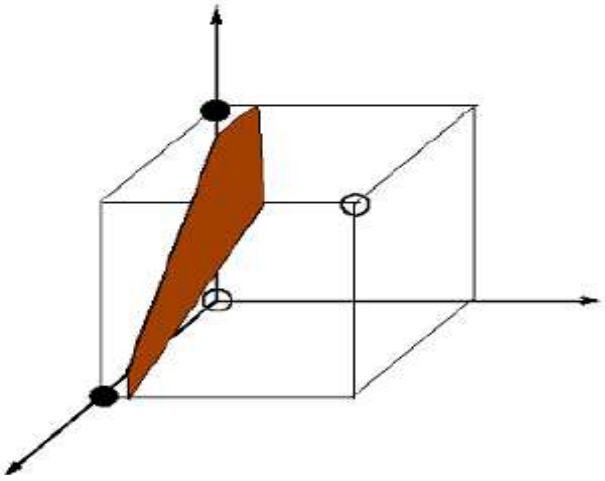


Figure 3.12 : Illustration de passage d'un espace 2D à un espace 3D [27].

4. Fondements mathématiques

Nous allons détailler dans les paragraphes ci-dessous les principes mathématiques sur lesquels repose SVM.

4.1 Problème d'apprentissage

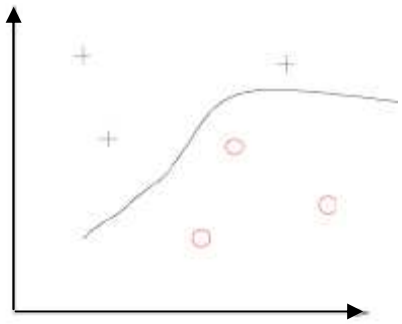
On s'intéresse à un phénomène f (éventuellement non déterministe) qui, à partir d'un certain jeu d'entrées x, produit une sortie y = f(x).

Le but est de retrouver cette fonction f à partir de la seule observation d'un certain nombre de couples entrée-sortie {(xi; yi) : i = 1,…, n} afin de « prédire » d'autres évènements.

On considère un couple (X, Y) de variables aléatoires à valeurs dans X x Y.
Seul le cas Y = {-1, 1} (classification) nous intéresse ici (on peut facilement étendre au cas
card(Y) = m > 2 et au cas $Y = \Re$. La distribution jointe de (X, Y) est inconnue.

Sachant qu'on observe un échantillon S = {(X1, Y1),..., (Xn, Yn)} de n copies indépendantes de (X, Y), on veut: construire une fonction h : $X \rightarrow Y$ telle que P(h(X) != Y) soit minimale [39].

Illustration :
Trouver une frontière de décision qui sépare l'espace en deux régions (pas forcément connexes).

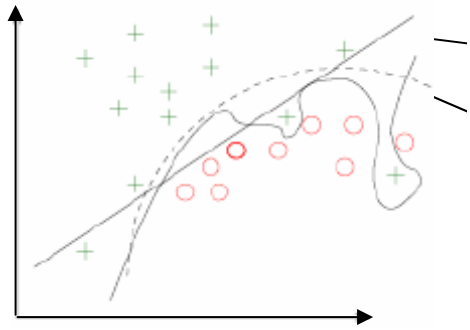


Connaissant h, on peut en déduire la classification des nouveaux points c'est à dire trouver une frontière de décision.

Le problème est de trouver une frontière assez éloignée des points de différentes classes. C'est ce qui constituera l'un des problèmes majeurs de classification grâce aux SVMs [39].

Figure 3.13 : Problème détermination de frontière assez éloignée des points de différentes classes [39].

Sur et sous- apprentissage :



Si les données sont générées par un modèle quadratique :

Le modèle linéaire est en situation de sousapprentissage

Le modèle de haut degré est en situation de surapprentissage (apprentissage par coeur)

Il faut donc trouver un compromis entre adéquation aux données et complexité pour pouvoir généraliser.

Figure 3.14 : Illustration des sous et sur apprentissage [39].

4.2 Classification à valeurs réelles

Plutôt que de construire directement h : $\boldsymbol{X}\rightarrow$ {-1, 1}, on construit :
f : $X \rightarrow R$ (ensemble des réels).La classe est donnée par le signe de f ;
h = signe (f).

L'erreur se calcule avec P (h(X) != Y) = P (Yf(X) ≤0). Ceci donne une certaine idée de la confiance dans la classification. Idéalement, |Yf(X)| est proportionnel à P (Y|X).Yf(X) représente la marge de f en (X, Y). Le but à atteindre est la construction de f et donc h. Nous allons voir comment y parvenir [27].

4.2.1 Transformation des entrées

Il est peut être nécessaire de transformer les entrées dans le but de les traiter plus facilement. X est un espace quelconque d'objets. On transforme les entrées en vecteurs dans un espace F (feature space) par une fonction: $\Phi : X \rightarrow F$; $\boldsymbol{F}$ n'est pas nécessairement de dimension finie mais dispose d'un produit scalaire (espace de Hilbert). L'espace de Hilbert est une généralisation de l'espace euclidien qui peut avoir un nombre infini de dimensions. La non linéarité est traitée dans cette transformation, on peut donc choisir une séparation linéaire (on verra plus loin comment on arrive à ramener un problème non linéaire en un problème linéaire classique) [27].

Dès lors, il s'agit de choisir l'hyperplan optimal qui classifie correctement les données (Lorsque c'est possible) et qui se trouve le plus loin possible de tous les points à classer.

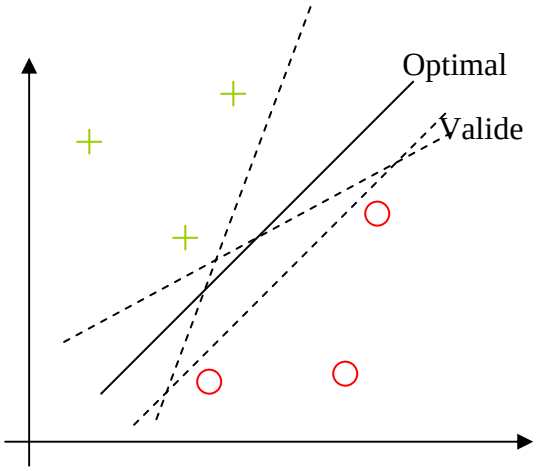


Figure 3.15 : Exemple de recherche d'un hyperplan optimal [27].

Mais l'hyperplan séparateur choisi devra avoir une marge maximale.

4.2.2 Maximisation de la marge

La marge est la distance du point le plus proche à l'hyperplan.

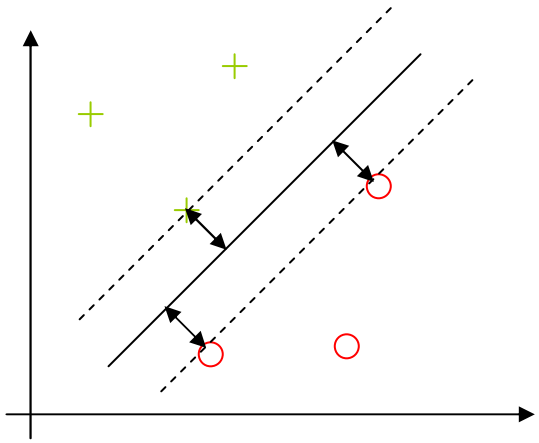


Figure 3.16 : Illustration de la relation entre marge, points de vecteurs de support et hyperplan optimal [39].

Dans un modèle linéaire (Figure ci-dessus), on a f(x) = w.x + b. L'hyperplan séparateur (frontière de décision) a donc pour équation w.x + b = 0.
La distance d'un point au plan est donnée par d(x) = |w.x + b|/||w| L'hyperplan optimal est celui pour lequel la distance aux points les plus proches (marge) est maximale. Soient x1 et x2 eux points de classes différentes (f(x1) = +1 et f(x2) = -1)($w.x1$) + b = +1 et ($w.x2$) + b = -1 donc (w. ($x1$ - $x2$)) = 2 D'où : (w/||w||. ($x1$ - $x2$)) = 2/||w|| [39].

On peut donc en déduire que maximiser la marge revient à minimiser ||w|| sous certaines contraintes que nous verrons dans les paragraphes suivants.

4.2.3 Problème primal

Un point (x, y) est bien classé si et seulement si yf(x) > 0 . Comme le couple (w, b)

est défini à un coefficient multiplicatif près, on s'impose yf(x) ≥ 1. On en déduit (en s'appuyant également sur le paragraphe précédent), le problème de minimisation sous contraintes suivantes :

$$\begin{cases} \min \frac{1}{2}\|w\|^2 \\ \forall i, y_i(w \cdot x_i + b) \geq 1 \end{cases}$$

Il peut être en effet plus aisé de minimiser ||w||² plutôt que directement ||w|| [48,49].

4.2.4 Problème dual

On passe du problème primal au problème dual en introduisant des multiplicateurs de Lagrange pour chaque contrainte.

Ici on a une contrainte par exemple d'apprentissage :

$$\begin{cases} \max \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{ij} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \\ \forall i, 0 \leq \alpha_i \leq c \\ \sum_{i=1}^{n} \alpha_i y_i = 0 \end{cases}$$

C'est un problème de programmation quadratique de dimension n (nombre d'exemples).On défini ainsi la matrice suivante appelée **«** *matrice hessienne* **»** : (xi.xj) i, j qui représente la matrice des produits des entrées X (La notation matricielle permettant de résoudre plus facilement le problème en informatique) [27].

On montre que si les α_i^* sont solutions de ce problème alors on a :

$$w^* = \sum_{i=1}^{n,} \alpha_i^* y_i x_i \text{ [49]}$$

Seuls les α_i correspondant aux points les plus proches sont non nuls. On parle de vecteurs de support.

La fonction de décision associée est donc :

$$f(x) = \sum_{i=1} \alpha^*{}_i y_i x_i \cdot x + b$$

Il existe néanmoins des cas où on ne peut pas classer les entrées de façon linéaire [49].

4.3 La non linéarité (cas non séparable/ marge molle)

On part du problème primal linéaire et on introduit des variables « ressort » pour assouplir les contraintes [49] :

$$\begin{cases} \min \frac{1}{2}\|w\|^2 + c\sum_{i=1}^{n} \varepsilon_i \\ \forall i, y_i(w \cdot x_i + b) \geq 1 - \varepsilon_i \end{cases}$$

On pénalise par le dépassement de la contrainte.
On en déduit le problème dual qui a la même forme que dans le cas séparable [49]:

$$\begin{cases} \max \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{ij} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \\ \forall i, 0 \leq \alpha_i \leq c \\ \sum_{i=1}^{n} \alpha_i y_i = 0 \end{cases}$$

La seule différence est la borne supérieure C sur les α.

4.3.1 Fonction noyau (kernel)

Dans ler cas linéaire, on pouvait transformer les données dans un espace où la classification serait plus aisée. Dans ce cas, l'espace de redescription utilisé le plus souvent est R (ensemble des nombres réels). Il se trouve que pour des cas non linéaires, cet espace ne suffit pas pour classer les entrées. On passe donc dans un espace de grande dimension [39].

$$\phi : \begin{array}{l} R^d \rightarrow F \\ x \rightarrow \phi(x) \end{array}$$

Avec card (***F***) > d.

Exemple:

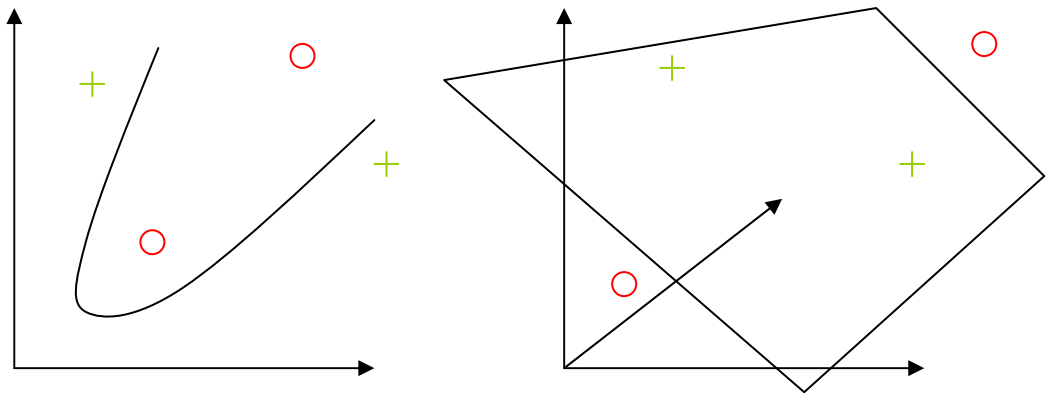


Figure 3.17 : Illustration de passage à R^3 [39]

Le passage dans $F = R^{3.}$ rend possible la séparation linéaire des données. On doit donc résoudre :

$$\begin{cases} \max \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \phi(x_i) \cdot \phi(x_j) \\ \forall i, 0 \le \alpha_i \le c \\ \sum_{i=1}^{n} \alpha_i y_i = 0 \end{cases}$$

Et la solution a la forme:

$$f(x) = \sum_{i=1}^{n} \alpha_i^* y_i \phi(x_i) \cdot \phi(x) + b$$

Le problème et sa solution ne dépendent que du produit scalaire $\phi(x) \cdot \phi(x')$. Plutôt que de choisir la transformation non-linéaire $\phi : X \to F$ **,** on choisit une fonction $K : XxX \to R$ (nombres réels) appelée *fonction noyau*.

Elle représente un produit scalaire dans l'espace de représentation intermédiaire. Du coup k est linéaire (ce qui nous permet de faire le rapprochement avec le cas linéaire des paragraphes précédents). Cette fonction traduit donc la répartition des exemples dans cet espace *k(x,x')=Φ(x).Φ(x')*. Lorsque k est bien choisie, on n'a pas besoin de calculer la représentation des exemples dans cet espace pour calculer *Φ*.
Exemple:

$$\text{Soit } \; x = (x_1, x_2) et \phi(x) = \left(x_1^{\,2}, \sqrt{2} x_1 x_2, x_2^{\,2}\right)$$

Dans l'espace intermédiaire, le produit scalaire donne

$$\begin{aligned} \phi(x) \cdot \phi(x') &= x_1^{\,2} x_1^{',2} + 2 x_1 x_2 x_1' x_2' + x_2^{\,2} x'^2_2 \\ &= \left(x_1 x_1' + x_2 x_2'\right)^2 \\ &= \left(x \cdot x'\right)^2 \end{aligned}$$

On peut donc calculer $\phi(x) \cdot \phi(x')$ sans calculer ϕ : *k(x,x')=(x.x')²*. k représentera donc le noyau pour les entrées correspondantes mais devra néanmoins remplir certaines conditions [42].

4.3.2 Condition de Mercer

Une fonction k symétrique est un noyau si (k (xi,xj)) i,j est une matrice définie positive. Dans ce cas, il existe un espace ***F*** et une fonction ϕ tels que k (x, x') = $\phi(x) \cdot \phi(x')$ [39].

Problèmes :

- Cette condition est très difficile à vérifier
- Elle ne donne pas d'indication pour la construction de noyaux
- Elle ne permet pas de savoir comment est ϕ

Exemples de noyaux:

- Linéaire $k(x, x') = x.\ x'$
- Polynomial $k(x, x') = (x.\ x')^d$ ou $(c+x.\ x')^d$
- Gaussien $k(x, x') = e^{-||x-x'||2/\sigma}$
- Laplacien k (x, x') = $e^{-||x-x'||1/\sigma}$.[48]

On remarque en général que le noyau gaussien donne de meilleurs résultats et groupe les données dans des paquets nets. En pratique, on combine des noyaux simples pour en obtenir de plus complexe [39].

4.4 Temps de calcul et convergence

4.4.1 Complexité

Nous allons évaluer la complexité (temps de calcul) de l'algorithme SVM. Elle ne dépend que du nombre des entrées à classer (d) et du nombre de données d'apprentissage (n).

On montre que cette complexité est polynomiale en n.

$$dn^2 \leq \text{Complexité} \leq dn^3$$

Taille de la matrice hessienne = n^2

En effet, on doit au moins parcourir tous les éléments de la matrice ainsi que toutes les entrées.

Pour un très grand nombre de données d'apprentissage, le temps de calcul explose. C'est Pourquoi les SVMs sont pratiques pour des « petits » problèmes de classification [27].

4.4.2 Pourquoi SVM marche?

Les noyaux précédents qui sont les plus utilisés, remplissent les conditions de mercer (facile à vérifier une fois qu'on a le noyau).

Normalement, la classe (le nombre) des hyperplans de R^d est de dH = d + 1. Mais la classe des hyperplans de marge 1/||w|| tels que $||w||^2 \leq c$ est bornée par : $dH \leq \text{Min}(R^2 c, d) + 1$ Où R est le rayon de la plus petite sphère englobant l'échantillon d'apprentissage S Donc dH peut être beaucoup plus petit que la dimension d de l'espace d'entrée X ; il est donc toujours possible d'en trouver un c'est la raison pour laquelle [27].

5. Les domaines d'applications

SVM est une méthode de classification qui montre de bonnes performances dans la résolution de problèmes variés. Cette méthode a montré son efficacité dans de nombreux domaines d'applications tels que le traitement d'image, la catégorisation de textes ou le diagnostiques médicales et ce même sur des ensembles de données de très grandes dimensions.

La réalisation d'un programme d'apprentissage par SVM se ramène à résoudre un problème d'optimisation impliquant un système de résolution dans un espace de dimension conséquente. L'utilisation de ces programmes revient surtout à sélectionner une bonne famille de fonctions noyau et à régler les paramètres de ces fonctions. Ces choix sont le plus souvent faits par une technique de validation croisée, dans laquelle on estime la performance du système en la mesurant sur des exemples n'ayant pas été utilisés en cours d'apprentissage.

L'idée est de chercher les paramètres permettant d'obtenir la performance maximale. Si la mise en oeuvre d'un algorithme de SVM est en général peu coûteuse en temps, il faut cependant compter que la recherche des meilleurs paramètres peut requérir des phases de test assez longues [27].

Conclusion

Dans ce chapitre, nous avons tenté de présenter de manière simple et complète le concept de système d'apprentissage introduit par Vladimir Vapnik, les Machines à Vecteurs de Support. Nous avons donné une vision générale et une vision purement mathématiques des SVM.

Cette méthode de classification est basée sur la recherche d'un hyperplan qui permet de séparer au mieux des ensembles de données. Nous avons exposé les cas linéairement séparable et les cas non linéairement séparables qui nécessitent l'utilisation de fonction noyau (kernel) pour changer d'espace. Cette méthode est applicable pour des tâches de classification à deux classes, mais il existe des extensions pour la classification multi classe.

Nous nous sommes ensuite intéressé aux différents domaines d'application. Il existe des extensions que nous n'avons pas présentées, parmi lesquelles l'utilisation des SVM pour des tâches de régression, c'est-à-dire de prédiction d'une variable continue en fonction d'autres variables, comme c'est le cas par exemple dans la prédiction de consommation électrique en fonction de la période de l'année, de la température, etc. Le champ d'application des SVM est donc large et représente une méthode de classification intéressante.

Chapitre 4

Reconnaissance de caractères manuscrits arabes par les SVM

- **Introduction**
- **Bilan des méthodes de segmentation**
- **Choix et proposition de méthodes**
- **Mise en œuvre, résultats et bilan**
- **Conclusion**

Introduction

Dans le chapitre précédent nous avons présenté la méthode de classification binaire SVM, inspirées de la théorie statistique de l'apprentissage de Vladimir Vapnik introduite en 1995. Dans ce chapitre nous allons présenter un bilan des méthodes de segmentation de caractères manuscrits existantes, l'approche de segmentation structurelle que nous proposons, ainsi qu'une conception par affinement successif du système en donnant son architecture générale, puis nous détaillons en étudiant séparément chacun de ses composants, ensuite nous allons voir les résultats obtenus, bilan et comparaison avec d'autres méthodes.

1. Bilan des méthodes existantes

Après la phase de prétraitement, la majorité des systèmes OCR isolent les caractères individuels avant de les reconnaître. Ceci est effectué durant la phase de segmentation. Segmenter une page de texte peut être divisé en deux étapes [1]:

- la décomposition de la page, et
- la segmentation des mots.

La décomposition de la page consiste à séparer les différents éléments de la page, produisant ainsi à partir des blocs de texte, des lignes et des pseudo mots (PAWs) [1]. Lorsqu'on travaille avec des pages contenant différents types d'objets tels que les graphiques, formules mathématiques, blocs de texte …etc.

La segmentation des mots consiste à séparer les caractères d'un pseudo mot (PAW). Les performances d'un système d'OCR dépendent généralement de comment sont isolés les caractères [1]. Ensuite, une étape de classification est faite dans le but de reconnaître chaque caractère segmenté en donnant un étiquette de classe suivant la décision d'un classifieur [90].

1.1 Approches de segmentation

Les méthodes de segmentation de l'écriture latine cursive ont été étudiées de façon extensive. Néanmoins, il est difficile d'appliquer les algorithmes et méthodes de segmentation utilisés pour les écritures latines sur l'écriture arabe.

Les méthodes de segmentation des mots arabes peuvent être classées selon cinq approches [1,88]:

1. la première approche, suppose que le mot en entrée est déjà segmenté en caractères.
2. la seconde approche, consiste à segmenter le mot en entrée en primitives plus petites que le caractère.
3. la troisième approche, segmente le mot en entrée en caractères.
4. la quatrième approche, considère est le mot est reconnu de telle sorte que la segmentation soit un sous module du module de reconnaissance.
5. la cinquième approche, traite le mot comme un tout, sans segmentation.

1.1.1 Première approche : caractères isolés

L'approche est basée caractères isolés, comme elle traite principalement les écritures en caractères isolés. Bien que, les caractères isolés sont rarement utilisés dans l'écriture arabe, sauf dans quelques mots et dans les formules mathématiques. Cette approche n'est utilisée que pour des cas particuliers, et de tels systèmes nécessitent un sous système de segmentation qui identifie les caractères à l'intérieur du mot, avant de le reconnaître [1,84].

1.1.2 Deuxième approche : primitives petits que le caractère

Dans l'approche basée primitves un PAW est segmenté à tous les endroits qui paraissent être des points de connexion. Alors, il est possible qu'il soit découpé en primitives plus petites que le caractère tels que les traits, points d'intersection, les points d'inflexion et les boucles. Le schéma habituel de reconnaissance ici est de reconnaître les primitives puis les combiner en caractères [1,82].

C'est approche utilisée, pour la segmentation les deux types décritures en-ligne et hors-ligne amincie. Elle pssède l'avantage, qu'il est plus facile d'identifier un ensemble potentiel de points de connexion qui peuvent inclure tous les points de connexion actuels que d'identifier directement les points de segmentation. Ensuite, C'est à la phase de classification de décider quels sont les points de segmentation, suivant la connaissance à priori qu'elle possède. Particulièrement, Cette méthode est appropriée pour la reconnaissance de l'écriture manuscrite où les bordures des caractères sont ambiguës [1,82].

1.1.3 Troisième approche : caractères

L'approche basée caractères essaye de segmenter correctement un mot en caractères puis de les reconnaître. Donc, l'étape de segmentation est devenue l'étape la plus critique dans le processus de la reconnaissance. Beaucoup de techniques utilisées dans cette approche sont similaires à celles utilisées dans l'approche précédente, mais modifiées pour prévenir de la dissection du caractère en plus d'une partie [1,80].

Il y'a des méthodes qui segmentent un mot aminci en caractères en suivant la ligne de base du mot, en détectant quand les pixels commencent à monter au dessus ou descendre en dessous de cette ligne [1]. Le système IRAC II par exemple proposé dans [80], segmente à la fin d'une dent (**سن**) Mais comme un caractère peut comporter plusieurs dents (exemple 3 dents dans le cas de la lettre (**س**) et comme les caractères comportant une seule dent doivent avoir des points diacritiques au dessus ou en dessous. Le système examine les points se trouvant autour de la dent pour décider de segmenter ou non [80,81].

D'autres méthodes, cherchent les points de segmentation le long de la ligne de base à l'aide des histogrammes de projection verticale. Ces points sont définis comme étant les endroits où l'histogramme descend en dessous d'un certain seuil.

les chercheurs utilisent différentes méthodes, parce que le même caractère peut avoir à l'intérieur plusieurs descentes, pour prévenir de rupture d'un caractère en plus d'une partie. Par exemple certains chercheurs utilisent des règles heuristiques qui préviennent la segmentation de caractères de largeur inférieure à une certaine valeur.

D'autres modifient l'histogramme de projection verticale en multipliant chaque entrée par la hauteur de la colonne relative à la ligne de base. Ceci a pour effet d'amplifier la distance de la ligne de base de manière à ne pas considérer les points loin de la ligne de base comme points de connexion [80,86].

1.1.4 Quatrième approche : segmentation sous module du module de reconnaissance

Cette approche reconnaît les caractères d'un mot connectés sur place (sans segmentation préalable). Quelques systèmes qui adoptent cette méthode commencent à l'extrême droite d'un pseudo mot et examinent un ensemble de colonnes (de largeur égale au caractère le plus proche) et essayent de le reconnaître. Si la reconnaissance échoue. Ils ajoutent itérativement d'autres colonnes, jusqu'à la reconnaissance du caractère. Une fois un caractère reconnu, il est enlevé du sous mot et le processus est répété [1,88].

Ce type d'approches, pose un problème quand le système échoue à la reconnaissance d'un caractère dans n'importe quelle partie du mot et spécialement au début le reste du mot ne sera pas traité. La solution est d'utiliser une reconnaissance arrière de gauche à droite, déclenchée lorsque le système n'arrive pas à reconnaître un caractère au milieu [1,88].

1.1.5 Cinquième approche : sans segmentation

Le mot est reconnu comme une entité par les systèmes de cette approche. Généralement, des techniques de comparaisons globales sont utiliées pour comparer les mots en entrée à d'autres stockés dans une base de données. Cependant, cette approche est limitée à la reconnaissance d'un ensemble de mots prédéfini (exemple : commandes d'ordinateurs dans les ordinateurs basés stylo), et n'est pas pratique pour la reconnaissance générale de texte à riche vocabulaire [1,88].

Après avoir présenté les approches existantes, nous remarquons que le domaine de segmentation en caractères des mots est très vaste, et il existe des classes de méthodes pour pouvoir balayer une multitude de facettes de la calligraphie de l'écriture.

En examinant la bibliographie concernant la segmentation en caractères de l'écriture arabe, nous avons constaté cinq classes de méthodes. Une des classe utilise comme base de sa segmentation la squelette des mots du texte, une seconde utilise comme base le contour des mots pour les segmenter en caractères, une troisième les histogrammes de projection verticale et horizontale, une quatrième classe utilise les réservoirs, et une cinquième utilisant les fenêtres glissantes.

Approche	Principe	Inconvénients
Première approche	- utilisée dans le cas des caractères isolés.	- ne peut être utilisée pour l'écriture arabe, - nécessite un sous système de segmentation qui identifie les caractères à l'intérieur du mot.
deuxième approche	- découpage du PAW en des primitives plus petites que le caractère (tel que les traits, points d'intersection, les points d'inflexion et les boucles) - fait la reconnaissance des primitives (pour les combiner ensuite en caractères).	- il est du rôle de la phase de classification de décider quels sont les points de segmentation, suivant la connaissance à priori qu'elle possède.
Troisième approche	- le mot est segmenté correctement en caractères, - les caractères sont reconnus.	- il faut modifier des méthodes de segmentation de la deuxième approche pour prévenir de la dissection du caractère en plus d'une partie.
Quatrième approche	- le mot est reconnu sur place (sans segmentation préalable), - en commençant à l'extrême droite d'un pseudo mot et en examinant un ensemble de colonnes (de largeur égale au caractère le plus proche), - si la reconnaissance échoue, il ajoute itérativement d'autres colonnes jusqu'à la reconnaissance d'un caractère.	- si le système échoue à la reconnaissance d'un caractère dans n'importe quelle partie du mot et spécialement au début le reste du mot ne sera pas traité.
Cinquième approche	- le mot est reconnu comme une entité en utilisant des techniques de comparaisons globales (pour comparer les mots en entrée à d'autres stockés dans une base de données).	- limitée à la reconnaissance d'un ensemble de mot prédéfini, - n'est pas pratique pour la reconnaissance générale de texte à riche vocabulaire.

Tableau 4.1 : Tableau récapitulatif des avantages et inconvénients des approches de segmentation.

1.2 Classification

Dans la littérature, on trouve une variété de classifieurs tels que: KPV, réseaux de neurones, et MMC…etc. quelque soit le classifieur choisit, le système de reconnaissance garde qu'il faut faire deux phase importantes:

- Apprentissage, et
- Décision.

Suivant le type de classifieur on note une différence en terme de vecteur caractéristique, temps d'apprentissage, temps d'exécution ou de reconnaissance et par conséquent, une variation dans la certitude de classification [38,55].

1.2.1 Apprentissage

En intelligence artificielle, l'apprentissage est représenté par deux courants- numérique et symbolique - qui exploitent respectivement et majoritairement des formalismes statistiques et logiques. C'est principalement l'aspect numérique que nous considérerons ici [44]. Les exemples particuliers sont représentés par un ensemble de couples d'entrée/sortie. Le but est d'apprendre une fonction qui correspond aux exemples vus et qui prédit les sorties pour les entrées qui n'ont pas encore été vues. Ce qui nécessite de bien choisir:

- De bons exemples,
- La fonction noyau et les paramètres adéquats,…etc.

L'apprentissage est une phase cruciale car les résultats de décision se basent sur les paramètres fixés durant cette phase [27].

1.2.2 Décision

Effectuer une classification consiste à déterminer une règle de décision capable, à partir d'observations externes, d'assigner un objet à une classe parmi plusieurs. Le cas le plus simple consiste à discriminer deux classes [34].

Le tableau suivant illustre les taux de reconnaissance pour quelques classifieurs et avec des différents ensembles d'apprentissage.

	Nombre d'exemples d'apprentissage				
	15	100	1000	10000	60000
Naïve bayes- multinomial	48.90%	67.90%	80.67%	83.41%	83.64%
Naïve bayes- multivariate	28.21%	66.36%	80.52%	83.55%	84.13%
Kpv - Euclidien - K=1	50.48%	67.94%	86.90%	94.64%	96.91%
Kpv - Euclidien - K=3	32.04%	64.76%	86.22%	96.63%	97.05%
Kpv - angulaire - K=1	51.56%	71.57%	88.54%	95.41%	97.23%
Kpv - angulaire - K=3	33.42%	67.79%	87.84%	95.34%	97.33%
Régression logistique	32.10%	52.11%	68.91%	34.95%	32.46%
SVM	41.53%	64.68%	84.88%	90.28%	91.83%

Tableau 4.2 : Taux de reconnaissance pour des différents classifieurs et avec des différents ensembles d'apprentissage [38].

2. Choix et proposition de méthodes

Dans cette section, nous allons voir la méthode de segmentation proposée, ainsi que le classifieur choisit pour qu'il soit utilisé dans notre système.

2.1 Approche de segmentation proposée

Nous avons vu dans le paragraphe I.1 que pour la segmentation des mots arabes il existait cinq approches différentes. La première approche assumait que les mots étaient déjà segmentés à l'entrée. La seconde segmentait les mots en primitives plus petites que le caractère et qu'elle était plus appropriée au cas du manuscrit. La troisième segmentait le mot en caractères pour les reconnaître, c'est l'approche adoptée pour les cas de caractères imprimés. La quatrième approche reconnaissait les mots sans segmentation préalable, en utilisant des primitives morphologiques et des modèles pour la comparaison. Le problème avec cette approche est que la définition des primitives dépendait de la taille des caractères et de leur fonte ce qui limitaient le nombre de fontes et les tailles de caractères. Dans la cinquième approche des mots entiers étaient reconnus sans segmentation. Ce qui posait le problème de limite du vocabulaire.

Dans ce travail, nous présentons une méthode de segmentation structurelle simple qui appartient à la deuxième classe d'approches. Le principe est simple, en faisant un balayage de l'image du PAW source (voir figure 4.3) suivant la direction de lecture (de droite à gauche pour l'arabe) plus un autre balayage de bas en haut et en respectant les règles suivantes:

1. le premier segment ou le dernier segment détecté est marqué comme segment porteur de caractère,
2. une opération de division ou de fusion avec un segment marqué comme porteur de caractère simple (niveau 1, voir figure 4.1) ,élimine la marque,
3. une opération de division ou de fusion avec un segment marqué comme porteur de caractère complexe (niveau 2), lance la division en deux du segment indiqué et fait évalué l'ensemble de segments en traitement comme des segments de niveau 3, alors un caractère est obtenus.

La séquence de figure 4.2-4.5 illustre bien ces étapes.

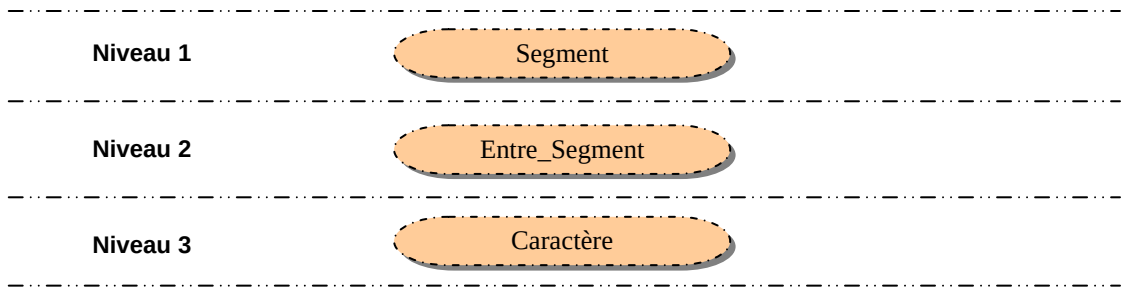


Figure 4.1 : Différents niveaux d'extraction de caractéristiques.

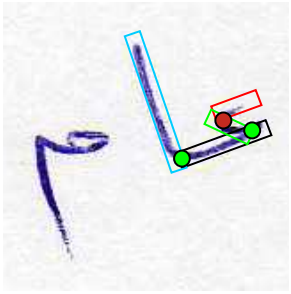
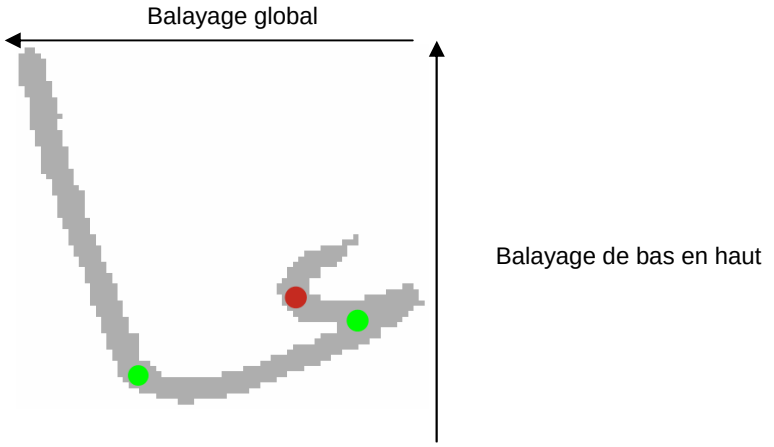


Figure 4.2: Etat initial du mot "عام".

Pour chaque étape du cycle de balayage globale, tout un cycle de balayage de bas en haut est parcouru dans le but d'extraire la définition des points de division ou de fusion, comme la montre la figure suivante.



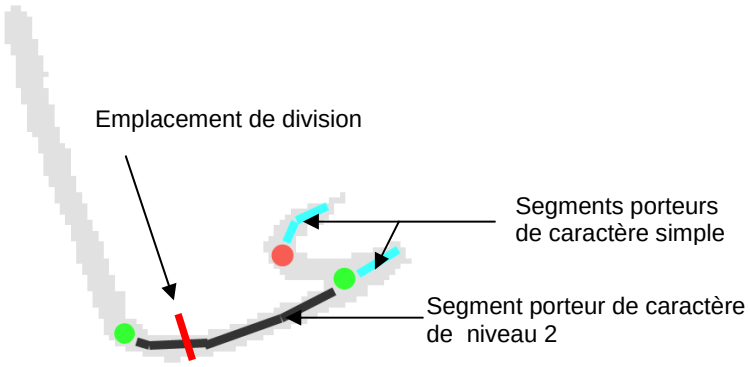


Figure 4.4: Le PAW "عا" après marquage des segments porteur de caractères.

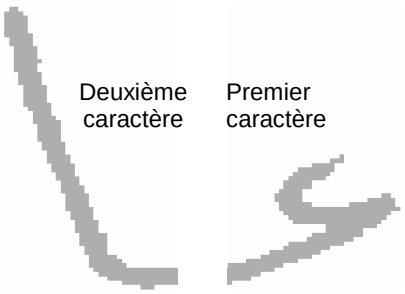


Figure 4.5: Le résultat de segmentation.

L'extraction de caractéristiques des caractères se fait en calculant des valeurs donnant des définitions de description (de segment, entre segments, caractère) qui respectent les trois axes présentés par la figure 4.6. Alors, les définitions obtenus commence de niveau 1, en donnant des caractéristiques de chaque segment séparément, ensuite les caractéristiques entre segments de niveau 2 et finalement une description du caractère de niveau 3 (voir figure 4.1).

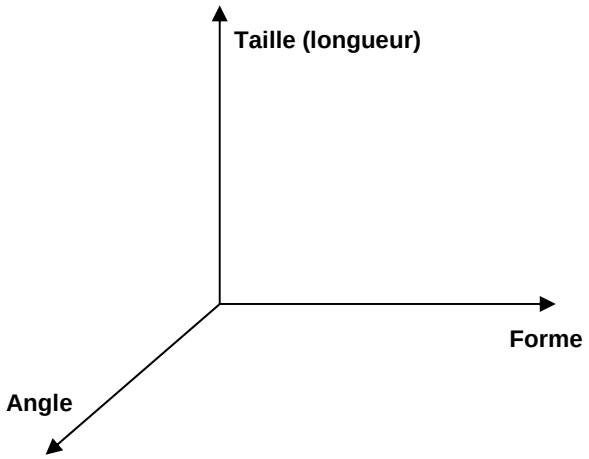


Figure 4.6 : Axes de base pour l'extraction des caractéristiques.

Le présent travail, ne suit pas un algorithme de squelettisation, mais, une technique de suivi très simple, alors:

- pas de perte d'information de représentation;
- gain du temps d'exécution;
- grande possibilité de faire une reconnaissance efficace de scripteurs, et styles d'écriture.

La figure suivante illustre cette technique.

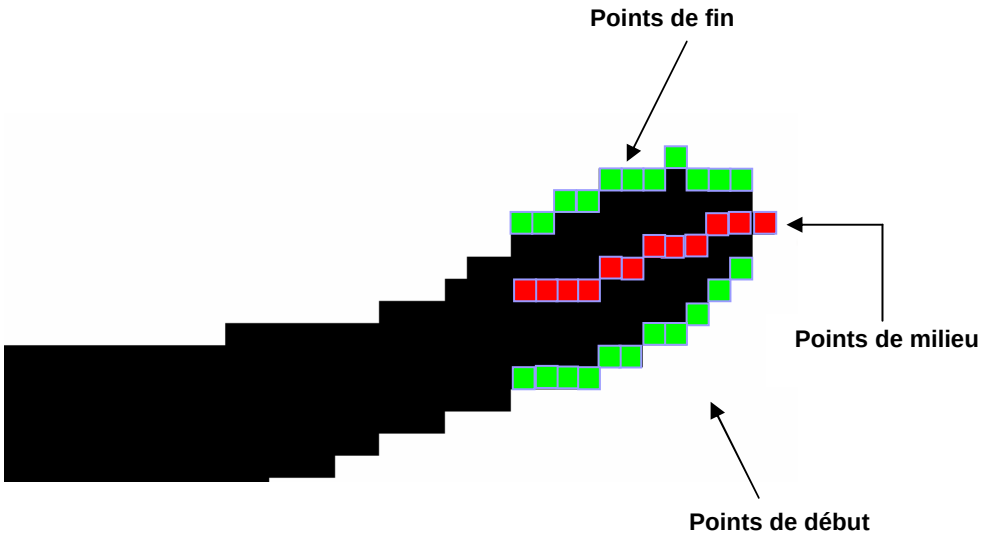


Figure 4.7 : Illustration du technique de semi squelettisation utilisée.

Les points de milieu extraits sont utilisés pour calculer au cours du balayage la longueur et l'angle de déviation du segment en cours de traitement. Les points d'extrémités sont utilisés pour extraire des informations sur la forme du segment.

2.2 Méthode de classification choisie

Dans ce travail, nous avons choisis d'utiliser un classifieur SVM dans le but d'avoir un compromis de choix entre: un vecteur caractéristique de taille important, temps d'apprentissage plus au moins réduit, et un temps d'exécution ou de reconnaissance raisonnable, également, une précision de classification est ciblée.

3. Mise en œuvre, résultats et bilan

Dans cette section, nous allons présenté une conception du système en donnant une vue globale du système, ensuite nous allons détaillé chaque module composant le système séparément. Après cela, nous allons voir : les différentes interfaces et fenêtres principales du système, les tests et résultats obtenus.

3.1 Mise en œuvre du système

Par analogie aux différentes étapes de l'OCR, le système peut être vu comme étant 06 modules complétant chacun une tâche du processus de reconnaissance.

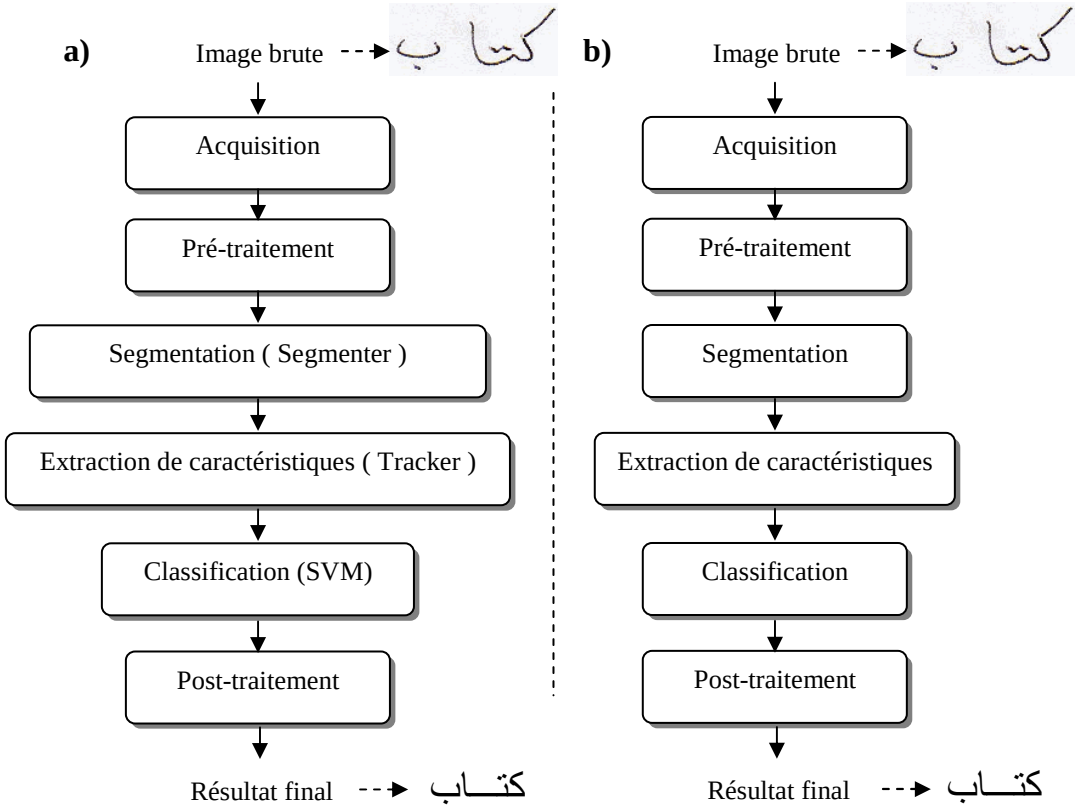


Figure 4.8 : Illustration du processus de reconnaissance :
a)les modules du système,
b) les étapes de l'OCR.

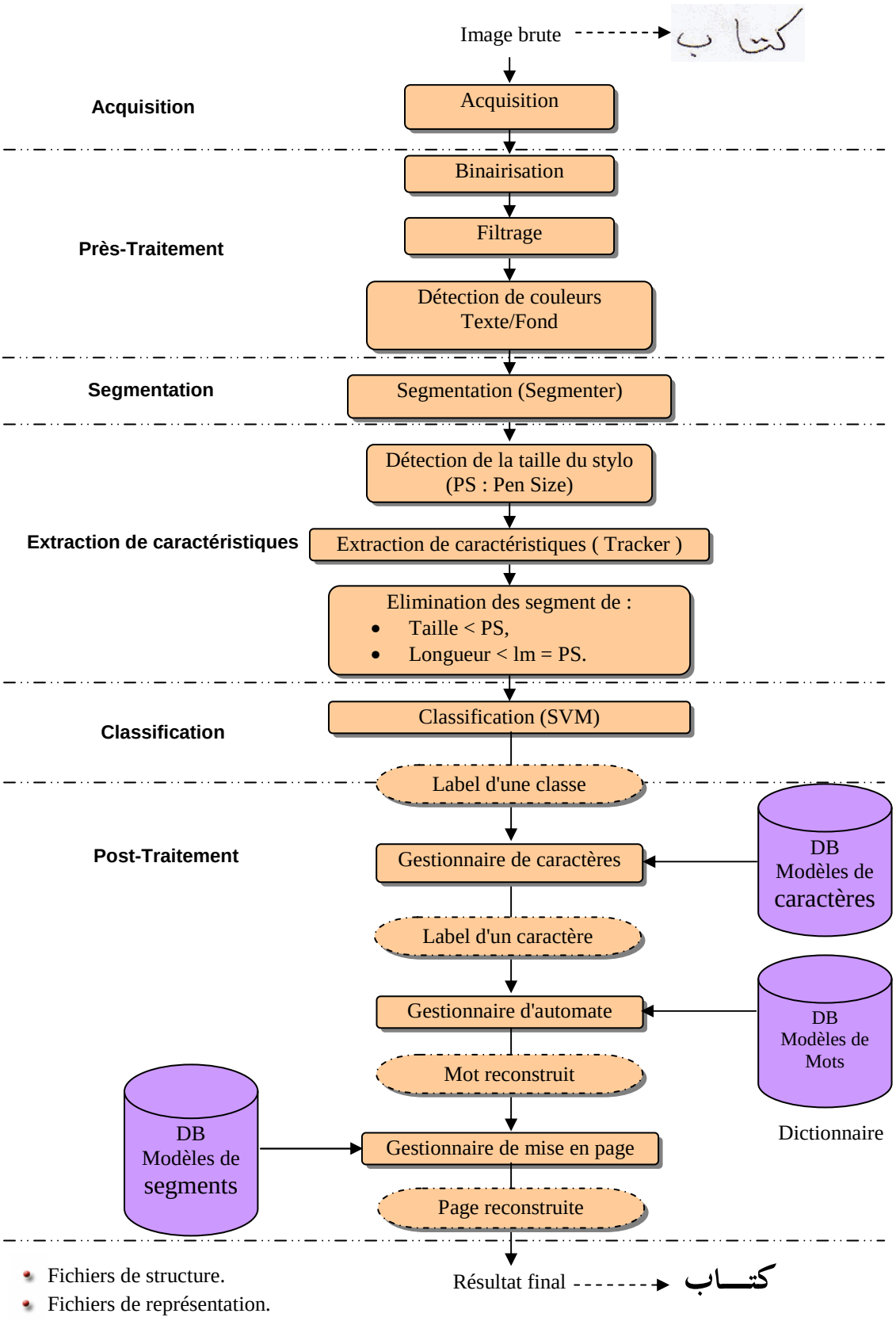


Figure 4.9 : schéma détaillé du processus de reconnaissance utilisé par le système Aya.

3.1.1 Acquisition

Ce module consiste tout simplement à acquérir ou charger l'image du papier et la sauvegarder en un format d'image connu (par exemple : jpeg, gif, bmp…etc.) et /ou la transformer en un matrice de pixels dans le but de l'utiliser directement. Pour cela on utilise généralement des interfaces d'acquisition telle que:

- Twain (JTWain pour java),
- Morena Framework,
- …etc.

Dans ce qui suit nous allons présenté l'interface d'acquisition TWain.

A. Création du TWain

Twain est créé par le petit groupe de compagnies software et hardware comme réponse au besoin de spécification proposée par l'industrie d'imagerie. Le but du groupe été de fournir une solution multi-paltformes et ouvertes pour interconnecter les différents outils d'acquisition avec les applications. Le groupe original est représenté par les 5 compagnies : *Aldus*, *Caere*, *Eastman Kodak*, *Hewlett-Packard*, et *Logitech*. Trois autres compagnies, *Adobe*, *Howtek,* et *Software Architects* sont aussi contribuées significativement.

Le design de TWain est commencé en janvier, 1991. Une revue est survenue du Toolkit de développeurs originaux de TWain d'Avril, 1991 au Janvier, 1992.le Toolkit original est revu par la coalition de TWain. La coalition inclut approximativement 300 individus représentant 200 compagnies pour influencer et guider le développement de TWain.

La version courante de TWain est écrite par les 11 membres courants du Groupe du Travail TWain. Les membres inclus: *Adobe*, *Canon*, *Eastman Kodak Company*, *Fujitsu Computer Products of America*, *Genoa Technology, Inc.*, *Hewlett-Packard Company*, *Intel Corporation*, *J.F.L. Peripherals*, *Kofax Image Products*, *Ricoh Corporation*, et *Xerox*.

En Mai 1998, un agrément est annoncée entre Microsoft et le Groupe de Travail TWain ce qui fournit l'inclusion de TWain Data Source Manager en Microsoft Windows 98 et Microsoft Windows NT 5.0 [61,62].

B. Eléments du TWain

TWain définit un protocole de software standard et un API (Application Programming Interface) pour la communication entre les applications software et les périphériques d'acquisition d'image (la Source de données) [61,62].

Les trois éléments clés dans TWain sont :

- **Application software** – une application doit être modifiée pour utiliser TWain.
- **Source Manager software** – ce software maintien la gestion d'interactions entre l'application et la source.
- **Source software** – ce software contrôle le périphérique d'acquisition d'image et il est écrit par le développeur du périphérique en respectant les spécifications du TWain.

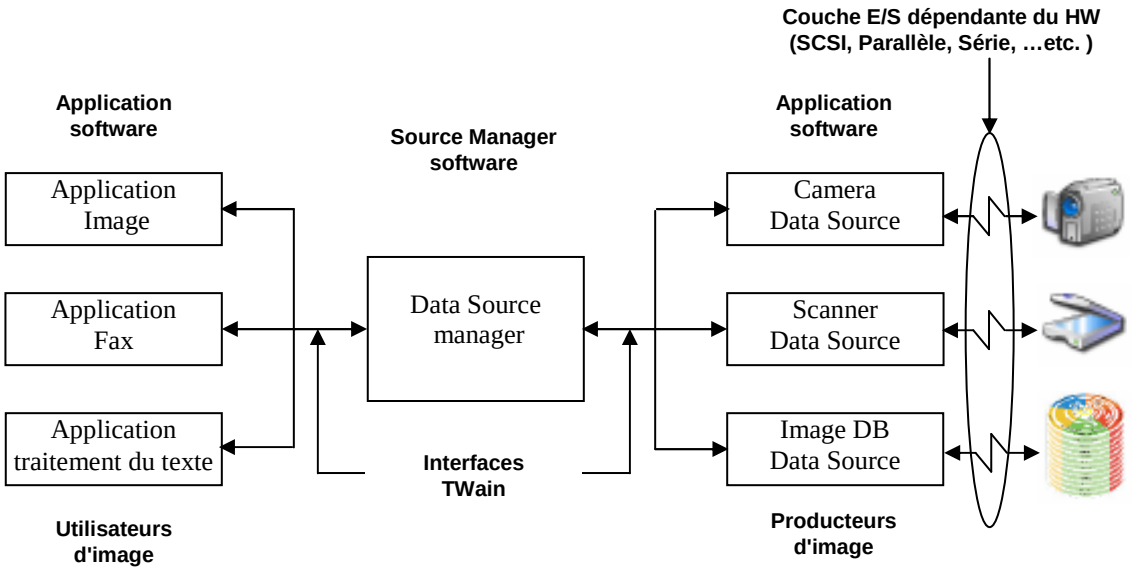


Figure 4.10 : Eléments du TWain.

C. Architecture du TWain

Le transfert de données est possible par trois éléments du logociel qui travail ensemble en TWain : l'*Application*, le *Manager de la Source* (Source Manager) et la *Source*.

Ces éléments utilisent l'architecture de TWain pour communiquer. L'architecture de TWain est constitue de 04 couches:

- Application,
- Protocole,
- Acquisition,
- périphérique.

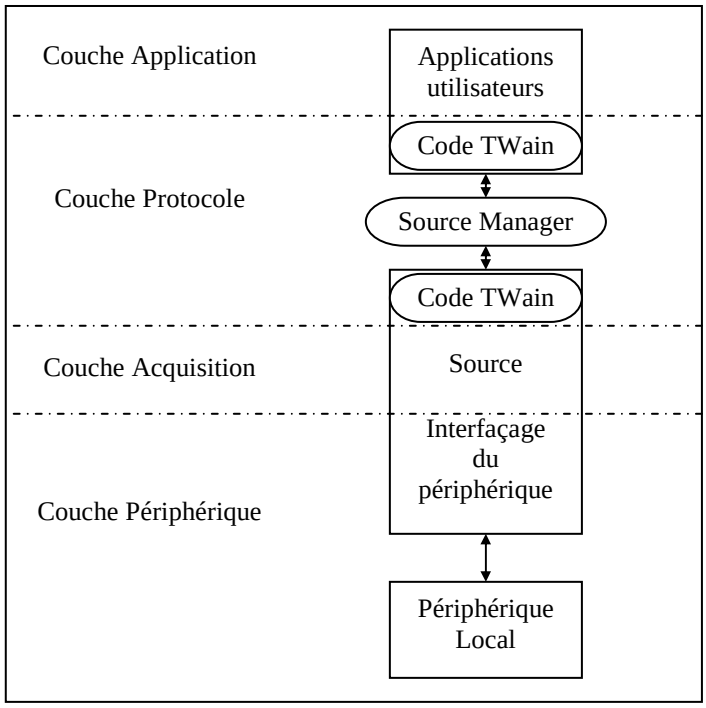


Figure 4.11 : Architecture du TWain.

C.1 Couche Application

Les applications logicielles exécutent dans cette couche. Twain décrit l'interface utilisateur au développeur d'application mais pas la manière d'accès aux fonctionnalités du Twain et comment une source est sélectionnée.

Twain n'est pas concerné par l'implémentation d'application. Twain n'a pas d'effet sur aucun schéma de communication inter-application que l'application utilise.

C.2 Couche Protocole

Le protocole est le "langage" parlé et la syntaxe utilisée par Twain. Il implémente les instructions et les communications requises pour le transfert de données. La couche Protocole inclut :

- la portion d'application software qui fournit l'interface entre l'application et Twain,
- Twain Source Manager fournit par TWain,
- Le software inclut avec le Source Device pour recevoir des instructions du Source Manager et renvoyer des données et retourner du code.

C.3 Couche Acquisition

Les périphériques d'acquisition peut être physique (comme un scanner ou une caméra numérique) ou logique (comme une image de base de données). Les éléments du software écrits pour contrôler les acquisitions sont appelés *Sources* et réside principalement dans cette couche.

La Source transfert les données pour l'application. Elle utilise le format et le mécanisme de transferts acceptés par la Source et l'application.

La source fournit toujours une interface utilisateur de Built-in qui contrôle le(s) périphérique(s). Elle est écrite pour guider (Drive). Une application peut annuler cette interface et présenter son interface utilisateur pour l'acquisition.

D.4 Couche Périphérique

C'est la location traditionnelle des drivers de périphériques de bas niveau. Ils converties les commandes des périphériques spécifiques en commandes hardware et actions spécifiques au périphérique spécifiques. Les applications qui utilisent TWain ne sont au besoin de drivers de cartes car elles sont une partie de la Source.

TWain n'est pas concerné du tout par la couche périphérique car la Source cache la couche périphérique à l'application.

3.1.2 Pré-traitement

L’image obtenue dans la phase d’acquisition n’est qu’une image brute (bruitée). Durant la phase de prétraitement nous allons appliqués des filtres afin de rendre cette image prête pour des traitements futurs.
La figure 4.4 illustre cette phase.

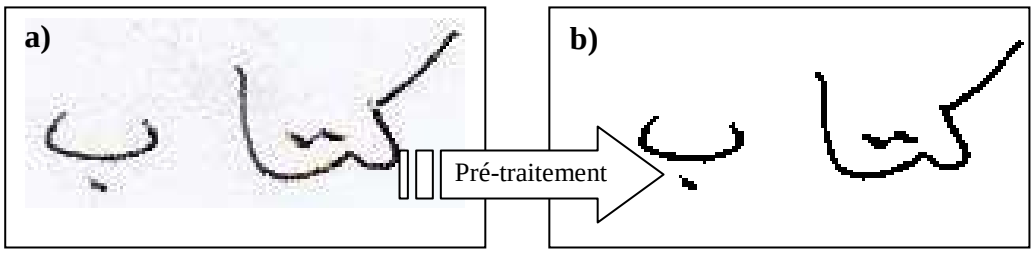


Figure 4.12 : Illustration d'un exemple d'image prétraitée:
a) image brute,
b) image épurée.

Pour réaliser cette phase on passe pratiquement par trois étapes :

- Binairisation,
- Filtrage,

A. Binairisation

L'étape de binairisation consiste à transformer la matrice de l'image brute obtenue par un procédé d'acquisition et représentée en couleurs (espace de couleur RGB) en une image représentée en deux couleurs, noir et blanc.

Pour le faire on utilise les formules de conversion RBG/YUV, à savoir :

$Y = 299 * R + 0.587 * G + 0.114 * B$
$U = 0.492 (B – Y)$
$V = 0.877 (R – Y)$

Comme il est possible de faire le passage inverse, YUV/RGB, à savoir :

$R = Y + 1.140 V$
$G = Y– 0.395 U – 0.581 V$
$B = Y + 2.032 U$

Dans la réalité, en pratique on utilise seulement la première équation, ensuite en définissant un seuil S on peut décider si une pixel est noir ou blanc.

La figure 4.5 illustre cette étape.

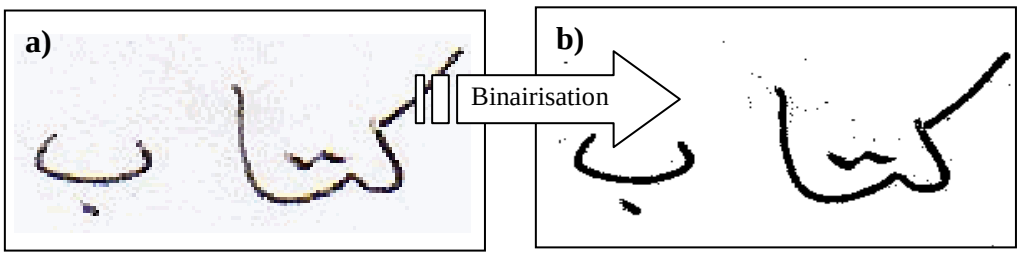


Figure 4.13 : Application de binairisation:
a) image brute,
b) image après binairisation .

B. Filtrage

L'image représentée maintenant en niveau de grille contient des taches (image bruitée), alors le filtrage consiste à suivre certains processus afin d'éliminer ces taches et de représenter en mieux les points d'intérêt.

Pour réaliser l'étape de filtrage, en éliminant tous les pixels qui respectent l'un des conditions suivants :

1) un pixel noir entouré par des pixels blanc,

```
Pour (tous les pixels de l'image) Faire
        Si (pixel noir entouré par des pixels blancs)
Alors
             Eliminer le pixel
        FinSi
FinPour
```

2) un pixel blanc entre deux pixels noir que ce soit horizontalement ou Verticalement.

L'étape de filtrage est illustrée par la figure suivante:

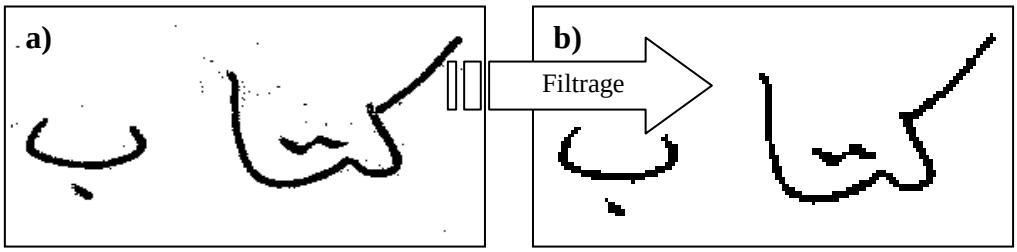


Figure 4.14 : Illustration d'un exemple de filtrage:
a) image après binairisation,
b) image filtrée.

3.1.3 Segmentation [Segmenter]

Dans la phase de segmentation on utilise la nuance en couleur - noir/blanc - de l'image épurée pour obtenir les différents segments composants le mot ou le caractère. La figure 4.7 illustre cette tâche.

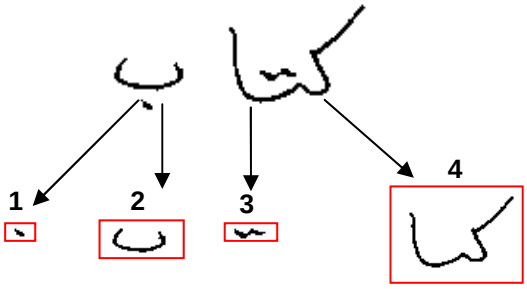


Figure 4.15 : Exemple de segmentation de l'image du mot "كتاب"

Cette tâche est réalisée à l'aide de la procédure récursive suivante:

```
Procédure ConstructSegment(Point p)

     FULL_MARKER=false: booléen;
     m: matrice d'entier [3,3];
     x,y,h,k: entier;

Début
     // Initialiser l'indicateur du marquage

     FULL_MARKER=faux;
     Pour h de 0 à 3 pas 1 Faire
            Pour k de 0 à 3 pas 1 Faire

               x=(p.x-1)+h;
               y=(p.y-1)+k;

               Si((x>=0)et(x<=img.getWidth()))Alors
          Si ((y>=0)et(y<=img.getHeight()))Alors
          Si (IN_BF((img.getRGB(x,y))) Alors

                         Si ((x<>p.x)ou(y<>p.y)) Alors
                    m[h,k]=1;
               Sinon
                    m[h,k]=0;
               FinSi
                          FULL_MARKER=vrai;
               Si(x<Xmin)Alors Xmin=x; FinSi
               Si (y<Ymin) Alors Ymin=y; FinSi
               Si (x>Xmax) Alors Xmax=x; FinSi
               Si (y>Ymax) Alors Ymax=y; FinSi

// changer le pixel correspondante de l'image temporaire
// en noir=0

                         tmpimg.setRGB(x,y,0);

// changer le pixel correspondante de l'image source
// en blanc = 16777215

                    img.setRGB(x,y,16777215);

       Sinon
            m[h,k]=0;
       FinSi
       FinSi
       FinSi

       FinPour
```

```
        Si (FULL_MARKER)
        Pour h de 0 à 3 pas 1 faire
        Pour k de 0 à 3 pas 1 faire

        Si(m[h,k]=1)Alors

                x=(p.x-1)+h;
                y=(p.y-1)+k;

                     ConstructSegment(nouveau Point(x,y));

        FinSi
        FinPour
        FinPour
        FinSi
Fin
```

Où:
`Img`: image source,
`Tmpimg`: image résultante de la segmentation,
`getWidth`: fonction permet d'obtenir la largeur d'une image (ex: img.),
`getHeight`: fonction permet d'obtenir la hauteur d'une image (ex: img.),
`getRGB(x,y)` : fonction permet d'obtenir les composantes du couleur du pixel en position (x,y),
`setRGB(x,y,v)` : fonction permet de changer les composantes du couleur du pixel en position (x,y) à la valeur v (v appartient à [0, 16777215]),
`IN_BF`: fonction permet de connaître si la couleur d'une pixel est dans la plage du couleur noir ou non.
Le résultat de cette étape est un ensemble d'images nommée « Image_NumSéq.gif » et des fichiers nommés « Segment_NumSeq.pos » contenant la position de chaque image produit dans l'image initiale, tout ça , à l'aide d'une classe Java appelée Segmenter (Segmenter(nom_Image)).

A. Segmentation de la page

Dans la réalité, ce type de segmentation n'existe pas dans notre travail parce qu'on ne traite que des documents simples qui contient seulement des segments de texte.

B. Segmentation de texte en lignes

En utilisant les fichiers Segment_NumSeq.pos, nous pouvons utilisés les coordonnées de chaque segments d'image, de plus, en faisons un balayage de droite à gauche nous allons trouvés que pour les segments de la même ligne: chaque segment possède un chevauchement maximal avec le segment qui est à son droite, ce qui permet de reconstruire une représentation logique des lignes.

La figure suivante illustre cette idée.

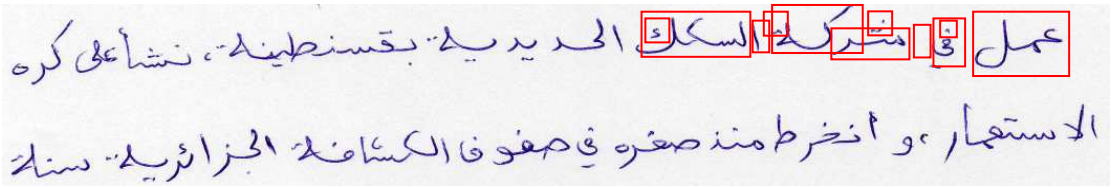


Figure 4.16 : Illustration d'un exemple de chevauchement entre segments d'image.

C. Segmentation de lignes en mots

La segmentation de lignes en mots n'est pas possible à ce niveau car nous n'avons aucune information sur l'organisation réelle des mots. Dans l'étape précédente, nous pouvons voir que les segments d'images générées présentent une représentation de lignes en séquences de PAWs.

Cette technique ne permet pas de segmenter complètement le mot, comme c'est le cas pour le cadre quatre (voir figure 4.15). Pour palier à ce problème l'utilisation d'un tracker (collecteur d'information) est nécéssaire.

3.1.4 Extraction de caractéristiques [Tracker]

La phase d'extraction de caractéristiques consiste à utiliser une méthode de suivi (tracking) dans le but d'extraire :

- les caractères (segmenter le reste du mot),
- les caractéristiques de ces caractères.

En définissant deux types de segments, comme suit:

1. **Segment de type 1** : c'est un segment qui commence par un vecteur de pixels ne possède pas de liaisons avec d'autres segments.
2. **Segment de type 2** : c'est un segment qui commence par un vecteur de pixels lié à un autre segment.

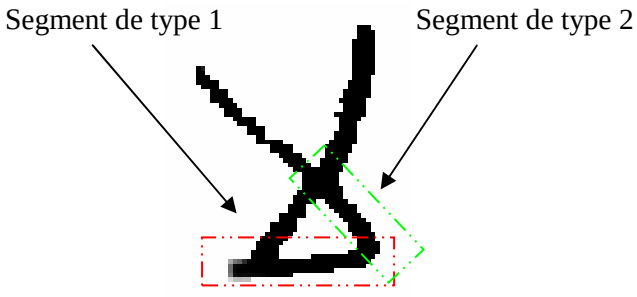


Figure 4.17: Illustration de types de segments

En prenant aussi, que chaque segment est étiqueté comme suit:

S (Seg_Type, Marker, Num_Seq, Num_Seg_Prec)

Tel que:

Seg_Type : le type du segment,
Marker : un marqueur qui va contenir la description et les caractéristiques du segment,
Num_Seq : le numéro de séquence du segment,
Seg_Prec: le numéro du segment qui précède ce segment.

Avec les deux opérations suivantes :

1. *Division*, notée: D(s) → s1,s2;
2. *Fusion*, notée: F (s1, s2) → s.

On aura par conséquent les trois types de points suivants :

1. Point de Division,
2. Point de Fusion,
3. Point de Division et de Fusion.

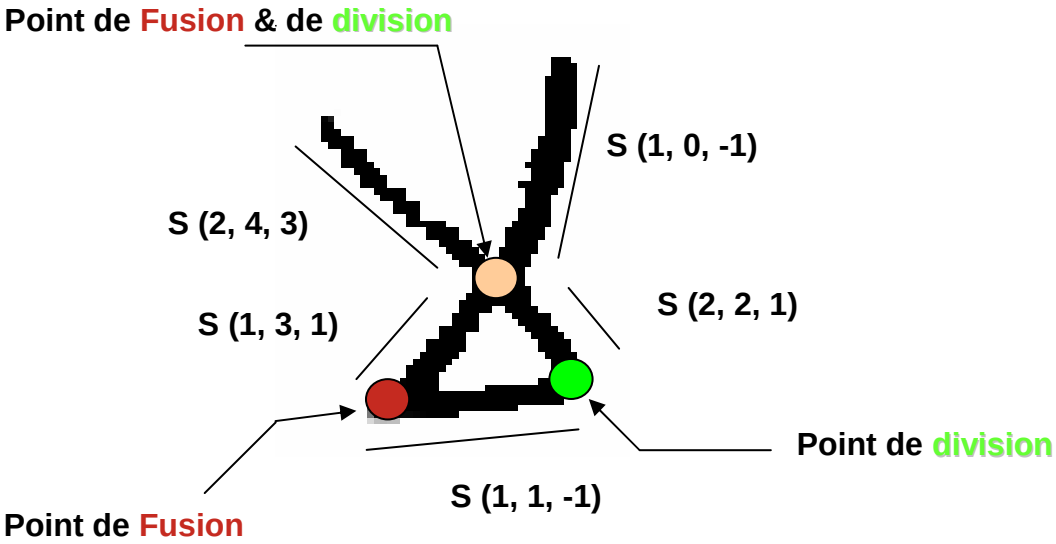


Figure 4.18: Illustration de points de Division et de Fusion.

Ces segments sont sauvegardés dans une liste où chaque élément possède la structure suivante:

```
TElement
{
   Segment_Type : Entier;  // 02 types de segments (1, 2)
   marker : Marker;   //   Caractéristiques du segment
   Num_Seq : Entier;
   Num_Seg_Prec : Entier;   // Numéro du segment père
}
```

Alors, le résultat va être une liste, comme suit:

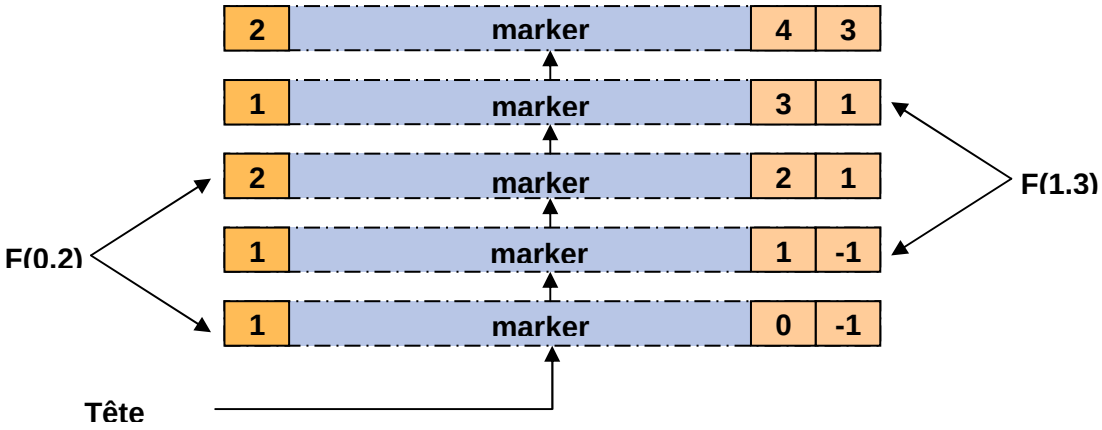


Figure 4.19: Illustration d'une liste de segments.

Chaque marqueur (marker) va contenir la description des caractéristiques suivantes:

- La direction,
- La taille relative des segments entre eux,
- Le nombre de boucle,
- La taille du segment d'image,
- La forme des boucles,
- L'angle des segments,
- Les secteurs des angles,
- Le type de différents points constituants la boucle.

A. Direction

La direction est définit à l'aide de 03 descripteurs:

1. direction sur l'axe des X,
2. direction sur l'axe des Y,
3. direction générale, résulte de la composition des deux précédentes.

Les figures suivantes montre bien cette idée:

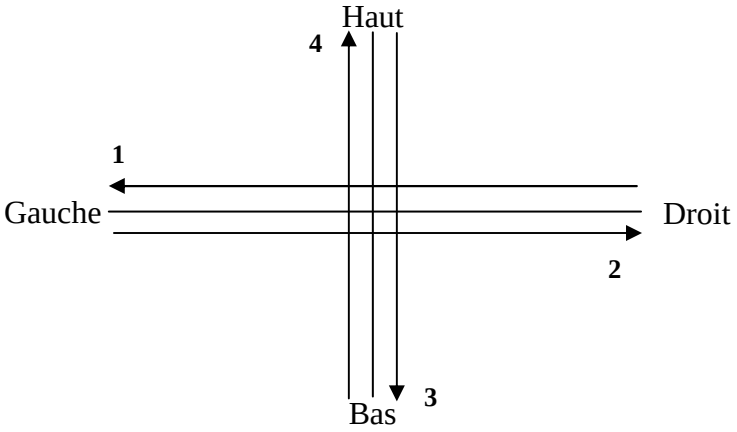


Figure 4.20: Illustration des 04 directions utilisées

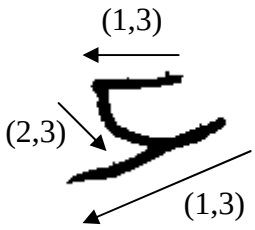


Figure 4.21: Illustration d'un exemple d'extraction de direction.

Exemple:

Direction en X = 121
Direction en Y = 333
Direction générale = 131

B. Taille relatif des segments

Avant de discuter comment calculer la taille relative il faut calculer premièrement la taille de chaque segment séparément, et cela est réalisée à l'aide de la fonction suivante:

```
Fonction getDistance(Point p1,Point p2):Entier

    Distance, dx, dy:Entier;

Début

     dx=p1.x-p2.x;
     dy=p1.y-p2.y;

     Distance=round(sqrt(dx*dx)+(dy*dy)));

     getDistance = distance;
Fin
```

Maintenant après avoir la taille de chaque segment il est facile de calculer la taille relative entre segment par le calcul du rapport des tailles des segments.

C. Nombre de boucles

Le nombre de boucles est simple à calculer, tout simplement on incrémente un compteur de boucles à chaque fois qu'on fait une opération de fusion avec une égalité de point de départ et de fin entre les deux segments à fusionner.

D. Forme de la boucle

La forme de la boucle est définit en utilisant 04 points de contrôle, le calcule de distance et angle entre ces points permet d'obtenir une description sur la forme de la boucle.

La figure suivante illustre ce principe.

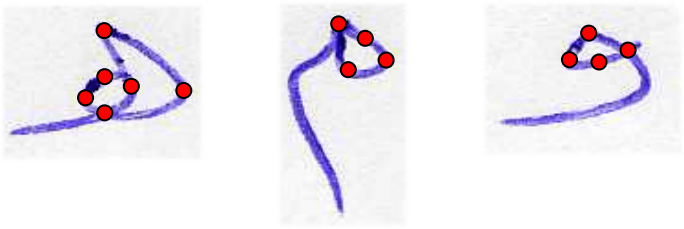


Figure 4.22: Illustration d'un exemple de points de contrôle.

E. Angle de déviation

L'angle de déviation de chaque segment est calculé à chaque fois qu'on fait une opération de fusion ou nous voulions extraire les caractéristiques de la forme d'une boucle. Pour la calculer on fait appel à la fonction suivante:

```
Fonction getAngle(Point a, Point b):double

     angle,dx,dy:double;

Début
     dx = b.x - a.x;
     dy = a.y - b.y;
     angle=atan2(dy,dx);
     Si(dy<0)Alors
       angle=angle+(2*PI);
     FinSi
     getAngle  = angle;
Fin
```

F. Secteurs des angles

Les secteurs sont utilisés pour avoir une description présentant la relation entre les angles de déviation des segments, et au même temps en garde un certain degré de liberté.

La figure suivante illustre ce principe.

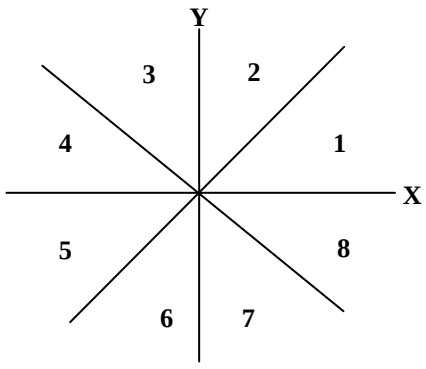


Figure 4.23: Illustration des secteurs.

Après la calcule de l'angle avec l'algorithme précédent, nous pouvons obtenir le secteur de cette angle en appelant cette fonction:

```
Fonction  getAngleSector(angle:double):entier

     rem:double;
     sector:entier;

Début
      rem=angle mod (2*PI);
      sector=1+floor(rem/(PI/4)));
      getAngleSector = sector;
Fin
```

Exemple: on prenant qu'un secteur est noté comme suit: s(num_secteur) nous obtenons la figure suivante:

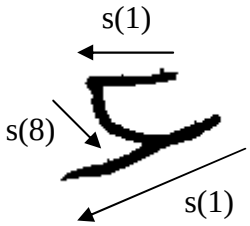


Figure 4.24: Illustration d'un exemple de secteurs d'angles.

Et par la suite, nous pouvons avoir la description suivante:

SADescriptor = 181, (Sector Angle Descriptor).

G. Taille du segment d'image

La taille du segment d'image est définit par la dimension de l'image (largeur, hauteur) et il est facilement calculable par les deux fonctions (fournit par le langage de programmation – JAVA-) suivantes:

- getWidth:fonction permet d'obtenir la largeur d'une image,
- getHeight: fonction permet d'obtenir l'hauteur d'une image.

La figure suivante illustre cette idée.

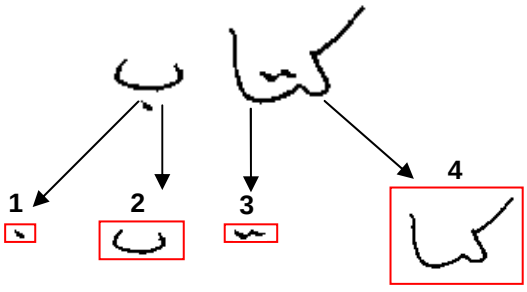


Figure 4.25 : Exemple de segmentation de l'image du mot "كتاب"

En voyant bien, la figure 4.25 nous pouvons déduire que la taille du segment d'image peut être un élément clé pour différencier entre les différents formes (par exemple entre les cadre:1, 2 et 3).

H. Type des points constituants la boucle

Le type des points constituant la boucle est obtenu au fur et à mesure qu'on fait des opérations de division ou fusion. Cette étape est réalisée à l'aide d'une classe Java appelée Tracker (nom_Image), qui va prendre chaque image et faire un suivi afin d'obtenir les caractéristiques ou vecteur caractéristique (taille, nb_Cycles, Direction…etc) de chaque caractère.

A la fin de la phase d'extraction de caractéristiques, nous disposons d'un vecteur de coefficients réels qui caractérisent un/des caractère(s). Dans la phase suivante, la méthode SVM est utilisée pour se servir de ces coefficients et reconnaître le(s) caractère(s).

3.1.5 Classification des caractères [SVM]

Le vecteur caractéristique obtenu dans l'étape précédente est prêt maintenant pour être classer, mais avant de discuter ce point, nous allons voir premièrement :

- quelles sont les classes utilisées ou plus précisément quelle hiérarchie de classes nous allons utiliser,
- la méthode de classification SVM en détail.

A. Hiérarchie de classes utilisé par Aya

Dans la réalité les classes sont beaucoup plus des formes que des lettres, parce que en examinant les lettres de la langue arabe on peut remarquer que pas mal de lettres sont composés de deux formes (voir figure 4.26) ce qui permet d'avoir une *variabilité* importante dans le style d'écriture.

A.1 Variabilité de style d'écriture manuscrite

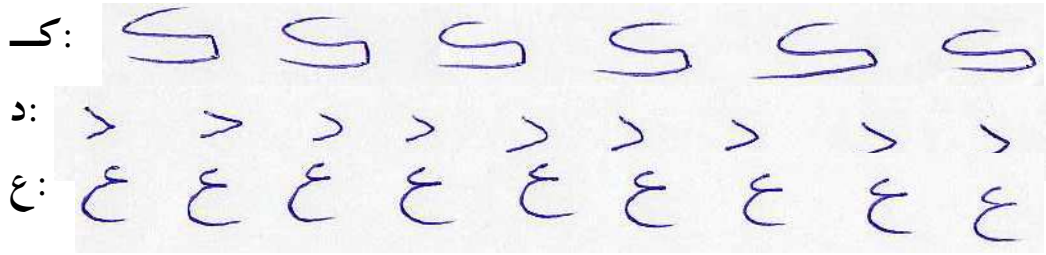
C'est la différence en écriture de la même séquence de caractères. On note pratiquement deux types de variabilité, à savoir :

- variabilité locale, au niveau caractère;
- variabilité globale, au niveau mot.

A.1.1 Variabilité locale

Elle résulte de la différence en représentations du même caractère en terme de position des différents formes composants ce caractère ainsi que les différentes manières d'écrire (de dessiner) chaque forme.

Exemple: voir les allographes des lettres suivantes:



A.1.2 Variabilité globale

Elle résulte de la somme des variabilités locales: des caractères composants le mot et les différentes manières de liaisons entres ces caractères. Nous sommes alors au niveau mot.

A.2 Formes primitives

Afin de minimiser l'effet de variabilité on procède à une séparation entre les formes composantes les mêmes caractères (sous classes), on parle donc d'une hiérarchie de classes. Le résultat c'est des formes primitives plus des règles de gestion de position ou positionnement.

Exemple :

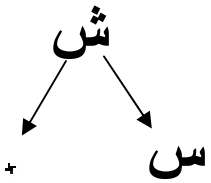


Figure 4.26 : Exemple de génération de formes primitives.

On aura par conséquent les formes primitives suivantes :

ا،ب،ح،د،ر،س،ص،ط،ع،ف،ق،كـ،ل،م،ن،هـ،ه،و،ـه،
ى،ء،.

Figure 5.27 : Illustration des formes primitives.

On peut classer ensuite les lettres de la langue arabe comme suit :

1. **Caractères simples CS :** ـكـ ، ل ،م
2. **Caractères composés CC :** أ،إ،آ، ب ، ت،ث، ج ، ذ ،ز،ش،ض،ظ،غ،ف،ق،ن،ي
3. **Caractères ambigus CA :** ح،د،ر،س،ص،ط ،ع، و،ه،ى

La figure suivante montre bien comment ces caractères sont obtenus:

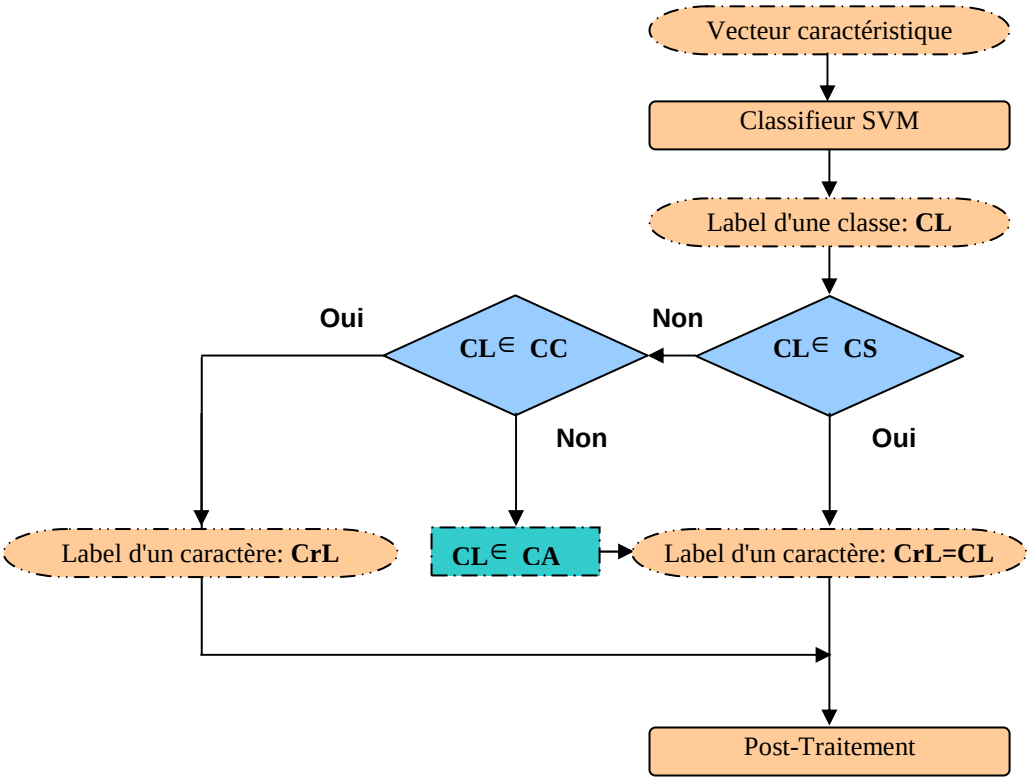


Figure 4.27 : Processus de post-classification.

B. Classifieur SVM

Un classifieur SVM est chargé de prendre le vecteur de caractéristiques (taille, nb_Cycles, Direction…etc) généré par le module précédent comme une donnée d'entrée, rechercher un hyperplan séparateur qui sépare les exemples dans la phase d'apprentissage et faire une décision de classification dans la phase d'identification.

Dans le module SVM, il y a deux phases : une pour l'apprentissage et l'autre pour la classification. La figure suivante représente la relation entre ces deux phases.

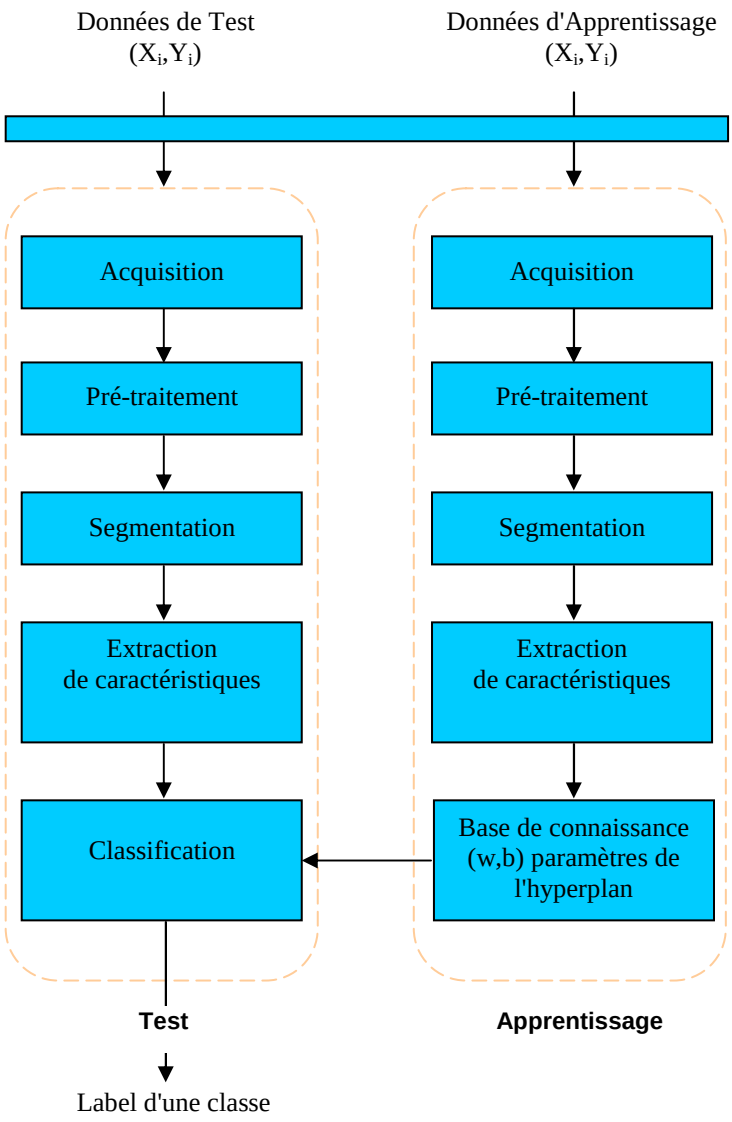


Figure 4.28 : Illustration des deux phases utilisées d'un classifieur SVM.

Où :
Xi : représentent le vecteur caractéristique d'un caractère ;
Yi : l'étiquette d'une classe**.**

B.1 Phase d'apprentissage

A présent nous devons résoudre le problème dual :

$$\text{Min } \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j K(x_i,x_j) \qquad (1)$$

Sous les contraintes :

$$0 \le \alpha_i \le c \qquad (2)$$

$$\sum_i \alpha_i = 1 \qquad (3)$$

Faire converger ce problème d'optimisation revient à trouver une solution respectant les conditions de Karush-Kuhn-Tucker (KKT). En effet, on sait d'une part que ces conditions sont satisfaites à la solution de tout problème d'optimisation(convexe ou non), et d'autre part que ce sont des conditions nécessaires et suffisantes pour un problème convexe, ce qui est le cas pour les SVM [48].

Pour le problème primal évoqué précédemment, ces conditions sont d'une part :

$$\frac{\partial L}{\partial w} = 0 \Leftrightarrow w = \sum_i \alpha_i \phi(x_i)$$

$$\frac{\partial L}{\partial b} = 0 \Leftrightarrow \sum_i \alpha_i = 1$$

$$\frac{\partial L}{\partial \xi_i} = 0 \Leftrightarrow \alpha_i = c, \beta_i \leq c .$$

Ce qui nous garantissons comme on l'a vu précédemment que l'on a équivalence entre les problèmes primal et dual ; et d'autre part :

$$\alpha_i (\langle w, \phi(x_i) \rangle - b + \xi_i) = 0$$

(Avec toujours les contraintes de positivité sur les $(\alpha_i, \beta_i et \xi_i)$.Qu'est ce que cela donne pour nos trois cas de figure évoqués précédemment (sous l'hypothèse que les différentes contraintes sont vérifiées et qu'on n'a par conséquent que la dernière condition à vérifier) ?

Si $\alpha_i = 0$ alors les conditions sont vérifiées.
Si $0 \leq \alpha_i \leq c$ alors on peut démontrer que l'on a $\xi_i = 0$ et comme $\langle w, \phi(x_i) \rangle = b$ les conditions sont vérifiées.
Si $\alpha_i = c$ alors $\xi_i = b - \langle w, \phi(x)_i \rangle (\neq 0)$ et les conditions sont vérifiées. Que l'on a C'est donc grâce à de la que l'on va définir l'algorithme d'optimisation : on va chercher *b* et des coefficients $\alpha_i (i \in [1, l]$ (où *l* le nombre des exemples) tels que l'ensemble des points de l'ensemble d'apprentissage vérifie les conditions de KKT.
L'algorithme présenté est une variante de l'algorithme « *Sequential Minimum Optimisation* » *(SMO)* développé par John Platt. Nous allons le présenter dans ses grandes lignes.

B.1.1 Algorithme SMO

Il n'est pas facile de résoudre l'optimisation du problème dual des SVM du fait du volume de données mis en jeu. En effet, on doit pour cela considérer une matrice $Ker(i, j) = k(x_i, x_j)$ de taille *N*N*. L'optimisation peut rapidement s'avérer assez longue lorsque la taille de la base d'apprentissage augmente [48].

En 1997, Osuna et al ont démontré qu'on pouvait décomposer le gros problème quadratique en une suite de sous problèmes de taille plus petite. La condition étant

qu'il y ait au moins une variable violant KKT ajoutée au problème à chaque étape. L'algorithme SMO applique le théorème de Osuna et décompose le gros problème quadratique en une suite de sous problèmes les plus petits possibles : optimiser les multiplicateurs de Lagrange deux à deux à chaque étape.
Même si le nombre d'itérations doit être plus important, l'optimisation est nettement plus rapide puisqu'elle peut se faire de manière totalement analytique. En effet, si on considère les α_i deux à deux, alors compte tenu :

- d'une part des contraintes sur le domaine des α ($0 \le \alpha_i \le c$),
- et d'autre part de $\sum_i \alpha_i = 1$, ou de $\sum_i y_i \alpha_i = 0$ dans le cas générale, (ce qui implique que la somme des deux coefficients est la même avant et après optimisation), on ne recherche le minimum que sur une portion de droite (une ligne diagonale dans le carré défini par les bornes des α_i. L'avantage est que l'on n'a ainsi pas besoin d'effectuer de calculs matriciels (voir figure 4.29).

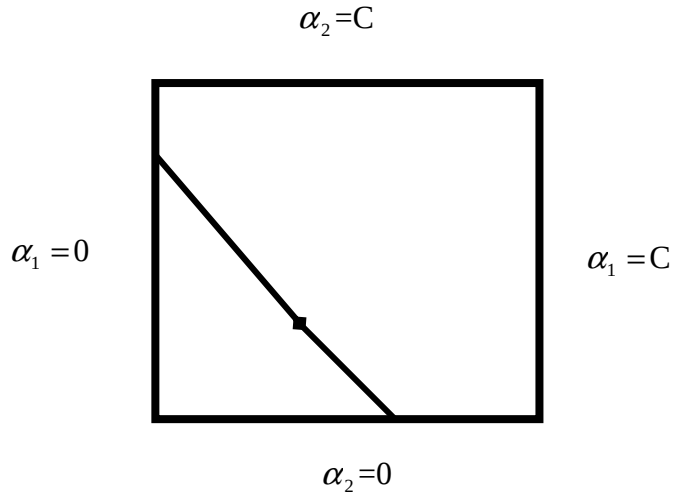


Figure 4.29 : Illustration du domaine à considérer pour l'optimisation jointe des deux multiplicateurs de Lagrange.

L'algorithme SMO est donc le plus efficace pour résoudre le problème des SVM. Il y a deux choses à prendre en compte pour définir l'algorithme SMO :

- le choix des deux α_i à optimiser.
- la méthode pour optimiser.

B.1.2 Choix des α_i à optimiser

La première étape de l'algorithme est de définir à chaque itération quels vont être les coefficients α_i à optimiser. Pour cela l'algorithme utilise deux heuristiques :

- La première concerne le premier multiplicateur : elle vise à choisir un multiplicateur correspondant à une donnée ne vérifiant pas les conditions de KKT.
- La seconde concerne le choix du second multiplicateur : elle vise à maximiser l'optimisation à être effectuée une fois que l'on a choisi le premier multiplicateur.

L'algorithme se compose de deux boucles : la première applique ces deux heuristiques sur l'ensemble des données, et la seconde s'intéresse uniquement aux coefficients correspondant aux vecteurs de support non bornés, c'est à dire pour lesquels n'est à la fois ni nul ni égal à la contrainte de poids maximal. En effet, ces coefficients sont plus susceptibles d'être modifiés que les coefficients situés aux bornes. Un passage dans la première boucle est suivi de multiples passages dans la seconde jusqu'à ce que l'ensemble des vecteurs de support non bornés défini précédemment satisfasse KKT. Concrètement, à chaque passage dans la première boucle, on modifie (lors de l'optimisation) l'ensemble des vecteurs de support non bornés, et la seconde boucle modifie les coefficients correspondant jusqu'à ce que cet ensemble soit en accord avec KKT [10].

En appliquant cette méthode, on est sur qu'à chaque itération on présente au moins un coefficient violant KKT à l'algorithme, et la méthode est donc en accord avec le théorème d'Osuna. On assure ainsi la convergence de l'algorithme, c'est à dire qu'à chaque itération on est sur de diminuer la fonction objective à minimiser. L'optimisation se termine lorsque toutes les données de l'ensemble d'apprentissage vérifient KKT, c'est à dire lorsque la première boucle ne trouve plus de violateurs. En ce qui concerne la seconde heuristique, une fois que l'on a choisi notre premier multiplicateur, on choisit le second de manière à maximiser le pas d'optimisation. Cela revient à choisir α_2 tel qu'on ait une distance maximale entre les sorties du classifieur :

$$\left|O_1 - O_2\right| où O_i = \sum_j \alpha_j k(x_i, x_j)$$

B.1.3 Méthode d'optimisation

Une fois qu'on connaissant les deux coefficients que l'on veut optimiser, il reste à définir une méthode d'optimisation jointe. Comme on l'a déjà évoqué précédemment, cette optimisation peut être réalisée de manière analytique, et de plus sur un domaine assez restreint (une ligne diagonale dans le carré défini par les contraintes sur les bornes des α_i) [48].

Etant donné que l'on ne considère que deux coefficients, on peut réécrire le problème d'optimisation (1) comme :

$$Min_{\alpha_1,\alpha_2} \frac{1}{2}\sum_{i,j=1}^{2} \alpha_i \alpha_j k(i,j) + 2\sum_{i=1}^{2} \alpha_i c_i + c \,. \qquad (4)$$

Où $\quad c_i = \sum_{j=3}^{l} \alpha_j k(i,j)$ et $c = \sum_{i,j=3}^{l} \alpha_i \alpha_j k(i,j)$.

De la même façon, la contrainte linéaire (3) peut se réécrire $\sum_{i=1}^{2} \alpha_i = \Delta$ où $\Delta = 1 - \sum_{i=3}^{l} \alpha_i$ de façon à exprimer α_1 en fonction de α_2. c étant indépendant de α_1 et de α_2 minimiser la fonction objective revient à minimiser la fonction suivante :

$$Min_{\alpha_2} \frac{1}{2}(\Delta-\alpha_2)^2 k_{1,1} + (\Delta-\alpha_2)k_{1,2} + \frac{1}{2}(\alpha_2)^2 k_{2,2} + (\Delta-\alpha_2)c_1 + \alpha_2 c_2 \qquad (5)$$

Le minimum correspond au α_2 annulant la dérivée de (5)

$$\alpha_2 = \frac{\Delta(k_{1,1}-k_{2,2}) + c_1 + c_2}{k_{1,1}+k_{2,2}+k_{1,2}}. \qquad (6)$$

Une fois que α_2 est calculé, on prend $\alpha_1 = \Delta - \alpha_2$ .Pour alléger les notation, on note $O_i = \sum_{j=1}^{l} \alpha^*{}_j k_{i,j}$ où $\alpha^*{}_j$ désignent les multiplicateurs de Lagrange avant le pas d'optimisation et $\eta = k_{1,1} + k_{2,2} - 2k_{1,2}$ .On obtient alors : $\alpha_2 = \alpha_2^* + \frac{O_1 - O_2}{\eta}$. (7)

On peut remarquer que η est la dérivée seconde de la fonction objective. Comme cette dernière est définie positive, η sera toujours strictement supérieur à zéro.

Nous avons donc α_1 et α_2 qui vérifient la contrainte linéaire $\sum \alpha_i = 1$(3), il ne reste plus qu'à faire en sorte qu'ils préservent la contrainte $0 \le \alpha_i \le c$ (2), c'est à dire qu'ils restent sur un segment diagonal.

Les extrémités de ce segment s'expriment assez simplement. Nous avons vu que sans perte de généralité l'algorithme peut d'abord optimiser le second multiplicateur de Lagrange, α_2, et calculer les bornes du segment diagonal en terme d' α_2. Comme nous n'avons qu'une classe, nous avons les bornes suivantes appliquée à α_2 :

$$L = Max(0, \alpha_2 + \alpha_1 - c)$$

$$M = Min(c, \alpha_2 + \alpha_1)$$

La contrainte minimum α_2^{final} est donc trouvée en maintenant α_2 entre les extrémités du segment:

$$\alpha_2^{final} = \begin{cases} HSi\,\alpha_2 \ge H \\ \alpha_2 SiL \le \alpha_2 \le H \\ LSi\,\alpha_2 \le L \end{cases} \qquad (8)$$

Et donc la valeur d' α_1 est calculée avec la valeur finale de α_2 :

$$\alpha_1 = \alpha_1^* + (\alpha_2^* + \alpha_2^{final}) \qquad (9)$$

B.1.4 Calcul de b

La valeur de *b* est recalculée après chaque pas d'itération de sorte que le couple de multiplicateurs optimisé vérifie les conditions de KKT. Nous avons vu dans le

deuxième chapitre que *b* est calculé à partir des points appartenant à la frontière. Nous n'avons donc b_1 valide que si α_1 n'est pas une borne et alors b_1 a pour valeur :

$$b_1 = \sum_{j=1}^{l} \alpha_j k_{1,j} .$$

de même b_2 n'est valide que si α_2 n'est pas une borne et a pour valeur :

$$b_2 = \sum_{j=1}^{l} \alpha_j k_{2,j}$$

Quand b_1 et b_2 sont valides, ils sont égaux et $b = b_1 = b_2$, *b* est alors le nouveaux seuil pour l'itération suivant. Quand un de deux seulement est valide en le prend comme nouvelle valeur de *b* .Lorsque aucun de deux n'est valide, mais que L ≠ M, i.e. $\alpha_1 = c$ et $\alpha_2 = 0$ ou alors $\alpha_1 = 0$ et $\alpha_2 = c$ alors l'intervalle entre b_1 et b_2 contient tous les seuils consistant avec KKT . L'algorithme SMO choisit alors la moyenne entre b_1 et b_2 [48].

B.2 Phase de Classification

Dans la phase précédente nous résolvons un problème dual. Nous connaissons alors *w et b* et nous pouvons définir la fonction de décision pour une nouvelle donnée *x* :

$$f(x) = signe(\langle w.\phi(x)\rangle - b .$$

B.3 Algorithme générale de SVM

Avant la discussion des détails de l'algorithme SVM, Nous notons que lors de l'implémentation, on est contraint de considérer une précision pour le zéro numérique. Ceci est de toutes façons vrai pour toute implémentation. Ici nous avons considéré que deux données sont égales si elles diffèrent de moins de 10^{-3} . Ce seuil proposé par Platt dans la présentation de l'algorithme est raisonnable car il revient dans le cas des SVM à considérer pour une marge théorique de 1, une marge effective comprise entre 0.999 et 1.001.

Cette précision nous sert en particulier lorsque l'on recherche les données violant les conditions de KKT et pour contrôler l'optimisation des couples de multiplicateurs de Lagrange. Notons que cette valeur a son influence sur la vitesse de convergence de l'algorithme : plus elle est faible et plus l'algorithme est lent à converger. Cette précision est appelée *zéro_ tolérance* dans nos programmes [48].

B.3.1 Algorithme de résolution

Entrée:

Q,b,y,Cp,Cn, et an des points initial faisable alpha;
l: la taille des vecteurs et matrices;
eps: critère d'arrêt.

Sortie:

alpha: vecteur contenant la solution;
obj: la valeur objective;

Variables locales:

G,G_bar: vecteurs utiliser pour les calcules du gradient;
alpha_status: vecteur contenant l'état de alpha;
active_size: variable contenant la taille du vecteur active_set;
active_set: vecteur contenant l'ensemble de travail courant;

Début

1. Initialisation des paramètres : taille de l'échantillon (l), paramètre c, largeur de bande du noyau (gaussien), précision numérique eps (zéro_tolérance).
2. Initialiser le vecteur alpha_status (vecteur contenant l'état d'alpha).
3. Initialiser la variable active_size et le vecteur active_set (utilisés durant l'opération de shrinking).
4. Initialiser le gradient.
5. Commencer l'optimisation:
    a. Faire l'opération de shrinking.
    b. Reconstruire tout le gradient.
    c. Réinitialiser active_size et choisir un ensemble de travail.
    d. Mis à jour d'α_i, α_j, G, G_bar et alpha_status.
6. Calculer la valeur objective.
7. Affichage des résultats.

Fin

L'algorithme présenté ci-dessus ne montre que les grandes étapes de résolution des SVM; pour plus d'informations il faut consulter l'ad-doc jointe avec le package libsvm-2.83 [40].

B.3.2 Classification multi classes

Pour une classification multi-classes nous utilisons une stratégie de vote: chaque classification binaire est considérée à être un vote, où les votes sont considérés comme un cast pour les points des données X. la classe choisi est celle qui a le nombre max de votes [40].

B.4 Bibliothèque LIBSVM utilisée

La bibliothèque LIBSVM est développée dans le but de simplifier l'utilisation des SVM comme un outils, dans ce travail nous utilisant la version 2.83 qui est présentée par le package java libsvm-2.83.

La relation entre une application utilisateur et le package libsvm-2.83 peut être vu selon un modèle de trois couches:

- **Application:** qui représente l'application utilisateur,
- **Interface:** qui permet la communication entre l'application et les modules de calcule, et
- **Calcule:** qui permet de réaliser les calcules nécessaire.

La figure suivante illustre cette relation.

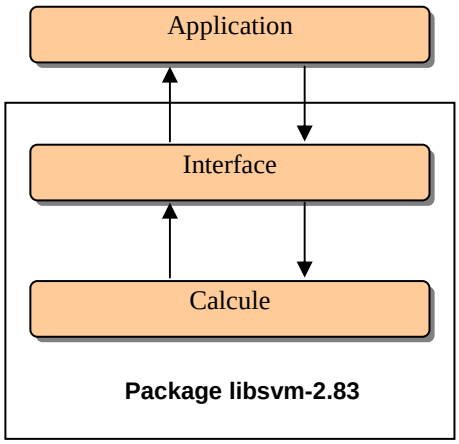


Figure 4.30 : Illustration de la relation entre l'application utilisateur et le package SVM.

La couche interface est présentée par les deux modules:

- svm_train(CarVec.txt) : pour la phase d'apprentissage, et
- svm_predict (CarVec.txt, CarVector.model, classLabels.txt) : pour la phase de test (prédiction).

Où:
CarVec.txt: fichier contenant les caractéristiques extraites.
CarVector.model: fichier contenant les paramètres (w,b) après la phase d'apprentissage.
classLabels.txt: fichier contenant les étiquettes des classes après la phase de test ou décision.

Autant que la couche calcule est présentée par les modules:

- svm: comme module principale,
- svm_node, svm_parameter, et svm_problem: comme des modules complémentaire.

Les modules de la couche calcule sont invisible a l'utilisateur du package. Un utilisateur ne peut se communiquer avec le package qu'à travers les deux modules de la couche interface présentée ci-dessus.

La relation entre l'application et le package libsvm-2.83 ainsi que les différents modules du package est présentée en détail par la figure suivante:

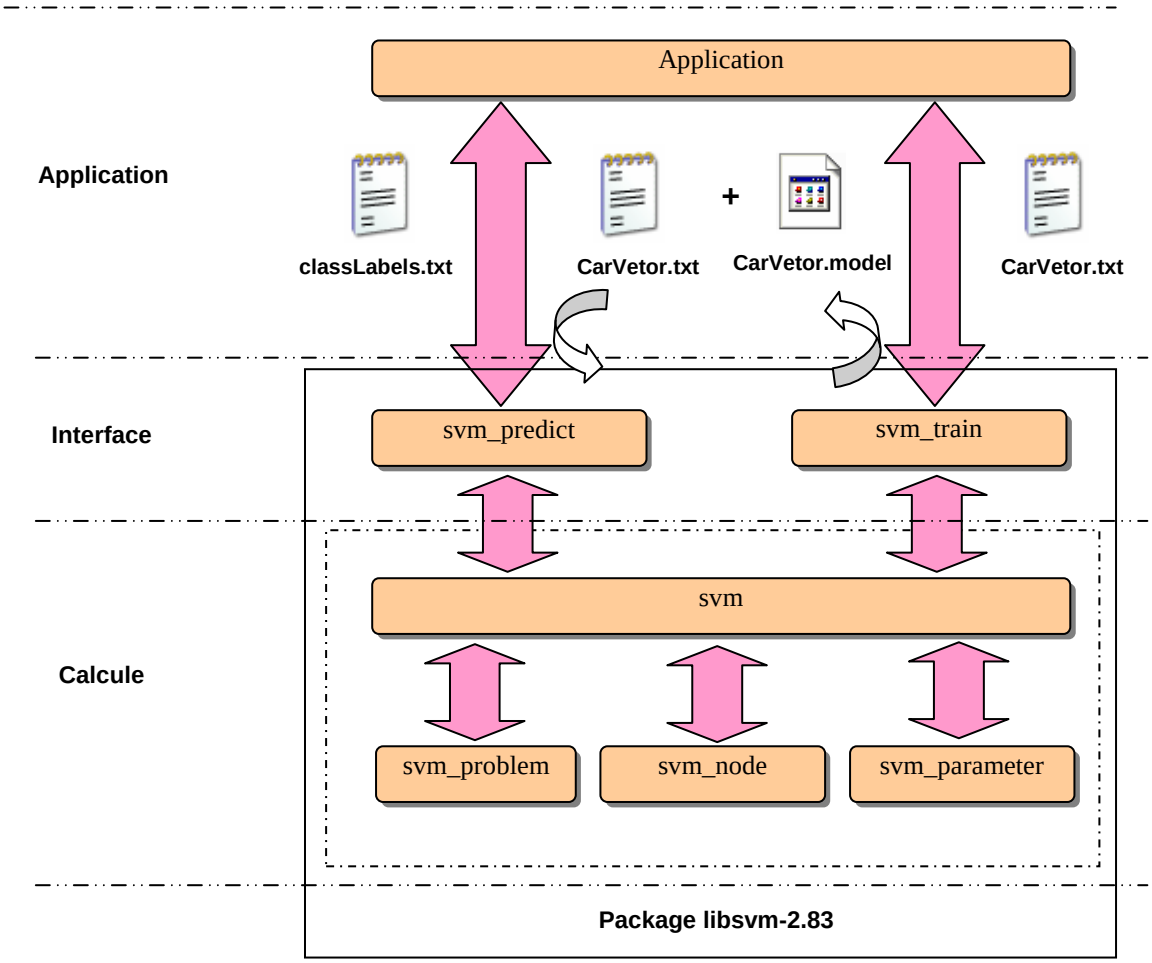


Figure 4.31 : Relation en détail entre l'application utilisateur et le package libsvm-2.83 .

Premièrement, dans la phase d'apprentissage la couche application génère un fichier texte (CarVector.txt) contenant les vecteurs caractéristiques (Xi) et les étiquettes des classes (Yi), en utilisant ce fichier comme entrée au module svm_train nous obtenant un fichier de modèle (CarVector.model) qui contient les paramètres (w,b) nécessaire à la phase de test ou prédiction.

En revanche, dans la phase de test on utilise le fichier de modèle généré précédemment plus un fichier texte (CarVector.txt) généré par l'application nous obtenant comme résultat un autre fichier texte (classLabels.txt) qui contient les étiquettes des classes.

3.1.6 Post-traitement

Dans le cas de la langue arabe, l'étape de post-traitement peut être considérer prémordiale, du fait qu'il est presque impossible de bien associer les caractères du même mot sans avoir passer par un dictionnaire ou bien règles lexicales. Le mot reconstruit est donné à un gestionnaire de mise en page afin de reconstruire le résultat final (page entière), donc on passe par trois étapes:

1. reconstruction du caractère,
2. reconstruction des mots et lignes,
3. reconstruction de la page.

La figure suivante illustre ces étapes:

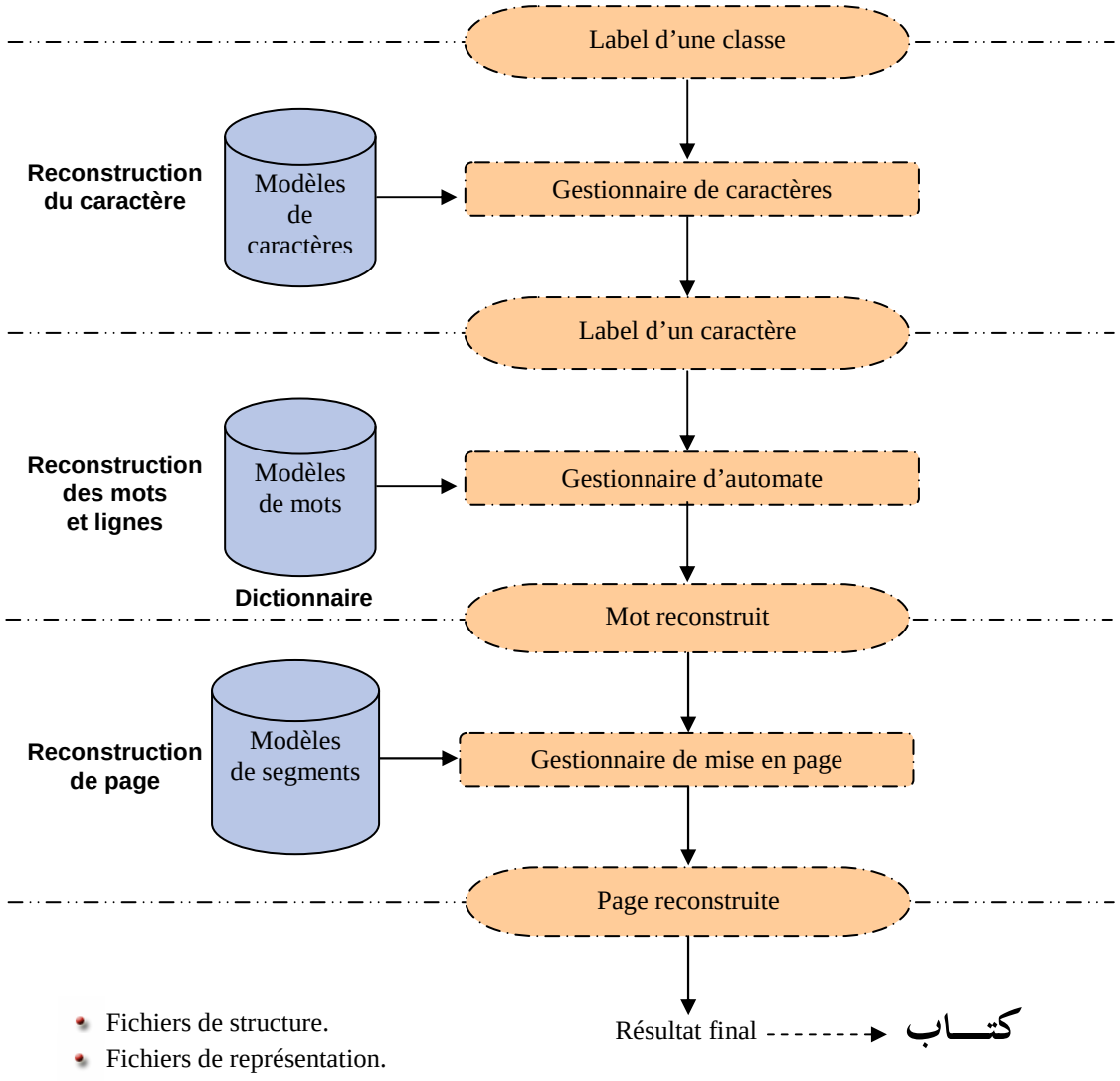


Figure 4.32 : Illustration des étapes de Post-Traitement

A. Reconstruction du caractère

Après avoir obtenir le label d'une classe de l'opération de classification, il faut avoir maintenant le label du caractère adéquat, et ça se fait en appelant la fonction suivante:

```
Fonction getCharacterLabel(CL:entier):entier
    // Character Label: label d'un caractère
    CrL:entier;

Début
    Si(CL<3)Alors // caractère simple
        CrL=CL;
    Sinon
        Si(CL<20)Alors // caractère composé
            // rassembler le caractère
            CrL=reassembleCharacter(CL);
        Sinon
            Si(CL<22)Alors // caractère de gestion
                // rien faire
            FinSi
        FinSi
    FinSi
    getCharacterLabel=CrL;
Fin
```

Dans la figure 4.26 nous avons vu un exemple de génération de formes primitives, la fonction `reassembleCharacter`(CL:entier) permet de faire le processus inverse.

La figure suivante illustre ce principe.

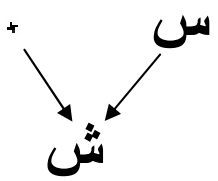


Figure 4.33 : Exemple de composition de formes primitives.

```
Fonction reassembleCharacter(CL:entier):entier
        // Character Label: label d'un caractère
        CrL:entier;
Début
     Cas CL De
         3:Si(getUpperSegmentClassLabel()==19)Alors
                CrL=34;  // أ
           Sinon
                Si(getLowerSegmentClassLabel()==19)Alors
                     CrL=35;  // إ
                Sinon
                 Si(getUpperSegmentClassLabel()==21)Alors
                      CrL=36;  // آ
                 Sinon
                      Ecrire("Error code ?...");
                 FinSi
                FinSi
           FinSi
         4:...
     Fin
     reassembleCharacter=CrL;
Fin
```

Où:

`getUpperSegmentClassLabel()`: permet de retourner la classe du segment d'image en haut du segment traité.

`getLowerSegmentClassLabel()`: permet de retourner la classe du segment d'image en bas du segment traité.

Le reste des caractères est reconstruit de la même façon en donnant un code d'entrée et en obtenant un code de sortie pour le caractère adéquat, suivant les codes présentés dans le tableau B.3 (voir Annexe B).

B. Reconstruction des mots et lignes

Dans la phase de segmentation, pour un texte donner nous allons extraire un ensemble de lignes. Maintenant, pour chaque ligne nous allons extraire la suite de caractères en passant par les étapes précédemment présentées (tracking, classification). Ensuite, cette suite est lu de droite à gauche pour construire un segment de mot qui est comparé après avec un autre mot chargé d'un dictionnaire pour vérifier son existence.

Ces étapes sont présentées par l'algorithme suivant:

```
1. Extraire les lignes du  texte sous format d'image;
2. Pour chaque ligne Faire
3.     Extraire la séquence de caractères;
4.     Pour i de 0 à nbr_caractères pas 1 Faire
5.         mot=mot+caractère[i];
6.         Si(existe(mot,Dictionnaire)Alors continuer
7.         Sinon
8.              mot=mot- caractère[i];
9.              Si(mot<>null)Alors
10.               écrire(mot,HTML_File);
11.             Sinon
12.                  // gestion d'érreur
13.                  écrire("code d'érreur?...");
14.             FinSi
15.        FinSi
16.   FinPour
17.FinPour
```

C. Reconstruction de page

Parce qu'on fait la reconnaissance de document simple où il y'a seulement du texte cette étape est pratiquement réalisée automatiquement dans l'étape précédente. Les lignes obtenues sont écrites dans un fichier (HTML par défaut).

Le résultat final (page reconstruite) va être représenter par des:

1. Fichiers de structure: XML,…etc.
2. Fichiers de représentation: HTML, DOC,PDF,Ps, …etc.

3.2 Résultats et bilan

Cette section, présentera le choix du langage de programmation, les différentes interfaces et fenêtres principales du système, les tests et résultats obtenus.

3.2.1 Choix du langage de programmation

Dans ce travail, nous avons choisis comme environnement de programmation le langage JAVA qui possède une richesse et offre une grande simplicité de manipulation d'images, soit en acquisition ou en génération des fichiers images.

Ce langage possède des avantages très intéressants tel que :

- La portabilité des logiciels ;
- La réutilisation de certaines classes déjà développées ;
- La possibilité d'ajouter à l'environnement de base des composants fournis par l'environnement lui même ;
- La quasi-totalité de contrôle de windows (boutons, boites de saisies, listes déroulantes, menus …etc.) qui sont représentés par classes;

3.2.2 Interface et Fenêtres

On lançant le logiciel nous allons voir premièrement une image d'entrée (splash window) suivie de la fenêtre principale comme nous montre la figure 4.32.

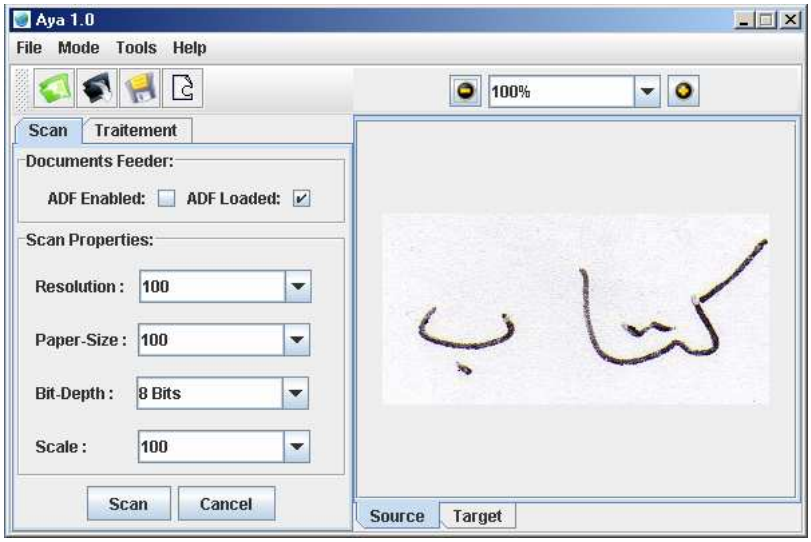


Figure 4.34: Illustration de la fenêtre principale de l'application.

L'application peut être utilisée suivant deux modes:

- Apprentissage: comme phase initiale pour aider le système à apprendre les différentes classes (Formes primitives),
- Test: pour tester et calculer le taux de reconnaissance, et éventuellement utiliser Aya.

A. Mode Apprentissage

Ce mode peut être vu comme phase initiale ou d'initialisation de la base de connaissance du système, pour le faire on procède comme suit:

1. choisir une classe cible (l'un des formes primitives), à l'aide du combo box comme le montre la figure suivante:

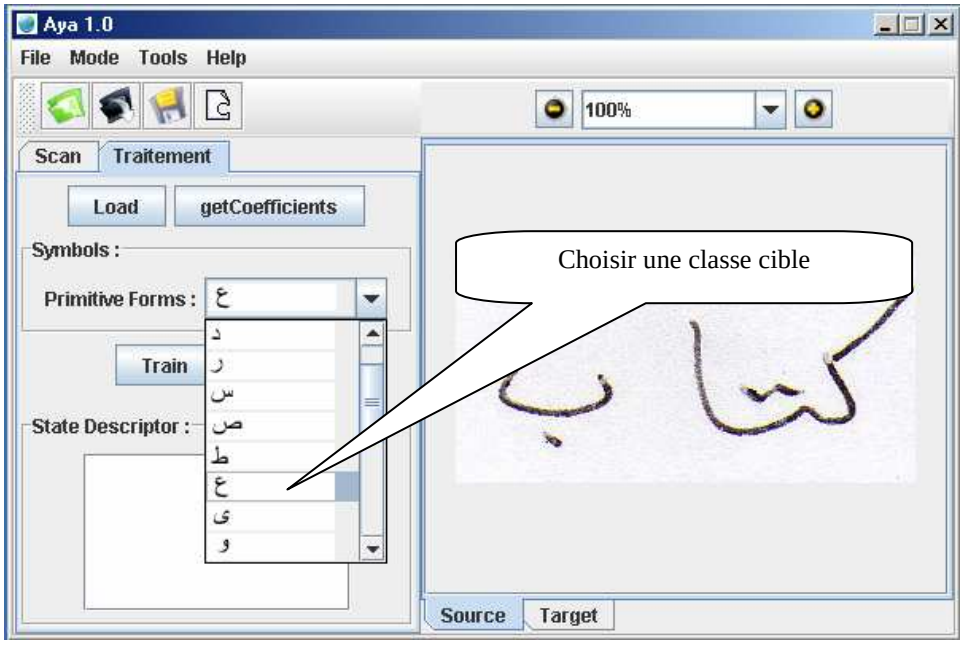


Figure 4.35: Illustration du chois d'une classe.

2. Charger une image contenant les exemples de la classe choisis (l'un des formes primitives), à l'aide du bouton *Load*, *File/Open* du menu principal ou de l'onglet *Scan* à l'aide d'un scannaire.

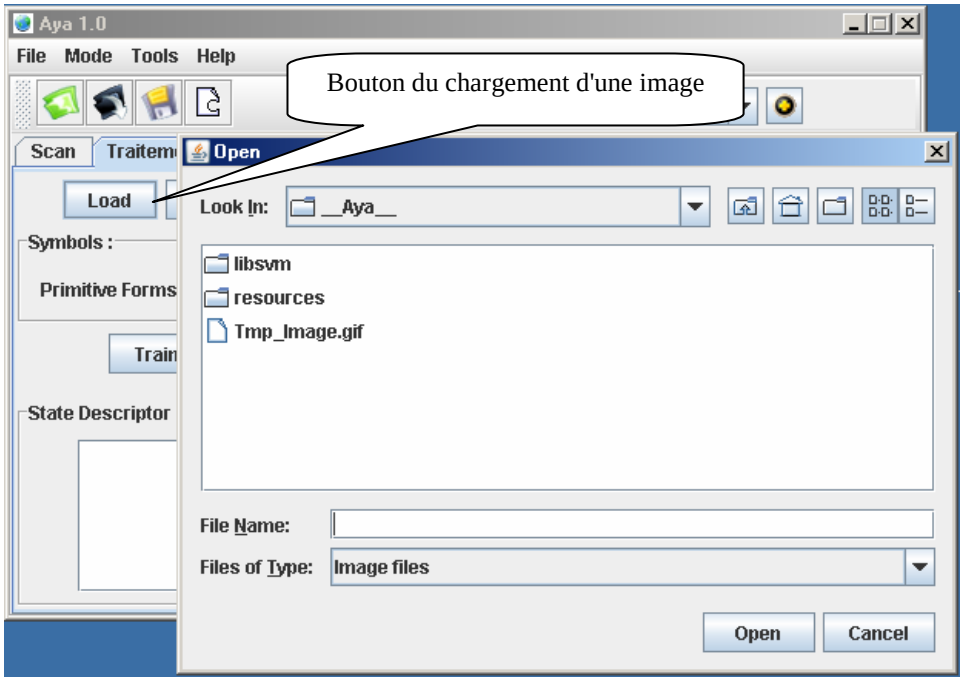


Figure 4.36: Illustration du chargement d'une image.

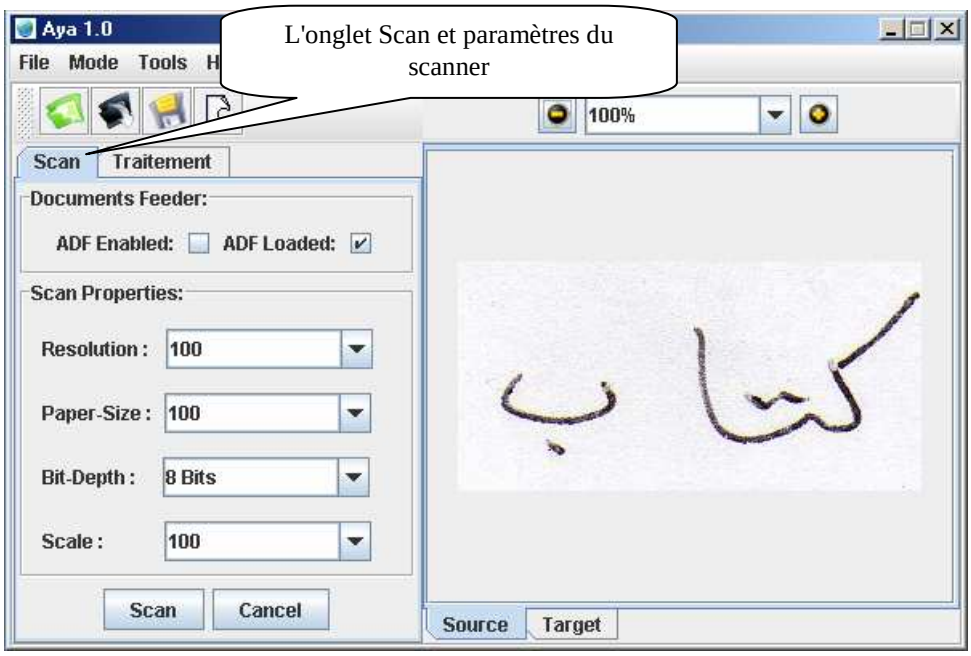


Figure 4.37: Illustration du chargement d'une image.

3. Cliquer sur le bouton *getCoeffients* pour extraire les caractéristiques des exemples et les écrire dans le block note du classifieur.

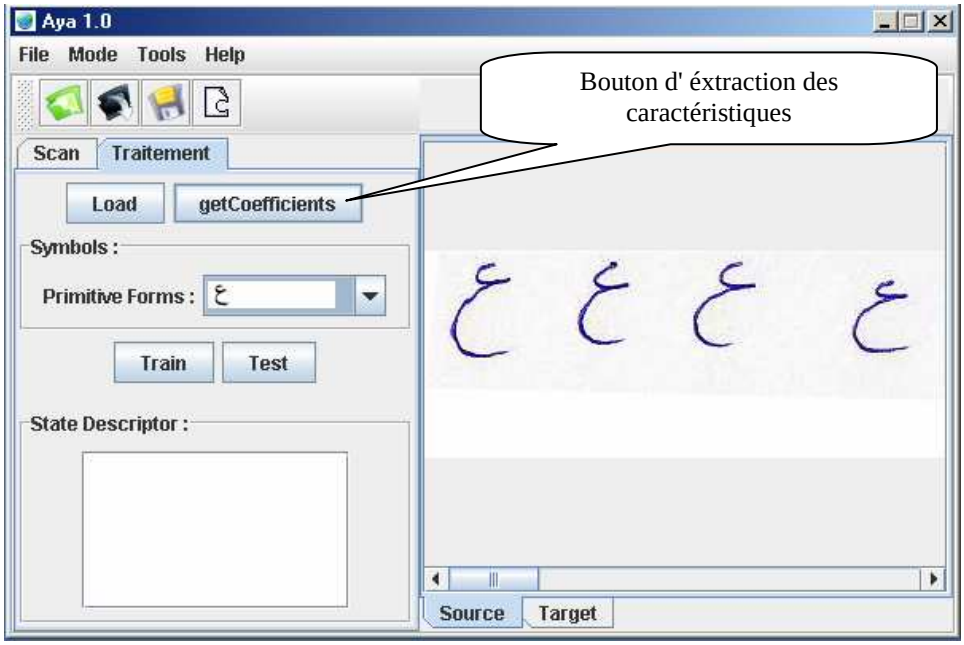


Figure 4.38: Extraction des caractéristiques des exemples d'apprentissage.

4. Cliquer sur le bouton d'apprentissage *Train*.

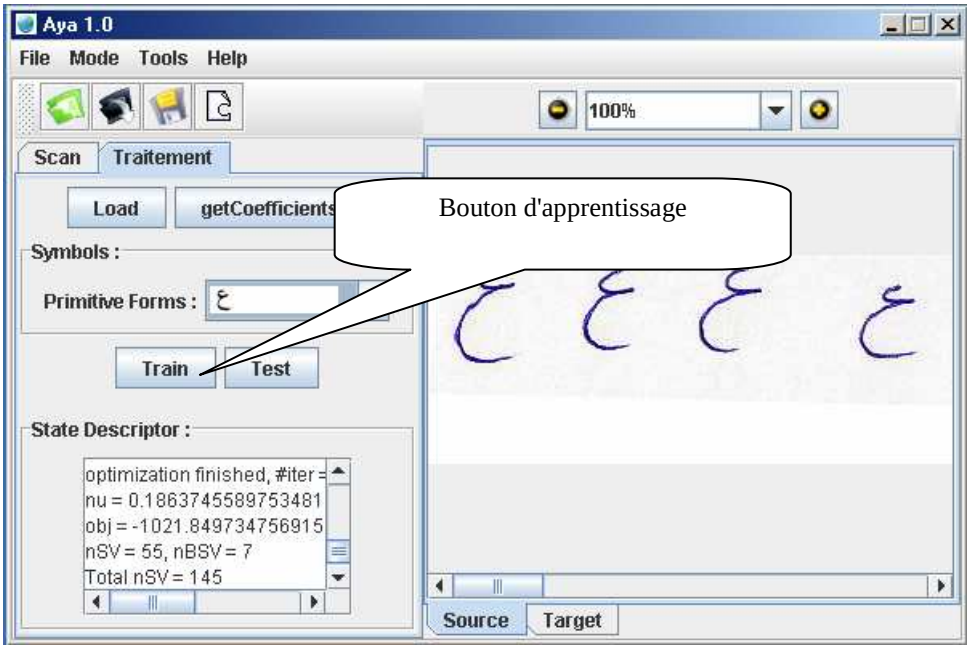


Figure 4.39: Illustration du mode d'apprentissage.

Ces étapes sont répétées pour toutes les classes (Formes primitives) .

B. Mode Test

Ce mode ne peut être exploitable qu'après avoir terminer la phase d'apprentissage et il suit presque les mêmes étapes avec des différences légères. Les étapes 1 et 2 sont les mêmes, pour l'étape 3, on clique sur le bouton de test *Test*.

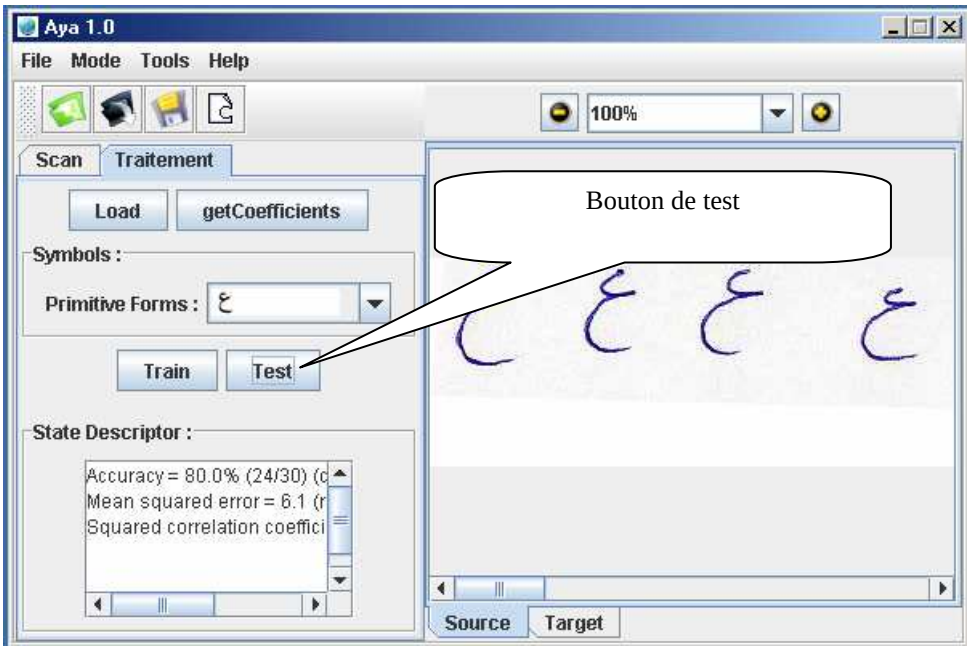


Figure 4.40: Illustration du mode de test.

Dans la réalité, ce mode peut être étendu en mode d'utilisation normal en annulant tout simplement l'étape 1, ce qui veut dire qu'on s'intéresse pas à calculer le taux de reconnaissance mais seulement à reconnaître ce qui est écrit dans le fichier source (on garde les étapes 2 et 3).

Le résultat du traitement va être présenté dans l'onglet *Target* sous forme d'un fichier html.

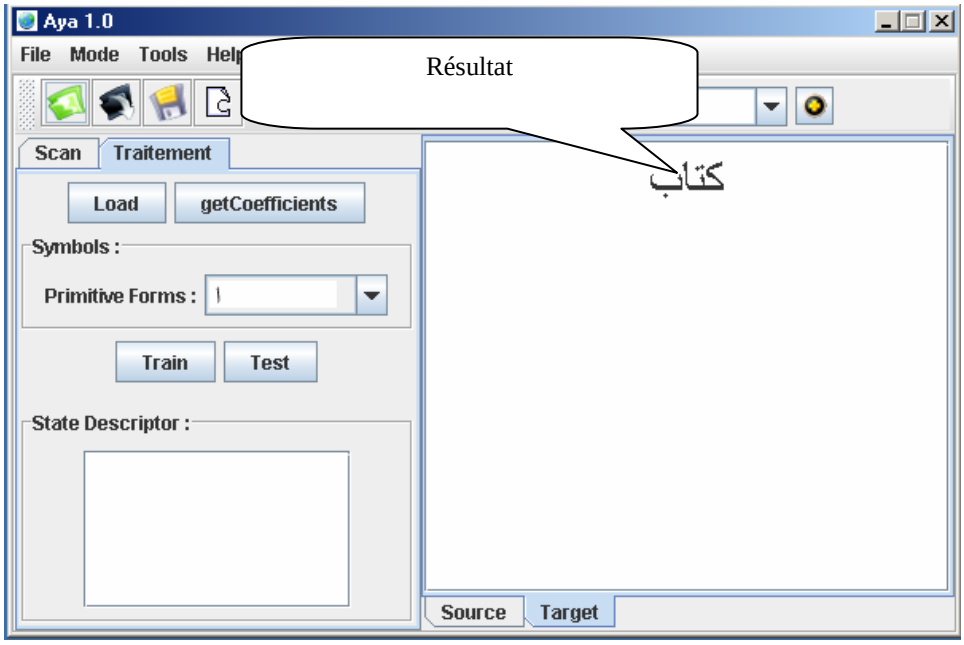


Figure 4.41: Illustration du résultat du traitement.

Dans ce qui est passé, nous avons présentés seulement les fenêtres principales, mais d'autres détailles comme le changement des filtres, les paramètres du classifieur et les formats d'export possible il faut consulter l'aide de l'application.

3.2.3 Test et Résultats

Suivant ce que nous avons vus (voir section 3.1.5) précédent, et vu les but (niveaux) de test, nous avons choisis quelques exemples de chaque classe de caractères: simples, composés et ambigus. Ces exemples sont scannés à l'aide d'un scanner EPSON CX3400 et avec une résolution entre 150 et 300 dpi. Les caractères (formes primitives) choisis sont au nombre de 08 et permet de générer jusqu'à 06 autres caractères composés.

Il est à remarquer que les caractères générés (composés) n'ont aucun effet sur le taux de reconnaissance parce qu'ils ne sont qu'une gestion de résultat de classification des formes primitives, située au niveau du code de l'application.

	Caractère	Nombre Exemples	Taux de reconnaissance
Caractères Simples	ـكـ	20	100%
	ء	20	91.66%
	.	24	100%
Caractères Ambigus	د	33	95%
	ع	21	100%
	ى	47	78.72%
	و	37	97.29%
	ح	29	96.47%
Caractères Composés	ج	20	90.32%
	ذ	33	93.10%
	غ	24	90.0%
	ي	16	87.93%
	ؤ	21	77.14%

Tableau 4.3: Résultats de test.

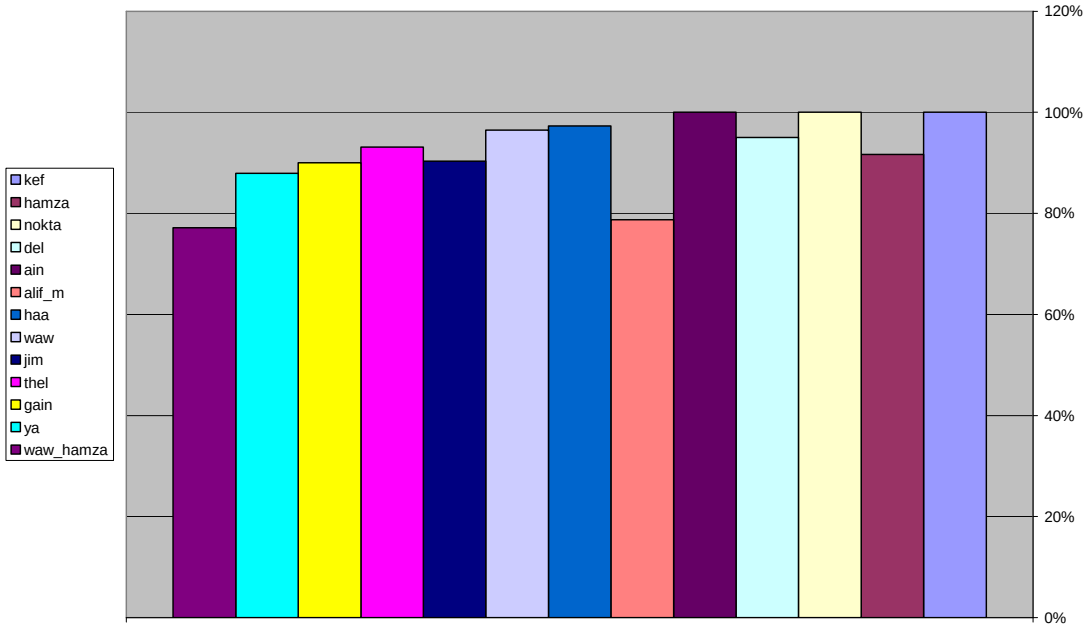


Figure 4.42: Histogramme des taux de reconnaissances des classes du Tableau 4.3.

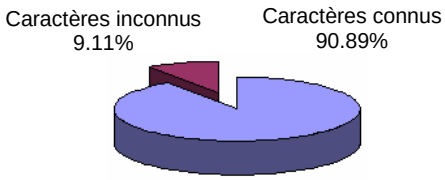


Figure 4.43: Taux de reconnaissance global.

3.2.4 Comparaison avec d'autres techniques d'AOCR

Les résultats de notre système sont comparé à d'autres systèmes de reconnaissance de caractère arabe hors-ligne. Somaya Alma'adeed et al ont proposé un système de reconnaissance qui se base sur les MMC, et qui a permis d'avoir un taux de reconnaissance de 45.0% sans post-traitement [77], qui est nettement très inférieur au taux de reconnaissance obtenus ici. Un autre système AOCR basé sur une compression ondelette, a donné pour un moyen de 80.0% au détriment du temps d'analyse [78].

Le système AHTR décrit en [80], basé en DWT a permis d'atteindre un taux de reconnaissance de l'ordre de 90.0%, avec un taux relativement faible pour les caractères au milieu. Ces résultats sont toujours au détriment de la vitesse du système due à l'utilisation des ondelettes.

On trouve en [81] un système appelé ASCA combiné avec un système déjà existant RECAM (ASCA-RECAM), basé sur une analyse morphologique du contour du mot. Les caractéristiques topologiques du texte sont exploitées pour extraire des règles morphologiques. Des résultats de segmentation sont présentés dans le tableau suivant:

Sous-segmentation	Sur-segmentation	Segmentation juste
5%	9%	86%

Tableau 4.4: Illustration des taux de segmentation du système ASCA-RECAM.

Le système ASCA-RECAM utilise une technique de segmentation qui suit presque le même principe que le notre; néanmoins, dans notre travail on cherche pas à faire une analyse morphologique du contour, mais seulement on fait une semi-squeletisaton ce qui permet d'avoir une accélération en temps d'exécution. En grosso modo, les résultats de segmentation obtenus par ASCA-RECAM sont proches de celles de notre système.

En résumé, notre système a pu atteindre un taux de reconnaissance élevé, sans avoir sacrifié beaucoup de la simplicité du système ou du temps d'exécution.

Conclusion

Nous avons présenté dans ce chapitre les différentes étapes qui peuvent conduire à une conception convenable d'un système de reconnaissance de caractères arabe manuscrit. Notre système multi fonte, mais il prend en charge la gestion des différents styles, nous avons choisis une méthode de suivi (tracking) pour l'extraction des caractéristiques et la méthode SVM pour la reconnaissance des différentes classes.

Les résultats de la classification par SVM obtenus peuvent être améliorés en optimisant les paramètres suivants :

- Les paramètres C et gamma ;
- Le nombre d'exemples ;
- …etc.

Comme solution au problème des classes qui ne sont pas facilement séparables, nous pouvons augmenter le nombre d'exemples.

Conclusion générale

Malgré les efforts et les travaux intensifs réalisés dans le domaine de la reconnaissance optique de l'écriture, aucun système OCR n'est jugé fiable à 100%. Mais au fur et à mesure les auteurs essayent d'améliorer les scores pour de meilleurs résultats.

Dans le cas de notre étude, nous avons présenté une méthode de segmentation structurelle qui a prouvé sa performance du point de vue taux de reconnaissance.

Cependant les problèmes majeurs influençant la recherche en AOCR sont:

- le manque de normalisation des calligraphies des caractères arabes
- l'absence d'études approfondie relative à la classification des fontes du point de vue calligraphie et corps,
- l'absence d'outils tels que les dictionnaires, bases de données et statistiques se rapportant à l'écriture arabe.

La résolution de ces problèmes serait d'un apport considérable, tant au niveau simplification de la tâche de l'AOCR, qu'aux nivaux validation et portabilité des produits réalisés.

Par ce travail nous espérons avoir couvert une grande partie concernant le domaine de recherche en segmentation des caractères arabes, et pouvoir contribuer à l'évolution des recherches, malgré que les efforts de nos jours s'intensifient dans ce domaine et chaque jour de nouveaux articles sont publiés, traitant du sujet.

Perspectives

Le travail que nous avons réalisé durant ce mémoire, constitue une étape et un premier apport pour la segmentation structurelle l'écriture manuscrite arabe, cependant nous pensons qu'il peut être amélioré, et étendu par les points suivants:

- Ajouter une technique de filtrage dynamique au lieu d'une méthode de seuillage fixe utilisée (seuillage adaptatif),
- Prendre en considération la gestion de l'écriture coupée afin d'arriver à un modèle de reconnaissance plus efficace (surtout pour l'écriture cursive).
- Les résultats obtenus dépendent de la lisibilité de l'écriture, pour des taux de reconnaissance plus représentant en mieux l'efficacité du système d'autres tests sont nécessaires.

Dans la réalité, il y'a deux types de caractéristiques:

- Caractéristiques de base ou générales,
- Caractéristiques liées à chaque écrivain.

Dans notre système nous avons utilisés seulement des caractéristiques de premier type, alors pour d'autres détails comme quel écrivain?, quel style?...etc. il faut ajouter d'autres caractéristiques de seconde type (gestion détaillée de l'épaisseur des segments du caractère).

Annexes

A & B

- **Organisation de la fiche d'apprentissage**
- **Tableaux de correspondance**

Introduction

Dans le but de simplifier la manière de rassembler les exemplaires des différents scripteurs, d'apprentissage et de test nous avons proposés une fiche d'apprentissage qui permet de bien représenter les informations nécessaire pour le test des différents composants du système Aya.

Cette fiche offre aussi la possibilité de tester l'effet des scripteurs sur le taux reconnaissance, ainsi que l'effet des différents styles, et ages des différents scripteurs. Dans ce qui suit nous allons présenter l'organisation de la fiche utilisée dans notre travail.

1. Niveaux de test

Depuis la figure 5.2 présentant l'archirecture détaillée du système, nous pouvons déduire qu'une fiche d'apprentissage doit permêttre de tester les trois composants suivantes:

- ***Le classifieur SVM* :** où nous devons avoir des exemples des formes primitives,
- ***Le gestionnaire de post-classification* :** où nous devons disposer de l'ensembles de caractères utilisés,
- ***Le gestionnaire de mise en page*** : où il faut avoir des mots, et texte pour tester l'extraction de ligne de base, et relations entre mots.

Par analogie à cette idée, nous avons proposés une fiche d'apprentissage qui permet de tester ces trois points ou niveaux de tests, comme le montre la figure suivante.

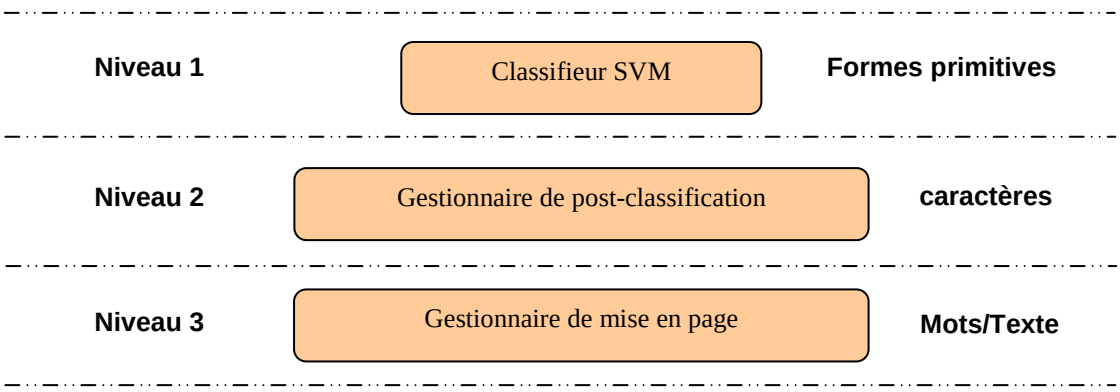


Figure A.1: Illustration des trois niveaux de tests.

2. Exemplaire d'une fiche d'apprentissage

La figure B-2 montre un exemplaire d'une fiche d'apprentissage qui respecte ces trois points de test:

Fiche d'apprentissage N:-	**و.تمرين رقم:-**
Age:-	**العمر:-**
Chiffre:-	**الأرقام:-**
	:0
	:1
	:2
	:3
	:4
	:5
	:6
	:7
	:8
	:9
Formes primitives:-	**الأشكال الأساسية:-**
	ا:
	ب:
	ح:
	د:
	ر:
	س:
	ص:
	ط:
	ع:

ف:

ق:

كـ:

ل:

م:

ن:

هـ:

و:

ى:

ـه:

ة:

.:

ء:

Lettres / Mots:- **الحروف/الكلمات:-**

ا:

ب:

ح:

د:

ر:

س:

ص:

ط:

ع:

ق:

كـــ:

ل:

م:

ن:

هـــ:

و:

ى:

ـــه:

ه:

:.

_____

:ء

Texte:- نص:-

Num	Forme	Code HTML
0000	ـكـ	223
0001	ـلـ	225
0002	ـمـ	227
Caractères Simples		
0003	ا	199
0004	ـحـ	205
0005	د	207
0006	ر	209
0007	ـسـ	211
0008	ـصـ	213
0009	ـطـ	216
0010	ـعـ	218
0011	ى	236
0012	و	230
0013	ه	229
Caractères Ambigus		
0014	ـفـ	/
0015	ـقـ	/
0016	ـكـ	/
0017	ـنـ	/
0018	ـيـ	/
0019	ء	193
Formes de Caractères Composés		
20	.	/
21	~	/
Caractères de gestion		

Tableau B.1 : Code de formes primitives.

Num	Caractère	Code HTML
0000	ـكـ	223
0001	ـلـ	225
0002	ـمـ	227
Caractères Simples		
0003	ا	199
0004	ـحـ	205
0005	د	207
0006	ر	209
0007	ـسـ	211
0008	ـصـ	213
0009	ـطـ	216
0010	ـعـ	218
0011	ى	236
0012	و	230
0013	ه	229
Caractères Ambigus		
0014	ـبـ	200
0015	ـتـ	202
0016	ـثـ	203
0017	ـجـ	204
0018	ـخـ	206
0019	ـذ	208
0020	ز	210
0021	ـشـ	212
0022	ـضـ	214
0023	ـظـ	217
0024	ـغـ	219
0025	ـفـ	221
0026	ـقـ	222
0027	ـنـ	228
0028	ـيـ	237
0029	ؤ	196
0030	أ	195
0031	إ	197
0032	آ	194
0033	ئ	198
0034	ك	223
0035	ة	230
0036	ء	193
Caractères Composés		

Tableau B.2: Code de caractères en HTML.

Remarque :

les caractères simples et les caractères de rejet garde la même entrée de la table des formes primitives, et par conséquence les mêmes codes de caractères en sorties .

Code d'entrée	Forme	Codes possible	Formes de composition	Code de sortie
0003	ا	0019	أ	0034
			إ	0035
		0021	آ	0036
0004	ح	0020	ج	0017
			خ	0018
0005	د	0020	ذ	0019
0006	ر	0020	ز	0020
0007	س	0020	ش	0021
0008	ص	0020	ض	0022
0009	ط	0020	ظ	0023
0010	ع	0020	غ	0028
0011	ى	0020	ي	0032
		0019	ئ	0037
0012	و	0019	ؤ	0033
0013	ه	0020	ة	0035
0014	ف	0020	ف	0025
0015	ق	0020	ق	0026
0016	ك	0019	ك	0034
0017	ن	0020	ن	0027
0018	ب	0020	ب	0014
			ت	0015
			ث	0016

Tableau B.3: Illustration de possibilités de composition.

Bibliographie

[1] S. Haitaamar :" *segmentation de texte en caractère pour le reconnaissance optique de l'écriture arabe*". Université EL-HADJ LAKHDHAR Batna, Juillet 2007.

[2] P. Smrž et al :" *Off-line Recognition of Cursive Handwritten Czech text*". Université de Masaryk, Février 1998.

[3] Z. Lu et al :"*A Robust, Language-Independent OCR System*", BBN Technologies, GTE Internetworking, Cambridge, MA 02138, Décembre 1998.

[4] E.Kavallieratou et al :" *Slant estimation algorithm for OCR* Systems ". Wire Communications Laboratory, University of Patras, 26500 Patras, Greece, Novembre 2001.

[5] N. Ben Amara et al :" *Utilisation des modèles markoviens en reconnaissance de l'écriture arabe : Etat de l'art*". Ecole Nationale d'Ingénieurs de Monastir - 5019 Monastir – TUNISIE, LORIA-CNRS, Tunisie, Avril 2001.

[6] S. Chevalier et al : "*Étude de primitives spectrales pour la reconnaissance de caractères manuscrits dans le cadre d'une approche markovienne 2D*". DGA/Centre d'Expertise Parisien, France, Novembre 2005.

[7] A. Madaan, R. Mehta :"*Comparative Evaluation of Learning Algorithms on Hand written Character Data*". Cours CS 698, Mars 2004.

[8] L. S. Oliveira, E. Lethelier, F. Bortolozzi et R. Sabourin :" *Segmentation de caractères manuscrits basée sur une approche structurelle* ". Ecole de Technologie Supérieure, Laboratoire d'Imagerie, de Vision et d'Intelligence Artificielle 1100, QUEBEC, Mai 2000.

[9] A.Bennasri, A.Zahour, B. Taconet :" *Extraction des lignes d'un texte manuscrit* arabe".Vision Interface '99, Trois-Rivières, Canada, 19-21 Mai 1999.

[10] F. Bougamouza, S. Hazmoune, Benmohammed M :" Nouveaux prétraitements et extraction des caractéristiques pour la reconnaissance de mots manuscrits arabes ". IMAGE'2009,5ème Symposium iternational, Biskra, Algérie, 03-05 Novembre 2009.

[11] A. Boutarfa :"*Reconnaissance de formes 3D par approche neuronale associant la transformée de Hough en robotique mobile: application à la productique*". Thèse de doctorat Es-Sciences en électronique industrielle, Université de Batna, 2006.

[12] J. Park :" *Hierarchical character recognition and it's use in handwritten word/phrase recognition*". Thèse de phd, Université de New York, Novembre 1999.

[13] M. Li :" *Sequence and Text Classification: Features and Classifiers*". Thèse de phd, Ecole de sciences d'informatique, Université d'est d'Anglia, Norwich NR4 7TJ, Juillet 2006.

[14] T. Paquet, L. Heutte, Y. Lecourtier :"*Problématique de la Reconnaissance de l'Écriture*". ASTI'2001 des Sons, des Images et des Documents à leur Interprétation, France, 2001.

[15] L. Likforman-Sulem, A. Zahou, B. Taconet :" *Text Line Segmentation of Historical BarraultDocuments: a Survey*". submitted to Special Issue on Analysis of Historical Documents, International Journal on Document Analysis and Recognition, Springer, France, 2006.

[16] C. Bahlmann, B. Haasdonk, H. Burkhardt :" *On-line Handwriting Recognition with Support Vector Machines—A Kernel Approach*". publ. in Proc. of the 8th Int. Workshop on Frontiers in Handwriting Recognition (IWFHR), pp. 49–54, Germany, 2002.

[17] V. Vapnik :"*Universal Learning Technology : Support Vector Machines*". NEC Journal of Advanced Technology, Vol 2, No 2.

[18] M. Marchand :" *Les SVMs (Support Vector Machines)*". IFT-65764 (H-2006), 2006.

[19] Chih-Jen Lin :" *Support Vector Machines for Data Classification* ". National Taiwan University, February 9, 2004.

[20] J. Callut :" *Implémentation efficace des Support Vector Machines pour la classification*". Mémoire présentée en vue de l'obtentiondu grade de Maître en Informatique, Université Libre de Bruxelles, 2002-2003.

[21] Steve R. Gunn :" *Support Vector Machines for Classification and Regression* ". University of Southamptan, May 1998.

[22] F. Markowetz :" *Classification by Support Vector Machines* ". Max-Planck-Institute for Molecular Genetics - Computational Molecular Biology –Berlin, 2003.

[23] Marti A. Hearst :"*Support Vector Machines*".IEEE Intelligent Systems,University of California, Berkeley, 1998.

[24] Shigeo Abe :" *Support Vector Machines for Pattern Classification* ". Springer, 2005.

[25] N. Cristianini, J. Shawe-Taylor :"*An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*". Cambridge University Press, 2000.

[26] M. Jardino :" *Fouille de données dans les corpus de textes Classification supervisée : SVM* ". Groupe Langues, Information et Représentations.

[27] Mohamadally Hasan,Fomani Boris :" *SVM machine à vecteurs de support ou séparateur à vaste marge* ". BD Web, ISTY3,Versailles St Quentin, France, janvier 2006.

[28] Thanh-Nghi Do, F. Poulet :" *Interprétation des résultats de SVM*". ESIEA Recherche 38, Universitaire de Laval-Changé, Septembre 2008.

[29] A Rakotomamonjy, F. Suard :" *Sélection de variables par SVM: application à la détection de piétons* ". Laboratoire Perception, Systèmes et Informations, FRE CNRS 2645 ,INSA de Rouen, France, 2004.

[30] G. Loosli et al :" *Une boîte à outils rapide et simple pour les SVM** ". CAP, RSISE&NICTA à Canberra, Australie, Novembre 2004.

[31] A. Cornuéjols :" *Les SVM : Séparateurs à Vastes Marges* ". IIE & CNRS - Université de Paris-Sud, Orsay, France.

[32] E. Viennet :" *Reconnaissance des formes et Machines à Vecteurs de Support*". Laboratoire d'Informatique de Paris Nord, CNRS URA, Université Paris XIII, 93430 Villetaneuse France.

[33] M. Christophe :" *SVM Support Vector Machine : une méthode de classification binaire par apprentissage* ". LIC2M / Support Vector Machine, Août 2004.

[34] P. Mahé :" *Noyaux pour graphes et Support Vector Machines pour le criblage virtuel de molécules* ". Rapport de stage, DEA MVA 2002/2003,Septembre 2003

[35] M. Pom H. Wu :" *Handwritten Character Recognition* ".Queensland University, October 2003.

[36] Ba-Quy Vuong a, Siu-Cheung Hui a, Yulan He :" *Progressive structural analysis for dynamic recognition of On-line handwritten mathematical expressions* ". Computer Sciences Department, University of Wisconsin-Madison, Decembre 2007.

[37] A. Dhawan, A. R. Ganesan :" *Handwritten Signature Verification* ". ECE 533 – Project Report, Wiconsin Madison University.

[38] M. Eldawy :" *A Survey of Classification techniques* ". Presentation, May 2006.

[39] O. Bousquet :" *Introduction aux ' Support Vector Machines '(SVM)* ". Centre de Mathématiques Appliquées Ecole Polytechnique, Palaiseau, Orsay, Novembre 2001.

[40] Chih-Chung Chang et Chih-Jen Lin :" *LIBSVM: a Library for Support Vector Machines* ". Technical_Report, Septembre 2006.

[41] R. Nilsson, J. Bjrkegrenm J.Tegné :" *A Flexible Implementation for Support Vector Machines* ". The Mathematica Journal 10:1 © 2006 Wolfram Media, Inc.

[42] P. Vincent :" *Modèles à noyaux à structure locale* ". Thèse présentée à la Faculté des études supérieures en vue de l'obtention du grade de Philosophiæ Doctor (Ph.D.) en informatique, Université de Montréal, Octobre 2003.

[43] A. Karatzoglou, D. Meyer, K. Hornik :" *Support Vector Machines in R* ". Journal of Statistical Software Issue 9, Volume 15, April 2006.

[44] P. Gallinari, H. Zaragoza, M. Amini :" Apprentissage et données textuelles ". LIP6, Université Paris 6, 4 Place Jussieu, 75252 Paris cedex 05, France.

[45] V. Guigue, A. Rakotomamonjy, S. Canu :" SVM et k-ppv pour la reconnaissance d'émotions ". PSI - CNRS FRE 2645 - INSA de Rouen, France, Juin 2003.

[46] G. Lebrun, C. Charrier, O. Lezoray :" *Réduction du temps d'apprentissage des SVM par Quantification Vectorielle* ". LUSAC EA 2607, groupe Vision et Analyse d'Image, 120 Rue de l'exode, F-50000 Saint-Lô, France.

[47] J. Mary :" *Méthodes d'Apprentissage Avancées, SVM* ". equipe TAO, LRI, Janvier 2006.

[48] P. Mahé, L. Ait-Ali :" *Projet d'apprentissage statistique SVM pour l'apprentissage non supervisé*". DEA MVA, Février 2003.

[49] A. Cornuéjols :" *Une nouvelle méthode d'apprentissage : Les SVM. Séparateurs à vaste marge*". Université de Paris-Sud, Orsay, France, Juin 2002.

[50] Christopher J. C. Burges :" *A Tutorial on Support Vector Machines for Pattern Recognition* ". Bell Laboratories, Lucent Technologies, 1998.

[51] Pai-Hsuen Chen, Chih-Jen Lin, et Bernhard Schõlkopf :" *A Tutorial on v-Support Vector Machines* ". Department of Computer Science and Information Engineering National Taiwan University Taipei 106, Taiwan, Max Planck Institute for Biological Cybernetics, Tübingen, Germany

[52] S. Carbonnel E. Anquetil :" *Modélisation et intégration de connaissances lexicales pour le post-traitement de l'écriture manuscrite en-ligne*". IRISA, INSA, France, Mars 2009.

[53] A. Nosary, L. Heutte, T. Paquet :" *Modélisation et intégration de connaissances lexicales pour le post-traitement de l'écriture manuscrite en-ligne*". Laboratoire Perception Systèmes Information, UFR des Sciences, Université de Rouen, France, 2002.

[54] M. Crucianu et al :" *Détection et usage des composantes typographiques et graphiques dans les pages de documents* ". Laboratoire d'Informatique – Université de Tours, France, Octobre 2002.

[55] H. Oulhadj, J. Lemoine, E. Petit, H. Wehbi :" *Combinaison d'algorithmes pour la reconnaissance des chiffres et des lettres batons dans un environnement multiscripteur d'ecriture courante mixte* ". Laboratoire d'Etude et de Recherche en Instrumentation, Signaux et Systèmes , Université Paris XII, France, May 1999.

[56] S. Quiniou :" *Intégration de connaissances linguistiques pour la reconnaissance de textes manuscrits en-ligne* ". Thèse de docteur de l'INSA de Rennes mention Informatique, IMADOC – IRISA, MATISSE, France, 17 décembre 2007.

[57] J.Y. Ramel, S. Leriche :" *Segmentation et analyse interactives de documents anciens imprimés* ". Laboratoire *d'Informatique, Ecole Polytechnique de l'Université de Tours, 64, avenue Jean Portalis 37200 Tours, France.*

[58] Y. Rangoni, A. Belaïd :" *Data categorization for a context return applied to logical document structure recognition* ". Loria Research Center - Read Groupm, Vandoeuvre-l`es-Nancy, Francem 2005.

[59] L. Robadey :" *2(CREM): Une méthode de reconnaissance structurelle de documents complexes basée sur des patterns bidimensionnels* ". Thèse de doctorat soumise à la Faculté des Sciences de l'Université de Fribourg, Suisse, 2001.

[60] A. Soudi, et al :" *Arabic Computational Morphologyknowledge – based and Empirical Methods* ". Springer, Volume 38, 2007.

[61] P. Atzeni, F. Trimoldi, S. Muzzarelli :" *Prototipo per l'acquisizione e la manipolazione avanzata delle immagini* ". Università degli Studi "Roma Tre" Dipartimento di Informatica ed Automazione Facolta di Ingegneria, Corso di Laurea in Ingengeria Informatica, Italia, 2005.

[62] TWAIN Working Group Committee:" *TWAIN Specification, Version 1.9* ". TWAIN Working Group Committee, January 20, 2000.

[63] G. Almouzni :" *Traitement numérique des images* ". EISTI, 2007.

[64] A. Boukharouba , A. Bennia :" *Reconnaissance de Caractères Imprimés Omni-fonte* ". 3rd International Conference: Sciences of Electronic, Technologies of Information and Telecommunications, Tunisia, March 27-31, 2005.

[65] F. B. Samoud, S. S. Maddouri, K. Hamrouni :" *Segmentation de chèques bancaires arabes* ". 3rd International Conference: Sciences of Electronic, Technologies of Information and Telecommunications, Tunisia, March 27-31, 2005.

[66] S. Adam, et al :" *Utilisation de la transformée de Fourier-Mellin pour la reconnaissance de formes multi-orientées et multi-échelles : application à l'analyse automatique de documents techniques* ". Laboratoire PSI, La3I, Université de Rouen ,France, 2001.

[67] J. Y. Ramel :" *Lecture automatique des partitions musicales* ". Mémoire de DEA ingénierie informatique, LISPI - Equipe de Reconnaissance des Formes et Diagnostics ,Université Lyon 1, France, 1993.

[68] F. S. Douzidia :" *Résumé automatique de texte arabe* ".
Mémoire de M.Sc en informatique, Université de Montréal, Canada, Septembre 2004.

[69] T. Lelore, J. Martinez :" *Segmentation d'image Application aux documents anciens* ".
Mémoire de Master de recherche, Université de Nante, France, Mai 2007.

[70] F. Menasri :" *Segmentation d'image Application aux documents anciens* ".
Thèse Docteur de l'Université Paris Descartes en Informatique, France, Juin 2008.

[71] L. Najman :" *Morphologie mathématique, systèmes dynamiques et applications au traitement des images* ".
Mémoire d'Habilitation à Diriger des Recherches en Informatique, Université de Marne-la-Vallée France, Mai 2006.

[72] A. Zidouri, M. Sarfraz :" *On optical character recognition of Arabic text* ".
The 6th Saudi Engineering Conference, Volume 4, KFUPM, Dhahran, December 2002.

[73] J. Y. Toumit, S. G. Salicetti, H. Emptoz :" *Caractérisation d'objets mathématiques et redondance graphique pour la lecture automatique de documents mathématiques* ".
Vision Interface '99, Trois-Rivières, Canada, 19-21 May.

[74] A. Belaïd :" *Reconnaissance automatique de l'écriture et du document* ".
LORIA-CNRS, Campus scientifique B.P. 239, 54506 Vandoeuvre-Lès-nancy, France.

[75] G. Stamon, N. Vincent :" *Bio-Imagery – Image Analysis Document Analysis – Pattern Recognition* ".
Rapport d'activité SIP, Laboratoire CRIP5, Université Paris Descartes, France, Avril 2008.

[76] M. Côté :" *Utilisation d'un modèle d'accès lexical et de concepts perceptifs pour la reconnaissance d'images de mots cursifs* ". Thèse de Docteur de l'école Nationale Supérieur de Télecommunications, France, Juin 1997.

[77] S. Alma'adeed, C. Higgens, D. Elliman :" *Recognition of Off-Line Handwritten Arabic Words Using Hidden Markov Model Approach* ". ICPR, University of Nottingham, 2002.

[78] A. Aburas, S. M. A. Rehiel :" *Off-line Omni-style Handwriting Arabic Character Recognition System Based on Wavelet Compression* ". International Islamic University Malaysia, Electrical and Computer Engineering, Malaysia, 2007.

[79] N. Kharma, R. Ward :" *A Novel Invariant Mapping Applied to Hand-written Arabic Character Recognition* ".
E.C.E. Department, (#289) 2366 Main Mall, University of British Columbia, Vancouver BC, Canada. V6T 1Z4.

[80] I. A. Jannoud :" *Automatic Arabic Hand Written Text Recognition System* ". Damascus University, Damascus, Syria and Al-zaytoonah University, Amman, Jordan, American Journal of Applied Sciences 4 (11): 857-864, 2007.

[81] T. Sari, L. Souici, M. Sellami :" *Off-line Handwritten Arabic Character Segmentation Algorithm: ACSA* ".
Laboratoire LRI, Université Badji Mokhtar – Annaba, Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition, Algérie, 2002.

[82] Y. Al-Ohali, M. Cheriet, C. Suen :" *Databases for recognition of handwritten arabic cheques* ". In: L.R.B. Schomaker and L.G. Vuurpijl (Eds.), Proceedings of the Seventh International Workshop on Frontiers in Handwriting Recognition, Amsterdam, September 2000.

[83] P. Dreuw, S. Jonas, H Ney :" *White-Space Models for Offline Arabic Handwriting Recognition* ". Human Language Technology and Pattern Recognition, RWTH Aachen University, 2008.

[84] A. Rachidi , M. El Yassa, D. Mammass :" *A Pretopological approach for handwritten isolated Arabic characters recognition* ". IRF – SIC, Ecole Nationale de Commerce et de Gestion, IRF-SIC, Faculty of Sciences, Ibn Zohr University Agadir, Morocco.

[84] A. Rachidi , M. El Yassa, D. Mammass :" *A Pretopological approach for handwritten isolated Arabic characters recognition* ". IRF – SIC, Ecole Nationale de Commerce et de Gestion, IRF-SIC, Faculty of Sciences, Ibn Zohr University Agadir, Morocco.

[85] N. M. Wanas, M. R. El-Sakka, M. S. Kamel :" *Multiple Classifier Hierarchical Architecture for Handwritten Arabic Character Recognition* ". Pattern Analysis and Machine Intelligence Laboratory, Department of Systems Design Engineering, University of Waterloo, Waterloo, Ontario Canada N2L-3G1, 1999.

[86] A. J. Alnsour, L. M. Alzoubady :" *Arabic Handwritten Characters Recognized by Neocognitron Artificial Neural Network* ". University of Sharjah Journal of Pure & Applied Sciences Volume 3, No. 2, June 2006.

[87] M. Z. Khedher, G. A. Abandah, A. M. Al-Khawaldeh :" *Optimizing Feature Selection for Recognizing Handwritten Arabic Characters* ". World Academy of Science, Engineering and Technology 4 2005.

[88] T. Klassen:" *Towards Neural Network Recognition Of Handwritten Arabic Letters* ". A Project Submitted to the Faculty of Computer Science In Partial Fulfillment of the Requirements For the Degree of MASTER OF COMPUTER SCIENCE (M.C.Sc.), Dalhousie University, 2001.

[89] Gherghout Y., Souici-Meslati L:" *Reconnaissance de Caractères Arabes Manuscrits par les Séparateurs à Vastes Marges (SVM)* ". ICAI'09,International Conference on Applied Informatic,Traitemet d'image, Bordj Bou Arréridj, Algérie,15-17 Novembre 2009.

[90] Al-Rashaideh H:" *Preprocessing phase for Arabic Word Handwritten Recognition* ". Institut d'informatique et autimatisme, Tom 6, N^0 1,2006, cmp.11-19, Russie, February 26, 2006.

[91] Shubair A et al.:" *Off-line Arabic handwritten word segmentation using rotational invariant segments features* ". The international Arab journal of information technology, Vol. 5, No. 2, April 2008.

[92] Jawad H. et al.:" *Component-based Segmentation of Words from Handwritten Arabic Text* ". International Journal of Computer Systems Science and Engineering 5:1 2009.

[93] M. Volker, E. Haikal, M. Pechwitz :" *Offline Handwritten Arabic Word Recognition Using HMM - a Character Based Approach without Explicit Segmentation* ". Technical University of Braunschweig Institut for Communications Technology (IfN), Germany, Mars 2008.

[94] S. Mozaffari et al.:" *IfN/Farsi-Database: A Database of Farsi Handwritten City Names* ". ICFHR, Iran, 2008.

[95] M. Volker, E. Haikal, M. Pechwitz:" *Comparison of Two Different Feature Sets for Offline Recognition of Handwritten Arabic Words* ". Technical University of Braunschweig Institut for Communications Technology (IfN) , Germany, Juin 2006.

[96] A.Gheith, N. Anssari:" *Novel Moment Features Extraction for Recognizing Handwritten Arabic Letters* ". Journal of Computer Science 5 (3): 226-232, 2009.

[97] J. André:" *Courrier: Histoire d'un caractère de la machine à écrire aux fontes numériques* ". Version fac-similée, France, Janvier 2010.