Approche markovienne bidimensionnelle d'analyse et de reconnaissance de documents manuscrits

Melanie Lemaitre

Université Paris 5 René Descartes
UFR de Mathématiques et Informatique

THÈSE DE DOCTORAT

présentée en vue de l'obtention du grade de
Docteur de l'Université de Paris 5

Discipline : Informatique

Approche markovienne bidimensionnelle d'analyse et de reconnaissance de documents manuscrits

Mélanie Lemaitre

Présentée le 16 novembre 2007 devant le jury composé de :

Edouard Geoffrois	Examinateur
Laurent Heutte	Rapporteur
Laurence Likforman	Examinateur
Françoise Prêteux	Directrice de thèse
Georges Stamon	Examinateur
Karl Tombre	Rapporteur
Nicole Vincent	Présidente du jury

Remerciements

En premier lieu je souhaiterais remercier ma directrice de thèse Françoise Prêteux pour m'avoir permis de conduire ces travaux, pour ses encouragements et pour ses conseils toujours très précieux pour la rédaction des différents écrits et la préparation de la soutenance de thèse. Un grand merci également à Edouard Geoffrois, encadrant au quotidient à la DGA, pour sa confiance, sa bienveillance et son enthousiasme. Merci en particulier de m'avoir sensibilisée à l'importance des campagnes d'évaluation et de m'avoir permis de participer au projet RIMES, première campagne d'évaluation dans le domaine de la reconnaissance de l'écriture manuscrite en France.

Je tiens ensuite à adresser mes plus sincères remerciements à Emmanuèle Grosicki qui a joué un rôle très important dans cette thèse. Qu'elle soit assurée de ma reconnaissance pour son soutien technique mais surtout humain tout au long de ces trois années.

Je remercie chaleureusement Karl Tombre et Laurent Heutte pour avoir accepté le difficile rôle de rapporteur de ces travaux ansi que Nicole Vincent, Laurence Likforman et Georges Stamon d'avoir accepté de prendre part à mon jury.

Je tiens ensuite à remercier l'ensemble de mes collègues du CEP de la DGA pour leur sympathie et bonne humeur.

Merci à mes amis et à ma belle-famille pour leurs encouragements.

Merci à mes parents et à ma soeur pour leur soutien et pour leur aide dans la préparation du pot de thèse.

Enfin merci à Yann pour m'avoir supportée dans tous les sens du terme tout au long de cette aventure que représente la thèse...

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Depuis son invention il y a plus de 5300 ans, l'écriture reste un moyen de communication privilégié entre les êtres humains. Bien que l'imprimerie créée il y a plus de 550 ans puis l'informatique aient permis son automatisation, l'écriture manuscrite est loin d'avoir disparu de notre société et les individus émettent et reçoivent une grande quantité de documents manuscrits. Le traitement de masse de ces documents apparaît alors incontournable. C'est ainsi que les recherches concernant la lecture automatique des documents manuscrits ont débuté il y a plus de 55 ans. Les premiers OCR (Optical Character Recognition) ont fait leur apparition dans les années 1950 et les premiers systèmes de lecture d'adresses postales utilisés pour le tri du courrier sont apparus en 1965. La lecture automatique de l'écriture manuscrite dans les formulaires a fait son apparition dans les années 1980 grâce au développement de moyens de calculs toujours plus puissants.

Malgré les efforts et progrès réalisés dans le domaine, on est encore loin du rêve d'un monde "sans papier". Hormis pour des applications très précises telles que la lecture automatique de montants de chèques bancaires, d'adresses postales ou de formulaires, la reconnaissance de l'écriture manuscrite reste un problème complexe et non résolu dans des cas de reconnaissance de documents manuscrits moins contraints. De la reconnaissance de quelques caractères, mots, signatures ou adresses, on envisage maintenant celle de documents de plus grande envergure.

La lecture automatique des documents manuscrits est actuellement un sujet de recherche très actif et a pour objectif principal de "convertir les images de documents manuscrits en fichiers informatiques en vue de l'archivage, la recherche, la modification, la réutilisation et la transmission de l'information que ces images contiennent" [91]. Les applications sont aussi nombreuses que variées : tri automatique de courrier, transcription de documents manuscrits anciens, lecture automatique et tri de formulaires, mise au propre de brouillon ou d'agenda, extraction d'informations utiles d'un document comme par exemple les numéros de téléphone dans un courrier manuscrit, indexation de documents, routage automatique de fax, etc.

Le sujet traité dans cette thèse concerne deux phases essentielles de l'analyse de documents : la reconnaissance de l'écriture manuscrite et la structuration de documents. La reconnaissance de l'écriture manuscrite, étape clé de la lecture automatique de document, est au cœur de nos recherches : nous avons ainsi abordé dans un premier temps la reconnaissance de caractères puis celle de mots isolés. Nous avons ensuite étudié la structuration de documents manuscrits non contraints. Nous nous sommes interessée en

particulier au cas des courriers manuscrits destinés à des entreprises pour lequel l'objectif est de localiser des champs utiles à leur traitement automatique (coordonnées de l'expéditeur ou l'objet de la lettre par exemple).

Pour traiter ces deux problématiques, nous préconisons une approche s'appuyant sur les modèles markoviens connus pour leur capacité d'intégration du contexte et d'absorption du bruit.

Fort de leurs succès en traitement de la parole, les HMM (Hidden Markov Models) ont été utilisés dans bon nombre de problèmes de reconnaissance de formes et plus particulièrement en reconnaissance de l'écriture manuscrite. Du fait de la complexité des algorithmes de décodage, la plupart des approches proposées étaient 1D ou pseudo 2D. L'apparition de la programmation dynamique 2D en 1998 [33] a permis d'ouvrir la voie à la modélisation purement bidimensionnelle de l'écriture manuscrite. En effet, en 2003 Chevalier *et al.* ont proposé une approche bidimensionnelle markovienne basée sur la programmation dynamique 2D et les primitives spectrales locales pour la reconnaissance de l'écriture manuscrite [17]. Si cette méthode a été appliquée avec succès à une tâche de reconnaissance de chiffres manuscrits de la base MNIST, elle est encore à l'état d'ébauche pour la reconnaissance de mots (un taux de reconnaissance de 40% avait été obtenu en ne traitant que les mots de taille inférieure ou égale à 4 lettres) [16]. La méthode mise en œuvre par Chevalier apparaissait alors comme exploitable en même temps qu'améliorable.

Repartant de cette approche, un premier apport de nos travaux a donc été de proposer une approche générale de reconnaissance et de segmentation d'images appliquée à la reconnaissance de l'écriture manuscrite et à la structuration de documents. L'approche générale ainsi définie est appellée AMBRES ou "Approche Markovienne Bidimensionnelle pour la Reconnaissance Et la Segmentation d'images".

L'amélioration de l'approche initiale de reconnaissance de caractères manuscrits a permis d'apporter des contributions dont les points forts sont : l'étude approfondie des primitives spectrales locales, l'étude du système et de ses performances, la combinaison originale de systèmes obtenus avec différents vecteurs de primitives spectrales locales et l'approche de division et fusion d'états pour l'optimisation de la topologie du champ de Markov.

L'amélioration drastique des performances obtenues en reconnaissance de mots constitue un autre apport significatif de cette thèse. Cela a notamment été rendu possible grâce à la proposition d'un nouveau mode de segmentation dépendant du type de caractère considéré.

Enfin, AMBRES a permis d'aborder avec succès la structuration de documents manuscrits. Un système complet de structuration de courrier manuscrit a été proposé.

Pour l'évaluation de nos algorithmes, nous nous sommes attachée à utiliser des bases de données publiques pour comparer les performances à celles d'autres méthodes. Même si les bases utilisées dans nos travaux sont assez classiques dans le domaine (bases MNIST et Senior & Robinson), la comparaison des résultats n'est pas toujours évidente. En effet, la comparaison objective de deux résultats suppose des protocoles d'évaluation identiques. Ainsi s'impose-t-il la nécessité de participer à des campagnes d'évaluation

sur des bases de données communes. En 2004, les ministères français en charge de la recherche et de la défense ont lancé le programme Techno-Vision destiné à créer une dynamique de l'évaluation dans le domaine du traitement de l'image. Le projet RIMES (Reconnaissance et Indexation de données Manuscrites et de fac-similES) s'inscrit dans ce cadre et vise à évaluer des algorithmes de reconnaissance et d'indexation des courriers manuscrits, notamment en créant une base de données de grande ampleur. La participation à ce projet nous a ainsi permis de tester nos algorithmes sur une nouvelle base de données et de comparer nos résultats avec ceux obtenus par d'autres méthodes.

Cette thèse se décompose en 6 chapitres.

Dans le premier chapitre nous présentons un état de l'art afin de situer le contexte de nos travaux concernant la reconnaissance de l'écriture manuscrite. Nous proposons un bref panorama des techniques classiques d'extraction de primitives et de reconnaissance de l'écriture manuscrite. Les approches markoviennes, cœur de AMBRES, sont analysées en détail et leur évolution de la modélisation monodimensionnelle à la modélisation véritablement bidimensionnelle est mise en relief.

Le chapitre 2 est consacré à la présentation de l'approche AMBRES fondée sur les champs de Markov, outils statistiques très puissants pour une modélisation bidimensionnelle des images. Après en avoir rappelé le cadre théorique, nous détaillons l'approche AMBRES. Nous montrons comment résoudre les deux problématiques que sont la reconnaissance et la segmentation d'images en créant pour chaque classe d'image un modèle markovien reliant des états cachés à des observations extraites des images. Dans le cadre de la reconnaissance d'images, ces modèles serviront à déterminer la classe la plus probable et dans le cadre de la segmentation d'images, ils serviront à déterminer la configuration optimale des étiquettes recherchées. Les paramètres du modèle markovien sont explicités, des densités d'observations classiques au modèle original de position en passant par le modèle de transition. Enfin, le choix des observations dans le cadre de AMBRES pour la modélisation de l'écriture manuscrite et pour la structuration de documents est décrit. Les primitives spectrales locales utilisées pour extraire une information de direction des traits sont en particulier étudiées.

Le chapitre 3 est dédié à la reconnaissance de caractères manuscrits. Le système fondé sur l'approche AMBRES est détaillé. Une étude des paramètres du modèle est réalisée à partir de la base publique MNIST de chiffres manuscrits. En particulier, le choix des coefficients pour le vecteur de primitives spectrales locales est analysé. Une étude des erreurs et des performances du système conduit à envisager de nouvelles stratégie d'amélioration : la combinaison de systèmes s'avère très performante et la stratégie de division et fusion d'états apparaît comme une perspective à explorer pour l'optimisation de la topologie du champ de Markov. Enfin, la reconnaissance de lettres majuscules manuscrites est abordée afin de montrer la généralisation de AMBRES à tout type de caractères manuscrits.

Le chapitre 4 s'intéresse ensuite à la reconnaissance de mots avec un grand vocabu-

laire. L'approche retenue consiste à construire des modèles de mots par concaténation de modèles de lettres. Pour cela, nous nous appuyons sur la méthode de reconnaissance de caractères décrite précédemment. Ce chapitre étudie ainsi le passage du système de reconnaissance de caractères à un système de reconnaissance de mots. Toute la difficulté de cette démarche consiste dans un premier temps à obtenir suffisamment d'images de lettres pour construire des modèles de lettres précis. En effet, on dispose de très peu de mots de une lettre et on va donc devoir les extraire des imagettes de mots. Effectuer un découpage manuel des imagettes de mots s'avérerait fastidieux et présenterait donc peu d'intérêt : un découpage automatique est donc envisagé et mis en oeuvre. La seconde grande difficulté est de réaliser correctement la concaténation des modèles de lettres pour obtenir des modèles de mots. Le chapitre 4 décrit ainsi l'extension à la reconnaissance de mots en détaillant en particulier les deux problématiques citées ci-dessus. Le système est testé sur la base de données publique Senior et Robinson.

Dans le chapitre 5, nous nous plaçons au niveau du document entier et abordons la structuration de document manuscrit, c'est-à-dire que l'on cherche à extraire sa structure physique. Le système fondé sur AMBRES est appliqué au cas des courriers manuscrits de la base RIMES. Après le réglage des différents paramètres, une étude des erreurs est menée. Ainsi, après avoir étudié la possibilité d'augmenter le nombre d'états du modèle, une phase de post-traitement est introduite. Celle-ci se décompose en plusieurs étapes correspondant chacune à un type d'erreur en particulier. Enfin, nous proposons, en perspective, de considérer une étape supplémentaire de reconnaissance de mots clés.

Le chapitre 6, après avoir rappelé les objectifs du projet RIMES, présente les résultats obtenus par AMBRES lors de la première campagne d'évaluation. La généralité de notre approche a permis de proposer un système pour 5 des 6 tâches évaluées couvrant des domaines aussi variés que la reconnaissance de l'écriture manuscrite, la structuration de courriers manuscrits ou encore la reconnaissance de logos isolés.

Enfin, nous terminons par des conclusions sur les travaux réalisés et présentons des perspectives d'évolution.

Les travaux de thèse ont permis d'apporter les contributions suivantes :

- **Chapitre 2**
  S. Chevalier, E. Geoffrois, F. Prêteux et M. Lemaitre : "A generic 2D approach of handwriting recognition", ICDAR 2005 (prix du meilleur papier étudiant).

- **Chapitre 3**
  M. Lemaitre, S. Chevalier, E. Geoffrois, E. Grosicki et F. Prêteux : "Étude de primitives spectrales pour la reconnaissance de caractères manuscrits dans le cadre d'une approche markovienne 2D", RFIA 2006.

- **Chapitre 4**
  M. Lemaitre, E. Grosicki, E. Geoffrois et F. Prêteux : "Modélisation bidimensionnelle Markovienne de l'écriture manuscrite : De la reconnaissance de caractères à

la reconnaissance de mots", article à soumettre.

- **Chapitre 5**
  M. Lemaitre, E. Grosicki, E. Geoffrois et F. Prêteux : "Preliminary experiments in layout analysis of handwritten letters based on textural and spatial information and a 2D Markovian approach", ICDAR 2007.

- **Chapitre 6**
  Article en préparation.

Chapitre 1

Reconnaissance de l'écriture manuscrite hors-ligne : état de l'art

1.1 Introduction

Cœur de tout système de lecture automatique de documents manuscrits, la reconnaissance de l'écriture manuscrite est toujours un domaine de recherche très actif. Malgré les nombreuses applications possibles, les systèmes industrialisés correspondent à des tâches restreintes telles que la lecture d'adresses postales, le tri automatique de courrier ou l'analyse de formulaires pour lesquels les informations écrites sont précasées, principalement pour des grandes administrations (INSEE, CAF, Trésor Public, URSSAF...). Cette constatation peut s'expliquer par la grande variabilité inhérente à la nature même de l'écriture manuscrite. Ainsi, devant cette difficulté à établir un système universel traitant tous les cas d'écriture, une solution à court terme a été de restreindre la tâche et de construire un système pour cette tâche.

L'introduction d'une modélisation statistique par champ de Markov [31] a permis des avancées importantes dans bon nombre de problèmes classiques en analyse d'images. Les champs markoviens ont été utilisés avec succès dans des tâches aussi diverses et variées que l'analyse de textures, de restauration et de débruitage, de segmentation ou encore de reconnaissance de formes. Pourtant, si les modèles markoviens 1D ou pseudo 2D sont utilisés depuis assez longtemps en reconnaissance de l'écriture manuscrite, les modèles réellement 2D qui permettraient de mieux prendre en compte la nature 2D de l'écriture ne le sont que depuis plus récemment.

Ce premier chapitre aborde tout d'abord la problématique de la reconnaissance de l'écriture manuscrite en explicitant les facteurs de difficulté pour son traitement, puis détaille les différentes étapes d'un système de reconnaissance. Les méthodes classiques d'extraction de primitives et de classification sont en particulier étudiées. La seconde partie de ce chapitre effectue un état de l'art des différentes approches markoviennes et montre en particulier l'évolution des approches de la modélisation purement monodimensionnelle à la modélisation purement bidimensionnelle.

1.2 Reconnaissance de l'écriture manuscrite hors-ligne

On distingue communément deux secteurs d'application en reconnaissance de l'écriture manuscrite suivant le mode d'acquisition :

- la reconnaissance d'écriture dite en-ligne pour laquelle le texte est saisi avec un stylet sur une surface sensible, fournissant ainsi une information temporelle,
- la reconnaissance d'écriture dite hors-ligne pour laquelle aucune information temporelle n'est disponible : la page de texte est scannée afin d'obtenir une image numérique.

C'est la reconnaissance de l'écriture hors-ligne qui va nous intéresser dans cette étude.

1.2.1 Degrès de difficulté du traitement de l'écriture manuscrite

On distingue plusieurs degrés de difficulté du traitement de l'écriture manuscrite (figure 1.1) suivant le type d'écriture et ses contraintes, le nombre de scripteurs ou la taille du vocabulaire.

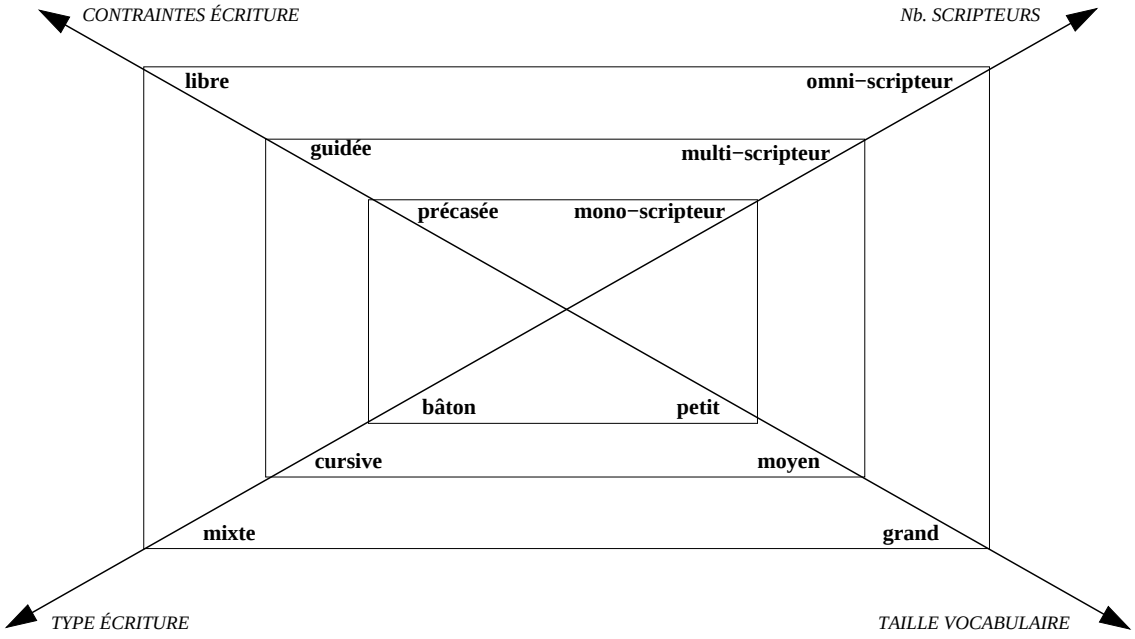


Fig. 1.1. Difficultés de l'écriture manuscrite

1.2.1.1 Styles d'écriture et contraintes

Chaque individu écrit d'une façon qui lui est propre, son écriture est différente de celle des autres même si l'alphabet de base est le même. Toute la difficulté de la reconnaissance de l'écriture manuscrite réside en majeure partie dans cette grande variété

d'écritures : tracés différents, enchaînements différents entre les lettres à l'intérieur d'un mot, etc.

Les variabilités dans l'écriture peuvent être dues à des contraintes externes ou à des contraintes internes provenant des habitudes propres au scripteur [87].

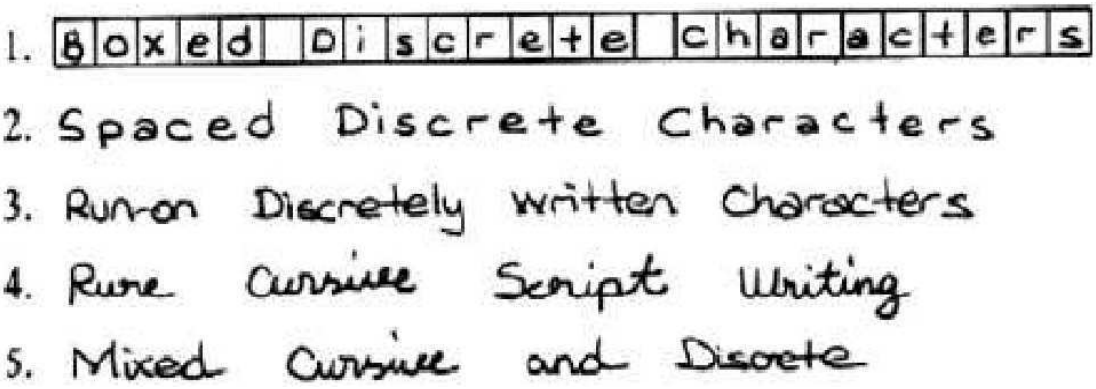


Fig. 1.2. Classification de l'écriture selon Tappert [87]

- *Contraintes externes* : le texte peut être précasé (cas 1 de la figure 1.2), guidé ou libre.
- *Contraintes internes* : elles sont propres à chaque scripteur. L'écriture peut être à lettres séparées, à lettres groupées liées, script (écriture bâton), purement cursive ou mixte (cas 2 à 5 de la figure 1.2). De plus, d'une personne à l'autre, la forme donnée à chaque caractère diffère, l'ensemble de ces différentes formes constitue ce que l'on appelle des allographes (figure 1.3).

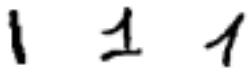


Fig. 1.3. Allographes du chiffre "1"

1.2.1.2 Taille du vocabulaire

La taille du vocabulaire est également un facteur influant sur les performances d'un système de reconnaissance ainsi que sur la méthode adoptée. En effet, alors que pour des tâches nécessitant un petit vocabulaire (une trentaine de mots dans le cas de la reconnaissance de montants de chèques) on cherchera à reconnaître le mot comme une entité à part entière, pour des tâches de très grand vocabulaire on cherchera plutôt à reconnaître les lettres au sein du mot.

Il est habituel de considérer que :

- un petit vocabulaire est consitué d'une dizaine de mots,
- un vocabulaire de taille moyenne comporte une centaine de mots,
- un grand vocabulaire rassemble des milliers de mots,
- enfin, un très grand vocabulaire renvoie à plus d'une dizaine de milliers de mots.

1.2.1.3 Nombre de scripteurs

La difficulté de reconnaissance augmente avec le nombre de scripteurs. En effet, plus il y a de scripteurs (droitier/gaucher, enfant/adulte, différentes professions), plus il y a de styles d'écritures différents (figure 1.4).

Fig. 1.4. Le mot "Rome" écrit par différents scripteurs

Un premier niveau est la reconnaissance *mono-scripteur* avec apprentissage de l'écriture propre à l'utilisateur considéré. Un niveau plus général est atteint par les systèmes *multi-scripteurs* et enfin les systèmes *omni-scripteurs* sont capables de reconnaître l'écriture de n'importe quel scripteur.

1.2.2 Architecture des systèmes de reconnaissance de caractères et de mots isolés

Dans le cadre de la reconnaissance de l'écriture manuscrite, ce sont les systèmes de reconnaissance de caractères isolés qui ont été le plus l'objet de recherche. Les techniques ont beaucoup évolué depuis la commercialisation du premier OCR dans les années 50 et surtout depuis l'apparition de moyens de calculs puissants dans les années 80 [2]. On est passé de méthodes structurelles où chaque caractère était comparé à des patrons à des méthodes statistiques. Depuis les années 90, les sytèmes de reconnaissance de l'écriture manuscrite ont profité des progrès du traitement de l'image et de la reconnaissance de formes. Les techniques statistiques telles que les réseaux de neurones et les modèles de Markov ont permis d'obtenir des résultats satisfaisants pour la reconnaissance de caractères isolés ou pour la reconnaissance de mots isolés mono-scripteur avec un petit lexique.

L'architecture d'un système de reconnaissance de caractères ou de mots isolés est composée de trois grandes étapes (figure 1.5) : le prétraitement, l'extraction des primitives et la classification.

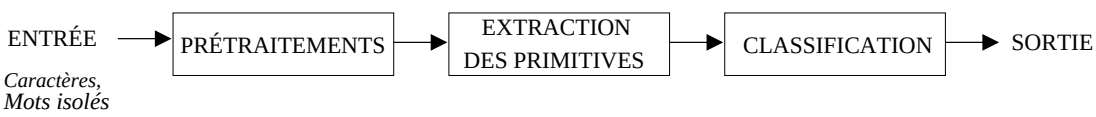


Fig. 1.5. Système de reconnaissance de mots ou de caractères isolés

1.2.2.1 Prétraitement

La phase de prétraitement est une étape facultative. Suivant les systèmes on trouve plus ou moins de prétraitements parmi lesquels [5] :

- le filtrage : il vise à réduire le bruit et les imperfections de l'image,
- la binarisation / le seuillage : il s'agit de convertir l'image en noir et blanc ou en niveaux de gris,
- le redressement de la ligne de base : il rend horizontaux les mots,
- le redressement des écritures penchées : il corrige l'inclinaison de l'écriture,
- la squelettisation : elle vise à obtenir une épaisseur du trait d'écriture égale à 1 pixel,
- la normalisation : elle ramène les images de mots à des tailles standards. Certaines méthodes cherchent même à normaliser localement les différentes parties d'un mot : par exemple, on veut que la partie centrale, les hampes et les jambes occupent chacun un tiers de la hauteur [77].

Image originale *Image après correction de la pente*

Fig. 1.6. Exemple de prétraitement : redressement d'écriture penchée

1.2.2.2 Extraction des primitives

La phase d'extraction des primitives (encore appelées caractéristiques), c'est-à-dire la représentation des données d'entrée, est une étape très importante pour un système de reconnaissance de formes. En effet, même si on utilise un classifieur très performant celui-ci ne peut compenser un mauvais choix des primitives. On peut définir l'extraction de primitives comme un *"problème d'extraction à partir de l'image, de l'information la plus pertinente, pour un problème de classification donné, c'est-à-dire celle qui minimise la variabilité intra-classe et qui maximise la variabilité inter-classe"* [25].

Dans la littérature, les primitives sont généralement classées de trois façons différentes.

Une première distinction est faite entre les primitives selon qu'elles sont extraites d'une image en niveaux de gris, d'une image binarisée, du contour ou du squelette de la forme [90, 102] résultant de la phase de prétraitement (figure 1.7).

- Pour une image en niveaux de gris, on extraira plutôt des primitives du type zoning, moments géométriques ou de Zernike, des primitives discrètes (largeur, hauteur, épaisseur moyenne des traits, etc.).
- Pour une image binarisée, il existe deux approches possibles : l'analyse indirecte et l'analyse directe selon que l'on passe par une extraction des composantes connexes

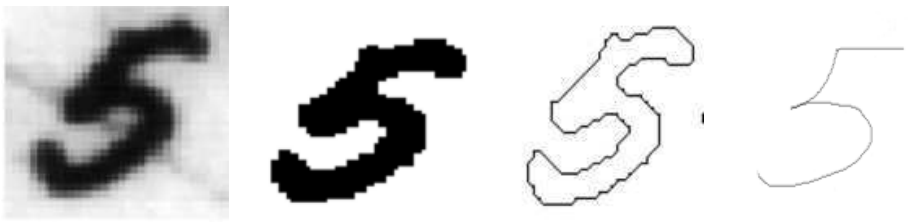


Fig. 1.7. Image en niveaux de gris, binaire, contour et squelette [90]

(les méthodes couramment utilisées sont le marquage des composantes connexes et les diagrammes de Voronoï) ou non [23, 24]. Les primitives seront entre autres des projections, des histogrammes, des moments géométriques ou de Zernike, des primitives du type zoning, des primitives discrètes (largeur, hauteur, épaisseur moyenne des traits, etc.).

- Sur un contour on extraira d'avantage des profils, des descripteurs de Fourier ou d'ondelettes, des splines et également des primitives du type zoning, code de Freeman [26], contour code [94]. On utilise également des contours actifs tels que les snakes [80] ainsi que des primitives discrètes [102] (périmètre, compacité, excentricité, etc.).
- Enfin pour un squelette, les primitives extraites seront des descripteurs de Fourier, des descriptions de graphe, des primitives discrètes (nombres de boucles, de croisements, rapport largeur sur hauteur, présence d'un point isolé, etc.).

Une seconde distinction peut être effectuée entre les primitives globales et les primitives locales [23].

- Les primitives globales cherchent à représenter au mieux la forme générale d'un caractère et sont donc calculées sur des images relativement grandes. On trouve entre autres la transformée de Fourier, la transformée de Hough qui détecte les lignes dans les images [89], ainsi que des primitives discrètes décrites précédemment [41].
- Les primitives locales sont calculées dans une fenêtre glissante qui parcourt les 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. Parmi ces primitives, on trouve la transformée de Fourier fenêtrée (Gabor) et les dérivées qui extraient une information sur la direction des traits [79], les moments ainsi que des primitives discrètes décrites précédemment.

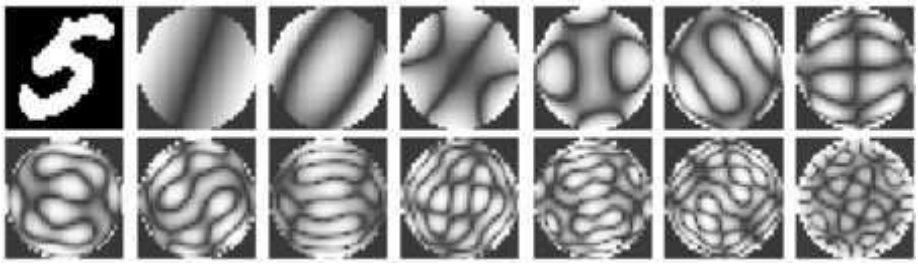
Une troisième distinction est effectuée entre les primitives topologiques, structurelles ou statistiques [90, 23].

- Les primitives topologiques ou métriques consistent à compter dans une forme ou un échantillon le nombre de trous, évaluer les concavités, mesurer 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, mesurer les surfaces et les périmètres, etc. [70, 40].
- Les primitives structurelles 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. Ainsi, on ne parle plus de trous, mais de boucles ou de cycles dans une représentation filiforme du caractère.
- Les primitives statistiques véhiculent une information distribuée sur toute l'image. L'histogramme [42, 73], qui représente le nombre de pixels sur chaque ligne ou colonne de l'image, en est un exemple classique et simple à calculer. On peut citer également l'approche fondée sur un moyennage des pixels situés à l'intérieur d'un masque rectangulaire (Zoning) : on construit une matrice de masques recouvrant la totalité de la forme qui permet une représentation statistique des valeurs correspondant à chaque masque.

Il est également intéressant de noter la classification proposée par Simon [83] en primitives régulières (invariantes aux translations) et singulières (appelées également accidents ou catastrophes).

Zoning Projections Profils



Moments de Zernike d'ordre 1 à 13

Fig. 1.8. Illustrations de certaines primitives [90]

1.2.2.3 Classification

Le rôle du classifieur est de déterminer l'appartenance d'une forme à une classe à partir des vecteurs de primitives extraits de cette forme. De nombreux classifieurs existent et sont plus ou moins bien adaptés à la reconnaissance de l'écriture. Ci-dessous, nous décrivons les principaux.

L'appariement de formes Cette approche a été l'une des premières proposées pour la reconnaissance de caractères : les formes sont comparées à un patron rigide via une mesure de similarité. Elle est peu adaptée à l'écriture du fait de sa grande variabilité qui impliquerait un grand nombre de représentants pour chaque classe.

L'appariement syntaxique Cette approche repose sur une représentation hiérarchique de la forme qui est vue comme un ensemble de sous-formes (patterns) eux-mêmes composées de patterns plus petits (primitives). Une analogie est ainsi faite avec la syntaxe du langage [37] : la forme est vue comme une phrase, les sous-formes comme des mots, les primitives sont les lettres de l'alphabet et les phrases sont obtenues par l'intermédiaire d'une grammaire. Grâce à la grammaire construite à partir de la base d'apprentissage, on peut donc définir un grand nombre de formes complexes. Toutefois, cette approche est très sensible aux problèmes de segmentation ainsi qu'au bruit.

Ces deux approches ne sont pas très adaptées à la reconnaissance de l'écriture manuscrite qui présente une grande variabilité. Les classifieurs les plus utilisés sont les SVM, les réseaux de neurones et les classifieurs statistiques par HMM.

SVM Les machines à vecteurs de support (SVM) appelées aussi classifieurs à marge optimale ou encore séparateurs à vaste marge ont été introduites par Vapnik [93]. Leur principe est de maximiser la marge entre les classes. Il faut donc déterminer l'hyperplan maximisant cette marge. Les SVM offrent des performances intéressantes pour la reconnaissance de caractères manuscrits [58], mais ils sont peu applicables à la reconnaissance de mots (sauf éventuellement avec une segmentation explicite des mots en lettres). En effet ils travaillent avec des données en dimension fixe et ne permettent donc pas d'introduire la variabilité de longueur des mots. De plus, ils ont l'inconvénient d'être assez lents en phase d'apprentissage comme en phase de reconnaissance.

Les réseaux de neurones L'idée principale est qu'un neurone formel est capable de réaliser des calculs élémentaires comme la séparation d'un vecteur en deux classes, chaque classe étant déterminée par le poids du neurone. Le problème est alors de choisir quels coefficients affecter aux poids pour réaliser une séparation optimale. La multiplication des neurones permet de séparer plusieurs classes : il faut donc réaliser un choix sur la topologie du réseau. Les réseaux de neurones sont bien adaptés à la reconnaissance de formes globales telles que des caractères isolés. Ils se limitent à la classification de formes simples car fondés sur une représentation en dimension fixe.

Les approches markoviennes Les modèles de Markov sont couramment utilisés pour la reconnaissance de l'écriture manuscrite. En effet, ce sont des outils statistiques puissants permettant de calculer la probabilité d'appartenance d'une forme à une classe. La forme est vue comme un ensemble d'observations émises par des états cachés. La modélisation markovienne est bien adaptée à l'écriture manuscrite car elle permet d'intégrer les variabilités de longueur des mots.

Dans notre étude, nous allons exploiter ce type d'approche décrite plus en détail dans la prochaine section.

1.3 Modélisation statistique de l'écriture manuscrite par des approches markoviennes

1.3.1 Introduction

Depuis quelques années, les modèles de Markov connaissent un essor important en reconnaissance des formes et plus particulièrement en reconnaissance de l'écriture manuscrite. Le traitement de la parole les utilise avec succès depuis longtemps et comme la parole, l'écriture manuscrite se prête à une modélisation markovienne.

Alors que la parole ne laisse pas de doute sur sa nature 1D celle de l'écriture manuscrite est plus discutable. En effet, suivant les points de vue et les méthodes, elle peut être interprétée comme étant purement 1D, ou elle peut être vue comme une image à part entière 2D, enfin elle peut être considérée comme un signal à la fois 1D (signal temporel de la gauche vers la droite) et 2D (image). On peut également noter que certaines méthodes cherchent à retrouver l'information temporelle à partir de l'image numérisée [76]. On peut constater que la tendance actuelle est la prise en compte du caractère 2D des images par une modélisation bidimensionnelle.

1.3.2 Modèles unidimensionnels : HMM ou Hidden Markov Models

Les HMM ou modèles de Markov cachés sont des outils statistiques puissants qui permettent de calculer la probabilité d'appartenance d'une forme à une classe. Cette forme est émise par des états cachés dont leur succession est modélisée par une chaîne de Markov pour laquelle quelques exemples sont donnés figure 1.9.

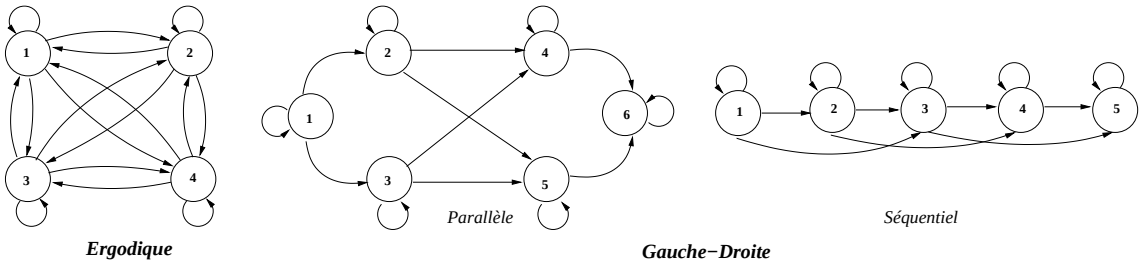


Fig. 1.9. Quelques exemples d'architecture de HMM

Une chaîne de Markov d'ordre n a la propriété suivante : la probabilité de se trouver dans un état dépend des n états précédents. Ainsi, si l'on considère une séquence de variables aléatoires $X_1, X_2, ..., X_T$ prenant leurs valeurs dans l'ensemble des états $S = \{s_1, ..., s_N\}$ et avec $n = 1$, on a :

$$P(X_i|X_1, X_2, X_3, X_4, ..., X_{i-1}) = P(X_i|X_{i-1}).$$

Une chaîne de Markov cachée d'ordre 1 est définie par :
- $A = \{a_{i,j}\}$ où $a_{i,j} = P(X_{t+1} = s_j | X_t = s_i)$. A est la matrice des probabilités de transition entre états.
- $B = \{b_j(o)\}$ où $b_j(o) = P(o_t = o | X_t = s_j)$. B est la matrice des probabilités d'observations dans les états.
- $\pi = \{\pi_j\}$ où $\pi_j = P(X_1 = s_i)$. π est le vecteur des probabilités initiales des états.

L'utilisation des HMM se décompose en trois problèmes [72] :

1. calcul de la probabilité d'une observation : elle est obtenue par l'algorithme *Forward* ;
2. calcul de la séquence d'états la plus probable compte tenu d'une observation (décodage) : elle est souvent obtenue grâce à l'algorithme de *Viterbi* ;
3. calcul des paramètres du modèle compte tenu des observations d'apprentissage : ils sont déterminés le plus souvent grâce à l'algorithme *Baum-Welch*.

En reconnaissance de l'écriture manuscrite, le HMM modélise un mot pour lequel chacune de ses lettres est un état caché ou une chaîne de Markov à N états (figure 1.10). Les observations extraites des images sont appelées primitives ou vecteurs caractéristiques. Un HMM est construit pour chaque mot du vocabulaire et les vraisemblances correspondantes sont calculées. On choisit le mot du vocabulaire qui maximise cette vraisemblance.

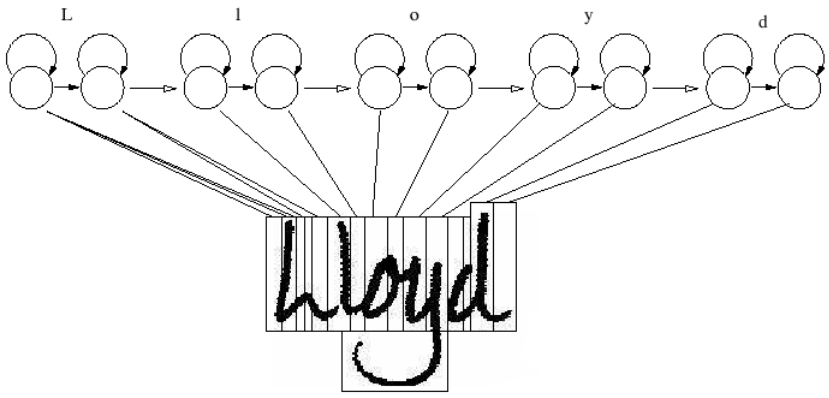


Fig. 1.10. Exemple d'application des HMM à la reconnaissance de mots (figure tirée de [96])

De nombreuses approches de reconnaissance de l'écriture à base de HMM ont été proposées. Elles varient entre autres suivant l'architecture du HMM et la nature des observations qui peuvent être continues [97, 8], discrètes [11] ou encore hybrides (c'est-à-dire que le HMM est combiné à des réseaux de neurones ou SVM par exemple). Quelques exemples de performances sont indiquées dans le tableau 1.1.

[1] Base de mots arabes concernant des noms de villes ou de villages en Tunisie

Tab. 1.1. Quelques résultats obtenus avec des HMM en terme de taux de reconnaissance au niveau mot

HMM	Taille du vocabulaire	Base	Base d'app. Nb images	Base de test Nb images	Résultats obtenus	Réf
continu	1334	Senior & Robinson	2360	1016	82.45%	[97]
continu	776	IAM	1769	443	71.34%	[61]
continu	7719	IAM	35215	8804	60.05%	[61]
continu	964	IFN/ENIT[1]	19844	6615	83.79%	[8]
discret	30k	Mots allemands	2000	200	86.7%	[11]
hybride	30k	Mots allemands	2000	200	89.2%	[11]

1.3.3 Modèles pseudo bidimensionnels : PHMM ou Planar HMM

Même si les HMM ont permis d'obtenir des résultats intéressants, l'extension des HMM au cas 2D laisse espérer des améliorations du fait de la nature bidimensionnelle de l'écriture. Selon Levin [53] une approche entièrement bidimensionnelle conduirait à une complexité excessive, c'est pourquoi il propose des simplifications, notamment une indépendance verticale : il définit ainsi les PHMM. L'étude des PHMM a par la suite été affinée par entre autres Agazzi [3] et Kuo [48].

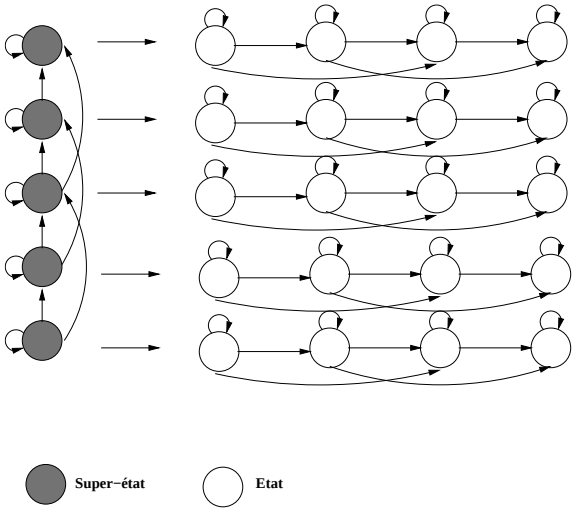


Fig. 1.11. Architecture d'un PHMM

Un PHMM est un HMM pour lequel la probabilité d'observation dans chaque état (super-état) est donnée par un HMM secondaire (figure 1.11). De la même façon que l'on avait défini un HMM, on peut définir un PHMM par :

- $A = \{a_{i,j}\}$ où $a_{i,j} = P(X_{y+1} = s_j | X_y = s_i)$. A est la matrice de transition entre super-états.
- $\pi = \{\pi_j\}$ où $\pi_j = P(X_1 = s_i)$. π est le vecteur des probabilités initiales des super-états.
- $\Lambda = \{\lambda^k\}$, l'ensemble des HMM associés aux super-états :

- $A^k = \{a^k_{i,j}\}$ où $a^k_{i,j} = P(X_{x+1,y} = s^k_j | X_{x,y} = s^k_i)$,
- $B^k = \{b_j(o)^k\}$ où $b_j(o) = P(o_{x,y} = o | X_{x,y} = s^k_j)$,
- $\pi^k = \{\pi^k_j\}$ où $\pi^k_j = P(X_{1,y} = s^k_i)$.

En reconnaissance de l'écriture, le modèle secondaire est souvent associé aux lignes où la forme est réellement observée, leurs architectures sont typiquement gauche-droite. Généralement, plusieurs lignes sont associées à chaque super-état (hypothèses de corrélation entre plusieurs lignes consécutives). Le nombre de super-états dépend de la morphologie de la forme et des principales zones horizontales d'observation que l'on veut mettre en évidence. Par exemple (figure 1.12), une zone horizontale peut correspondre à un ensemble de lignes pour lesquelles il y a les mêmes transitions noir blanc.

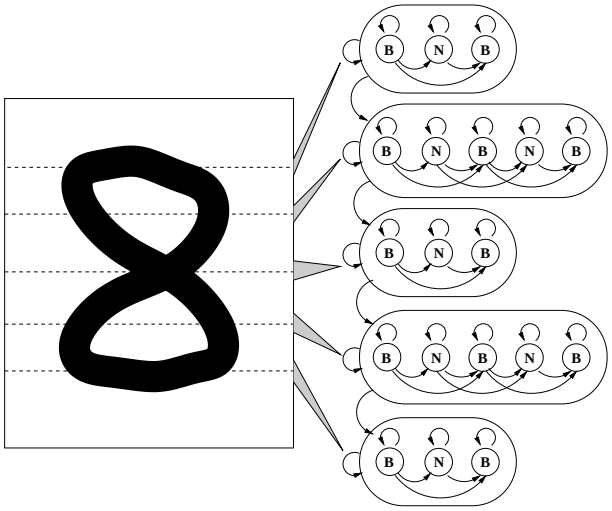


Fig. 1.12. Exemple d'application des PHMM pour la reconnaissance de chiffres

Quelques exemples de résultats obtenus avec les PHMM en reconnaissance de l'écriture manuscrite sont donnés dans le tableau 1.2.

Tab. 1.2. Quelques résultats obtenus avec des PHMM

Taille du vocabulaire	Base	Base d'app. Nb images	Base de test Nb images	Résultats obtenus	Réf.
100 PAW[2]	Noms de ville	*non connu*	33168 PAW	99.84%	[64]
21	Chèques allemands	47000	1634	84.2%	[10]
10	UNIPEN	12600	1400	86.29%	[77]

1.3.4 Modèles bidimensionnels

Contrairement aux PHMM, les champs de Markov modélisent une vraie structure 2D. Connus et étudiés par les statisticiens depuis une quarantaine d'années, les champs

[2]PAW=Piece of Arabic Word

de Markov ont fait leur apparition dans le domaine de l'analyse d'images en 1984 avec les travaux de Geman et Geman [31]. Ils sont utilisés avec succès dans des domaines très variés tels que l'extraction de contours, la segmentation, la stéréovision, la reconstruction tomographique, etc. En revanche ils sont moins utilisés en reconnaissance de l'écriture manuscrite, même s'ils sont de plus en plus étudiés.

Un champ de Markov posséde une vraie structure 2D du fait de la dépendance bidimensionnelle locale des sites au sein d'un voisinage. Ainsi, la probabilité de la réalisation d'une variable aléatoire connaissant toutes les réalisations des autres variables aléatoires ne dépend que du voisinage fixé :

$$P(X_{i,j}|\{X_{m,n}\}_{(m,n)\in\Omega}) \quad = \quad P(X_{i,j}|\{X_{m,n}\}_{(m,n)\in V_{i,j}}),$$

où $V_{i,j}$ est le voisinage (figure 1.13) du site (i,j) et $\Omega \subset L$ où $L = \{(i,j)|1 \leq i \leq m, 1 \leq j \leq n\}$ est l'ensemble des sites.

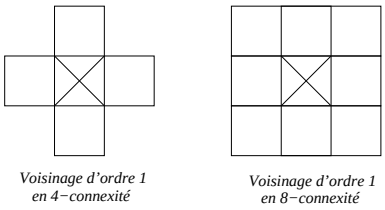


Fig. 1.13. Exemples de voisinage

La plupart des champs de Markov utilisés en reconnaissance de l'écriture manuscrite font une hypothèse de causalité introduisant un sens de parcours comme les réseaux de Markov ou les champs de Markov unilatéraux et utilisent un décodage monodimensionnel ; cependant une nouvelle méthode a été récemment proposée pour décoder bidimensionnellement la séquence d'états la plus probable.

Dans le cas d'un décodage monodimensionnel l'étiquetage d'un site à un certain état se fait en fonction d'une partie des sites de l'image. Un décodage bidimensionnel va lui tenir compte de l'ensemble des sites.

1.3.4.1 Décodage monodimensionnel

Réseaux de Markov Les réseaux de Markov ou Markov Mesh Random Fields ont été introduits par Abend en 1965. Ce sont des champs de Markov faisant une hypothèse de causalité.

On peut définir le réseau de Markov de la façon suivante :

$$P(X_{i,j}|\{X_{m,n}\}_{(m,n)\in\Omega_{i,j}}) \quad = \quad P(X_{i,j}|\{X_{m,n}\}_{(m,n)\in V_{i,j}}),$$

où $V_{i,j} = \{(i,j-1),(i-1,j),(i-1,j-1)\}$ et $\Omega_{i,j} = \{(k,l)|1 \leq k \leq i \text{ ou } 1 \leq l \leq j\}$, avec $(i,j) \in L$ où $L = \{(i,j)|1 \leq i \leq m, 1 \leq j \leq n\}$ est l'ensemble des sites.

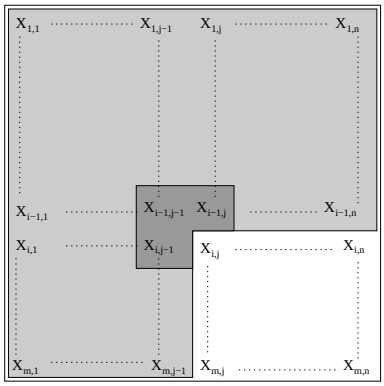


Fig. 1.14. Architecture d'un réseau de Markov

Park et Lee ont appliqué les réseaux de Markov à la reconnaissance de l'écriture manuscrite [71]. Pour la phase de décodage un algorithme rapide *look-ahead* et plus particulièrement un algorithme "one-row-one-column" *look-ahead* est utilisé : l'étiquetage d'un site (m, n) n'est pas définitif tant que le site $(m+1, n+1)$ n'a pas été traité par l'algorithme. Les résultats obtenus sont donnés dans le tableau 1.3.

Tab. 1.3. Résultats obtenus avec un réseau de Markov et un décodage *look-ahead* [71]

Taille du vocabulaire	Base	Base d'apprentissage Nb images	Base de test Nb images	Résultats obtenus
10	Chiffres de l'Université Concordia	4000	2000	90.8%

Champs de Markov unilatéraux Ils ont été introduits par Preuss en 1975. Ils différent des réseaux de Markov de par le "passé" des états.

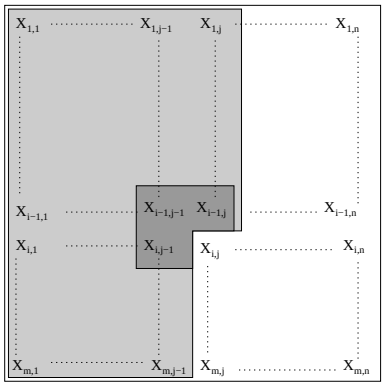


Fig. 1.15. Architecture d'un champ de Markov unilatéral

Le champ de Markov unilatéral est défini de la façon suivante :

$$P(X_{i,j}|\{X_{m,n}\}_{(m,n)\in\Omega_{i,j}}) \quad = \quad P(X_{i,j}|\{X_{m,n}\}_{(m,n)\in V_{i,j}}),$$

où $V_{i,j} = \{(i, j-1), (i-1, j), (i-1, j-1)\}$ et $\Omega_{i,j} = \{(k, l) | l < j \text{ ou } (l = j, k < i)\}$, avec $(i, j) \in L$ où $L = \{(i, j) | 1 \leq i \leq m, 1 \leq j \leq n\}$ est l'ensemble des sites.

Saon [77] propose une simplification grâce à l'utilisation de NSHP-HMM (Non-Symmetric Half-Plane-HMM ou HMM à demi-plan non symétrique) pour la reconnaissance de l'écriture manuscrite. La réalisation du champ aléatoire (c'est-à-dire l'image du mot) est vue comme une observation d'une séquence de colonnes (une colonne représente donc un état du HMM) ; le décodage est donc monodimensionnel. La probabilité d'observation d'un état du HMM est le produit des probabilités d'observation des pixels de la colonne. NSHP étant un champ de Markov causal, la probabilité d'observation d'un pixel dépend de celle de ses voisins. L'image est donc analysée colonne par colonne ce qui permet une élasticité horizontale qui rend son adaptation facile à différentes largeurs d'image. En revanche, NSHP-HMM présente une rigidité verticale (le nombre de lignes est fixé).

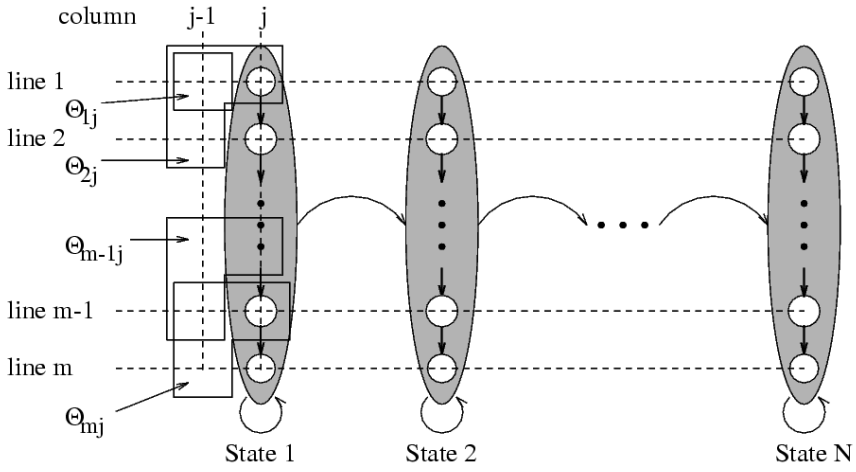


Fig. 1.16. Architecture d'un champ de Markov unilatéral selon Saon (figure tirée de [77])

Les résultats obtenus par G. Saon pour la reconnaissance de montants de chèques sont donnés dans le tableau 1.4.

Tab. 1.4. Résultats obtenus avec NHSP-HMM [77]

Taille du vocabulaire	Base d'apprentissage		Base de Test		Résultats obtenus
	Base	Nb images	Base	Nb images	
26	SRTP[3] (2/3)	4653	SRTP (1/3)	2378	90.08%
26	A2iA[4]	36829	A2iA	4098	82.50%
26	A2iA	>100000	LIX[5]	15892	91.10%

[3]SRTP=Service de Recherche Technique de la poste

[4]Les chèques de la base A2iA (nommée 0503) proviennent de différentes banques.

[5]Base LIX : base fournie par le laboratoire du même nom. Les scripteurs ont simulés des montants de chèques réels de sorte que les mots soient représentés le plus uniformément possible

Champs de Markov avec décodage bidimensionnel Le décodage bidimensionnel permet à la fois de parcourir les sites de façon quelconque et de tenir compte de l'ensemble de l'image pour l'assignation d'un état à un site. En effet, contrairement au décodage 1D, l'étiquetage des sites n'est pas définitif tant que toute l'image n'a pas été traitée par l'algorithme de décodage 2D.

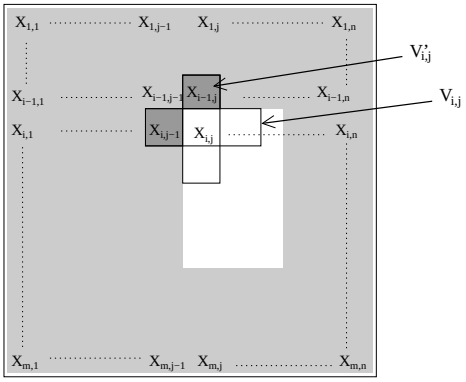


Fig. 1.17. Architecture d'un champ de Markov avec parcours quelconque des sites

Dans le cas d'un parcours quelconque de l'image, on définit le champ de Markov de la façon suivante :

$$P(X_{i,j}|\{X_{m,n}\}_{(m,n)\in\Omega_{i,j}}) \quad = \quad P(X_{i,j}|\{X_{m,n}\}_{(m,n)\in V'_{i,j}}),$$

où $V'_{i,j} = \Omega_{i,j|V_{i,j}}$ avec $(i,j) \in L$ où $L = \{(i,j)|1 \leq i \leq m, 1 \leq j \leq n\}$ est l'ensemble des sites. $\Omega_{i,j}$ est défini par le sens de parcours de l'image. Pour un parcours gauche-droite (du haut vers le bas) on a $\Omega_{i,j} = \{(k,l)|l < j \text{ ou } (l = j, k < i)\}$.

Chevalier [19] a récemment proposé une application des champs de Markov par décodage bidimensionnel à la reconnaissance de chiffres manuscrits. L'algorithme de décodage utilisé est la programmation dynamique 2D proposée par Geoffrois [33, 32]. L'objectif de l'algorithme proposé est de segmenter les images suivant la position et la direction des traits dans l'image : cela est rendu possible grâce aux primitives spectrales locales. Les performances obtenues sont indiquées dans le tableau 1.5.

Tab. 1.5. Résultats obtenus avec un champ de Markov et la programmation dynamique 2D [19]

Taille du vocabulaire	Base	Base d'apprentissage Nb images	Base de test Nb images	Résultats obtenus
10	Chiffres MNIST[49]	60000	10000	97.26%
364	Senior et Robinson Mots $\leq$ 4 lettres	2304	329	40%

1.4 Conclusion

Nous avons décrit les approches markoviennes utilisées pour la reconnaissance de l'écriture manuscrite suivant leur nature 1D, pseudo 2D ou réellement 2D. De notre point de vue, l'écriture manuscrite hors-ligne a une nature réellement 2D ; en effet seule l'information bidimensionnelle de l'image numérique est disponible puisque l'information temporelle n'est pas connue. Ainsi, nous souhaitons pouvoir modéliser l'écriture par une approche entièrement bidimensionnelle. La comparaison des 3 méthodes markoviennes bidimensionnelles présentées précédemment et résumées dans le tableau 1.6 montre que c'est l'apparition de la programmation dynamique 2D qui a permis une véritable modélisation 2D de l'écriture à tous les niveaux : observations, états et décodage.

Tab. 1.6. Comparaison des 3 méthodes de reconnaissance de l'écriture manuscrite par champs de Markov

Référence	Observations	États	Décodage
Saon, 1997 [77]	$2D$	$1D$	$1D$
Park et Lee, 1996 [71]	$2D$	$2D$	$1D$
Chevalier *et al.*, 2003 [17]	$2D$	$2D$	$2D$

L'approche bidimensionnelle markovienne proposée par Chevalier a été appliquée avec succès à une tâche de reconnaissance de chiffres manuscrits mais reste à l'état d'ébauche pour la reconnaissance de mots. Nous proposons dans cette thèse de repartir de cette approche en l'améliorant et en la généralisant à la reconnaissance et à la segmentation d'images. Cela nous permettra dans la suite d'aborder la reconnaissance de l'écriture manuscrite au niveau caractère comme au niveau mot ainsi que la structuration de documents manuscrits.

Chapitre 2

AMBRES : une approche bidimensionnelle markovienne générale pour l'analyse de documents manuscrits

2.1 Introduction

Dans cette thèse, nous traitons l'analyse de documents manuscrits dans son ensemble. C'est-à-dire que nous envisageons une tâche complète comme celle décrite par le projet RIMES (Cf chapite 6) englobant la reconnaisssance de caractères, la reconaissance de mots ou encore la structuration de documents. Alors que dans la littérature les systèmes proposés sont spécifiques à une tâche en particulier, nous proposons ici une approche générale.

L'introduction d'une modélisation statistique par champs de Markov a permis des avancées importantes en reconnaissance de l'écriture manuscrite. Du fait de la complexité des algorithmes de décodage, la plupart des approches proposées étaient 1D ou pseudo 2D. L'apparition de la programmation dynamique 2D en 1998 [33] a permis d'ouvrir la voie à la modélisation purement bidimensionnelle de l'écriture manuscrite. En 2003 Chevalier *et al.* ont proposé une approche bidimensionnelle markovienne dédiée à la reconnaissance de chiffres manuscrits fondée sur la programmation dynamique 2D et des primitives spectrales locales.

Dans nos travaux, nous proposons de partir de cette approche et de la généraliser à une tâche plus complète d'analyse de document pouvant traiter à la fois des tâches de reconnaissance (caractères, mots, ...) et de structuration. La nouvelle formulation ainsi généralisée de cette approche est appelée AMBRES (Approche Markovienne Bidimensionnelle pour la Reconnaissance Et la Segmentation d'images). Dans ce chapitre, nous rappelons et étudions en détail les différentes étapes de AMBRES et montrons sa généralisation à toute tâche de reconnaissance et de segmentation en introduisant notamment un modèle de position. Celui-ci est essentiel pour prendre en compte la

position spatiale des différents états et permet ainsi une modélisation plus fine de la structure physique d'un document.

La généralité de AMBRES sera démontrée tout au long de la thèse, en l'appliquant tout d'abord à la reconnaissance de caractères isolés (chapitre 3), à la reconnaissance de mots isolés (chapitre 4), à la structuration de courriers manuscrits (chapitre 5) ou encore à la reconnaissance de logos dans le cadre de la campagne d'évaluation RIMES (chapitre 6).

Une des originalités du travail présenté concerne la méthodologie de AMBRES mise en œuvre pour l'optimisation des paramètres et pour l'étude du système à travers ses performances et les erreurs commises amenant à des pistes d'amélioration (chapitre 3).

Dans ce chapitre, nous rappelons dans un premier temps le cadre théorique de la modélisation des images par champs aléatoires de Markov. Puis, nous présentons l'approche AMBRES et étudions en détail ses différentes étapes. Nous définissons en particulier un nouveau terme de position essentiel dans la modélisation de document. Nous montrons ensuite comment exploiter cette approche pour des tâches diverses de reconnaissance et de segmentation. Enfin, nous explicitons la phase d'extraction des primitives et étudions en détail les primitives spectrales locales.

2.2 Cadre théorique des champs aléatoires de Markov

Dans cette section, nous allons définir les champs aléatoires de Markov discrets. Dans toute la suite, on considère une grille rectangulaire L définie par : $L = \{(i,j)|0 \leq i < m, 0 \leq j < n\}$ constituée de $n \times m$ sites désignés par leurs coordonnées (i,j).

2.2.1 Définitions

Rappelons les définitions des champs aléatoires ainsi que des systèmes de voisinage et des cliques associées sur une grille L.

Définition 2.2.1. ***Champ aléatoire***
Un champ aléatoire X défini sur L est une collection de variables aléatoires : $X = \{X_{i,j}|(i,j) \in L\}$.

Définition 2.2.2. ***Système de voisinage***
Un système de voisinage V défini sur L est $V = \{V_{i,j} \subset L|(i,j) \in L\}$ où $V_{i,j}$ est appelé voisinage du site (i,j), tel que : $(i,j) \notin V_{i,j}$ et $(k,l) \in V_{i,j} \Rightarrow (i,j) \in V_{k,l}$.

Les voisinages les plus couramment utilisés sont les voisinages d'ordre 1 en 4 et 8-connexité respectivement notés :

- V^1, tel que $V^1_{i,j} = \{(i,j-1),(i,j+1),(i-1,j),(i+1,j)\}$,
- V^2, tel que $V^2_{i,j} = \{(i,j-1),(i,j+1),(i-1,j),(i+1,j),(i-1,j-1),(i+1,j+1),(i-1,j+1),(i+1,j-1)\}$.

Définition 2.2.3. ***Clique***
Une clique c associée à la grille L et au système de voisinage V est soit un singleton de V soit un ensemble de sites tels que chaque site est voisin des autres sites dans le système de voisinage V.

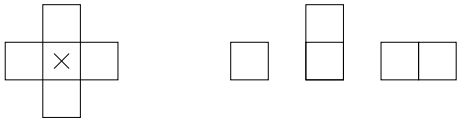


Fig. 2.1. Cliques associées à un voisinage d'ordre 1 en 4-connexité

2.2.2 Définition des champs aléatoires de Markov

Les champs de Markov exploitent une structure de graphe permettant une modélisation réellement bidimensionnelle des images. En effet, l'information contenue dans une image va au-delà de la seule donnée des niveaux de gris en chaque site. La description de l'image se fera plutôt en termes de zones, contours ou textures. C'est donc l'interaction entre les différents sites qui est significative. Le formalisme markovien permet de prendre en compte les interactions locales pour définir différentes régions de l'image. La définition d'un champ aléatoire de Markov est la suivante :

Définition 2.2.4. ***Champ aléatoire de Markov (Markov Random Fields ou MRF)***
X est un champ aléatoire de Markov si et seulement si :

$$\forall \Gamma \subseteq L, \forall (i,j) \in L \setminus \Gamma \quad P(X_{i,j}|X_\Gamma) = \quad P(X_{i,j}|X_{V_{i,j}}).$$

Cette définition exprime que la probabilité conditionnelle de la réalisation d'une variable aléatoire en un site, connaissant toutes les réalisations des autres variables aléatoires, ne dépend que des variables associées au voisinage du dit site. Cela illustre la propriété de contexte spatial et de dépendance locale utilisée dans le cadre de l'analyse d'images.

2.2.3 Équivalence entre champs de Markov et champs de Gibbs

Définition 2.2.5. ***Champ aléatoire de Gibbs (Gibbs Random Fields ou GRF)***
X est un champ aléatoire de Gibbs si et seulement si la probabilité d'une configuration est telle que :

$$P(X = x) \quad = \quad \frac{1}{Z} e^{-U(x)},$$

où Z est la constante de normalisation sur l'ensemble des réalisations de X et $U(x)$ est la fonction d'énergie exprimée par $U(x) = \sum_{c \in C} V_c(x)$. C est l'ensemble de toutes

les cliques de (L, V) et $V_c(x)$ représente le potentiel associé à la clique c et ne dépend que des sites appartenant à c. Ce potentiel de clique apporte des informations *a priori* sur l'image à modéliser en favorisant plus ou moins certaines configurations d'étiquettes.

Le théorème de Hammersley-Clifford établit l'équivalence entre les champs de Markov et les champs de Gibbs. Il s'énonce comme suit :

Théorème 2.2.1. ***Hammersley-Clifford***
X est un champ aléatoire de Markov sur la grille L avec le système de voisinage V et $P(X = x) > 0$ pour toute réalisation x si et seulement si X est un champ aléatoire de Gibbs sur la grille L avec le système de voisinage V.

Ce théorème est fondamental pour la modélisation par champs aléatoires de Markov. En effet, il établit une équivalence entre la caractère local des MRF et le caractère global des GRF. Plusieurs types de GRF ont ainsi été utilisés en analyse d'image. On peut notamment citer : le modèle d'Ising [31], aussi appelé modèle auto-logistique, utilisé pour modéliser la texture et dont la théorie a été inspirée de la physique statistique, et le modèle de Potts qui en est une généralisation.

2.2.4 Décodage d'un MRF

Pour le décodage d'un champ de Markov, c'est-à-dire la détermination de la séquence d'états la plus probable, trois algorithmes principaux existent dans la littérature : le recuit simulé, l'ICM et la programmation dynamique 2D.

- Le principe du recuit simulé [44] est d'introduire une notion de température qui sera diminuée progressivement. Entre chaque changement de température une configuration ω est simulée, notamment grâce à l'échantillonneur de Gibbs, avec la loi d'énergie $\frac{U(\omega)}{T}$. La température initiale doit être choisie grande et la décroissance doit être suffisamment lente pour converger vers un minimum global d'énergie. Les itérations sont arrêtées lorsque le taux de changement des configurations des sites entre deux étapes est suffisamment faible. Un des inconvénient du recuit simulé est qu'il est très lourd en temps de calculs et donc très lent.
- L'ICM (Iterated Conditional Modes) [9] est un algorithme itératif déterministe. À chaque étape, une configuration ω est simulée. Entre chacune de ces configurations, on choisit pour chaque site, la configuration maximisant la probabilité conditionnelle locale. Cet algorithme converge très rapidement, mais n'assure pas la convergence vers un minimum global.
- Pour obtenir à la fois rapidité et optimalité, un nouvel algorithme a été récemment proposé. Il s'agit de la programmation dynamique 2D [33] qui n'est autre que la généralisation de l'algorithme classique de programmation dynamique 1D au cas 2D. Il sera détaillé et explicité par la suite.

2.3 AMBRES

Dans cette section, nous présentons la formulation généralisée de l'approche dédiée à la reconnaissance de chiffres manuscrits proposée par Chevalier *et al.*. Cette approche

appelée AMBRES permettra d'aborder une tâche plus complète d'analyse de document en traitant à la fois la reconnaissance et la segmentation. Elle est fondée sur une modélisation par MRF modifié que l'on appelera MRF-P en raison de l'introduction d'un terme de position. Après avoir détaillé cette modélisation, nous explicitons les différents paramètres du modèle markovien. Nous voyons ensuite comment exploiter AMBRES dans le cadre de la reconnaissance de formes et de la segmentation d'image. Nous discutons enfin sur le choix des primitives et étudions les primitives spectrales locales.

2.3.1 Modélisation par MRF-P

L'idée générale est de créer un modèle markovien bidimensionnel pour chaque classe d'image : dans le cadre de la segmentation d'image, il n'y aura qu'une classe, en revanche pour la reconnaissance d'images il y aura autant de classes que de formes à reconnaître. Ces modèles seront utilisés lors de la phase de reconnaissance pour déterminer la classe la plus probable des différentes images de la base de test ou pendant la segmentation pour déterminer la configuration optimale des étiquettes recherchées.

On considère une approche bayésienne classique d'analyse d'image où l'image observée est supposée être générée par un processus caché que l'on cherche à déterminer (en l'occurrence une configuration d'étiquettes associées à chaque site). Les modèles correspondant à chaque classe d'images vont ainsi relier les états cachés $\omega_{i,j}$ aux primitives $o_{i,j}$ extraites des images (scalaires ou vecteurs).

Selon l'hypothèse markovienne de dépendance locale, la probabilité d'un état ne dépend que des états de son voisinage immédiat. Ici, le choix s'est porté sur un voisinage d'ordre 1, *i.e.* on utilise le voisinage V^1 précédemment défini. La structure du champ de Markov utilisée est illustrée figure 2.2.

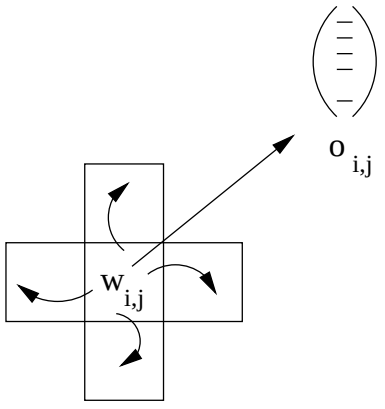


Fig. 2.2. Champ de Markov avec un voisinage d'ordre 1 en 4-connexité

L'objectif est de déterminer pour chaque image la configuration optimale $\hat{\omega}$ de la grille d'états connaissant les observations. Pour cela, on adopte une stratégie bayésienne,

qui va transformer la probabilité *a posteriori* $P(\omega|O)$ en probabilité *a priori* $P(O|\omega)$:

$$\begin{aligned}\hat{\omega} &= \arg\max_{\omega\in\Omega} P(\omega|O),\\ \hat{\omega} &= \arg\max_{\omega\in\Omega} \frac{P(O|\omega)P(\omega)}{P(O)},\\ \hat{\omega} &= \arg\max_{\omega\in\Omega} P(O|\omega)P(\omega),\end{aligned}$$

où :

- Ω est l'ensemble des configurations possibles ω,
- $O = \{o_{i,j}, (i,j) \in L\}$ est l'ensemble des primitives de l'image,
- $\omega = \{\omega_{i,j}, (i,j) \in L\}$ où $\omega_{i,j}$ prend ses valeurs dans $\{s_0, ...s_N\}$ et N est le nombre d'états du modèle.

Dans l'expression $P(O|\omega)P(\omega)$, le premier terme est appelé terme d'attache aux données, puisqu'il modélise le processus de formation de l'image en fonction d'une configuration d'états sous-jacente. Ce terme est aisément calculable du fait de l'hypothèse d'indépendance des observations conditionnellement aux étiquettes. Il se factorise en un produit de probabilités conditionnelles sur l'ensemble des sites de l'image :

$$P(O|\omega) = \prod_{(i,j)\in L} P(o_{i,j}|\omega_{i,j}).$$

Le second terme $P(\omega)$ représente la probabilité *a priori* de la configuration d'étiquettes. Il peut être assimilé à un terme de régularisation puisqu'il conduit à une certaine homogénéisation du champ d'étiquettes par la prise en compte du contexte local. Il s'interprête également comme un terme de transition puisqu'il est fonction de l'ensemble des probabilités de transition entre les sites au sein d'un même voisinage. Dans toute la suite, il sera noté $P^T(\omega)$.

À l'information de dépendance contextuelle locale introduite par les champs aléatoires de Markov et se matérialisant sous la forme du terme de transition, on ajoute une information de position exploitant des considérations spatiales : la probabilité d'un état va non seulement dépendre de la configuration de son voisinage immédiat $V(i,j)$, mais également de la position spatiale (i,j) considérée. $P(\omega)$ va s'exprimer de la façon suivante :

$$P(\omega) = P^T(\omega) \times \prod_{(i,j)\in L} P(\omega_{i,j}|(i,j)).$$

On note P^P le terme de position défini par :

$$P^P(\omega) = \prod_{(i,j)\in L} P(\omega_{i,j}|i,j).$$

On a ainsi $P(\omega) = P^T(\omega)P^P(\omega)$.

Originalité par rapport aux précédents travaux, le modèle de position est introduit essentiellement pour la segmentation d'images et plus particulièrement la structuration de documents qui nous interesse dans cette thèse. En effet, il permet de modéliser plus finement la structure physique d'un document en prenant en compte la position spatiale des différents états du modèle. Par exemple dans le cas des courriers manuscrits abordés au chapitre 5, l'état correspondant au champ "date, lieu" sera le plus souvent situé en haut de la page.

On introduit les MRF-P en modifiant la définition des MRF pour prendre en compte une dépendance spatiale.

Définition 2.3.1. ***Champ aléatoire de Markov-P (MRF-P)***
X est un champ aléatoire de Markov-P si et seulement si :

$$\begin{aligned} \forall \Gamma \subseteq L, \forall (i,j) \in L \setminus \Gamma \quad P(X_{i,j}|X_\Gamma) &= P(X_{i,j}|X_{V_{i,j}}, (i,j)), \\ &= P(X_{i,j}|X_{V_{i,j}}) \times P(X_{i,j}|(i,j)). \end{aligned}$$

La probabilité conditionnelle de la réalisation d'une variable aléatoire en un site, connaissant toutes les réalisations des autres variables aléatoires, ne dépend que des variables du voisinage du site et de la position spatiale du site considéré.

Trois jeux de paramètres interviennent donc pour la modélisation de l'image dans l'approche AMBRES : le premier correspond au terme d'attache aux données $P(O|\omega)$, le second au terme de position $P^P(\omega)$ et le dernier au terme de transition $P^T(\omega)$.

2.3.2 Paramètres du modèle markovien

2.3.2.1 Terme d'attache aux données

Le terme d'attache aux données ou densité d'observation permet de modéliser le processus de formation de l'image (observations) pour une configuration d'états donnée. Il est déterminé à partir des images déjà segmentées et est classiquement modélisé par des multigaussiennes calculées à partir d'un algorithme itératif EM. Elles sont de la forme :

$$P(o \mid \omega) = \sum_{k=1}^{M} c_k G(o, \mu_{k,\omega}, \Sigma_{k,\omega}),$$

où M est le nombre de gaussiennes, $G(o, \mu, \Sigma)$ est la valeur en o d'une gaussienne de moyenne μ et de matrice de covariance Σ (que l'on choisit diagonale en pratique), et où :

$$\sum_{k=1}^{M} c_k = 1.$$

L'algorithme d'estimation des paramètres c_k, $\mu_{k,\omega}$ et $\Sigma_{k,\omega}$ est un algorithme d'estimation par maximum de vraisemblance. Si on observe des échantillons $o = \{o_1, ..., o_N\}$

supposés générés par une loi de paramètre θ à déterminer, le problème s'exprime sous la forme :

$$\hat{\theta} = \arg\max_{\theta} \log[P(o|\theta)].$$

C'est un problème de calcul de maximum de vraisemblance dans le cas où les observations sont des données incomplètes. Une observation o est émise par l'un des M noyaux :

$$p(o) = \sum_{k=1}^{M} P(k)p(o|k).$$

– À l'étape E, on calcule la probabilité que le noyau k a émis o_i :

$$p_k^i = \frac{c_k G(o_i, \mu_k, \Sigma_k)}{\sum_{l=1}^{M} c_l G(o_i, \mu_l, \Sigma_l)}.$$

– À l'étape M, on obtient les nouveaux paramètres des noyaux :

$$\begin{aligned} c'_k &= \frac{1}{N}\sum_{i=1}^{N} p_k^i, \\ \mu'_k &= \frac{\sum_{i=1}^{N} p_k^i o_i}{\sum_{i=1}^{N} p_k^i}, \\ \Sigma'_k &= \frac{\sum_{i=1}^{N} p_k^i (o_i - \mu'_i)(o_i - \mu'_i)^t}{\sum_{i=1}^{N} p_k^i}. \end{aligned}$$

On itère les étapes E et M jusqu'à convergence.

L'apprentissage des paramètres liés au terme d'attache aux données consiste à déterminer les paramètres des multigaussiennes pour chaque classe. Ces paramètres sont : le nombre de gaussiennes M, les pondérations c_k associées et les paramètres de chaque gaussienne à savoir les moyennes et matrices de covariance.

2.3.2.2 Terme de transition

Le terme de transition $P^T(\omega)$ permet de prendre en compte l'information de dépendance locale. Il calcule ainsi les probabilités des sites d'être dans un certain état $\omega_{i,j} = s_k$ connaissant les états des sites voisins. Une solution pour calculer $P^T(\omega)$ est d'utiliser le théorème de Hammersley-Clifford qui établit l'équivalence entre un champ de Markov et un champ de Gibbs. Celui-ci permet d'écrire :

$$P^T(\omega) = \frac{1}{Z} e^{-\sum_{c \in C} V_c(\omega)},$$

où :

– C est l'ensemble des cliques associées au voisinage,
– V_c est le potentiel de la clique c,
– Z est la constante de normalisation telle que $\sum_{\omega} P^T(\omega) = 1$.

Si l'on se place dans le cadre d'un voisinage d'ordre 1, les cliques sont définies par :

$$\mathcal{C} = \mathcal{C}_0 \cup \mathcal{C}_1 \cup \mathcal{C}_2,$$

avec

$$\begin{array}{rcl} \mathcal{C}_0 & = & \{(i,j), 1 \leq i \leq n \text{ et } 1 \leq j \leq m\}, \\ \mathcal{C}_1 & = & \{(i,j),(i+1,j), 1 \leq i \leq n-1 \text{ et } 1 \leq j \leq m\}, \\ \mathcal{C}_2 & = & \{(i,j),(i,j+1), 1 \leq i \leq n \text{ et } 1 \leq j \leq m-1\}. \end{array}$$

Les potentiels de clique utilisés sont obtenus en calculant le logarithme négatif des termes d'interaction définis sur les cliques horizontale et verticale d'un voisinage d'ordre 1 et sont définis de la façon suivante :

$$V_c(\omega) = \begin{cases} -\log(P(\omega_k)) & \text{si } c \in \mathcal{C}_0 \text{ avec } c = (i,j) \text{ et } \omega_k = \omega_{i,j} \\ -\frac{1}{2}\log(I_v(\omega_k,\omega_l)) & \text{si } c \in \mathcal{C}_1 \text{ avec } \begin{cases} c = ((i,j),(i+1,j)) \\ \omega_l = \omega_{i,j} \text{ et} \\ \omega_k = \omega_{i+1,j} \end{cases} \\ -\frac{1}{2}\log(I_h(\omega_k,\omega_l)) & \text{si } c \in \mathcal{C}_2 \text{ avec } \begin{cases} c = ((i,j),(i,j+1)) \\ \omega_l = \omega_{i,j} \text{ et} \\ \omega_k = \omega_{i,j+1} \end{cases} \end{cases},$$

où I_v et I_h sont respectivement les termes d'interaction verticale et horizontale. Ils sont exprimés de la façon suivante :

$$I_v(\omega_k,\omega_l) = \frac{P\left(\begin{array}{|c|}\hline \omega_l \\ \hline \omega_k \\ \hline \end{array}\right)}{P(\omega_k)P(\omega_l)},$$

$$I_h(\omega_k,\omega_l) = \frac{P\left(\begin{array}{|c|c|}\hline \omega_l & \omega_k \\ \hline \end{array}\right)}{P(\omega_k)P(\omega_l)},$$

où

$$P\left(\begin{array}{|c|}\hline \omega_l \\ \hline \omega_k \\ \hline \end{array}\right) = P(\omega_{i,j} = \omega_k, \omega_{i-1,j} = \omega_l),$$

$$P\left(\begin{array}{|c|c|}\hline \omega_l & \omega_k \\ \hline \end{array}\right) = P(\omega_{i,j} = \omega_k, \omega_{i,j-1} = \omega_l).$$

Les paramètres à apprendre pour le modèle de transition sont donc les différents termes d'interaction obtenus grâce à un simple comptage des différentes transitions.

2.3.2.3 Terme de position

Le terme de position $P^P(\omega)$ a été introduit dans nos travaux afin de prendre en compte l'information de position spatiale. Il calcule les probabilités des sites d'être dans un certain état $\omega_{i,j} = s_k$ suivant la position spatiale du site (i,j) :

$$P^P(\omega) = \prod_{(i,j)\in L} P(\omega_{i,j}|(i,j)).$$

– Dans les cas de structuration de document, le modèle de position permet d'ajouter une information sur la position relative des différents états dans l'image. Notons que dans ce type d'application un état correspond à un champ du document à localiser. Par exemple, dans le cas de la structuration de courriers manuscrits la signature va se trouver très souvent en bas de l'image et donc cette information sera prise en compte dans le modèle de position.
Il est, dans ce cas, déterminé à partir des vérités terrain et nécessite donc une phase d'apprentissage.
Le modèle de position donne la probabilité d'apparition d'un état suivant la position spatiale horizontale et verticale du site considéré (i,j). On suppose qu'il y a une indépendance entre la direction horizontale et la direction verticale. On décompose ainsi $P^P(\omega)$ de la façon suivante :

$$P^P(\omega) = \prod_{(i,j)\in L} P(\omega_{i,j}|i) \prod_{(i,j)\in L} P(\omega_i|j).$$

On introduit ainsi les termes de position horizontale et verticale respectivement notés $P_h^P(\omega)$ et $P_v^P(\omega)$ et définis par :

$$P_h^P(\omega) = \prod_{(i,j)\in L} P(\omega_{i,j}|i),$$

$$P_v^P(\omega) = \prod_{(i,j)\in L} P(\omega_{i,j}|j).$$

Pour gérer les images de différentes tailles, on introduit les valeurs normalisées de i et j dans le calcul de $P_h^P(\omega)$ et $P_v^P(\omega)$. Ainsi, on a finalement :

$$P_h^P(\omega) = \prod_{(i,j)\in L} P(\omega_{i,j}|i'),$$

$$P_v^P(\omega) = \prod_{(i,j)\in L} P(\omega_{i,j}|j').$$

avec $0 \leq i' \leq 1$ et $0 \leq j' \leq 1$.
Classiquement, nous choisissons de modéliser ces deux termes de position par des multigaussiennes.
Les paramètres à déterminer lors de l'apprentissage du modèle de position sont ainsi les paramètres des multigaussiennes liés à chacun de ces deux termes à savoir les moyennes et matrices de covariance.

– Dans les cas de reconnaissance de caractères ou de mots manuscrits, nous verrons dans la section 2.4.1 que l'objectif est de segmenter les formes suivant la direction

et la position des traits dans l'image. Le modèle de position va permettre d'apporter cette information de position à chaque état du modèle. Les segmentations en états des images n'étant pas connues au départ, le modèle de position est imposé au système et consiste à limiter la propagation d'un état dans une zone restreinte de l'image au cours des itérations de l'algorithme d'apprentissage.

2.3.3 Décodage de la configuration la plus probable : programmation dynamique 2D

Pour obtenir la configuration optimale, *i.e.* réaliser la maximisation de $P(O|\omega)P(\omega)$, on utilise la programmation dynamique $2D$ proposée par Geoffrois en 1998 [32, 33]. C'est une extension naturelle de l'algorithme classique de programmation dynamique 1D dont le concept a été proposé dès 1957 pour une tâche de contrôle optimal [7]. Il a été introduit ensuite en reconnaissance de la parole en 1968 [98] et est toujours resté au cœur des meilleurs systèmes depuis.

2.3.3.1 Principe de la programmation dynamique 2D

La programmation dynamique 2D est un cas particulier de l'approche "diviser pour régner". Son principe général est le suivant : si R_1 et R_2 forment une partition de l'image I, pour déterminer la configuration optimale, au lieu d'explorer toutes les configurations, seule la configuration optimale des intérieurs de R_1 et R_2 doit être trouvée pour chaque configuration de la frontière. Cela réduit significativement le nombre de configurations à envisager.

Démonstration :

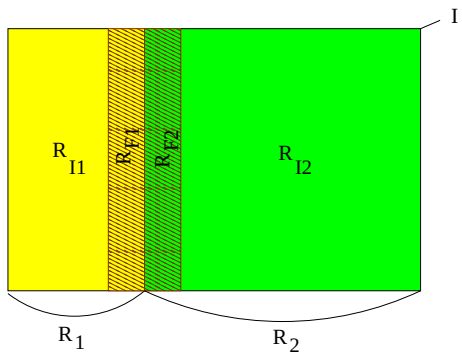


Fig. 2.3. Notations

Soient R_1 et R_2 telles que $R_1 \cup R_2 = I$ où I est l'image traitée.

Soient R_{F1} et R_{F2} les frontières de R_1 et R_2 respectivement. On appelle frontière d'une région l'ensemble des pixels de la région appartenant à une clique contenant à la fois des pixels de la région et des pixels de la région voisine.

Soient R_{I1} et R_{I2} les intérieurs de R_1 et R_2 respectivement. On appelle intérieur d'une région l'ensemble des pixels n'appartenant pas à la frontière.

Pour une configuration donnée ω de l'image, on note ω_i, ω_{Fi} et ω_{Ii} les restrictions de cette configuration à R_i, R_{Fi} et R_{Ii}. De plus, on note Ω_i la restriction de Ω à R_i, O_i la restriction de O à R_i et L_i la restriction de L à R_i.

On cherche à trouver la configuration optimale $\hat{\omega}$, telle que :

$$\begin{aligned} \hat{\omega} &= \arg\max_{\omega\in\Omega} P(O|\omega)P(\omega), \\ &= \arg\min_{\omega\in\Omega} -\log(P(O|\omega)P(\omega)), \\ &= \arg\min_{\omega\in\Omega} U(\omega). \end{aligned}$$

Pour $\omega = \omega_1 \cup \omega_2$, $U(\omega)$ s'écrit :

$$U(\omega) = U(\omega_1) + I(\omega_{F1}, \omega_{F2}) + U(\omega_2).$$

On considère maintenant une configuration ω' telle que :

$$\omega'_{F1} = \omega_{F1} \text{ et } \omega'_{F2} = \omega_{F2}.$$

On a donc $I(\omega_{F1}, \omega_{F2}) = I(\omega'_{F1}, \omega'_{F2})$, ainsi :

$$\text{si } U(\omega_1) < U(\omega'_1) \text{ et } U(\omega_2) < U(\omega'_2), \text{ alors } U(\omega) < U(\omega').$$

Pour $(\omega_{F1}, \omega_{F2})$ donnés, on obtient finalement :

$$\text{si } \hat{\omega_1} = \arg\min_{\omega_1\in\Omega_1} U(\omega_1) \text{ et } \hat{\omega_2} = \arg\min_{\omega_2\in\Omega_2} U(\omega_2), \text{ alors } \hat{\omega_1} \cup \hat{\omega_2} = \hat{\omega}.$$

Donc pour une configuration donnée de la frontière, il suffit de trouver les configurations optimales des intérieurs.

Ce processus est récursif. De même que la configuration optimale de I est aisément calculée à partir des configurations optimales de R_1 et R_2, la configuration optimale de R_1 peut être aisément calculée à partir des configurations optimales de $R_{1.1}$ et $R_{1.2}$ si on considère que $R_{1.1}$ et $R_{1.2}$ forment une partition de R_1. Plus généralement, pour une région R_k, sa configuration optimale peut être déterminée à partir des configurations optimales de deux sous régions $R_{k.1}$ et $R_{k.2}$, et ainsi de suite jusqu'aux régions élémentaires d'un seul pixel.

Appliquer l'algorithme de programmation dynamique 2D sur une image consiste :
- à initialiser la configuration de tous les sites, chacun ayant N configurations possibles (N états),
- à fusionner progressivement les sites entre eux en ne gardant à chaque fusion que la configuration optimale pour chaque configuration de la frontière de chaque région et la vraisemblance associée.

C'est la stratégie de fusion qui va déterminer la façon dont seront regroupés les sites.

2.3.3.2 Stratégie de fusion et complexité

La stratégie de fusion spécifie l'ordre dans lequel on fusionne les sites jusqu'à parcourir l'image entière pour obtenir la configuration optimale. En théorie, sans élagage cet ordre n'a aucune influence sur la solution trouvée. Il est cependant déterminant en pratique pour le temps de calcul et l'espace mémoire utilisé.

Dans notre approche, toutes les stratégies de parcours de pixels pour la programmation dynamique 2D sont envisageables. On définit, ci-dessous, les stratégies utilisées par la suite avec la complexité associée. On rappelle qu'en considérant toutes les configurations possibles on a une complexité en $O(N^{n^2})$.

La stratégie de fusion "escargot" Elle consiste à parcourir les pixels les uns après les autres du haut à gauche vers le centre, de l'extérieur vers l'intérieur (figure 2.4). La complexité de la programmation dynamique utilisant cette stratégie de fusion est en $O(N^{4n-4})$, si N est le nombre d'états et n^2 la taille de l'image.

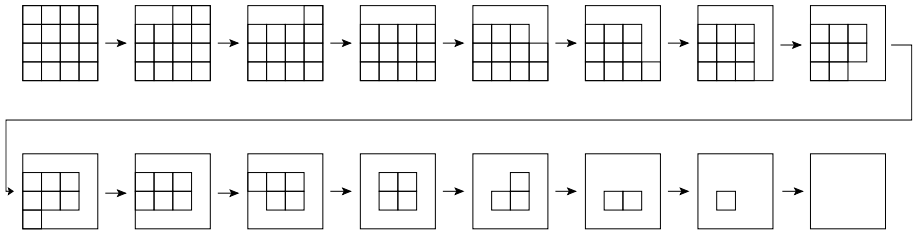


Fig. 2.4. La stratégie de fusion "escargot"

La stratégie de fusion "linéaire haut-bas" Elle consiste à parcourir les pixels les uns après les autres du haut à gauche vers le bas, ligne après ligne (figure 2.5). La complexité de la programmation dynamique utilisant cette stratégie de fusion est en $O(n^2N^{n+1})$, si N est le nombre d'états et n^2 la taille de l'image.

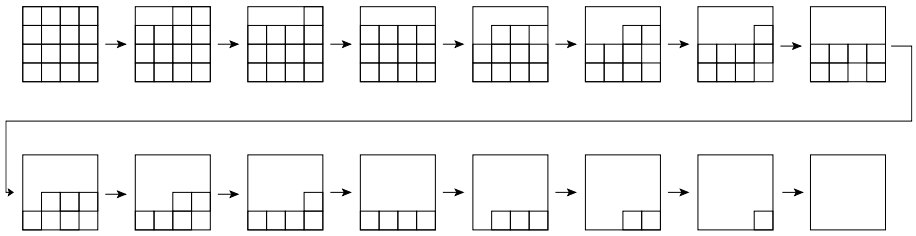


Fig. 2.5. La stratégie de fusion "linéaire haut-bas"

2.3.3.3 Élagage

L'algorithme de programmation dynamique 2D réduit drastiquement le nombre de configurations à tester mais il reste encore trop important pour que chaque configuration soit envisagée.

Une stratégie d'élagage est ainsi adoptée à plusieurs niveaux :
- les configurations dont la différence de coût avant et après fusion est au dessus d'un certain seuil sont élaguées,
- un nombre maximal de configurations est conservé à chaque étape.

Deux paramètres doivent ainsi être ajustés : le seuil d'élagage et le nombre maximum de configurations conservées après chaque fusion d'un nouveau site. Ce réglage est assez délicat et doit être bien réalisé. En effet, de lui dépend l'efficacité de la programmation dynamique 2D et donc de l'optimalité de la configuration finale.

2.3.4 Reconnaissance de formes

Dans cette section, nous montrons comment exploiter AMBRES pour la reconnaissance de formes.

Dans le cadre d'une tâche de reconnaissance de formes on cherche à déterminer à quelle classe appartient chaque image de la base de test (figure 2.6). Pour cela, on cherche dans un premier temps la configuration optimale $\hat{\omega_c}$ de la grille d'états pour chaque classe c. Puis, on détermine la classe la plus probable $\hat{c}$. On a ainsi :

$$\begin{aligned}
\hat{c} &= \arg\max_{c \in C} P(c|O), \\
\hat{c} &= \arg\max_{c \in C} P(O|c)P(c), \\
\hat{c} &= \arg\max_{c \in C} P(O|c,\hat{\omega})P(\hat{\omega}|c)P(c), \\
\hat{c} &= \arg\max_{c \in C} P(O|\hat{\omega_c})P(\hat{\omega_c})P(c).
\end{aligned}$$

Un modèle markovien doit ainsi être construit pour chaque classe c. Il est appris à partir des images de la base d'apprentissage. Comme les vérités terrain ne sont pas connues, un algorithme itératif de type EM est utilisé pour apprendre les paramètres du modèle. De la même façon que l'algorithme EM nous permet d'estimer les paramètres des multigaussiennes pour le calcul des densités d'observation et du modèle de position, il va nous permettre de déterminer la configuration optimale de la grille d'états de chacune des images de la base d'apprentissage et donc au final les différents paramètres des modèles.

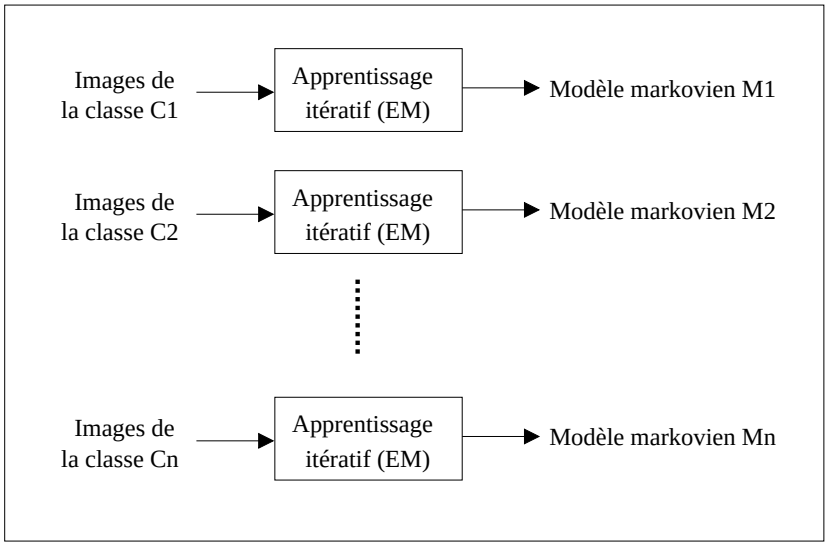
Soit $Q(\omega, \omega^i)$ la probabilité d'être dans la configuration ω sachant que la précédente configuration optimale était ω^i (c'est elle qui a permis le calcul des derniers paramètres). L'algorithme EM va donc se réaliser de la façon suivante :

- **Initialisation** : On initialise la configuration initiale ω^0, le seuil de convergence noté T et $i = 0$.

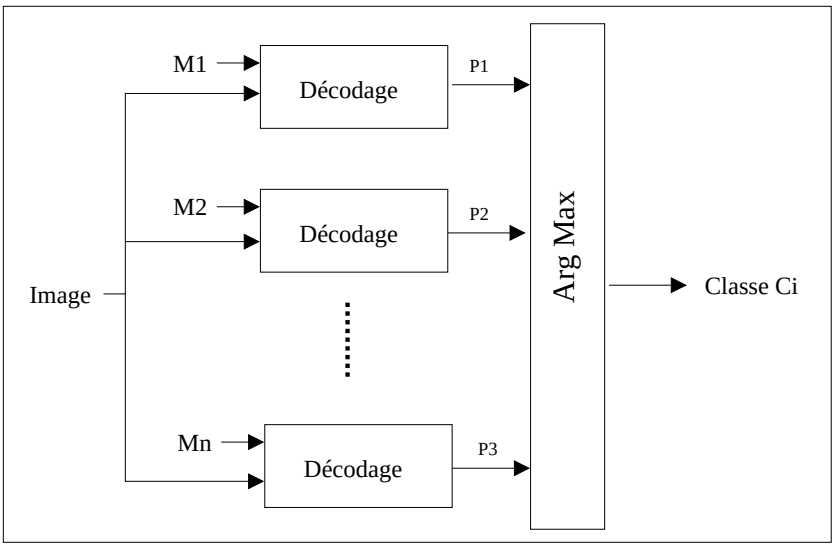
- **Itérations** : Tant que $Q(\omega^{i+1}, \omega^i) - Q(\omega^i, \omega^{i-1}) > T$ faire
  i+1 ;
  Étape E : calculer les $Q(\omega, \omega^i)$
  Étape M : $\omega^{i+1} \leftarrow \arg\min_\omega Q(\omega, \omega^i)$

– **Finalisation** : $\hat{\omega} \leftarrow \omega^{i+1}$

Dans le cadre de la reconnaissance de formes, AMBRES sera appliquée dans ces travaux à la reconnaissance de l'écriture manuscrite au niveau caractère (chapitre 3) et au niveau mot (chapitre 4). Nous verrons également dans le chapitre 6 une application à la reconnaissance de logos.



APPRENTISSAGE



RECONNAISSANCE

Fig. 2.6. Tâche de reconnaissance de formes

2.3.5 Segmentation d'image

Dans cette section, nous montrons comment exploiter AMBRES pour la segmentation d'image.

La segmentation d'image vise à rassembler les pixels de l'image entre eux suivant des critères prédéfinis. Par exemple dans le cadre de la structuration de courriers manuscrits on va chercher à segmenter les images en différents champs correspondant aux coordonnées de l'expéditeur, à la date, à la signature, ... Chacun de ces champs correspondra à un état dans le modèle markovien.

On construit ainsi dans un premier temps un unique modèle markovien appris à partir des vérités terrain et des images de la base d'apprentissage. Dans le cas de la structuration de documents celui-ci modélisera la structure physique du document. À partir de ce modèle, on peut déterminer pour chaque image de la base de test la configuration optimale du champ de Markov qui correspond à la segmentation recherchée (figure 2.7).

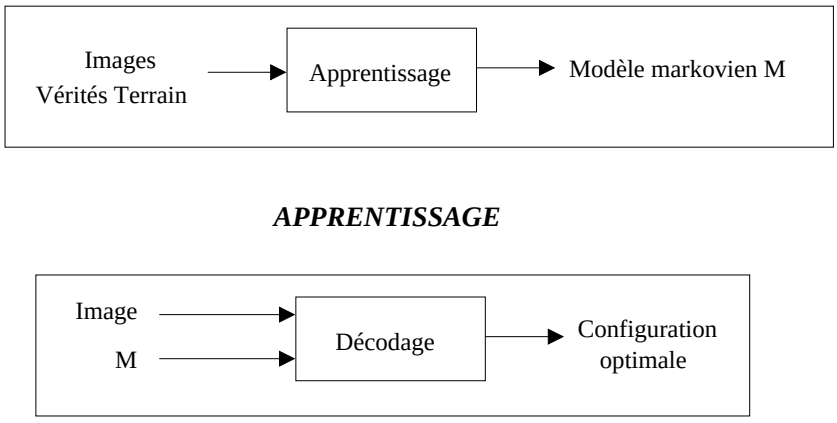


Fig. 2.7. Tâche de segmentation d'image

Dans le cadre de la segmentation d'image, AMBRES sera apliquée à la structuration de documents manuscrits dans le chapitre 5.

2.4 Choix des primitives dans le cadre de AMBRES

La phase d'extraction des primitives est une étape très importante autant pour la reconnaissance de formes que pour la segmentation d'images. Deux types d'approche peuvent être envisagés pour le choix des primitives :

- soit on utilise directement la valeur des pixels,
- soit on réalise une phase de prétraitement sur les pixels pour en extraire une information plus riche.

Alors que les champs de Markov utilisent classiquement la première approche, nous proposons d'exploiter la seconde. Nous allons décrire les primitives considérées dans le

cadre des deux applications de AMBRES étudiées dans cette thèse : la reconnaissance de l'écriture manuscrite et la structuration de documents manuscrits.

En particulier nous allons détailler l'étude des primitives spectrales locales utilisées dans le cadre de la reconnaissance de l'écriture manuscrite afin de sélectionner par la suite (chapitre 3) les paramètres et les coefficients les plus pertinents. Cette étape clé n'avait pas été approfondie dans les travaux de Chevalier et le choix des coefficients avait été fait de façon intuitive.

2.4.1 Cas de la reconnaissance de l'écriture manuscrite

L'écriture est une forme constituée d'un ensemble de traits de différentes directions et situés à différents endroits dans l'image. L'objectif va donc être de segmenter les images de caractères ou de mots en fonction de la direction et de la position des traits dans l'image. Les $n_x \times n_y$ zones ainsi obtenues correspondent aux différents états du champ de Markov qui seront donc caractéristiques d'une direction et d'une position dans l'image ; par exemple un état E_k sera caractéristique d'une direction horizontale située dans la partie haute à gauche dans l'image (figure 2.8).

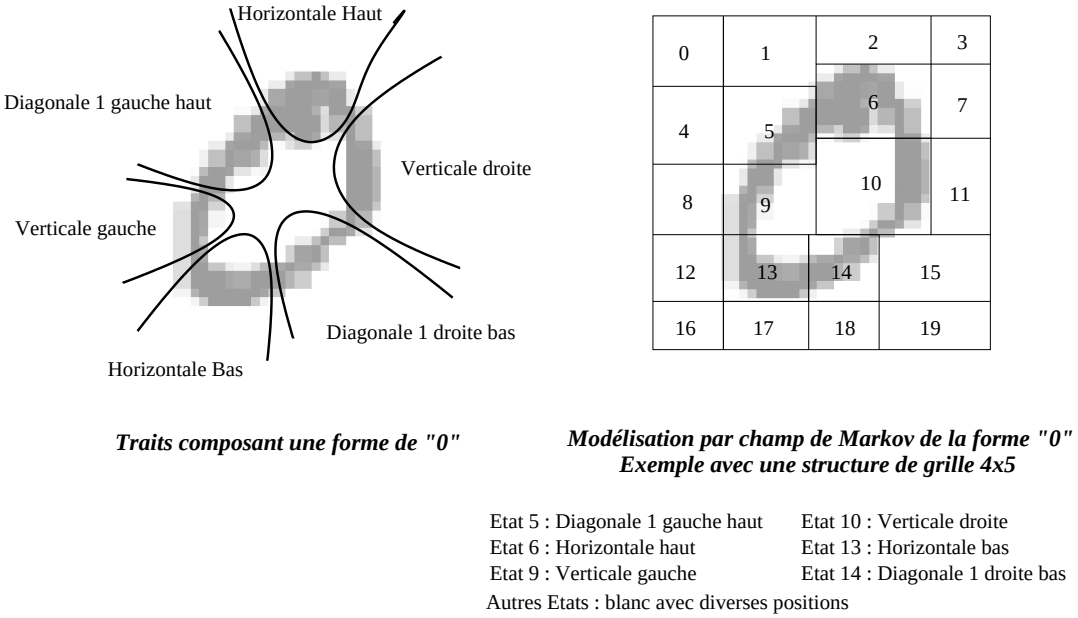


Fig. 2.8. Modélisation de la forme "0"

L'information de position est obtenue à partir du modèle de position : il devra permettre de conserver une structure de grille. L'information de direction est quant à elle extraite à partir de primitives directionnelles.

Nous avons choisi les primitives spectrales locales [17] qui outre leur capacité à extraire une information de direction, ont l'avantage d'être robustes au bruit et à la variabilité du signal.

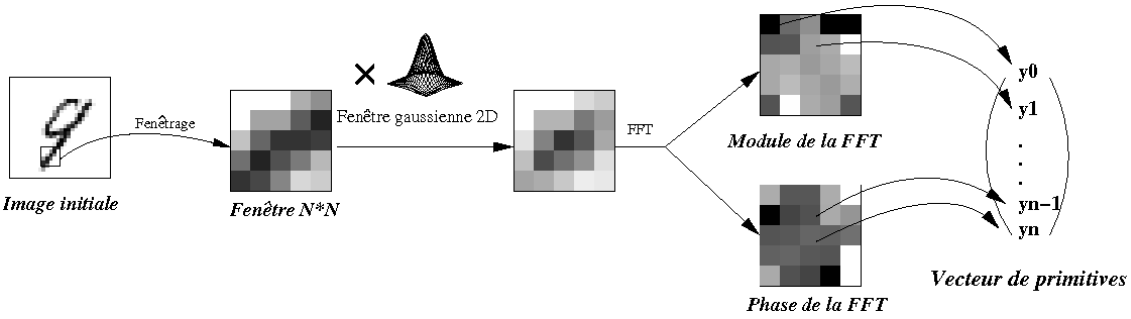


Fig. 2.9. Processus d'extraction des primitives spectrales locales

Les primitives spectrales locales consistent (figure 2.9) à calculer une FFT dans une fenêtre gaussienne glissante centrée autour de pixels uniformément répartis dans l'image (un sur deux par exemple) (figure 2.10) et à ne garder qu'un certain nombre de coefficients du module et de la phase. Le fenêtrage par une gaussienne a été introduit afin de réduire les effets de bords dus à la FFT (il est possible de choisir une autre fenêtre : Hamming, ...). En effet, ces effets de bords sont d'autant plus gênants que l'on cherche à extraire une information directionnelle (les bords de l'image ajoutent de l'information parasite sur les directions verticale et horizontale).

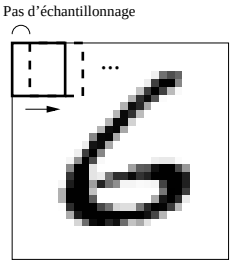


Fig. 2.10. Fenêtre glissante

Dans le cas d'une fenêtre 7×7, les différents coefficients du module de la FFT (même principe pour la phase) sont nommés comme présenté figure 2.11. On ne représente ici que les coefficients pour des fréquences en y négatives. En effet, il y a une symétrie centrale, c'est-à-dire :

$$D^M_{i,j} = D^M_{-i,-j} \text{ (modules)},$$
$$D^\varphi_{i,j} = D^\varphi_{-i,-j} \text{ (phases)}.$$

On note :

$$V^M_i = D^M_{i,0} \text{ et } V^\varphi_i = D^\varphi_{i,0},$$
$$H^M_j = D^M_{0,j} \text{ et } V^\varphi_j = D^\varphi_{0,j}.$$

Les différents coefficients peuvent être interprétés de la façon suivante :
– O est la moyenne sur l'imagette 7×7,

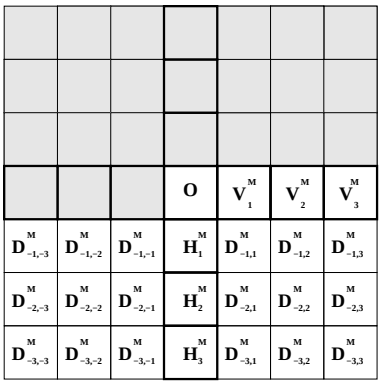


Fig. 2.11. Positionnement et nom des différents coefficients du module de la FFT

- V_1^M, H_1^M, $D_{-1,1}^M$ et $D_{-1,-1}^M$ sont les coefficients du module correspondant aux quatre directions principales (respectivement verticale, horizontale, première diagonale et deuxième diagonale),
- les coefficients V_k^M, H_k^M, $D_{-k,k}^M$ et $D_{-k,-k}^M$ $(k = 2, 3)$ correspondent aux mêmes directions, mais à une fréquence multiple,
- enfin, les autres coefficients correspondent à d'autres directions (figure 2.12).

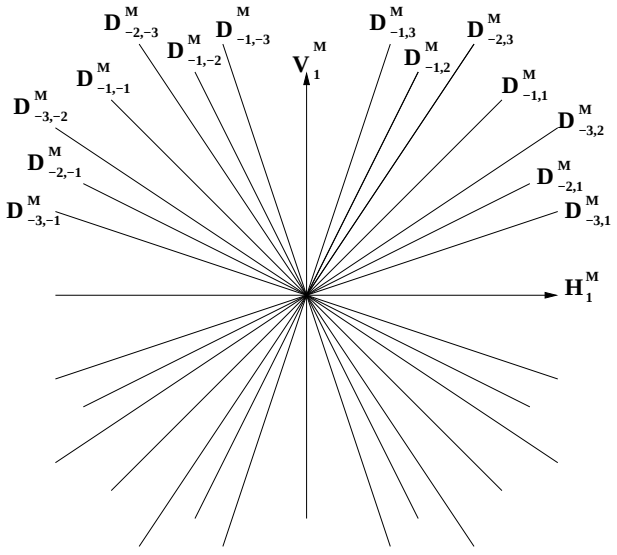
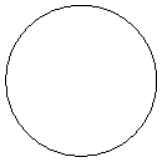


Fig. 2.12. Direction des coefficients

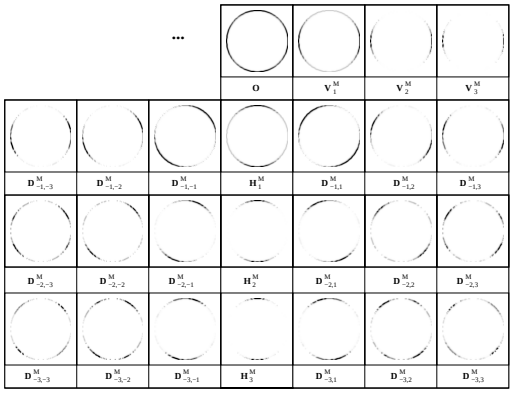
Remarque : un trait dans l'image d'origine se traduit dans le domaine de Fourier par un trait perpendiculaire passant par l'origine.

Illustration sur le cercle On considère l'image d'un cercle de faible épaisseur ainsi que celle d'un cercle assez épais afin de mieux constater les informations présentes dans

chacun des coefficients du module de la FFT. Les différents coefficients du module sont représentés figure 2.13 et 2.14.

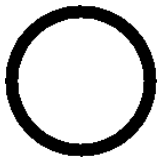


(a) Cercle d'épaisseur 1 pixel (Dim : 122 × 122)

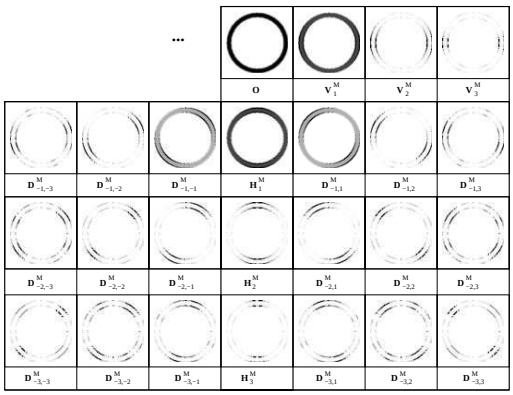


(b) Illustration des différents coefficients du module de la FFT

Fig. 2.13. Illustration des différents coefficients du module de la FFT du cercle



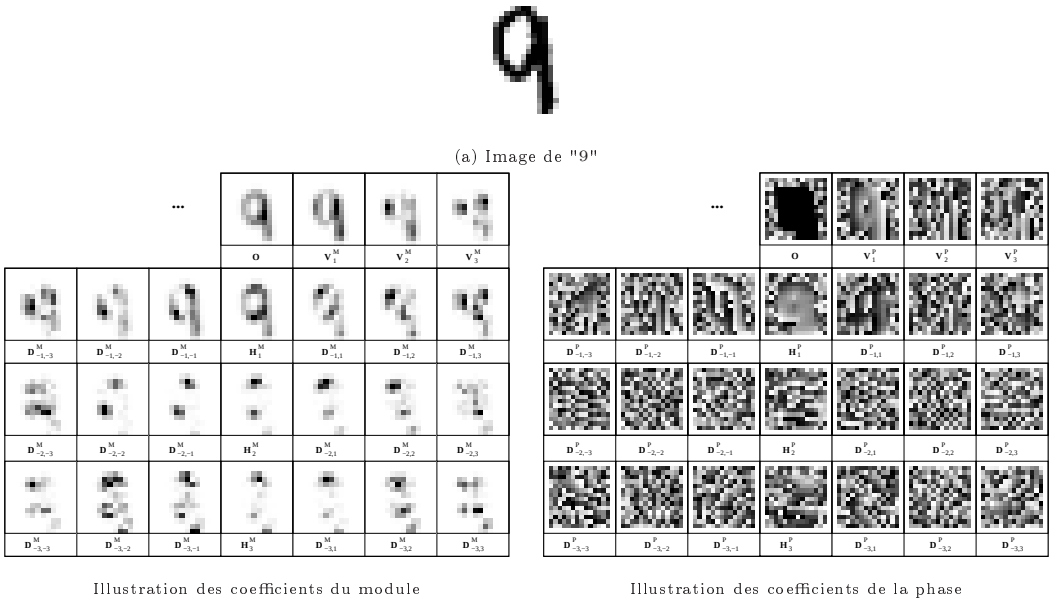
(a) Cercle d'épaisseur 10 pixels (Dim : 122 × 122)



(b) Illustration des différents coefficients du module de la FFT

Fig. 2.14. Illustration des différents coefficients du module de la FFT du cercle épais

Illustration sur une image de chiffres On considère une image du chiffre "9". Les différents coefficients du module et de la phase de la FFT sont représentés figure 2.15.



(a) Image de "9"

Illustration des coefficients du module

Illustration des coefficients de la phase

Fig. 2.15. Illustration des différents coefficients du module et de la phase de la FFT de l'image de "9"

On constate bien que chaque coefficient du module donne une information différente sur la direction des traits. On pourra, en particulier, observer les coefficients (V_1^M, H_1^M, $D_{-1,1}^M$ et $D_{-1,-1}^M$) correspondant aux quatre directions principales. Les coefficients de la phase semblent être moins facilement interprétables que les modules. On verra dans le chapitre 3 leur influence dans la reconnaisance.

Un certain nombres de paramètres rentrent ainsi en compte dans le calcul des primitives spectrales locales : la taille de la fenêtre, le pas d'échantillonnage et le choix des coefficients. Nous étudierons leur influence sur les performances du système dans le prochain chapitre.

2.4.2 Cas de la structuration de documents

Dans le cadre de la structuration de documents manuscrits, les primitives vont nous permettre de détecter l'écriture manuscrite. La détection de la direction des traits ne semble plus pertinente pour cette tâche. Nous avons donc choisi d'utiliser la valeur moyenne des niveaux de gris dans une fenêtre glissante et avons bien vérifié expérimentalement que les performances obtenues étaient meilleures qu'avec les primitives spectrales locales. La taille de la fenêtre et le pas d'échantillonnage sont les deux paramètres des primitives utilisées.

2.5 Conclusion

La méthodologie d'AMBRES est une généralisation de l'approche proposée par Chevalier [16] dédiée à la reconnaissance de chiffres manuscrits utilisant les champs de Markov et la programmation dynamique 2D. L' apport essentiel d'AMBRES est donc la généralité puisque nous avons montré que AMBRES pouvait être exploitée pour une tâche complète d'analyse de documents permettant d'aborder à la fois la reconnaissance et la structuration. De plus, la définition d'un champ de Markov-P permet maintenant d'introduire en plus d'une dépendance contextuelle une dépendance spatiale se matérialisant sous la forme d'un modèle de position. Celui-ci permettra essentiellement de modéliser plus finement la structure physique d'un document en modélisant la position spatiale des différents champs.

Les primitives utilisées pour chacune des applications de AMBRES réalisées dans ces travaux ont été présentées. En particulier les primitives spectrales locales choisies pour la reconnaissance de l'écriture manuscrite ont fait l'objet d'une étude approfondie, ce qui n'avait pas été réalisée jusqu'à maintenant. L'aspect expérimental concernant l'influence de chacun des paramètres liés à leur extraction sera considéré dans le cadre de la reconnaissance de caractères manuscrits dans le chapitre 3.

L'étude de l'approche AMBRES a montré qu'elle nécessitait le réglage de quelques paramètres liés à la topologie du champ de Markov, à la programmation dynamique 2D et à l'extraction des primitives. Ces paramètres devront faire l'objet d'une étude minutieuse pour garantir l'optimalité du système. Dans le prochain chapitre, nous allons donc présenter la méthodologie mise en œuvre pour réaliser cette optimisation. De plus une analyse approfondie des performances du système et des erreurs sera également réalisée afin d'envisager des pistes d'amélioration.

Chapitre 3

AMBRES appliquée à la reconnaissance de caractères manuscrits

3.1 Introduction

Dans ce chapitre, AMBRES est appliquée à la reconnaissance de caractères manuscrits. Afin de mettre au point le système de reconnaissance, la base MNIST de chiffres manuscrits a été exploitée. Cette base, en plus d'être de grande taille, a l'avantage d'être publique. Elle est couramment utilisée dans la littérature et on constate que la plupart des expériences sont effectuées directement sur les données de test ce qui engendre du surapprentissage. Dans notre étude nous n'exploitons la base de test que pour l'évaluation finale de notre système et utilisons une base de validation pour les évaluations intermédiaires.

Réalisé par un algorithme du type EM, l'apprentissage du modèle markovien associé à chaque classe de caractère nécessite au préalable le réglage d'un certain nombre de paramètres liés à la topologie du champ de Markov, à la programmation dynamique 2D et au vecteur de primitive. L'optimisation de l'ensemble des paramètres doit ainsi permettre l'optimisation des performances du système. L'étude du choix des coefficients du vecteur de primitives spectrales locales est en particulier essentielle : la bonne segmentation en états des formes suivant la position et la direction des traits dans l'image dépend de la pertinence de ce vecteur.

Une fois l'optimisation du système réalisée, une analyse des erreurs commises et des modèles obtenus conduit à énoncer des propositions d'amélioration et perspectives d'évolution. Ainsi, la combinaison de systèmes obtenus avec différents vecteurs de primitives est-elle exploitée pour diminuer le taux d'erreur. De plus, une technique de division et fusion d'états est proposée afin d'optimiser la topologie du champ de Markov.

Ainsi, l'apport principal de ce chapitre réside-t-il dans la mise en œuvre d'une méthodologie pour l'optimisation des paramètres et pour l'étude des performances du système

et des erreurs permettant ainsi l'amélioration du système de reconnaissance de caractère initialement proposé par Chevalier. Nous verrons en effet que nous parvenons à diminuer le taux d'erreur de 35%.

Le chapitre s'organise comme suit. Nous détaillons tout d'abord le système de reconnaissance de caractères manuscrits basé sur l'approche AMBRES. Puis, nous décrivons la base de chiffres manuscrits MNIST. Nous exposons ensuite les diverses expériences menées sur la base de développement pour la mise au point de l'algorithme avec notamment le réglage des différents paramètres. Cette étape est suivie d'une analyse des performances du système ainsi que d'une analyse des erreurs. Cela nous amène à proposer la combinaison de système comme amélioration du système et la division et fusion d'états comme perspective d'évolution. Enfin, les résultats sont donnés sur la base de test et une tâche de reconnaissance de lettres isolées est menée afin de montrer la généralité de l'approche à tout type de caractère manuscrit.

3.2 Mise en oeuvre de la reconnaissance de caractères manuscrits

Dans cette section, nous appliquons AMBRES à la reconnaissance de caractères et décrivons en particulier le déroulement des phases d'apprentissage et de reconnaissance.

3.2.1 Apprentissage

La phase d'apprentissage consiste à construire un modèle markovien pour chaque classe de caractères. Le principe est de partir d'une segmentation initiale qui va être affinée grâce à un algorithme itératif de type EM au cours duquel les paramètres associés aux différents états sont calculés (figure 3.1).

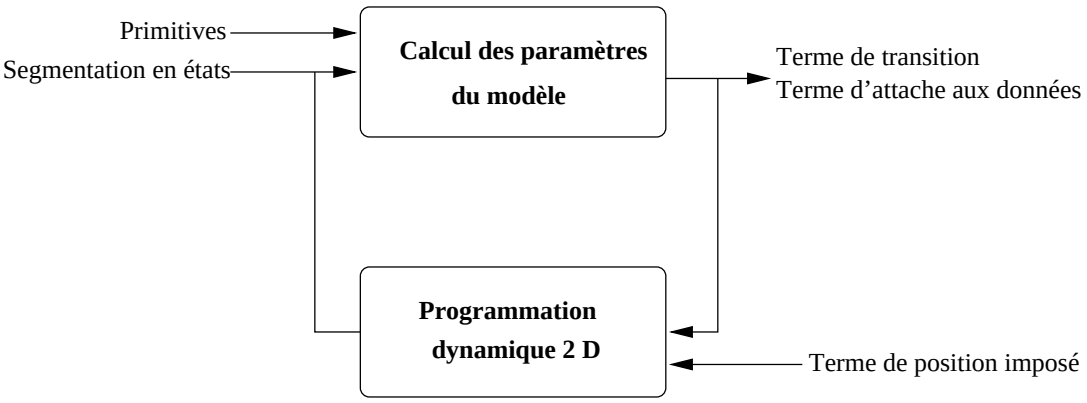


Fig. 3.1. Apprentissage d'un modèle de chiffre

L'algorithme d'apprentissage se divise donc en deux grandes étapes.

1. **Initialisation :**

   Pour l'initialisation des paramètres, on utilise une segmentation initiale uniforme $n_x \times n_y$ (figure 3.2). Celle-ci va permettre de calculer les premières probabilités de transition obtenues en comptant les transitions observées ; une valeur non nulle mais faible est cependant affectée aux transitions non observées. Elle va également permettre, associée aux primitives, de déterminer les densités d'observation. D'autre part, nous rappelons que le modèle de position est quant à lui imposé.

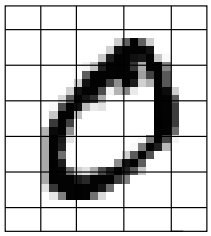


Fig. 3.2. Segmentation initiale uniforme avec l'exemple de la grille 5×7

2. **Itération :**

   Grâce aux paramètres du modèle déterminés à la précédente itération (ou initialisation), les images sont segmentées en états par la programmation dynamique 2D. Les paramètres du modèle sont alors réappris sur les nouvelles segmentations ainsi obtenues. Ce processus est répété jusqu'à convergence de l'algorithme.

3.2.2 Reconnaissance

Durant la phase de reconnaissance, on calcule les vraisemblances des images pour chacun des dix modèles obtenus dans la phase d'apprentissage. La vraisemblance des images est calculée sur la configuration optimale de la grille d'états donnée par la programmation dynamique 2D. Puis, on sélectionne la classe qui fournit le meilleur score.

L'optimisation des paramètres du système doit se faire à partir d'une base de données. Nous avons choisi pour cela d'exploiter la base MNIST de chiffres manuscrits qui est une base publique.

3.3 Base de données MNIST

3.3.1 Présentation

Nous utilisons la base de chiffres manuscrits MNIST disponible librement et gratuitement sur Internet [49]. C'est une base standard, publique, composée de 70000 images de chiffres extraits de la base NIST (voir figure 3.3) et divisée en une base d'apprentissage (60000 images) et une base de test (10000 images). Les spécificités de la base de données sont :

– toutes les images ont la même taille (28×28),

- les échantillons sont centrés et un cadre blanc de largeur 4 pixels est gardé autour des caractères,
- des niveaux de gris résultent des prétraitements de normalisation en taille,
- les bases d'apprentissage et de test ont été écrites par des scripteurs distincts.

Fig. 3.3. Échantillons de la base MNIST

L'optimisation des paramètres a été effectuée uniquement sur la base d'apprentissage découpée en deux parties : une partie de cette base appelée base de développement est utilisée pour l'apprentissage des modèles, et l'autre appelée base de validation pour leur validation. La base de test n'est donc pas utilisée pour l'optimisation des paramètres, mais uniquement pour l'évaluation finale des modèles. La partition ainsi réalisée de la base MNIST est illustrée figure 3.4.

3.3.2 État de l'art sur la base MNIST

Un tableau comparatif actualisé des principales méthodes testées sur la base MNIST est disponible sur le site de la base [49]. Nous nous proposons, ici, de donner un aperçu des performances réalisées sur cette base suivant la méthode employée (tableau 3.1). En l'absence de vraies campagnes d'évaluation il est assez délicat de comparer des résultats même obtenus sur une même base. En effet, les protocoles d'évaluation menés par les différentes équipes de recherche sont assez différents et il n'est pas rare de constater que l'optimisation du système se fait directement sur la base de test ce qui engendre du surapprentissage.

On constate néanmoins que les meilleures performances sont obtenues avec des réseaux de neurones ou des SVM qui semblent être très performants pour la reconnaissance

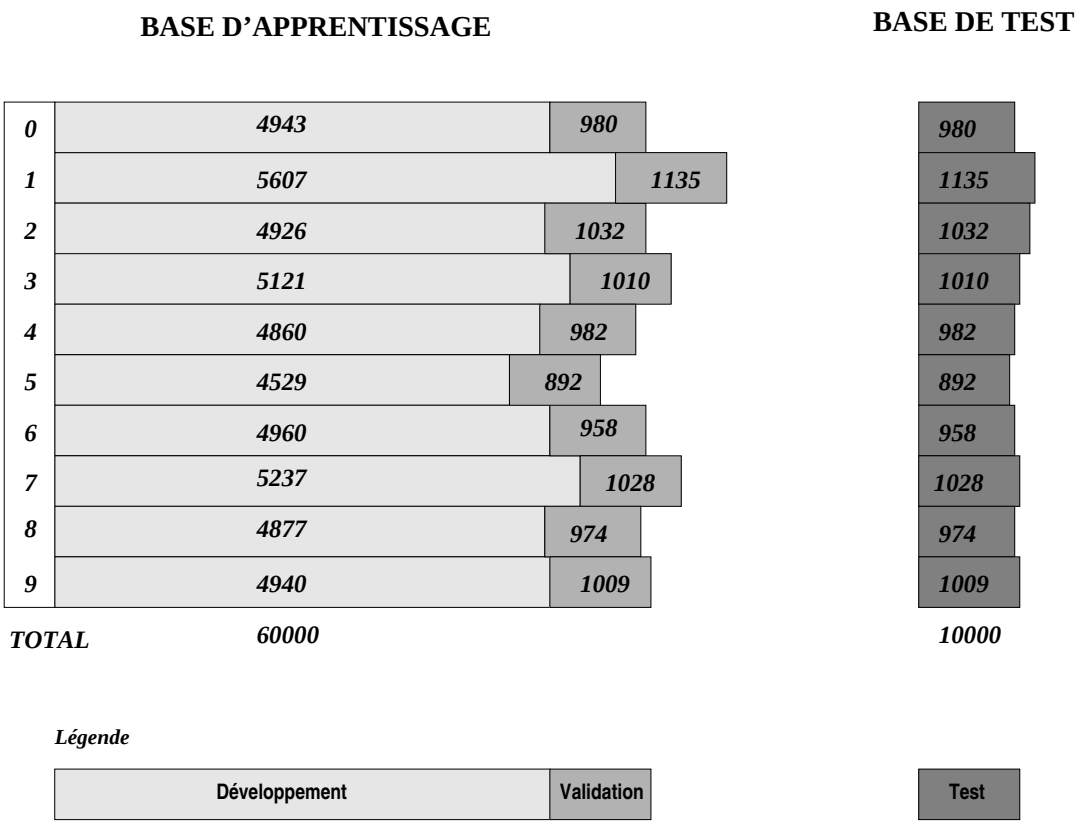


Fig. 3.4. Base de développement, validation et test

de caractères mais souvent non applicables et non appliqués à une tâche de reconnaissance de mots et ont donc l'inconvénient majeur d'être très spécifiques.

Tab. 3.1. État de l'art sur la base MNIST

Méthode	Taux d'erreurs	Référence
k-ppv par sous-voisinages emboîtés	1.72%	Taconet *et al.*, 2006 [85]
Champ de Markov et PD2D	2.74%	Chevalier, 2004 [16]
Méthode de l'Hyperplan ou HKNN	1.2%	Vincent et Bengio, 2001 [95]
Méthode de la distance tangente	1.1%	LeCun *et al.*, 1998 [50]
Réseau de neurones convolutionnel LeNet-1	1.7%	LeCun *et al.*, 1998 [50]
Réseau de neurones convolutionnel LeNet-4	1.1%	LeCun *et al.*, 1998 [50]
Réseau de neurones convolutionnel LeNet-5	0.8%	LeCun *et al.*, 1998 [50]

3.4 Choix des paramètres de l'algorithme

L'apprentissage des modèles markoviens nécessite au préalable le réglage d'un certain nombre de paramètres liés à la topologie du champ de Markov, à la programmation dynamique 2D ainsi qu'aux observations et à leur modélisation.

3.4.1 Topologie du champ de Markov

La topologie d'un champ de Markov est difficile à déterminer et elle se choisit souvent de façon intuitive en fonction de l'application considérée. Ici, nous cherchons à segmenter les caractères suivant la position et la direction des traits dans l'image. Le choix de la grille d'états $n_x \times n_y$ doit ainsi se faire en tenant compte des considérations suivantes :

- la grille d'états $n_x \times n_y$ doit être choisie suffisamment grande pour modéliser tous les types de traits (un type de trait correspondant à une certaine direction et à une certaine position),
- le nombre d'états ne doit pas être trop élevé car le temps de décodage est fortement dépendant de celui-ci et augmente rapidement avec lui.

Intuitivement en observant des formes relativement complexes tels que le "8" ou le "3" pour les chiffres ou le "B" ou le "f" pour les lettres, on constate que 7 états sont nécessaires en hauteur et 5 sont nécessaires en largeur (figure 3.5).

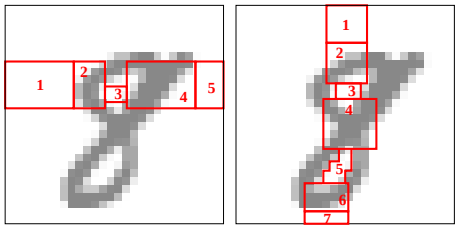


Fig. 3.5. Intuitivement pourquoi la grille 5×7 ?

Expérimentalement, on a pu vérifier ce constat puisque c'est cette grille 5×7 qui donne les meilleures performances (tableau 3.2). Une grille 5×7 a donc été adoptée pour la modélisation des différentes classes de caractères.

Tab. 3.2. Taux d'erreur en fonction de la topologie $n_x \times n_y$ choisie pour le champ de Markov

n_y \ n_x	4	5	6
8	2.52%	2.12%	2.06%
7	2.57%	**2.04%**	2.28%
6	3.21%	2.52%	2.50%

3.4.2 Programmation dynamique 2D

3.4.2.1 Stratégie de fusion

Nous avons choisi d'utiliser la stratégie de fusion "escargot" présentée dans le précédent chapitre afin de fusionner en premier les sites présentant le moins de doute sur leur configuration. C'est le cas des sites situés sur les contours de l'image représentant une information de blanc. En effet, les images de la base MNIST comportent un cadre blanc et la segmentation initiale a été prévue pour que les états du tour modélisent cette information de blanc.

3.4.2.2 Élagage

La stratégie d'élagage au cours de l'algorithme de décodage est déterminante pour la qualité des résultats obtenus. On peut ainsi observer l'influence du nombre de configurations gardées à chaque fusion sur les performances du système (figure 3.6). On constate que conserver moins de 10 configurations à chaque fusion entraîne des dégradations importantes et à l'inverse en garder plus de 30 est inutile. Ainsi, 30 semble être un bon compromis entre temps de calculs et performances.

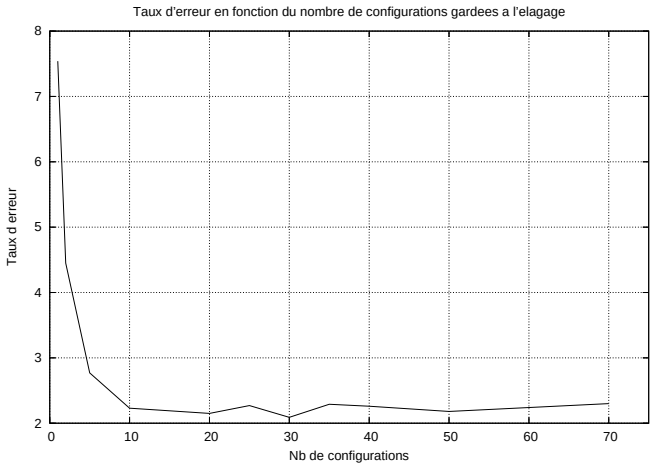


Fig. 3.6. Influence du nombre de configurations gardées dans l'élagage sur les performances du système

3.4.3 Primitives

Un certain nombre de paramètres rentrant en compte dans le calcul des primitives spectrales locales influe sur les performances du système ainsi que sur la complexité du modèle :

- la taille de la fenêtre gaussienne et le pas d'échantillonnage,
- la sélection des coefficients du module et de la phase de la FFT.

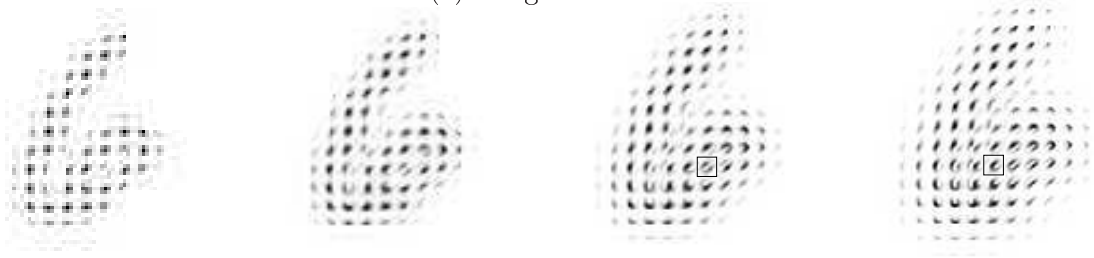
3.4.3.1 Choix de la taille de la fenêtre et du pas d'échantillonnage

La taille des fenêtres utilisées lors de l'extraction des primitives ainsi que le décalage entre deux fenêtres consécutives (ou pas d'échantillonnage) sont des paramètres fortement corrélés. Notre choix s'est effectué en tenant compte des considérations suivantes.

Premièrement, la taille de la fenêtre doit être choisie de manière à pouvoir extraire des informations locales et précises en terme de direction de traits. Elle doit donc être à la fois suffisament grande pour permettre de distinguer sans ambiguité des directions de traits et ne pas extraire simplement une information de niveau de gris et suffisament petite pour ne pas contenir des formes trop complexes composées de plusieurs directions de traits.

Pour illustrer ce point, nous pouvons observer les imagettes extraites d'une image de chiffre pour des tailles de fenêtres différentes (5×5, 7×7, 9×9, 11×11) et pour un même pas d'échantillonage fixé à 1 pixel sur 2 (figure 3.7).

(a) Image de 6



(b) 5×5 (c) 7×7 (d) 9×9 (e) 11×11

Fig. 3.7. Imagettes extraites pour différentes tailles de fenêtre

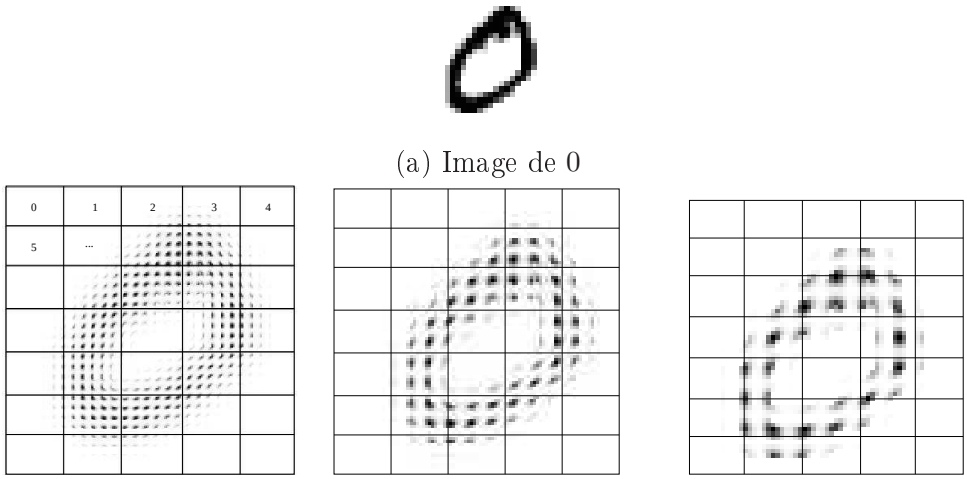
L'observation de ces figures montre que :

- la fenêtre 5×5 ne permet pas toujours de distinguer les directions des traits,
- les fenêtres 9×9 et 11×11 peuvent contenir des formes assez complexes contenant différentes directions de traits (cf. les encadrés noirs dans les figures correspondantes).

La fenêtre 7×7 semble représenter un bon compromis. Les expériences détaillées plus loin confirment ce choix.

Deuxièmement, le choix du pas d'échantillonage est lié à celui de la taille de la fenêtre. Il est effectué de façon à accorder la même importance à tous les pixels tout en limitant les redondances pour limiter le temps de calcul notamment au moment du décodage. L'analogie avec le traitement de la parole nous a conduit à adopter un pas d'échantillonnage égal à environ 1/3 de la taille de la fenêtre. Ainsi pour une fenêtre 7×7, le pas a-t-il été choisi égal à 2.

On peut observer que le choix de ces paramètres est en adéquation avec le modèle d'états développé, et en particulier avec le nombre d'états retenu (35). La figure 3.8.c montre qu'une fenêtre 7×7 et qu'un pas de 2 conduisent à un nombre suffisant d'observations par état comparé au cas d'une fenêtre 7×7 et d'un pas de 3 (figure 3.8.d). Le cas d'un pas égal à 1 (figure 3.8.b) n'a pas été retenu en raison des redondances introduites et du facteur 4 en temps de calcul de décodage par rapport au pas de 2.



(a) Image de 0

(b) Pas de 1 (c) Pas de 2 (d) Pas de 3

Fig. 3.8. Imagettes extraites avec une fenêtre 7×7 pour chaque état de la grille 5×7 avec différents pas d'échantillonnage

Étudions maintenant l'influence de la taille de la fenêtre d'analyse et du pas d'échantillonnage sur le taux d'erreur.

Comme nous l'avons vu précédemment, le pas d'échantillonnnage est lié à la taille de la fenêtre. Ainsi, pour une fenêtre 5×5, on prend un pas d'échantillonnage de 1 pixel sur 2, pour une fenêtre 7×7 un pas de 1 pixel sur 2, pour une fenêtre 9×9 un pas de 1 pixel sur 3 et pour une fenêtre 11×11 un pas de 1 pixel sur 4. Le tableau 3.3 synthétise les taux d'erreur obtenus pour ces quatre tailles de fenêtre en utilisant un vecteur de primitives constitué de 8 modules et de 2 phases. On constate que le meilleur résultat est obtenu avec une fenêtre 7×7.

Tab. 3.3. Influence de la taille de la fenêtre sur le taux d'erreur

Taille de la fenêtre	Taux d'erreur
5×5	2.88%
7×7	**2.04%**
9×9	3.06%
11×11	4.27%

3.4.3.2 Étude du choix des coefficients de la FFT pour une fenêtre 7×7

Dans ce paragraphe, nous étudions l'influence du choix des coefficients de la FFT sur le taux d'erreur.

Intuitivement, nous avons débuté nos tests en utilisant les coefficients du module correspondant aux quatre directions principales (horizontale, verticale et les 2 diagonales). En effet, ce sont celles qui, visuellement semblent apporter le plus d'information. L'apport de la phase est moins évident, c'est pourquoi, nous avons étudié l'influence de certains coefficients de celle-ci dans la reconnaissance (tableau 3.4).

Tab. 3.4. Comparaison des résultats suivant les phases utilisées

Coefficients	Taux d'erreur
4 dir principales	3.56%
4 dir principales + (V_1^φ, H_1^φ)	**2.38%**
4 dir principales + (V_1^φ, H_1^φ, $D_{-1,1}^\varphi$, $D_{-1,-1}^\varphi$)	2.61%

Plusieurs expériences ont montré que les deux phases V_1^φ et H_1^φ améliorent le taux d'erreur de 25% à 30% en relatif. De plus, on peut voir figure 2.15 que ce sont les deux phases qui semblent contenir une information intéressante. En revanche, l'ajout de phases supplémentaires n'améliorent plus les résultats.

Concernant l'influence du coefficient O (moyenne), des expériences ont montré que sa prise en compte ne permettait pas d'améliorer les résultats. De même les coefficients H_2 et V_2 ne semblent pas améliorer les performances.

Après nous être limités aux quatre coefficients du module correspondant aux quatre directions principales, nous allons étudier la prise en compte de directions supplémentaires.

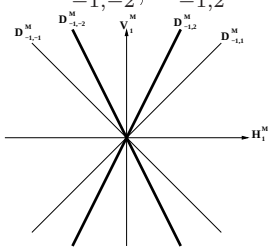
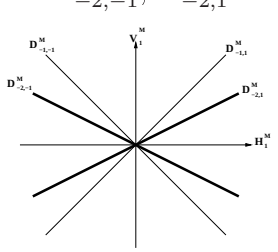
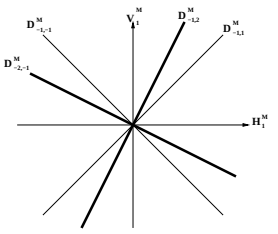
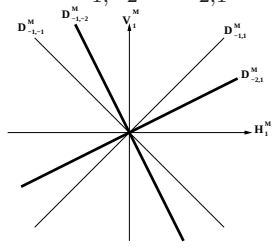
Remarque : les résultats donnés ci-dessous ont été obtenus en prenant en compte les deux phases V_1^φ et H_1^φ évoquées précédemment.

1. *Quatre directions principales et deux directions secondaires*

   On ajoute aux quatre directions principales, deux directions secondaires et on observe les résultats (tableau 3.5). On constate que la prise en compte de deux di-

rections supplémentaires améliore les résultats. En particulier, l'ajout des modules $D^M_{-2,-1}$ et $D^M_{-2,1}$ donne un taux d'erreur de 2.14%.

Tab. 3.5. Résultats obtenus avec 4 directions principales et 2 directions secondaires

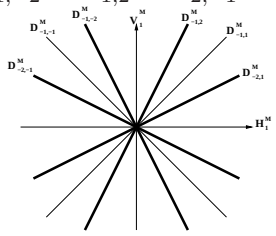
$D^M_{-1,-2}$, $D^M_{-1,2}$  Taux d'erreur : 2.42%	$D^M_{-2,-1}$, $D^M_{-2,1}$  Taux d'erreur : 2.14%
$D^M_{-2,-1}$, $D^M_{-1,2}$  Taux d'erreur : 2.27%	$D^M_{-1,-2}$, $D^M_{-2,1}$  Taux d'erreur : 2.17%

2. *Quatre directions principales et quatre directions secondaires*

   On ajoute aux quatre directions principales les quatre directions secondaires (tableau 3.6) et l'on obtient un taux d'erreur de 2.04%.

Tab. 3.6. Résultats obtenus avec 4 directions principales et 4 directions secondaires

$D^M_{-1,-2}$, $D^M_{-1,2}$, $D^M_{-2,-1}$, $D^M_{-2,1}$



Taux d'erreur : 2.04%

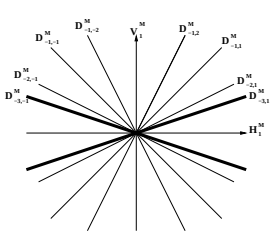
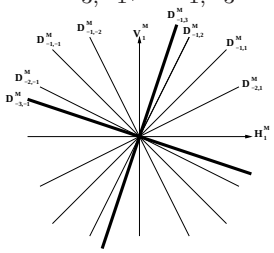
3. *Quatre directions principales et six autres directions*

   On ajoute aux huit directions précédentes, deux directions supplémentaires (tableau 3.7).

   L'ajout de directions supplémentaires ne semble plus améliorer les performances du système. Ceci peut s'expliquer par le fait que la taille des images traitées est

trop petite et la résolution n'est pas assez bonne pour faire la distinction entre deux directions assez proches l'une de l'autre.

Tab. 3.7. Résultats obtenus avec 6 autres directions

$D^M_{-3,-1}$, $D^M_{-3,1}$	$D^M_{-3,-1}$, $D^M_{-1,-3}$
	
Taux d'erreur : 2.25%	Taux d'erreur : 2.41%

Le meilleur résultat obtenu jusqu'à présent est un taux d'erreur de **2.04**% sur la base de validation en utilisant les quatre directions principales, les quatre directions secondaires et les deux phases.

3.4.4 Modélisation des primitives

Le calcul des paramètres liés à la modélisation des primitives est un élément clé dans la complexité totale de l'algorithme. En particulier si on appelle N_G le nombre maximum de gaussiennes possibles, le temps de calcul est presque linéaire en N_G. Il paraît donc nécessaire de correctement choisir ce nombre afin de limiter la complexité tout en modélisant finement les observations. Un nombre de 20 gaussiennes maximum a ainsi été choisi (figure 3.9).

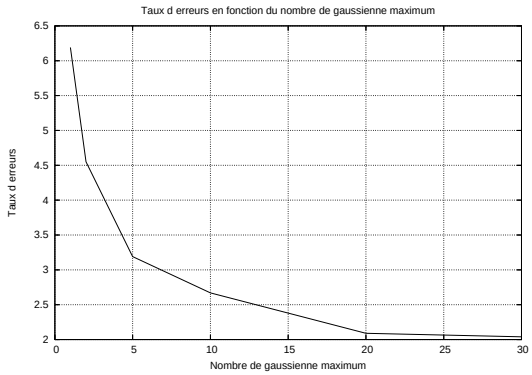


Fig. 3.9. Influence du nombre de gaussiennes maximum possibles sur les performances du système

3.4.5 Conclusion

L'étude conduite ici concernait le choix des paramètres liés à la topologie du champ de Markov, la programmation dynamique 2D et les primitives. Visant à l'optimisation

des performances du sytème, elle a permis de diminuer de 15% le taux d'erreurs obtenu initialement par la méthode proposée par Chevalier [16]. On obtient ainsi un taux d'erreur de 2.04% sur la base de validation.

3.5 Performances et analyse du système

3.5.1 Performances du système

Les performances obtenues jusqu'à présent sont très correctes puisqu'un taux d'erreurs de 2.04% est atteint. Comme nous l'avons précisé précédemment, il est très difficile de comparer les résultats obtenus par deux systèmes en dehors des campagnes d'évaluation. Si le taux d'erreurs obtenu par notre système n'atteint pas les meilleurs résultats qui semblent tourner autour de 1%, son avantage majeur par rapport à toutes les approches présentées dans la littérature est d'être très générale. De plus, nous verrons dans le chapitre 6 que lors de la campagne d'évaluation RIMES notre système a obtenu des résultats similaires à ceux obtenus par des systèmes spécifiquement dédiés à la reconnaissance de caractères.

Il est intéressant d'analyser les erreurs afin d'envisager des pistes d'amélioration.

L'analyse la plus simple renvoie à l'observation de la matrice de confusion (tableau 3.8).

Tab. 3.8. Matrice de confusion sur la base de validation de MNIST

Chiffre reconnu \ *Vérité terrain*	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
0	966	0	2	0	0	3	3	0	7	4
1	4	1117	1	0	3	0	2	0	1	0
2	7	9	1018	4	1	3	0	6	4	1
3	0	0	0	987	0	8	0	0	3	7
4	0	1	1	0	957	1	3	2	2	6
5	0	0	0	7	0	870	9	0	5	1
6	3	1	1	0	4	4	941	0	3	0
7	0	1	7	2	1	0	0	1014	0	5
8	0	6	2	6	2	3	0	0	946	5
9	0	0	0	4	14	0	0	6	3	980

Celle-ci peut aussi être rendue plus "visuelle" en représentant (figure 3.10) les images d'erreur et en indiquant pour chacune d'entre elles la confusion réalisée ($i \rightarrow j$ signifie que la classe i a été reconnue comme la classe j).

Plusieurs remarques s'imposent alors :

- les chiffres les plus "difficiles" à reconnaître par notre système sont le "8" et le "9",
- 20% des erreurs sont aisément compréhensibles dont environ 7% auraient pu faire l'objet d'une confusion par un homme,
- 80% des erreurs sont beaucoup moins justifiables et devraient donc être évitées.

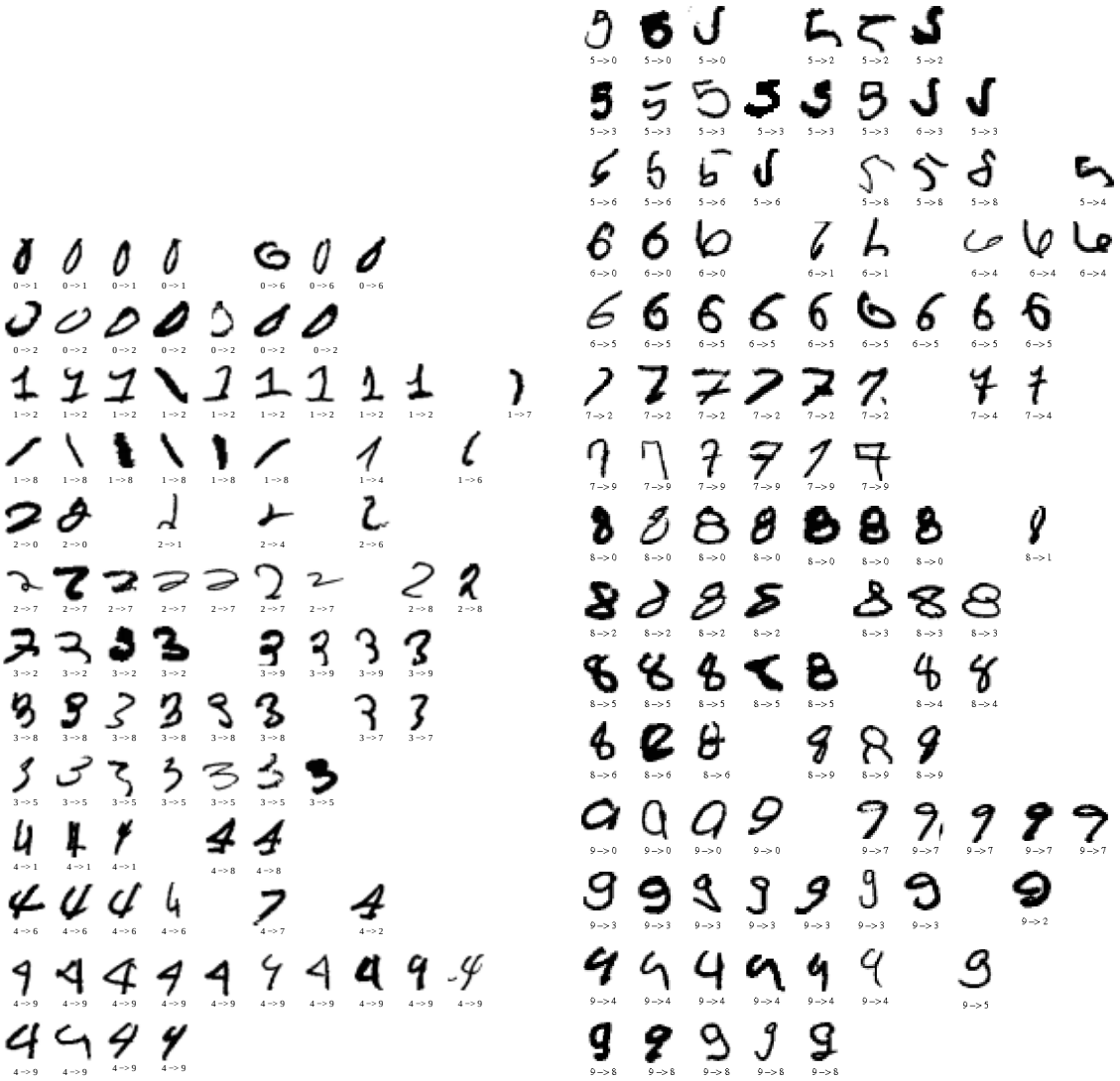


Fig. 3.10. Images mal reconnues et confusions réalisées sur la base de validation de MNIST

Une deuxième analyse assez simple est l'observation des images les mieux reconnues (figure 3.11) et des images les moins bien reconnues (figure 3.12) pour chaque modèle.

L'observation des 10 images les mieux reconnues nous donne une information sur la forme-type des modèles construits par notre système. Ainsi, par exemple, le "1" type est un "bâton" légèrement incliné sur la droite. On constate que les erreurs sont commises sur des formes assez éloignées de ces formes types. Une idée est donc d'utiliser plusieurs modèles pour une même classe : par exemple pour la classe "4" on pourrait utiliser un modèle de "4" fermé (souvent confondu avec un "9") et un modèle avec le reste des "4". Cela a été réalisé, mais les performances n'ont pas été améliorées de même que pour deux modèles de "1" et deux modèles de "7". Cela peut s'expliquer par le faible nombre d'échantillons disponibles pour la classe "4" fermé, "7" barré et "1" non bâton.

Fig. 3.11. Les 10 premières images bien reconnues pour chaque classe

Fig. 3.12. Les 10 dernières images bien reconnues pour chaque classe

Les 10 dernières images bien reconnues ont des formes très variées. Certaines d'entre elles ont des formes très proches de celles des images d'erreur. Si l'on observe la vraisemblance obtenue avec le modèle réel, on constate qu'elle n'est pas significativement plus grande que celle obtenue avec un autre modèle. C'est en quelque sorte un coup de "chance" si elles ont été bien reconnues, de même que c'était de la "malchance" que certaines confusions soient commises.

Cela peut être explicité en observant l'influence du rejet sur le taux d'erreur, l'idéal étant bien sûr que l'on ait un taux d'erreur nul avec un très faible taux de rejet.

On envisage deux politiques différentes de rejet :
- Rejet par seuil absolu : les probabilités supérieures à un certain seuil sont retenues ;
- Rejet par seuil relatif : la probabilité dont la différence avec la deuxième plus grande probabilité correspondant à une autre classe est supérieure à un certain seuil est retenue.

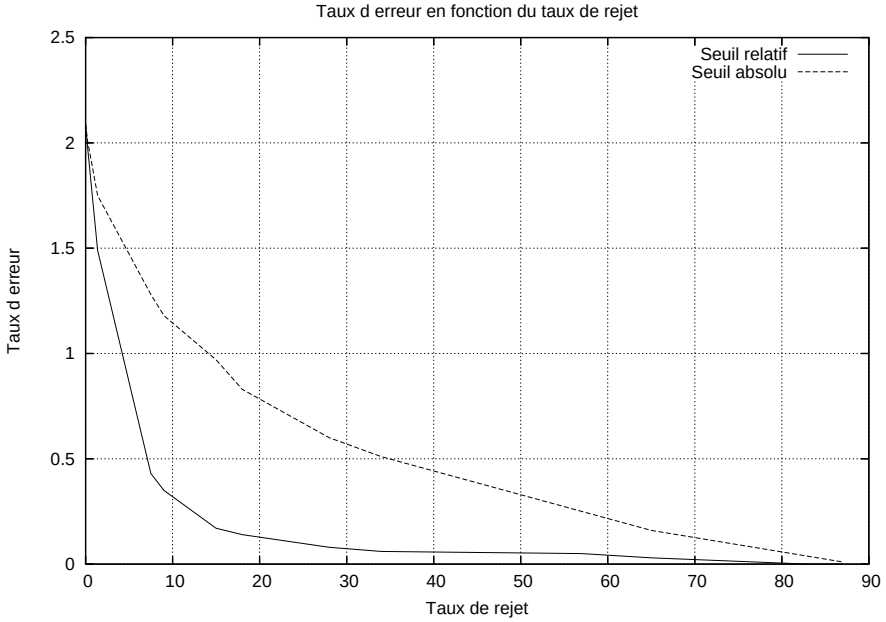


Fig. 3.13. Taux de rejet en fonction du taux d'erreur avec deux politiques différentes

Les meilleurs performances sont obtenues avec un rejet par seuil relatif. Ainsi avec un rejet de 10%, on obtient un taux d'erreur de 0.3%.

Si on veut diminuer le taux d'erreur sans utiliser de rejet, une analyse de la modélisation markovienne est nécessaire afin de proposer des stratégies d'amélioration. Nous allons tout d'abord commencer par analyser les segmentations en états afin de vérifier qu'elles sont bien conformes à ce qu'on attendait, c'est-à-dire qu'un état est bien caractéristique d'une direction et d'une position de trait dans l'image. Ensuite, nous pouvons analyser les modèles obtenus pour chaque classe afin de vérifier que le nombre d'états choisi est judicieux.

3.5.2 Validation de la segmentation en états

Une analyse des segmentations est effectuée pour vérifier la bonne segmentation des traits suivant leur direction et leur position dans l'image. On constate que ces conditions sont bien remplies et on donne figure 3.14 un exemple pour chacune des 10 classes.

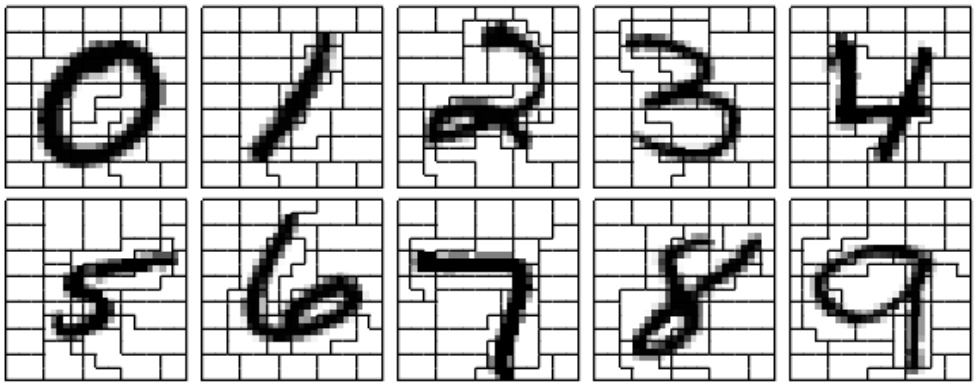


Fig. 3.14. Exemples de segmentations en états pour chacune des 10 classes de chiffres

Figure 3.15, on représente deux exemples de segmentations en états obtenues avec notre algorithme pour deux images de la classe "3". On observe que la répartition des états est similaire dans les deux cas, *i.e.* un même état dans les deux segmentations est associé à une direction de trait et à une position identiques. Par exemple, l'état 7 est associé pour les deux images à la direction horizontale positionnée sur le haut du "3".

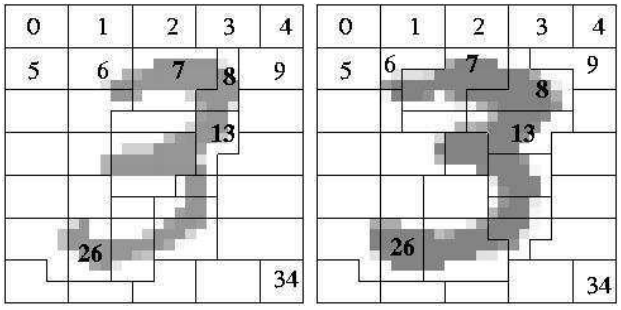


Fig. 3.15. Deux exemples de segmentations en état de la classe "3"

3.5.3 Analyse des modèles

Une représentation très informative est l'affichage pour chaque modèle de l'imagette moyenne des observations associée à chaque état (figure 3.16).

En regardant chacune des représentations de chaque classe, on parvient bien à identifier la forme de chaque chiffre. On remarque toutefois que certains modèles manquent de précision. Par exemple, pour le modèle de "8" aucun état ne semble caractériser les deux trous. Cela est en partie dû à la faible résolution des images (taille 28×28) et

donc il y a très peu de pixels caractérisant les "trous" du "8". Pour affiner les modèles, on pourrait envisager d'effectuer de la division d'états associée à de la fusion d'états.

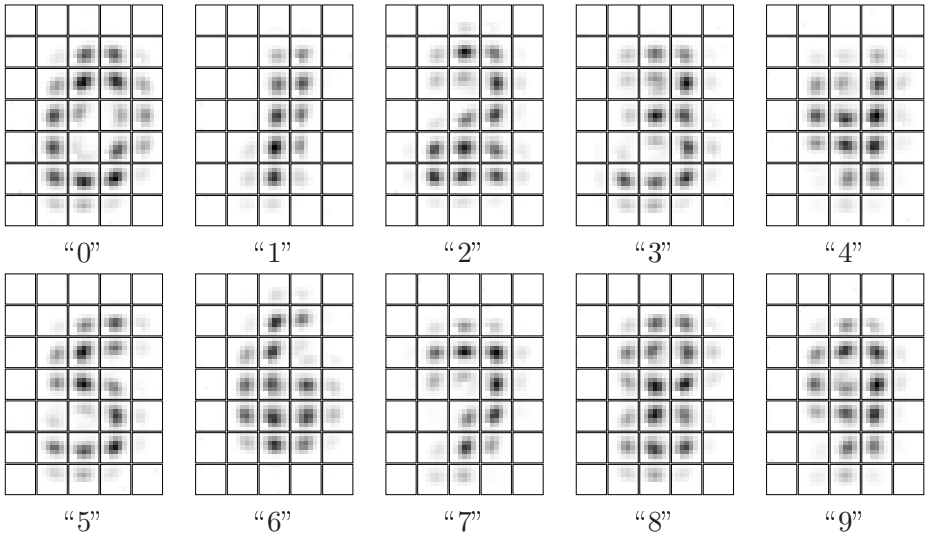


Fig. 3.16. Imagettes moyennes pour les 10 modèles

3.6 Amélioration des performances et perspectives

3.6.1 Combinaison de systèmes

3.6.1.1 Introduction

Lors de cette étude, on a pu constater que les ensembles d'erreur obtenus avec différents systèmes correspondant à différents vecteurs de primitives étaient différents, c'est-à-dire que les erreurs d'un système ne se retrouvent pas dans les erreurs d'un autre système et vice versa. Cette constatation nous a amenée à penser que la combinaison de systèmes pourrait être une piste interessante à explorer afin de diminuer le taux d'erreur. L'étude présentée ici a pour objectif de montrer tout l'intérêt de ce type d'approche pour notre application. L'approche développée est assez basique et les très bons résultats obtenus montrent qu'une étude plus approfondie pourrait permettre des améliorations plus conséquentes.

La combinaison parrallèle de systèmes a été proposée comme une voie de recherche permettant d'améliorer les performances de reconnaissance (taux de reconnaissance, fiabilité, ...) en exploitant la complémentarité qui peut exister entre les systèmes. La combinaison permet, à partir des sorties des différents classifieurs, d'élaborer une réponse finale unique.
En reconnaissance de la parole, la méthode ROVER a été proposée afin de combiner les sorties des différents systèmes lors des campagnes d'évaluation.
Plus généralement, la combinaison de systèmes est couramment utilisée en reconnaissance de formes pour laquelle de nombreuses méthodes ont été développées.

La méthode ROVER Au cours du programme d'évaluation sur la reconnaissance de la parole, le NIST a développé la méthode ROVER (Recognizer Output Voting Error Reduction) [29] qui permet de produire par combinaison automatique, une transcription d'un signal de parole à partir des transcriptions fournies par les différents systèmes de reconnaissance participant à l'évaluation. Le système composite résultant a toujours eu de meilleures performances (en précision) que n'importe lequel de ses constituants (les performances des systèmes étaient assez proches).

Dans l'approche ROVER, la sortie des systèmes de reconnaissance de la parole est d'abord combinée en un unique graphe de mots (suite de tous les mots possibles) au moyen d'une version simplifiée de l'algorithme d'alignement (programmation dynamique) utilisée par le NIST pour mesurer les performances des systèmes (outil sclite). Ce graphe est ensuite élagué à l'aide d'une stratégie de vote qui permet de sélectionner le meilleur choix à chaque point de décision.

Dans son article [29], Fiscus définit trois systèmes de vote différents :

- Vote par fréquence d'occurrence ;
- Vote par fréquence d'occurrence et moyenne du score de confiance pour chaque type de mot ;
- Vote par fréquence d'occurrence et score de confiance maximum.

Fiscus indique une réduction de l'erreur de 5.6% en absolu et de 12.5% en relatif. Remarquons toutefois que les trois systèmes de vote donnent des résultats sensiblement identiques.

Reconnaissance de formes Un système peut fournir différents types de sorties s_j :

- **Type classe :** $s_j = C_i$ où C_i est la classe renvoyée par le système j,
- **Type rang :** $s_j = \{r_1^j, ..., r_M^j\}$ où r_i^j est le rang de la classe i renvoyée par le système j,
- **Type mesure :** $s_j = \{m_1^j, ..., m_M^j\}$ où m_i^j est la mesure attribuée à la classe i renvoyée par le système j.

De la même manière, on peut définir trois types de méthode de combinaison de systèmes [105].

- **Type classe :** combinaison des sytèmes dont chacun attribue une classe unique à la forme à reconnaître. Parmi ces méthodes, on peut citer : le vote majoritaire, la théorie de Bayes, la théorie de Dempster-Shafer, la méthode BKS (Behaviour Knowledge Space).
- **Type rang :** combinaison des listes ordonnées de classes. On peut citer la méthode Borda Count.
- **Type mesure :** combinaison des listes ordonnées de classes associées à un taux de confiance. Parmi ces méthodes, on peut citer : les réseaux de neurones, les règles fixes (maximum, somme, moyenne, produit).

De plus, parmi ces méthodes, on pourra distinguer les méthodes avec apprentissage et les méthodes sans apprentissage. Le tableau 3.9 répertorie les méthodes de combinaison les plus utilisées.

Tab. 3.9. Taxonomie des méthodes de combinaison parrallèle de classifieurs

	Type classe	Type rang	Type mesure
Sans apprentissage	vote majoritaire vote à la pluralité vote pondéré vote notoire vote unanime	somme somme pondérée intersection union meilleur rang Borda count	maximum minimim mediane produit méthodes linéaires
Avec Apprentissage	Bayésienne Dempster-Shafer BKS	régression logistique	réseaux de neurones intégral flou

3.6.1.2 Techniques de combinaison des systèmes exploitées

Dans la littérature les combinaisons impliquent des sytèmes utilisant différents classifieurs. Ici, dans le cadre de la reconnaissance de chiffres manuscrits, on se propose de combiner différents systèmes correspondant à différents vecteurs de primitives spectrales locales (coefficients différents). Il existe de nombreuses méthodologies et techniques pour combiner des systèmes, mais suivant les applications une méthode sera meilleure ou moins bonne.

Pour cela, on se propose de tester différentes techniques :

1. **Type classe :**
   - *Méthode 1 : le vote à la pluralité*
     Il s'agit de conserver la classe qui a été donnée le plus de fois par les différents systèmes, en cas d'égalité on conserve la réponse du meilleur système.
2. **Type rang :**
   - *Méthode 2 : Borda count*
     Il s'agit d'attribuer à chaque classe un score correspondant au rang attribué à cette classe par chacun des systèmes (par exemple si la classe 0 est classée au rang 2 par le sytème 1 et au rang 4 par le système 2 son score sera égal à $4 + 2 = 6$) et de choisir la classe qui fournit le plus petit score.
3. **Type mesure :**
   - *Méthode 3 : la maximisation des vraisemblances*
     Il s'agit de conserver la classe C_k associée à une mesure P_k telle que :

$$P_k = \max_{i=1}^{N} \max_{j=1}^{L} m_{i,j},$$

avec

$$m_{i,j} = 1 - \frac{cout_{i,j}}{\sum_{i=1}^{N} cout_{i,j}}.$$

où L est le nombre de systèmes combinés, N est le nombre de classes et $cout_{i,j}$ est le coût associé à la classe i du j^{ime} système.

– *Méthode 4 : la maximisation d'un taux de confiance* Il s'agit de conserver la classe C_k associée à une mesure P_k telle que

$$P_k = \max_{j=1}^{L} m_j,$$

avec

$$m_j = 1 - \frac{coutmin_j}{coutmin_{2_j}},$$

où $coutmin_j$ est le plus petit coût associé à une classe C_k^j et $coutmin_{2_j}$ est le deuxième plus petit coût associé à une classe C_2^j.

3.6.1.3 Combinaison des systèmes obtenus avec différents vecteurs de primitives spectrales locales

Les systèmes utilisés pour différentes combinaisons sont décrits dans le tableau 3.10. Ils ont été sélectionnés en fonction de leur faible taux d'erreur, c'est-à-dire moins de 2.30% et ils ont été obtenus avec différents vecteurs de primitives spectrales locales. Il est délicat *a priori* de connaître la méthode et la combinaison qui donneront le meilleur résultat. Ainsi toutes les méthodes et combinaisons sont-elles envisagées. Les résultats obtenus par les différentes méthodes pour toutes les combinaisons possibles sont donnés dans le tableau 3.11.

Tab. 3.10. Caractéristiques des différents systèmes

Système	Primitives	Taux d'erreur
1	4 dir principales 4 dir secondaires 2 phases	2.04%
2	4 dir principales 2 dir secondaires (1) 2 phases	2.14%
3	4 dir principales 2 dir secondaires (2) 2 phases	2.17%
4	4 dir principales 2 dir secondaires (3) 2 phases	2.27%
5	4 dir principales 6 dir secondaires 2 phases	2.25%

La méthode Borda Count semble être la méthode la plus adaptée à notre application. C'est en effet elle qui donne généralement les meilleurs résulats pour chaque combinaison. Un taux d'erreur de 1.52% est ainsi obtenu en combinant les systèmes 1 et 5. On a ainsi une réduction de plus de 25% du taux d'erreur.

Tab. 3.11. Taux d'erreur obtenus par combinaison selon différentes méthodes

Combinaison	Méthode 1	Méthode 2	Méthode 3	Méthode 4
1 + 2	2.04%	1.53%	2.06%	1.87%
1 + 3	2.04%	1.60%	2.12%	1.95%
1 + 4	2.04%	1.61%	2.04%	1.96%
1 + 5	2.04%	**1.52**%	2.25%	2.03%
2 + 3	2.14%	1.56%	2.16%	1.89%
2 + 4	2.14%	1.62%	2.26%	2.08%
2 + 5	2.14%	1.54%	2.21%	1.98%
3 + 4	2.17%	1.56%	2.27%	2.03%
3 + 5	2.17%	1.55%	2.20%	2.06%
4 + 5	2.25%	1.57%	2.21%	2.10%
1 + 2 + 3	1.90%	1.95%	2.12%	1.85%
1 + 2 + 4	1.93%	1.97%	2.10%	1.92%
1 + 2 + 5	1.84%	1.88%	2.23%	1.86%
1 + 3 + 4	1.99%	1.99%	2.10%	1.95%
1 + 3 + 5	1.98%	1.91%	2.20%	1.95%
1 + 4 + 5	1.90%	1.98%	2.17%	2.25%
2 + 3 + 4	1.96%	2.04%	2.28%	1.92%
2 + 3 + 5	1.88%	1.91%	2.18%	1.91%
2 + 4 + 5	1.94%	2.01%	2.20%	2.20%
3 + 4 + 5	1.94%	2.05%	2.00%	2.00%
1 + 2 + 3 + 4	1.95%	2.17%	2.12%	1.88%
1 + 2 + 3 + 5	1.92%	2.15%	2.18%	1.87%
1 + 3 + 4 + 5	1.94%	1.88%	2.17%	2.17%
2 + 3 + 4 + 5	1.91%	1.86%	2.18%	1.90%
1 + 2 + 3 + 4 + 5	1.85%	2.15%	2.16%	1.88%

3.6.2 Optimisation du nombre d'états par division et fusion d'états : une perspective

3.6.2.1 Introduction

La topologie du HMM (grille d'états par exemple) à utiliser est assez difficile à déterminer, elle se choisit souvent de façon intuitive. Nous avons vu qu'une grille 5×7 donnait de très bons résultats pour la reconnaissance de caractères manuscrits, toutefois, on constate un manque de précision pour certains modèles (par exemple le "8").

La division et la fusion d'états sont assez souvent utilisées pour l'étude des protéines ou de l'ADN [101, 86], ou plus généralement quand la topologie du HMM est difficile à déterminer.

Les techniques standards utilisant des HMM supposent une connaissance *a priori* de la taille du modèle et de la topologie. Or, pour la plupart des applications, il est assez difficile de déterminer la topologie la mieux adaptée au problème. En outre, plus la complexité du HMM augmente (grand nombre d'états et donc de transitions), plus il est difficile de choisir la topologie du HMM "manuellement". C'est pourquoi plusieurs auteurs ont suggéré d'appliquer des algorithmes évolutifs pour déterminer les paramètres du modèle du HMM [84, 101, 86, 88].

Quelques méthodes heuristiques ont été introduites : la division et la fusion d'états apprennent la topologie de façon soit constructive, soit élagante, c'est-à-dire soit on ajoute des états au modèle, soit on fusionne des états similaires. Des algorithmes évolutifs plus élaborés ont été utilisés pour faire évoluer à la fois la topologie et les paramètres du modèle ; les opérations suivantes peuvent être utilisées : insertion ou suppression d'une transition ou d'un état, croisement entre deux modèles, etc.

Il est important de générer un HMM qui soit précis et robuste : la capacité de représentation est nécessaire pour la précision du modèle tandis que la simplicité l'est pour la robustesse du modèle [86].

Dans un premier temps, nous allons définir la distance de Kullback-Leibler qui va servir à mesurer la dissimilarité entre deux distributions de probabilité, puis nous expliquerons les processus de division et de fusion d'états fondés sur cette distance. Enfin, nous tenterons de mettre en oeuvre une stratégie de division et fusion d'états pour affiner la topologie de nos modèles markoviens.

3.6.2.2 La distance de Kullback-Leibler

La distance de Kullback-Leibler [47] est une mesure de dissimilarité entre deux distributions de probabilités. Par définition, la distance de Kullback-Leibler $D(p_1, p_2)$ entre $p_1(x)$ et $p_2(x)$ est donnée par :

$$D(p_1, p_2) = \int p_1(x) \times \log\left(\frac{p_1(x)}{p_2(x)}\right) dx.$$

Afin d'utiliser une distance au sens mathématique du terme, on utilise la distance de Kullback-Leibler symétrique $D_s(p_1, p_2)$ donnée par :

$$D_s(p_1, p_2) = \frac{D(p_1, p_2) + D(p_2, p_1)}{2}.$$

Dans le cas de deux distributions gaussiennes, on a :

$$\begin{aligned} p_1(x) &= \frac{1}{\sigma_1\sqrt{2\pi}} \int \exp^{-\frac{(x-m_1)^2}{2\sigma_1^2}} dx, \\ p_2(x) &= \frac{1}{\sigma_2\sqrt{2\pi}} \int \exp^{-\frac{(x-m_2)^2}{2\sigma_2^2}} dx. \end{aligned}$$

Finalement, on obtient l'expression suivante :

$$D_s(p_1, p_2) = \frac{(\sigma_1^2 - \sigma_2^2)^2 + (\mu_1^2 - \mu_2^2)^2(\sigma_1^2 + \sigma_2^2)^2}{4\sigma_1^2\sigma_2^2}.$$

3.6.2.3 Mise en oeuvre de la division d'état

Pour un modèle donné, pour déterminer l'état à diviser, on effectue une modélisation bi-gaussienne ($\alpha G_1 + \beta G_2$), et on choisit celui dont la divergence entre ses deux gaussiennes G_1 et G_2 est maximum. Pour cela, on utilise la divergence de Kullback-Leibler symétrique. La stratégie adoptée pour la division d'état est décrite figure 3.17.

Pour réaliser la division d'un état, on part du modèle obtenu par la modélisation bi-gaussienne. Pour l'état concerné E (c'est-à-dire celui dont la divergence entre ses deux gaussiennes G_1 et G_2 est maximum), on divise la bi-gaussienne en deux monogaussiennes G_1 et G_2. On attribue la première à l'état E_1 et la deuxième à l'état E_2. Ainsi les états E_1 et E_2 sont, à ce stade, chacun modélisés par une monogaussienne et les autres états sont modélisés par une bi-gaussienne. On utilise ensuite la programmation dynamique 2D pour obtenir les nouvelles segmentations associées qui disposent maintenant, non plus de N mais, de $N+1$ états. On calcule alors les nouveaux paramètres du modèle à partir de ces segmentations en états avec N gaussiennes maximum ($N > 2$).

3.6.2.4 Mise en oeuvre de la fusion d'état

Pour un modèle donné, pour déterminer les états voisins à fusionner, on effectue une modélisation mono-gaussienne, et on choisit les états qui minimisent la divergence entre leur deux mono-gaussiennes. Pour cela, on utilise également la divergence de Kullback-Leibler. La stratégie adoptée pour la fusion d'état est décrite figure 3.18.

Pour réaliser la fusion de deux états E_1 et E_2, on part des dernières segmentations obtenues et on fusionne tout simplement les deux états (pour chaque site si son état est E_1 alors il devient E, si son état est E_2 alors il devient E). On calcule alors les nouveaux paramètres du modèle avec 20 gaussiennes maximum.

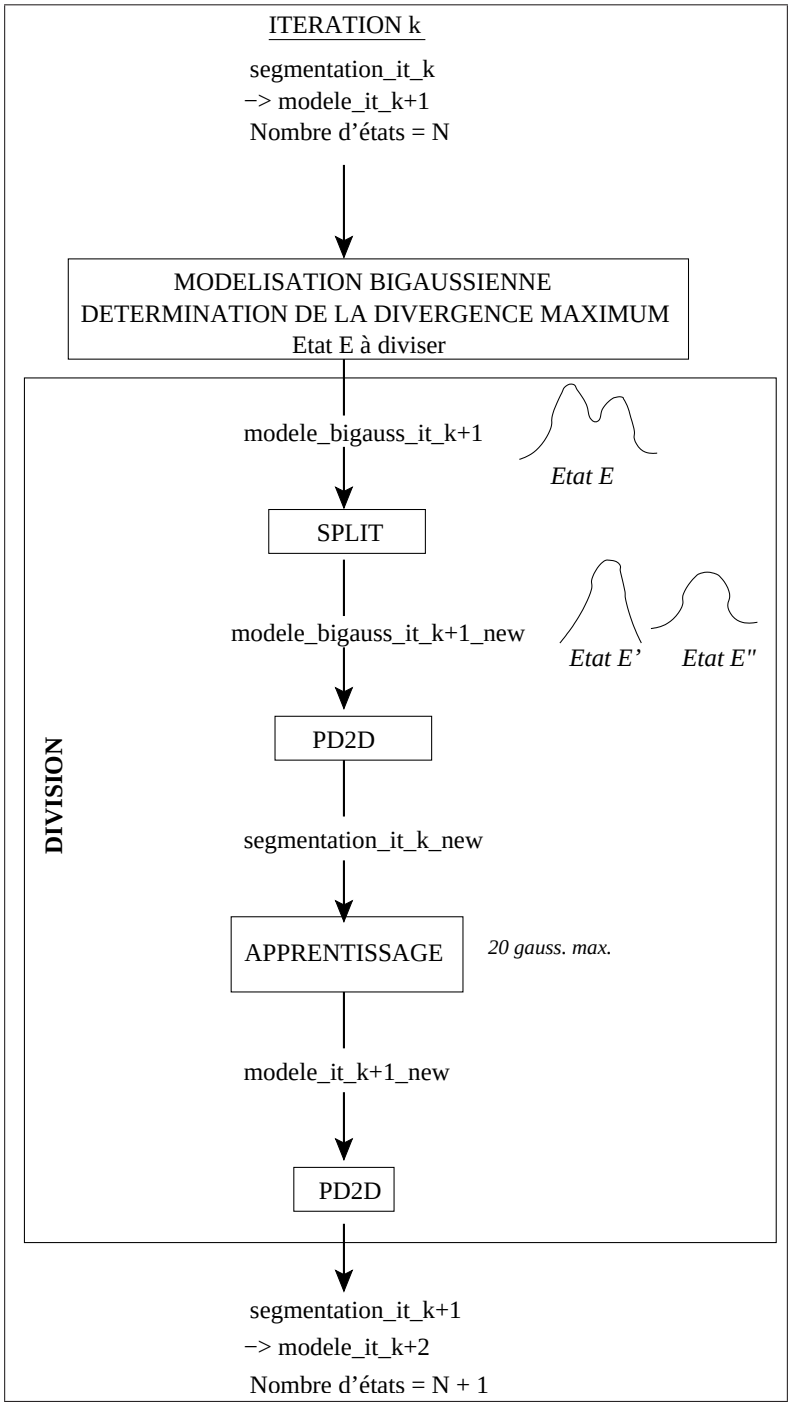


Fig. 3.17. Stratégie de division d'état

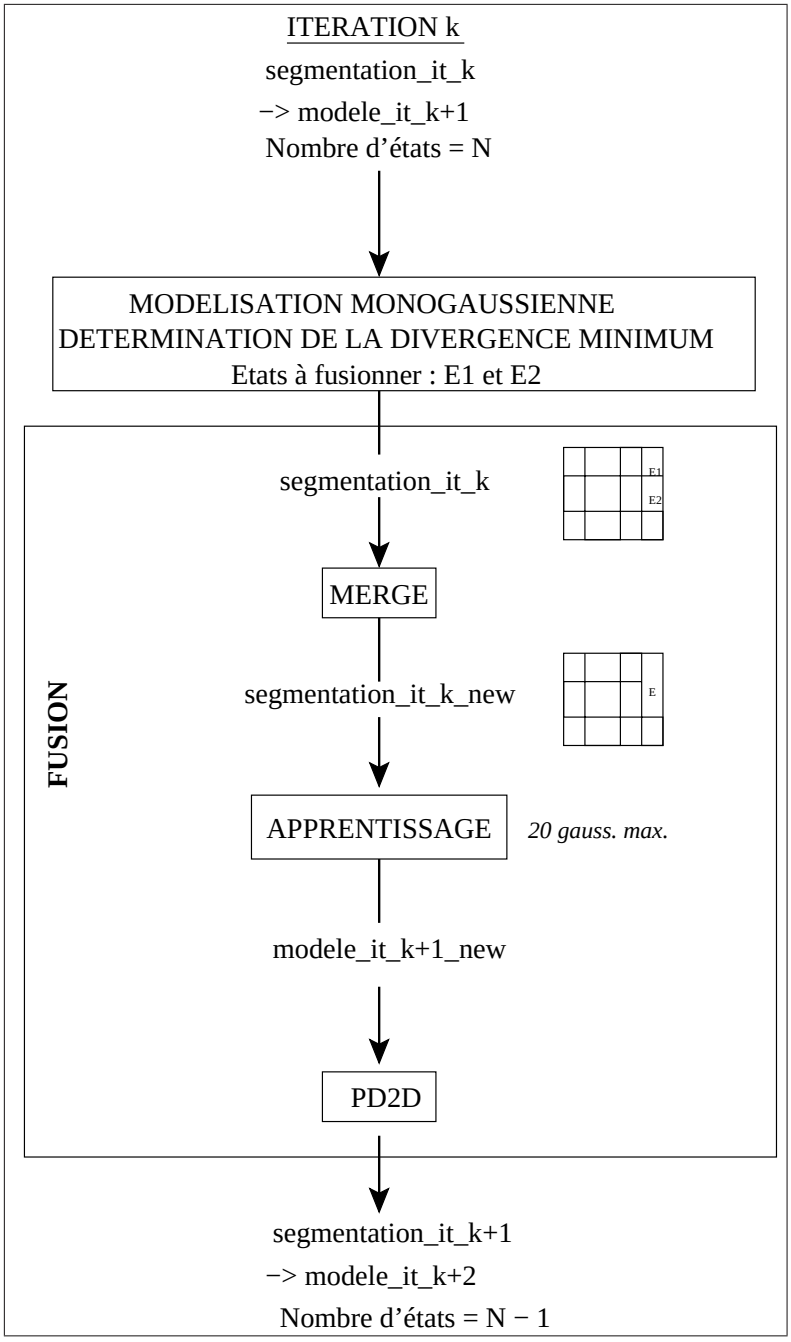


Fig. 3.18. Stratégie de fusion d'état

3.6.2.5 Division et fusion d'état

La difficulté est de trouver une stratégie déterminant l'enchaînement des opérations (division d'état, fusion d'état ou simple itération) à effectuer et pour quelle classe. Nous avons mis en œuvre dans un premier temps une stratégie consistant à selectionner l'opération qui donne le plus faible taux d'erreur sur la base de validation. On part du meilleur système (c'est-à-dire celui donnant 2.04%) et à chaque étape, on effectue soit une division parmi l'une des 10 possibles, soit une fusion parmi l'une des 10 possibles, soit une itération parmi l'une des 10 possibles.

Les résultats obtenus sont donnés figure 3.19. Les opérations effectuées sont indiquées sur le graphe ("I" pour itération simple, "D" pour division et "F" pour fusion). On parvient à un taux d'erreur de 1.8%.

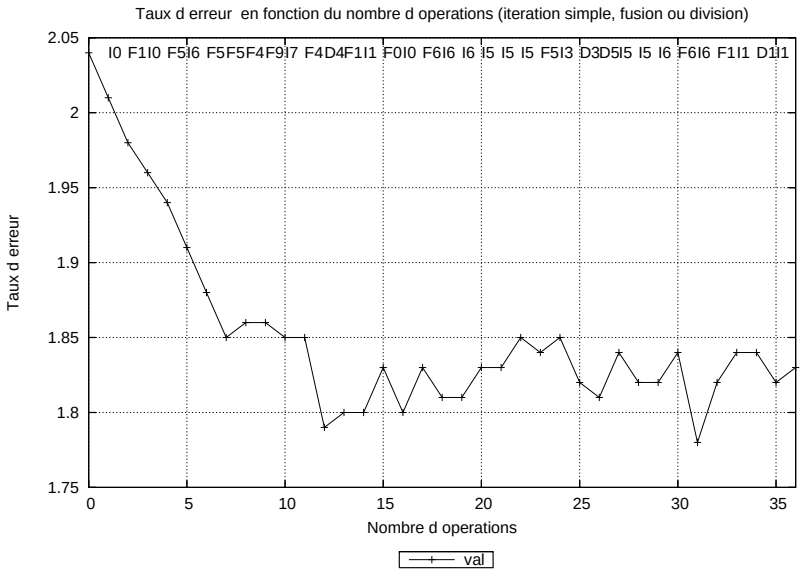


Fig. 3.19. Taux d'erreur en fonction du nombre d'opérations réalisées choisies selon *val*

La méthode proposée est assez élémentaire, mais elle conduit déjà à une diminution de 12% du taux d'erreur. D'autres approches sont envisageables. Par exemple une stratégie consistant à effectuer l'opération minimisant un coût à définir pourrait être mise au point. L'approche division et fusion d'état laisse entrevoir des perspectives d'amélioration plus importantes avec une stratégie plus adaptée que celle présentée ici.

3.7 Résultats sur la base de test

Le taux d'erreur obtenu sur la base de test est de 2.32%. Les résultats sont un peu moins bons que sur la base de validation qui est connue comme étant plus facile que la base de test. On donne dans le tableau 3.12, la matrice de confusion associée. On représente également figure 3.20 les images mal reconnues en indiquant les confusions réalisées. On constate les mêmes types d'erreur que sur la base de validation.

Fig. 3.20. Images mal reconnues et confusions réalisées sur la base de test de MNIST

Tab. 3.12. Matrice de confusions sur la base de test de MNIST

Chiffre reconnu \ *Vérité terrain*	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
0	970	0	1	0	0	1	6	0	13	1
1	1	1122	0	0	0	0	3	2	1	3
2	2	7	1010	7	0	0	0	13	4	5
3	0	0	2	984	0	3	0	0	1	7
4	0	4	2	0	963	0	1	4	2	6
5	0	1	0	6	0	885	9	0	9	2
6	4	0	3	0	2	1	936	0	2	0
7	1	0	6	6	0	0	0	998	3	11
8	2	1	8	4	3	1	3	2	934	8
9	2	0	0	3	14	1	0	9	5	966

3.8 Application à la reconnaissance de lettres manuscrites isolées

Afin de pouvoir aborder la reconnaissance de mots isolés dans le prochain chapitre, nous allons montrer préalablement que l'application de AMBRES aux chiffres manuscrits peut être transposée à la reconnaissance de lettres.

Nous allons pour cela utiliser la base ENST-FAX-CHAR [55] de lettres manuscrites isolées bâtons issus de télécopiés créée à l'ENST (Ecole Nationale Supérieure des Télécommunications).

3.8.1 Base de données ENST-FAX-CHAR

On donne figure 3.21 un exemple d'image pour chacune des 26 classes. Les images originales de la base sont binaires.

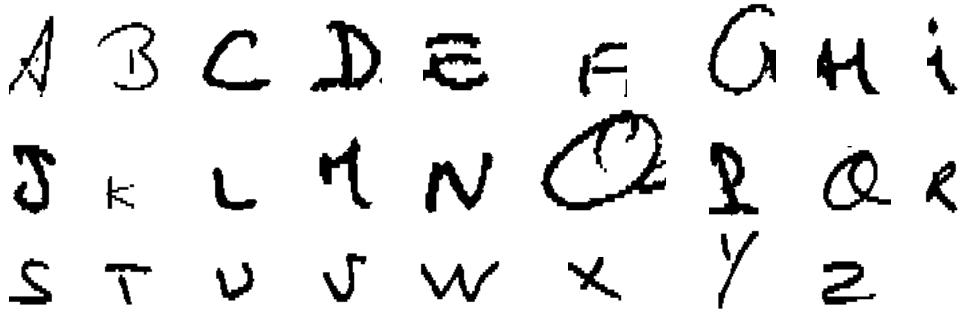


Fig. 3.21. Exemples d'images de la base ENST-FAX-CHAR

Comme elle n'est pas divisée en base de développement, validation et test, on divise ENST-FAX-CHAR en deux sous-bases : une base de développement et une base de validation. La base n'est en effet pas assez grande pour avoir en plus une base de test (5841 images en tout). Pour la répartition, on procède comme pour la base MNIST : 80% des images de chaque classe appartiendra à la base de développement et les 20%

restant à la base de validation. La répartition des différentes classes est donnée dans le tableau 3.13. On constate que celle-ci varie suivant les classes. Toutefois nous ne prendrons pas en compte cette remarque dans le calcul des vraisemblances, les classes sont considérées comme équiprobables.

Tab. 3.13. Répartition des images de la base ENST-FAX-CHAR

	A	B	C	D	E	F	G	H	I	J	K	L	M
Dév	204	215	215	217	217	150	183	217	217	134	92	216	148
Val	51	51	53	54	54	37	45	54	54	33	23	53	36
	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Dév	217	217	217	144	217	217	217	217	172	82	110	125	107
Val	54	54	54	35	54	54	54	54	43	20	23	31	26

3.8.2 Prétraitement

Au cours de l'étude et l'optimisation de l'ensemble des paramètres du modèle, nous avons constaté que la taille de la fenêtre d'analyse utilisée pour l'extraction des primitives spectrales locales dépendait de la taille des images. Or les images de la base ENST-FAX-CHAR ont des tailles différentes de celles de MNIST. Deux solutions s'imposent alors :

- soit on adapte la taille de la fenêtre d'analyse,
- soit on modifie la taille des images.

Les deux solutions ont été testées avec des résultats similaires. L'inconvénient de la première solution est que le temps de calcul est plus élevé du fait que les images de ENST-FAX-CHAR sont de plus grande taille que celles de MNIST. Nous choisissons donc d'utiliser une phase de prétraitement afin de normaliser la taille des images au format 28×28.

Ce prétraitement s'effectue en plusieurs étapes (figure 3.22) :

- on commence par détourer les images,
- on diminue ensuite la taille de l'image d'un facteur F tel que $F = \frac{20}{Max}$ où $Max = max(largeur, hauteur)$,
- enfin, on complète avec du blanc autour de l'image pour obtenir une taille égale 28×28.

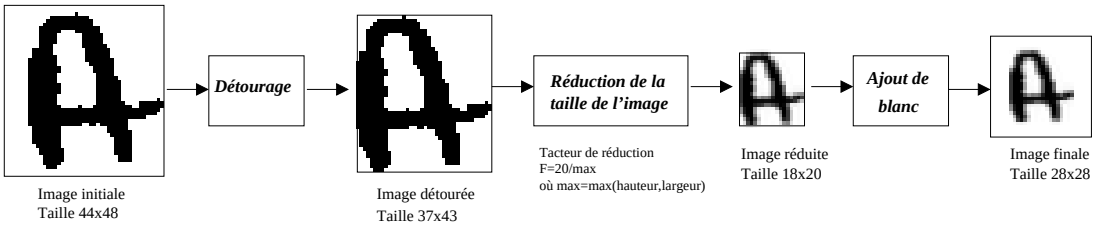


Fig. 3.22. Prétraitement : redimensionnement des images à la taille 28×28

On donne figure 3.23 le résultat obtenu après le prétraitement pour les images de la figure 3.21.

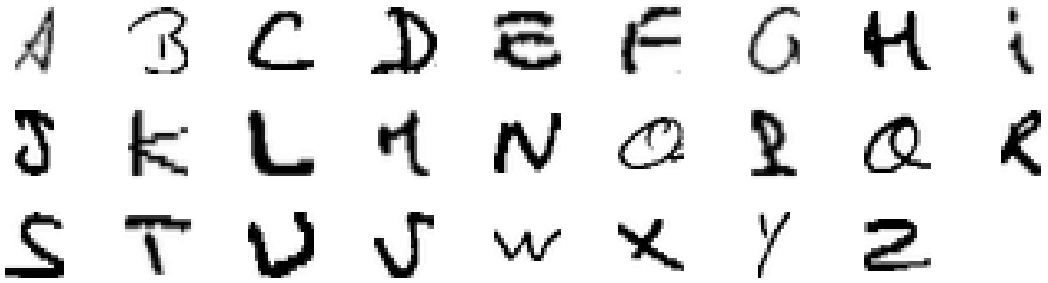


Fig. 3.23. Exemples d'images de la base ENST-FAX-CHAR après prétraitement

3.8.3 Résultats

On obtient un taux de reconnaissance d'environ 90%, ce qui est un assez bon résultat compte tenu du faible nombre d'images par classe pour l'apprentissage. En top 2, on a un taux de reconnaissance de 95%. La matrice de confusion est donnée tableau 3.14. On représente également figure 3.24 les images mal reconnues en indiquant les confusions réalisées.

3.9 Conclusion

Nous avons présenté un système de reconnaissance de caractères isolés appliqué avec succès à la reconnaissance de chiffres et de lettres majuscules isolés. Une étude approfondie des paramètres du modèle markovien a été réalisée. En particulier l'étude des primitives spectrales locales également décrite dans [51] a permis de diminuer de 15% le taux d'erreur obtenu initialement avec la méthode proposée par Chevalier [16]. On obtient ainsi un taux d'erreur de 2.04% contre 2.38%.

Une autre contribution est l'étude des performances du système par une analyse des modèles et des erreurs commises. Celle-ci nous a conduit à proposer des améliorations et perspectives d'évolution. Dans un premier temps le constat, selon lequel les ensembles des erreurs commises par deux systèmes ayant différents vecteurs de primitives spectrales locales étaient différents, nous a permis de mettre au point une stratégie de combinaison de ces systèmes. Plusieurs méthodes de combinaison ont été étudiées et l'une d'entre elles a conduit à une diminution de 25% du taux d'erreur. Un taux d'erreur de 1.52% ainsi obtenu permet d'atteindre les meilleures performances à l'état de l'art. Dans un second temps l'étude des modèles obtenus à l'issue de l'apprentissage a montré quelques imprécisions dans la modélisation : on a notamment constaté que les trous du "8" n'étaient pas pris en compte dans un état. Une perspective d'évolution consistant à effectuer de la division et de la fusion d'état a ainsi été proposée pour pallier à ce problème. La difficulté de cette approche consiste à trouver une bonne stratégie d'enchaînement des opérations de division et de fusion : une stratégie de base a été proposée

Fig. 3.24. Images mal reconnues et confusions réalisées sur la base ENST-FAX-CHAR

Tab. 3.14. Matrice de confusions sur la base ENST-FAX-CHAR

Chiffre reconnu \ *Vérité terrain*	*A*	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*
A	50					1	1						1					1							
B		46					1				1				1			1								5
C			44		1		1					2			1											
D				39					2	1					1											
E					48	1												1								
F						29																				
G		2	1		3		38																			
H								51			1		2								1				2	
I									51	1										1		1				
J					1					27																
K											18							2						1		
L			1						1			51														
M								2			1		30													
N								1					1	54								1	1			
O			1	13	1										48											
P	1			1		2							2			53		1								
Q		2		1			4			1					3	1	32									
R		1				3					1						2	48								
S						1				2									54							
T																				53						
U																					49	3				
V																					4	38	1			
W																							18			
X																								22	8	
Y																									21	
Z																										21

et a permis une diminution de 12% du taux d'erreur.

Notre système de reconnaissance de caractères manuscrits a été évalué dans le cadre de la campagne d'évaluation RIMES : les résultats sont donnés chapitre 6.

L'étape suivante de la reconnaissance de l'écriture manuscrite est la reconnaissance de mots abordée dans le prochain chapitre.

Chapitre 4

AMBRES appliquée à la reconnaissance de mots manuscrits

4.1 Introduction

Première étape avant d'aborder la reconnaissance de mots proprement dite, la reconnaissance de caractères isolés a été traitée avec succès grâce à l'approche AMBRES. Alors que la majorité des approches bidimensionnelles de la littérature s'intéressent essentiellement à la reconnaissance de caractères [71] ou de mots avec un petit vocabulaire (reconnaissance de montants de chèques par exemple [20]), nous nous intéressons dans ce chapitre à la reconnaissance de mots avec un grand vocabulaire.

L'approche générale que nous avons choisie pour aborder la reconnaissance de mots est d'apprendre des modèles de lettres puis de les concaténer pour pouvoir construire n'importe quel modèle de mot. L'intérêt de cette méthode est que l'on n'apprend pas uniquement les modèles de mots présents dans la base d'apprentissage et que l'on peut ainsi s'adapter à n'importe quel vocabulaire. De plus, alors qu'on n'a peu ou pas d'images pour chaque classe de mot, on dispose de nombreuses images de chaque classe de lettres. En effet, on peut utiliser celles présentes à l'intérieur des images de mots.

Deux difficultés liées à cette approche apparaissent alors :

- La première est de réaliser correctement la concaténation des modèles de lettres pour obtenir des modèles de mots. De plus, la segmentation initiale uniforme utilisée pour la reconnaissance de chiffres n'apparaît plus comme pertinente puisque les lettres ont des tailles différentes selon qu'elles sont des majuscules ou des minuscules ou qu'elles ont une hampe ou un jambage.
- La seconde difficulté consiste à obtenir suffisamment d'images de lettres pour construire des modèles de lettres précis. En effet, on ne dispose pas d'une base publique de lettres majuscules et minuscules isolées et il existe très peu de mots de une lettre. On va donc devoir les extraire des imagettes de mots. Effectuer un découpage manuel des imagettes de mots s'avérerait fastidieux et présenterait peu d'intérêt, un découpage automatique doit donc être envisagé.

Après avoir étudié ces deux problématiques, nous donnons les résultats obtenus sur la base publique Senior et Robinson.

4.2 Construction d'un modèle de mots à partir des modèles de lettres

4.2.1 Approche

Pour construire des modèles de mots, l'approche consiste à concaténer les modèles de lettres appris sur des images de lettres isolées (figure 4.1). Pour décrire cette procédure de concaténation, nous nous intéressons uniquement à l'obtention d'un modèle de mot de deux lettres c_1c_2, la procédure se généralisant facilement à un mot de n lettres. En effet, pour un mot de n lettres, on effectuera $n-1$ procédures de concaténation.

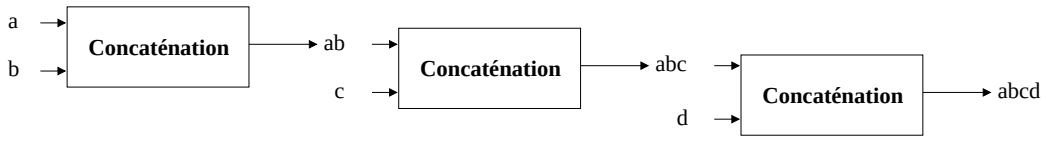


Fig. 4.1. Construction du modèle du mot "abcd" par concaténation des modèles des lettres a, b, c et d

Partant des modèles de deux caractères c_1 et c_2, pour obtenir le modèle du mot c_1c_2 l'idée est de supprimer la colonne d'états de blanc à droite du caractère c_1 (états 4, 9, 14, 19, 24, 29 et 34) et la colonne d'états de blanc à gauche du caractère c_2 (états 0, 5, 10, 15, 20, 25 et 30) afin de créer une liaison entre les deux caractères (voir figure 4.2). Ainsi à l'issue de la concaténation, si on appelle n_y et n_x respectivement le nombre d'états vertical et horizontal d'un caractère c (ici en l'occurrence $n_x = 5$ et $n_y = 7$), le modèle du mot c_1c_2 aura une topologie en $[2(n_x - 1)] \times n_y$. En généralisant à un mot de n lettres, la topologie sera en $[n(n_x - 2) + 2] \times n_y$.

Nous devons donc mettre au point une stratégie permettant d'obtenir $P_{c_1c_2}(O|\omega)$ à partir de $P_{c_1}(O|\omega)$ et $P_{c_2}(O|\omega)$, et $P_{c_1c_2}(\omega)$ à partir de $P_{c_1}(\omega)$ et $P_{c_2}(\omega)$.

Du fait de l'indépendance des observations conditionnellement aux états, $P_{c_1c_2}(O|\omega)$ est aisément calculable à partir de $P_{c_1}(O|\omega)$ et $P_{c_2}(O|\omega)$. On a ainsi :

$$P_{c_1c_2}(O|\omega) = \begin{cases} P_{c_1}(O|\omega) & \text{si } \omega \in c_1 \\ P_{c_2}(O|\omega) & \text{si } \omega \in c_2 \end{cases}.$$

Les probabilités de transition sont quant à elles plus complexes à déterminer. Plusieurs solutions peuvent être envisagées. Nous avons choisi l'approche suivante structurée en trois cas différents :

Cas 1 Les probabilités de transition entre deux états appartenant à une même lettre (c_1 par exemple) correspondent aux probabilités de transition du modèle de lettre

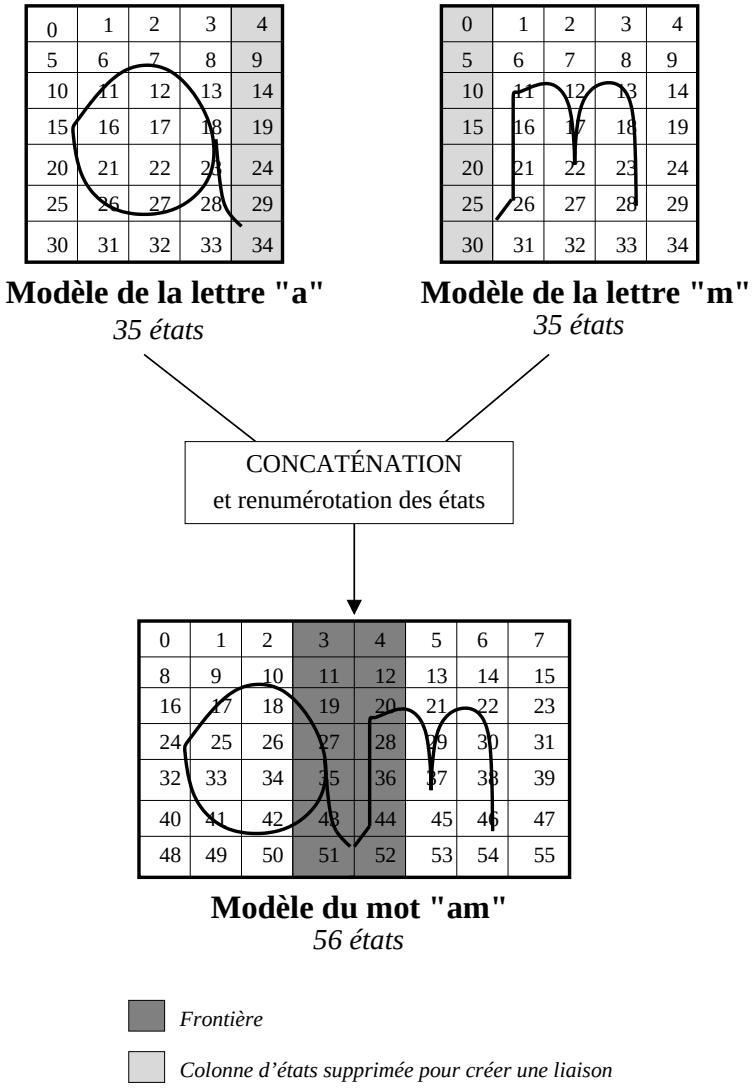


Fig. 4.2. Principe adopté pour la concaténation de deux modèles de caractères

correspondant.

Cas 2 Les probabilités de transition entre deux états appartenant à deux lettres différentes (c_1 et c_2) et situées à la frontière entre ces deux lettres sont obtenues en moyennant les probabilités de transition de la première lettre avec celles de la seconde lettre.

Cas 3 Les probabilités de transition entre deux états appartenant à deux lettres différentes (c_1 et c_2) et situées en dehors de la frontière entre ces deux lettres est fixée à zéro.

Un exemple pour chacun de ces trois cas possibles est donné figure 4.3.

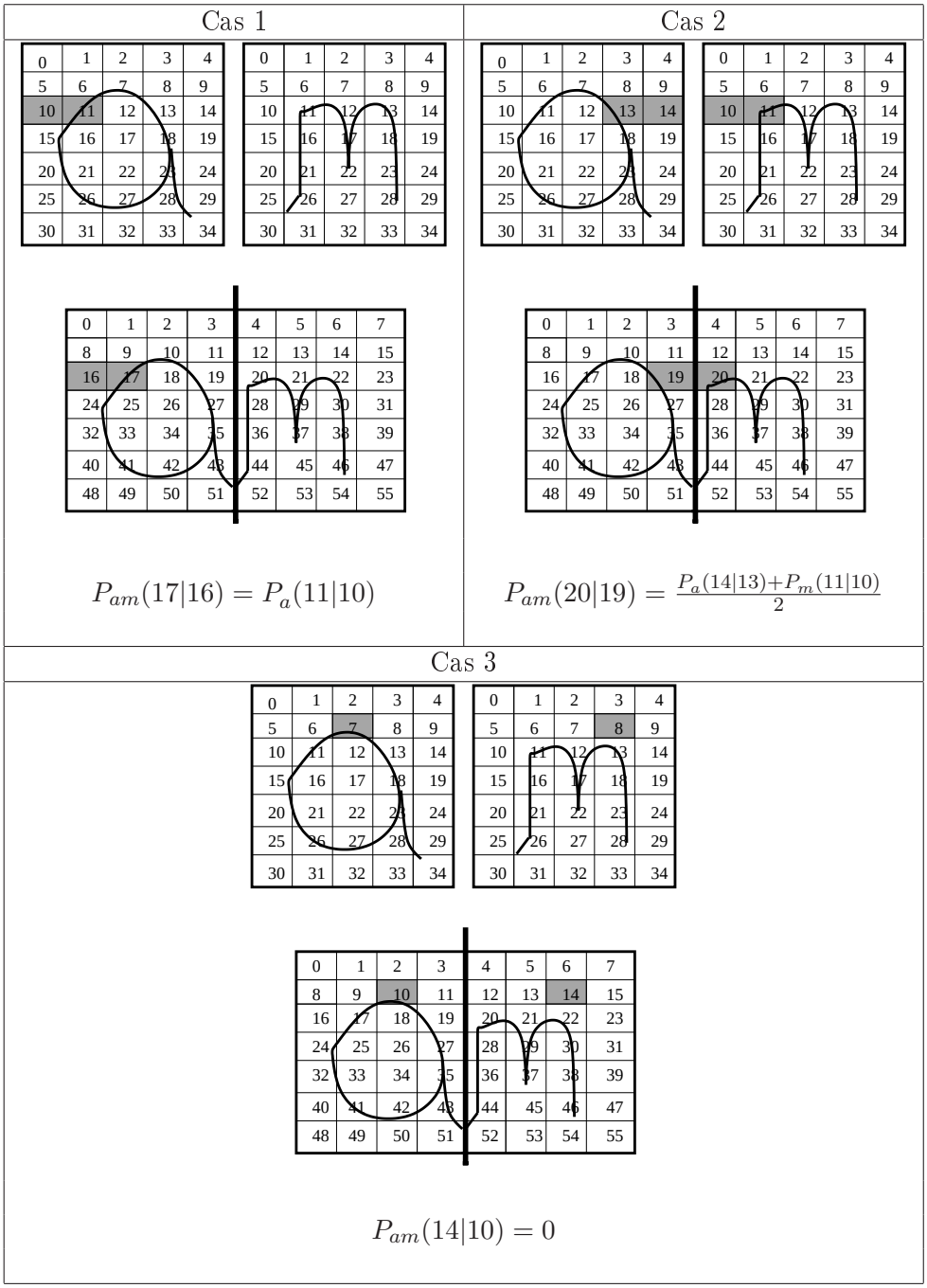


Fig. 4.3. Exemple pour chacun des trois cas possibles définis pour le calcul des probabilités de transition après concaténation

4.2.2 Validation de l'approche sur les nombres

Afin de valider la méthode de concaténation décrite ci-dessus, nous exploitons les modèles de chiffres obtenus sur la base MNIST pour effectuer de la reconnaissance de nombres de deux chiffres. Nous construisons pour cela une petite base de 80 images de nombres obtenues à partir des images de MNIST et considérons un vocabulaire de taille 90 (nombre de 10 à 99). Les images de nombres sont obtenues en "collant" deux images de chiffres de la base de validation. Pour simuler une liaison entre les deux chiffres un certain nombre de pixels est ôté respectivement à droite et à gauche de la première et de la deuxième image de chiffre (voir figure 4.4).

Fig. 4.4. Exemple d'imagette de nombre obtenue en "collant" deux images de la base MNIST

Les modèles des 80 nombres sont obtenus par concaténation des modèles de chiffres en adoptant la stratégie de concaténation décrite dans le précédent paragraphe. On peut ensuite procéder à la phase de reconnaissance. On obtient alors un taux d'erreur de 5% au niveau nombre et un taux d'erreur de 2.5% au niveau chiffre ce qui correspond au même taux d'erreur obtenu en effectuant la reconnaissance de caractères isolés de chacune des 160 imagettes de chiffre.

La procédure de concaténation donne donc les résultats attendus pour l'obtention de modèles de mots à caractères liés. On peut constater sur la courbe représentée figure 4.5 que la concaténation est d'autant plus efficace que les caractères sont collés. En effet, cette courbe représente le taux d'erreur obtenu au niveau chiffre pour différents degrés de rapprochement entre les deux chiffres (il s'agit en fait du nombre de pixels ôtés respectivement à droite et à gauche de la première et de la seconde image).

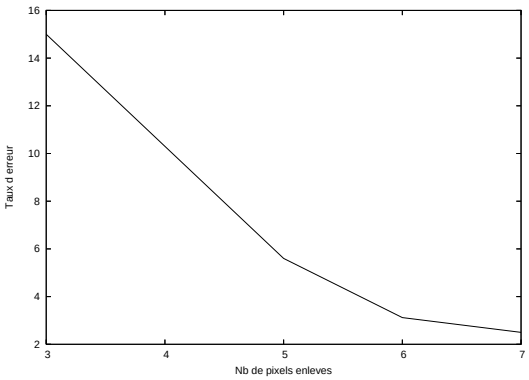


Fig. 4.5. Taux d'erreur obtenu au niveau chiffres pour différents nombres de pixels enlevés respectivement à droite et à gauche de la première et de la seconde image

4.2.3 Nouvelle définition de la segmentation initiale

Pour la concaténation de deux chiffres, il n'y pas de problème d'alignement car tous les chiffres sont de la même taille. Concernant les lettres, les tailles varient suivant si la lettre présente une hampe ou un jambage ou encore si elle est une majuscule ou une minuscule. La segmentation initiale uniforme utilisée pour les chiffres n'est donc plus applicable. En effet, prenons l'exemple de deux lettres "b" et "e" dont on souhaite concaténer les modèles. Si on utilise une segmentation initiale uniforme pour chacune des deux lettres (figure 4.6), on voit que pendant la concaténation on va juxtaposer un état haut du "b" avec un état haut du "e" (état 8 du "b" et état 6 du "e), ce qui n'est pas cohérent.

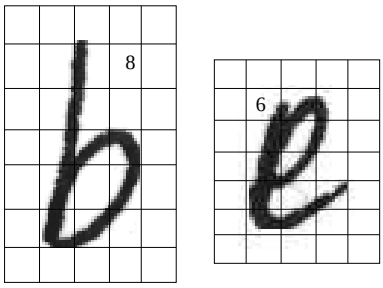


Fig. 4.6. Cas de segmentation initiale uniforme de lettres

Une nouvelle stratégie de segmentation doit ainsi être mise en oeuvre afin de réaliser correctement la concaténation de deux modèles de lettre. Pour ce faire, un point important est d'aligner leurs parties centrales. Pour conserver la topologie 5×7 optimisée sur les chiffres, on procéde de la façon suivante (figure 4.8) :

- pour les 5 états horizontaux : les deux états extérieurs sont conservés pour modéliser l'information de blanc et les 3 autres états pour modéliser le centre,

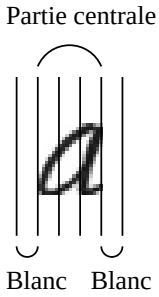


Fig. 4.7. Segmentation initiale non uniforme des lettres : répartition horizontale des états

- pour les 7 états verticaux : les états du haut et du bas sont utilisés pour modéliser l'information de blanc, les trois états du centre pour modéliser la partie centrale et les deux autres états pour modéliser la hampe et la jambe.

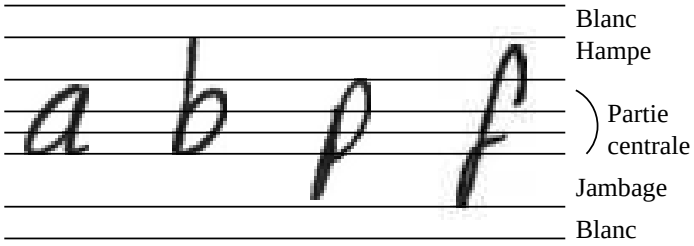


Fig. 4.8. Segmentation initiale non uniforme des lettres : répartition verticale des états

4.3 Apprentissage des modèles de lettres

Nous avons vu précédemment que l'on pouvait obtenir des modèles de mots par concaténation de modèles de lettres. Il reste donc à aborder l'apprentissage de ces modèles de lettres, phase essentielle pour l'obtention de modèles de mots précis. Ces modèles de lettres vont être appris sur des images de lettres automatiquement extraites des images de mots.

Pour le découpage automatique d'un mot en lettres, l'idée générale est d'utiliser sa segmentation en états en supposant que celle-ci soit connue. Cette segmentation va en effet permettre, en dessinant la frontière entre chacune des lettres, d'obtenir les segmentations en états de chacune des lettres constituant le mot ainsi que les observations associées.

Considérons le cas du mot "am" de la figure 4.9.

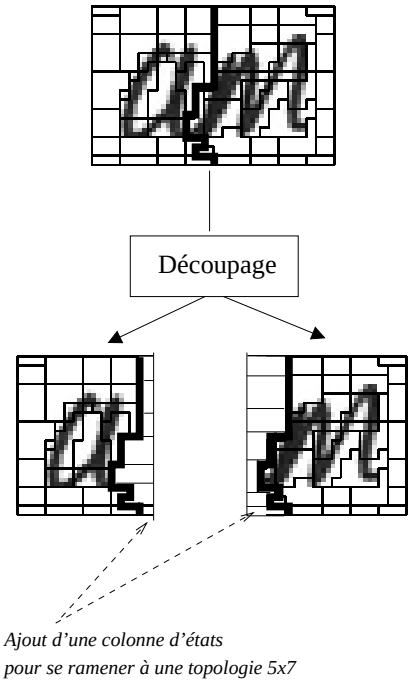


Fig. 4.9. Exemple de découpage d'une image de mot à partir de sa segmentation en états

Pour obtenir la frontière entre les deux lettres "a" et "m" à partir de la segmentation en états du mot "am", il suffit de tracer la limite entre les états étiquetés comme appartenant à la première lettre et les états étiquetés comme appartenant à la seconde lettre. Pour obtenir ensuite l'image de la lettre "a", par exemple, on place dans une nouvelle image les sites correspondant aux états étiquetés comme appartenant à la lettre "a" et on complète avec des pixels de fond afin d'obtenir une image rectangulaire.

Le problème qui apparaît alors est que l'on ne connaît pas les segmentations en états des différents mots de la base d'apprentissage. Nous allons donc utiliser un algorithme itératif du type EM tel qu'illustré figure 4.10 et décrit ci-après.

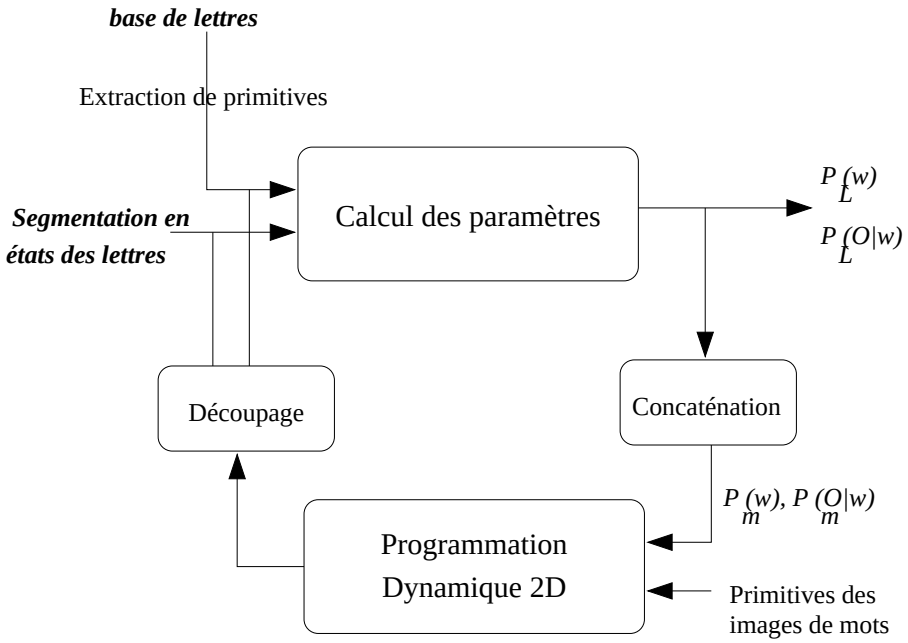


Fig. 4.10. Apprentissage des modèles de lettre à partir des mots

– **Initialisation**
Une petite base d'images de lettres est créée manuellement (environ 20 images par lettre). Elle permet d'initialiser les paramètres des modèles de lettres .

– **Itérations**
Les modèles de lettres sont concaténés pour obtenir les modèles de mots. La programmation dynamique permet ensuite, à partir de ces modèles, de déterminer les segmentations en états des différents mots de la base d'apprentissage. Puis on peut découper les images de mots en images de lettres comme décrit précédemment puisque leurs segmentations sont connues. Enfin, on apprend les modèles de lettre sur cette nouvelle base d'images de lettres ainsi créée.

Ce processus est itéré un certain nombre de fois jusqu'à convergence. On obtient ainsi, à l'issue de l'algorithme, les modèles de chaque lettre appris sur l'intégralité des lettres contenues dans tous les mots de la base d'apprentissage.

4.4 Reconnaissance de mots

Après avoir obtenu les modèles de lettres lors de la phase d'apprentissage, on peut les concaténer pour construire l'ensemble des modèles des mots contenus dans le vocabulaire.

Comme nous abordons une tâche de reconnaissance de mots avec un grand vocabulaire, tester l'ensemble des modèles de mots sur chaque image de la base de test serait trop coûteux en temps de calcul. Ainsi, nous adoptons une stratégie de réduction dynamique de la taille du lexique.

Les critères d'information globale sont souvent utilisés dans la littérature [46]. Ici, nous proposons d'exploiter la hauteur, la largeur ainsi que l'aire de l'image. On calcule ces trois valeurs sur toutes les images de la base d'apprentissage, puis on modélise par une gaussienne leur probabilité d'apparition pour chaque mot du dictionnaire ($P(hauteur = val|mot)$, $P(largeur = val|mot)$ et $P(aire = val|mot)$). Ainsi lors du test, pour une image donnée, on calcule sa hauteur h_1, sa largeur L_1 et son aire A_1. On ne lui teste ensuite que les modèles de mot tels que $P(hauteur = h_1|mot)$, $P(largeur = L_1|mot)$ et $P(aire = A_1|mot)$ soient supérieurs à un seuil fixé.

Les expériences menées sur la base Senior et Robinson ont montré que cette méthode permettait de diminuer le nombre total de modèles testés sur l'ensemble des images de test de 98%, et de réduire le taux d'erreur de 26%.

4.5 Base de données Senior et Robinson

4.5.1 Présentation

La base Senior et Robinson [82] est une base monoscripteur disponible gratuitement sur Internet. Elle est constituée de 25 pages de transcription des fichiers $B1$ et $G3$ du corpus LOB (London / Oslo-Bergen) écrites en anglais. 22 pages sont utilisées pour l'apprentissage des modèles et les 3 pages restantes pour le test. Le vocabulaire est composé de 1334 mots contenant le vocabulaire de la base d'apprentissage ainsi que le vocabulaire de la base de test (il n'y a donc pas de problème de mot hors vocabulaire). L'inconvénient de cette base est qu'elle comporte assez peu d'images de mots d'apprentissage (environ 4000) ce qui ne nous permet pas de la diviser en une base de développement et une base de validation.

Un exemple de page de cette base est donnée figure 4.14. On y a représenté les boîtes englobantes de chaque mot fournies avec la base.

Il est intéressant d'observer la répartition des lettres dans la base d'apprentissage et dans la base de test (figure 4.11). Les proportions de lettres sont les mêmes dans les deux bases. On peut également observer la répartition des mots suivant leur nombre de lettres (figure 4.12) et la répartition des lettres suivant la taille de mots dans lesquels elles se situent (figure 4.13).

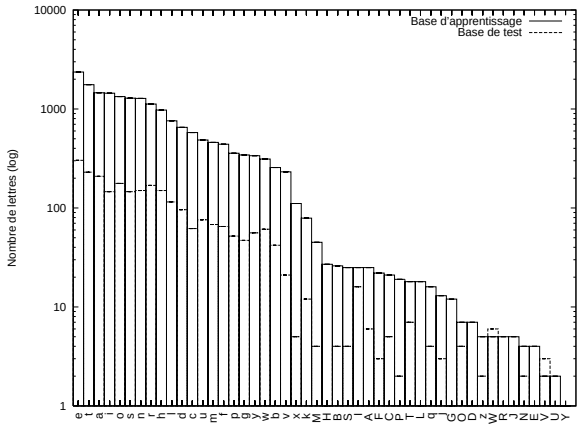


Fig. 4.11. Répartition des lettres dans la base Senior et Robinson

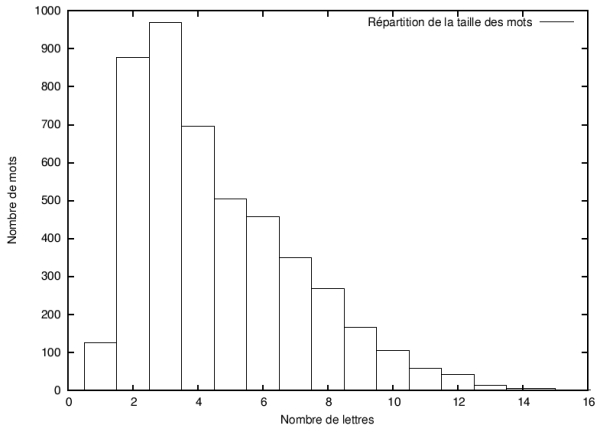


Fig. 4.12. Répartition des mots suivant leur nombre de lettres dans la base Senior et Robinson

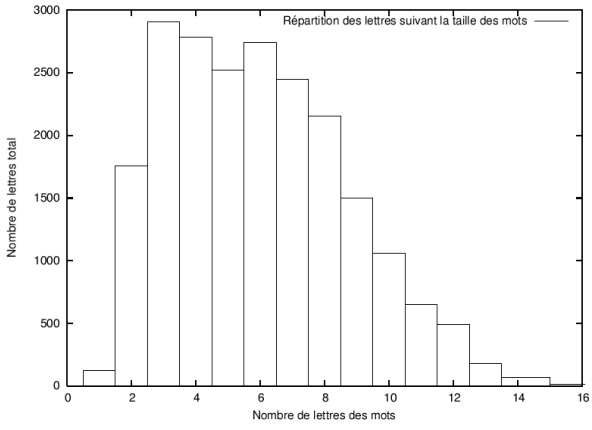


Fig. 4.13. Répartition des lettres suivant la taille des mots dans lesquels elles se situent dans la base Senior et Robinson

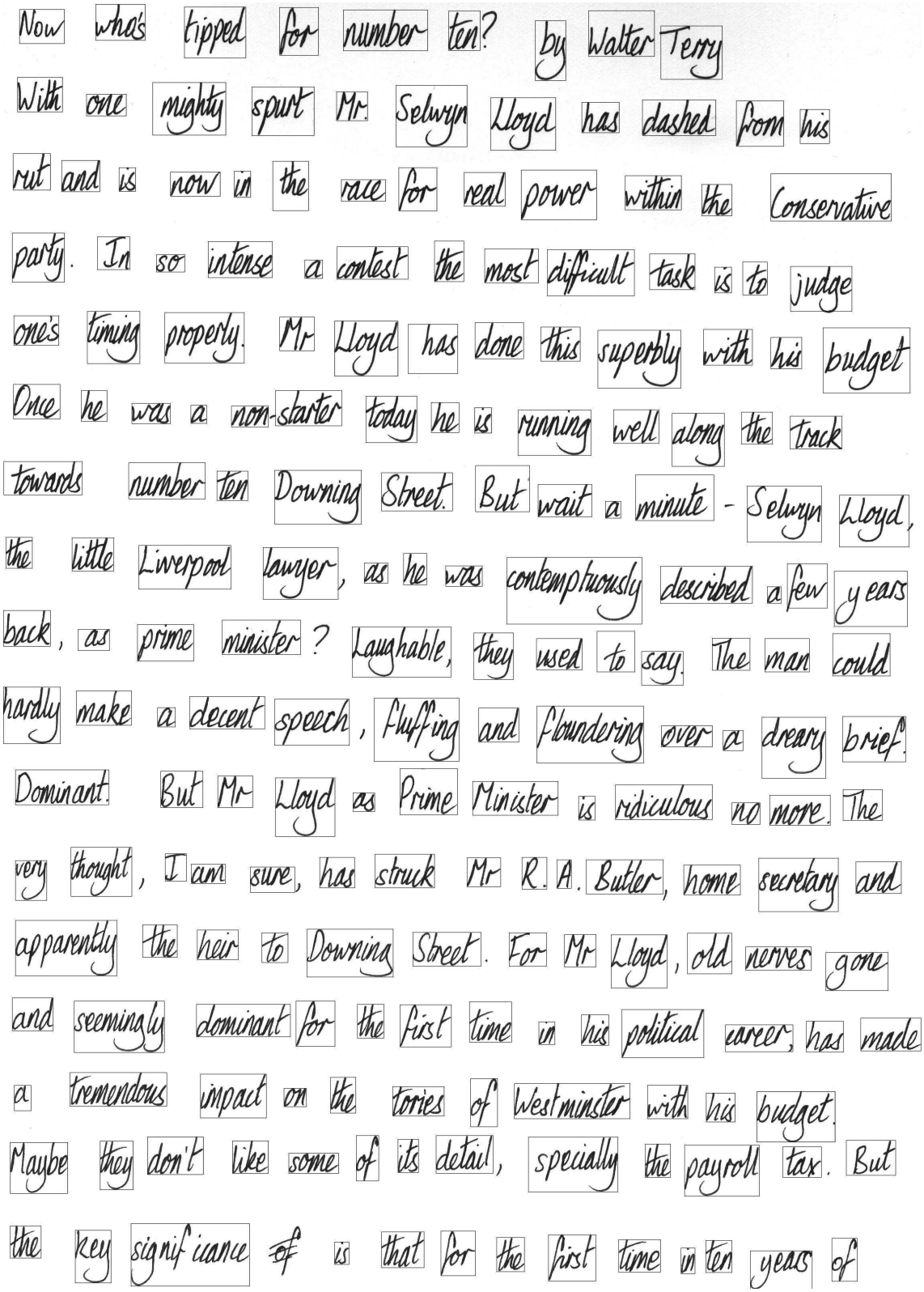


Fig. 4.14. Exemple de page de la base Senior et Robinson avec la représentation des boîtes englobantes

4.5.2 État de l'art sur la base Senior & Robinson

Dans la littérature, on trouve assez peu d'expériences réalisées sur cette base. En effet, la plupart des approches proposées pour la reconnaissance de mots sont testées soit sur des tâches de lecture de chèque ou de tri postal soit sur des bases non publiques. On donne néanmoins un résumé des principales performances, en terme de taux de reconnaissance au niveau mots, obtenues sur la base Senior & Robinson dans le tableau 4.1.

Tab. 4.1. État de l'art sur la base Senior et Robinson

Taux de reconnaissance niveau mots	Référence
93.4%	Senior *et al.*, 1992 [82]
85.5%	Senior, 1994 [81]
83.6%	Vinciarelli, 2002 [97]
89.5%	Wienecke *et al.*, 2002 [99]

4.6 Prétraitement

4.6.1 Débruitage des images

Une phase de filtrage des images est réalisée afin de réduire le bruit (voir exemple figure 4.15). Ce prétraitement permet de diminuer de 8% le taux d'erreur obtenu sans prétraitement.

Avant débruitage *Après débruitage*

Fig. 4.15. Exemple de résultat obtenu après débruitage des images

4.6.2 Redressement de la pente

Dans le cadre de notre approche, la segmentation des mots en lettres est une étape clé de la phase d'apprentissage. Aussi, dans ce paragraphe, nous étudions l'intérêt d'une phase préalable de correction de la pente de l'écriture qui pourrait éventuellement faciliter cette étape cruciale (exemple figure 4.16).

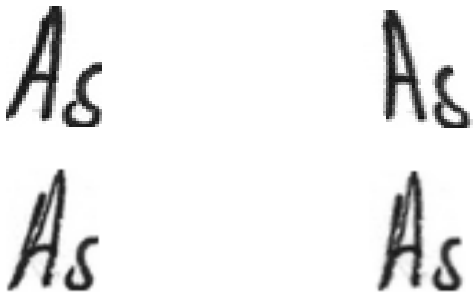
Nous avons réalisé l'expérience et avons constaté que cela n'améliorait pas les performances sans pour autant les dégrader fortement (diminution de 8% du taux de reconnaissance obtenu sans redressement).

Cela peut s'expliquer par le fait que les images de certains mots sont redressées différemment. Par exemple le mot "As" pose un problème puisqu'il va être redressé par

Image originale *Image après correction de la pente*

Fig. 4.16. Exemple de correction de la pente pour une image de la base Senior et Robinson

rapport à la pente du A majuscule : soit par rapport à son côté droit, soit par rapport à son côté gauche (voir figure 4.17). On obtient donc des lettres orientées différemment ce qui détériore les modèles associés. Si on n'effectue pas de redressement, dans le cas ici d'un unique scripteur, l'écriture et donc les lettres sont toujours penchées de la même façon : les modèles de lettres sont plus précis.



Images originales *Images après correction de la pente*

Fig. 4.17. Exemple de correction de la pente posant un problème : deux redressements différents pour deux images d'un même mot

4.7 Résultats

Le taux d'erreur obtenu au niveau mot est de 73.7% et au niveau caractère de 80%.

Pour analyser les résultats, on peut observer les segmentations des images de mots en lettres. En effet, l'obtention des imagettes de lettres par découpage automatique à partir des segmentations en états est une étape clé dans l'algorithme de reconnaissance de mots que nous avons mis au point.

Par exemple figure 4.18, on donne les segmentations en lettres de deux imagettes de mots "could" et "back". On a pour cela tracé la limite entre les états appartenant à chacune des lettres.

On peut remarquer que la frontière n'est pas une ligne verticale et qu'elle épouse l'inclinaison des lettres. Ceci est un des avantages de la modélisation bidimensionnelle. En effet, dans le cas d'une modélisation monodimensionnelle une phase de correction de la pente aurait été nécessaire puisque les frontières obtenues auraient été de simples droites verticales.

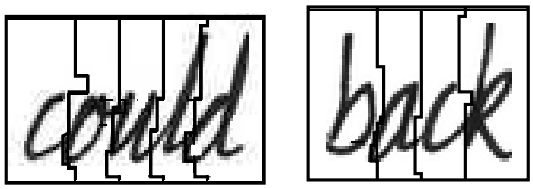


Fig. 4.18. Exemple de bonne segmentation de mots en lettres

Malgré la bonne segmentation en états de nombreux mots, on trouve également un certain nombre de mots mal segmentés en lettres. Par exemple la jambe du "y" se trouve associée à la lettre "b" dans le mot "by" (voir figure 4.19). Ce problème pourrait être réglé en introduisant des modèles de lettres en contexte, ici en l'occurrence un modèle de "b" suivi d'un "y" . Ce problème a commencé a être abordé mais on ne dispose pas assez d'exemples dans la base d'apprentissage pour obtenir un modèle de lettres en contexte assez précis. L'utilisation d'une plus grande base de données pourrait permettre d'introduire un certain nombre de modèles de lettres supplémentaires suivant leur contexte.

Fig. 4.19. Exemple de mauvaise segmentation de mots en lettres

4.8 Conclusion

Nous avons présenté un système de reconnaissance de mots manuscrits isolés avec un grand vocabulaire. Fondée sur l'approche AMBRES, la méthode proposée consiste à construire des modèles de mots par concaténation de modèles de lettres appris sur des imagettes automatiquement extraites des images de mots.

La redéfinition de la segmentation des lettres suivant si elles contiennent une hampe, un jambage ou si elles sont une majuscule ou minuscule a permis entre autre d'améliorer drastiquement les résultats. Dans nos travaux nous obtenons ainsi un taux de reconnaissance de 73.7% en considérant des mots de toute taille.

Les performances obtenues n'atteignent pas encore les performances à l'état de l'art mais de nombreuses améliorations sont encore possibles. En effet, la procédure de découpage automatique de mots en lettres montre quelques lacunes. L'utilisation d'une grande base de lettres isolées résoudrait le problème. De plus, la taille de la base Senior

et Robinson étant limitée, nous n'avons pu mettre au point une stratégie de modèles de lettre en contexte qui permettrait d'affiner les modèles de mots.

Chapitre 5

AMBRES appliquée à la structuration de documents manuscrits

5.1 Introduction

De nombreuses études ont été réalisées depuis longtemps pour l'analyse de la structure des documents dactylographiés [66, 1] tandis que l'analyse de la structure des documents manuscrits non contraints est plus récente [68, 14]. Du fait de la variabilité inhérente à l'écriture manuscrite, les techniques développées pour le dactylographié atteignent leurs limites. L'analyse de la structure physique d'un document manuscrit est donc un sujet de recherche très actif et reste encore un problème ouvert.

Nous nous intéressons à la structuration de documents manuscrits non contraints que sont les courriers manuscrits. Tout comme pour les fax, les entreprises reçoivent de nombreux courriers manuscrits. Pour optimiser le temps de traitement de ces courriers, un traitement automatique est envisagé. Ainsi la détermination de l'identité de l'expéditeur, de celle du destinataire, l'objet de la lettre, ... sont autant de tâches clés nécessaires au bon traitement d'un courrier. Pour parvenir à reconnaître ces différents champs, il est utile d'effectuer au préalable une structuration du document afin d'en extraire la structure physique, c'est-à-dire localiser les différents champs constituant un courrier (expéditeur, destinataire, date et lieu, objet, corps de texte, signature). Plutôt que d'effectuer la reconnaissance du document entier, l'extraction de la structure du courrier va permettre non seulement de réduire la zone où rechercher et reconnaître l'information, mais également d'adapter éventuellement le dictionnaire au champ traité.

La contribution majeure de ce chapitre se situe dans la mise en place d'un système complet de structuration de courriers manuscrits à partir de l'approche générale AMBRES présentée dans le chapitre 2. De plus une étude des erreurs a permis de mettre en place une phase de post-traitement dont l'objectif de chacune de ses étapes est de résoudre un type d'erreurs en particulier. En n'exploitant seulement des informations de

texture et de position, le système permet déjà d'obtenir de très bons résultats. La méthode laisse entrevoir des performances encore meilleures par la prise en compte d'une étape supplémentaire de reconnaissance de mots clés.

Le chapitre s'organise comme suit. Nous commencons par réaliser un bref état de l'art dans le domaine de l'analyse de la structure d'un document. Puis nous détaillons la méthode de structuration exploitant l'approche AMBRES. Nous donnons ensuite les résultats des expérimentations menées sur une partie de la base RIMES. Une étude des erreurs nous amène à proposer des améliorations grâce à une phase de post-traitement. Nous finissons enfin par proposer en perspective un système fondé sur une phase supplémentaire de reconnaissance de mots clés.

5.2 État de l'art

5.2.1 Types de documents

Il existe une grande variété de documents, on peut les classer de la façon suivante :

1. documents purement dactylographiés contenant plus ou moins d'images ou de graphiques (figure 5.1) ;
2. documents mixtes c'est-à-dire des documents contenant à la fois du dactylographié et du manuscrit ou des images et du manuscrit : ces documents sont plus ou moins contraints et parmi eux on trouve classiquement les questionnaires, les formulaires, les chèques, les enveloppes, les pages de garde de fax, les annotations de pages dactylographiées, etc. (figures 5.2 et 5.3) ;
3. documents purement manuscrits tels que les courriers manuscrits, les brouillons, les agendas, etc. : ces documents peuvent être également divisés en deux sous-catégories selon si on dispose d'informations *a priori* ou non comme cela peut être le cas pour un courrier manuscrit dont l'écriture est régie par des conventions culturelles plus ou moins différentes suivant le scripteur (figure 5.4).

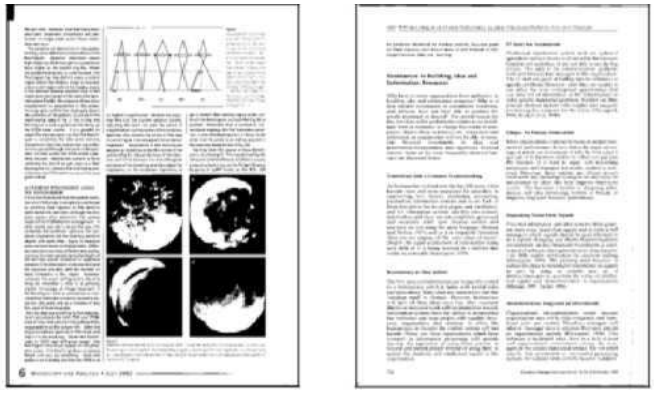
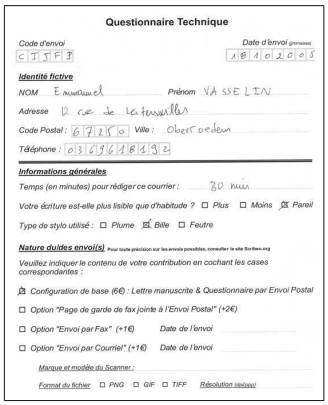
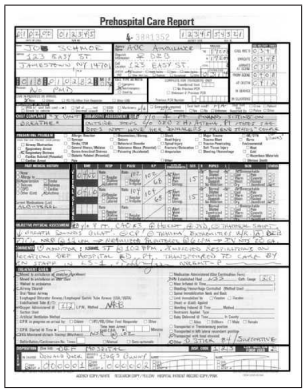


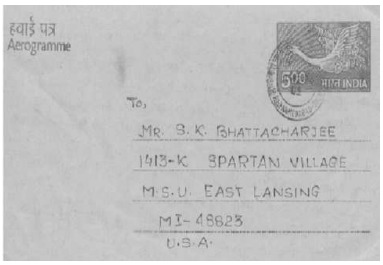
Fig. 5.1. Exemples de documents dactylographiés



Exemple de questionnaire



Exemple de formulaire médical [65]



Exemple d'enveloppe [36]

Fig. 5.2. Exemples de documents mixtes fortement contraints

Manuscrits et épreuves de "la Femme supérieure" Honoré de Balzac

Exemple de page de garde de fax

Fig. 5.3. Exemples de documents mixtes peu ou pas contraints

Brouillon de "Charmes" Paul Valéry

Exemple de courrier manuscrit

Fig. 5.4. Exemples de documents purement manuscrits non contraints

Nous nous intéressons dans le cadre de notre étude, à des documents purement manuscrits. Dans ce contexte, l'objectif de l'analyse vise à reconnaître des mots ou des lignes de texte du document entier ou d'un champ en particulier. Ainsi deux étapes préalables à la reconnaissance proprement dite sont nécessaires : la segmentation en mots ou en lignes et la structuration du document en différents champs.

5.2.2 Segmentation : découpage en mots ou lignes

Suivant le système de reconnaissance, les entités élémentaires à reconnaître peuvent être des lignes ou des mots. L'extraction des lignes de texte est souvent réalisée avant l'extraction des mots, c'est donc une étape importante pour la plupart des systèmes. Différentes méthodes ont été proposées dans la littérature suivant qu'elles sont fondées sur :

- un regroupement itératif des composantes connexes à partir d'images binaires [56],
- l'analyse des profils de projection où les minima locaux correspondent aux interlignes et les maxima locaux aux lignes elles-mêmes,
- les histogrammes de transition noir-blanc [39],
- la transformée de Hough [57],
- la recherche de minima locaux du contour externe des composantes connexes [28],
- les "level sets" après filtrage de l'image par une fenêtre gaussienne glissante [54].

La segmentation en mots se fait généralement après une phase de segmentation de lignes. La plupart de ces approches s'appuient sur l'analyse des espaces intra et inter mots, l'idée étant que les espaces entre deux mots sont plus grands que les espaces entre les lettres d'un mot. Ces espaces sont mesurés par une distance entre composantes connexes pour laquelle de nombreuses métriques existent.

5.2.3 Structuration : analyse de la structure physique

La structuration consiste à extraire la structure physique du document et ainsi permettre d'isoler les composantes d'un champ que l'on cherche à identifier. La difficulté de cette phase a été clairement évoquée par Sayre en 1973 et peut être résumée par le dilemme suivant [78] : "pour localiser des entités il faut au préalable les reconnaître, mais pour reconnaître les entités il faut au préalable les localiser". La solution principale pour contourner ce problème est de prendre en compte les informations *a priori* que l'on posséde au sujet des documents traités. Dans le cas des documents contraints, la structure physique est stable et permet de détecter aisément les informations sans reconnaissance. Dans le cas des documents non contraints, la structure physique n'est pas stable. Pour certaines tâches, on peut néanmoins utiliser des informations *a priori* liées à la culture, à la personne qui rédige, etc. : par exemple, la rédaction d'un courrier est régie par des conventions culturelles ; le brouillon d'un manuscrit littéraire rédigé par un unique scripteur aura une structure assez stable (par exemple les brouillons de Flaubert sont structurés de la façon suivante : texte à droite sur les 2/3 de la largeur, une marge avec des annotations à gauche sur 1/3 de la largeur [69]). Pour les documents manuscrits non contraints dont on ne dispose pas de connaissance *a priori*, la structu-

ration doit être menée conjointement avec la reconnaissance.

Même si la phase de structuration permet déjà de localiser un certain nombre d'information (par exemple pour un courrier les coordonnées de l'expéditeur), une phase d'extraction peut être envisagée pour détecter une information très précise telle que le nom de l'expéditeur, son adresse [27] ou son numéro de téléphone [14].

Les documents manuscrits contraints tels que les formulaires [65, 59], les chèques [34, 43, 100] ou encore les enveloppes postales [36, 92] ont bénéficié de nombreuses recherches et les sytèmes proposés sont déjà très performants. La structuration des documents purement manuscrits non contraints a quant à elle été peu étudiée puisque peu d'entre eux se prêtent à une phase de structuration. Les recherches actuelles concernent les documents régis par des règles de structuration implicites.

Les manuscrits de Flaubert ont notamment été étudiés dans le cadre du projet MADONNE [21], et l'extraction de la structure physique est réalisée [69] grâce aux champs de Markov avec différents algorithmes de décodage. Un exemple de résultat obtenu [67] est donné figure 5.5. Le taux moyen de pixels correctement étiquetés obtenu sur cette tâche est de 88.2%.

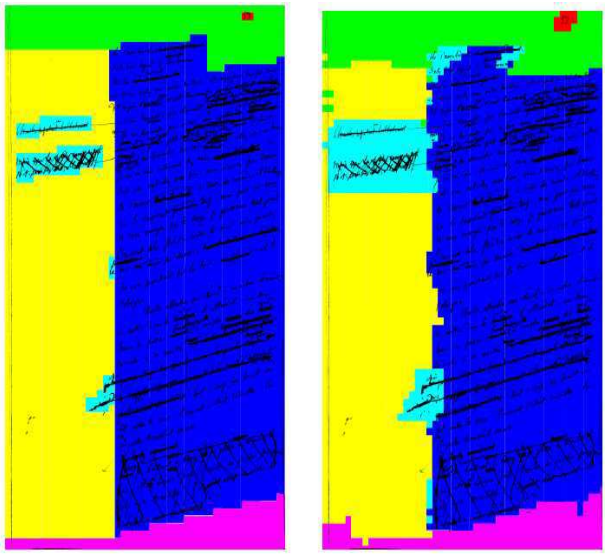


Fig. 5.5. Exemple de structuration obtenue sur les manuscrits de Flaubert [67] avec à gauche la vérité terrain et à droite la structuration obtenue

Le système DMOS (Description and Modification Of Segmentation) d'analyse de documents a été appliqué à l'analyse de courriers manuscrits. Cette approche avait déjà été mise en œuvre dans de nombreuses tâches d'analyse de documents telles que la reconnaissance de partition musicale ou de formules mathématiques [22]. Elle s'appuie sur une grammaire et la description d'un document est effectuée à l'aide du langage EPF (Enhanced Position Formalism). Les règles du système sont déterminées par l'utilisateur au vu des documents contenus dans la base d'apprentissage. Deux types de primitives sont utilisés : les lignes de texte et les composantes connexes.

5.3 Mise en oeuvre de AMBRES pour la structuration de courriers manuscrits

5.3.1 Approche

Dans cette étude, nous nous intéressons au cas des courriers manuscrits. L'objectif de la structuration est d'étiqueter les différents pixels de l'image suivant leur appartenance à un champ (figure 5.6). Nous utilisons dans ce contexte 8 champs différents qui correspondent chacun à un état du champ de Markov : "fond", "coordonnées expéditeur", "coordonnées destinataire", "objet", "ouverture", "signature", "corps de texte" et "date, lieu".

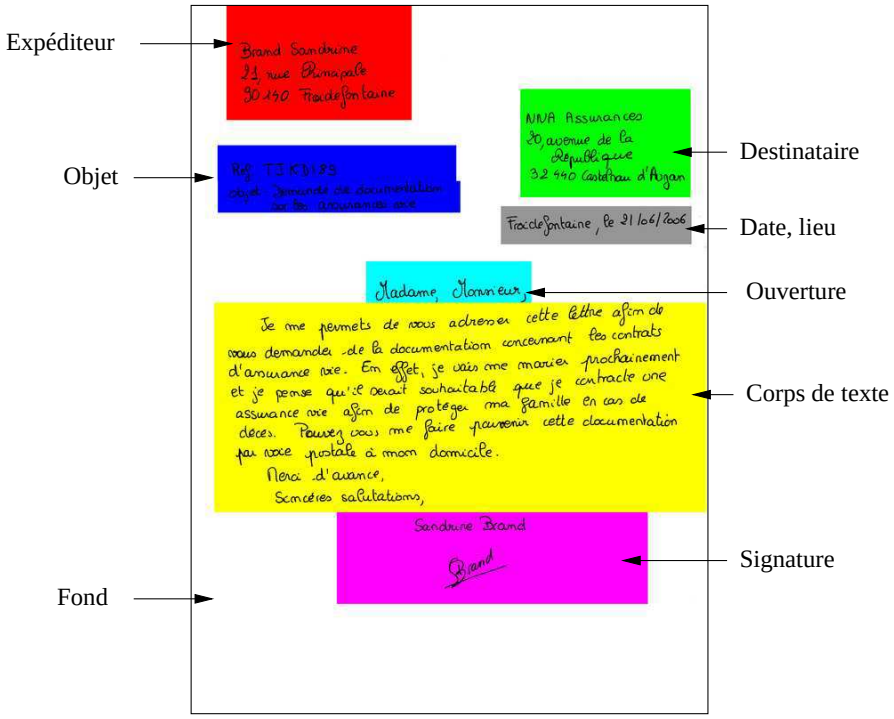


Fig. 5.6. Exemple de structuration de lettre manuscrite

Pour extraire la structure des courriers manuscrits, nous nous proposons dans un premier temps de n'utiliser que l'information de texture (présence de l'écriture) et de position spatiale des différents champs. En effet, l'écriture d'un courrier manuscrit est régie par certaines règles qui peuvent varier plus ou moins suivant le scripteur. Ces règles sont pour la plupart des règles de positionnement : par exemple l'adresse de l'expéditeur est en haut à gauche, la signature en bas de la page. Le modèle de position introduit dans AMBRES permet de tenir compte de la position spatiale des différents champs.

Connaissant les vérités terrain des segmentations en états des images de la base d'apprentissage, l'apprentissage des paramètres va se réduire à une unique itération qui consiste à calculer les paramètres du modèle en fonction des primitives extraites des images et des vérités terrain.

5.3.2 Extraction des primitives

La phase d'extraction de primitives est une étape importante puisqu'elle doit aboutir, dans le cadre de notre étude, à la détection de l'écriture manuscrite. Ici, nous avons retenu la valeur moyenne des niveaux de gris dans une fenêtre glissante. Le pas d'échantillonage détermine le nombre de sites : il a été choisi égal à un tiers de la taille de la fenêtre comme pour la reconnaissance de l'écriture manuscrite.

La variable étudiée pour le choix des primitives est le nombre de sites horizonal N_h, puisque le pas d'échantillonage et la taille de la fenêtre peuvent s'écrire en fonction de celui-ci :

- $pas_{ech} = E\left(\frac{N_h}{n_x}\right)$,
- $taille_{fen} = 3 \times pas_{ech}$,

où n_x est la largeur de l'image et $E()$ la fonction partie entière.

Comme toutes les images traitées sont issues de pages au format $A4$, pour un même N_h, toutes les images auront le même nombre de sites verticaux N_v. De plus, le ratio $\frac{N_v}{N_h}$ est une constante égale à $\sqrt{2}$.

On donne figure 5.7 quelques exemples de primitives extraites d'une image pour différents nombres de sites $N_s = N_h \times N_v$.

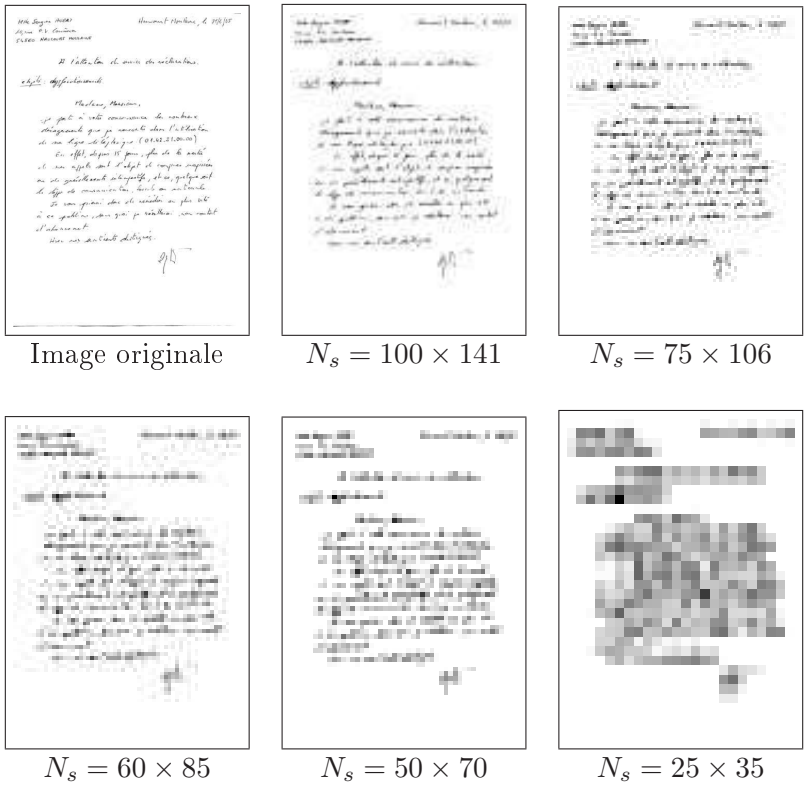


Fig. 5.7. Primitives extraites d'une image de courrier manuscrit pour différents nombre de sites

5.3.3 Définition d'une "zone utile"

Une première observation de l'ensemble des courriers manuscrits montre que la longueur des courriers est très variable (figure 5.8). Cela est dû essentiellement à la différence de taille de l'écriture du scripteur, de l'espacement plus ou moins important entre les lignes de texte et également à la quantité variable de lignes de texte contenues dans la lettre. Ainsi trouve-t-on la signature aussi bien au milieu de la page que tout en bas.

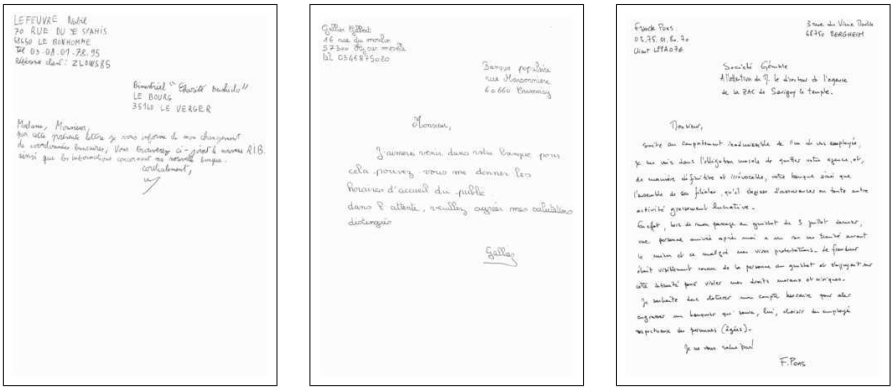


Fig. 5.8. Variabilité de la longueur d'une lettre

Cette observation pose un problème dans le cadre du calcul du modèle de position. Une zone utile est donc introduite afin que la signature (ou s'il n'y en a pas, la fin du corps de texte) soit toujours située en bas de cette zone utile. Cette dernière est une zone rectangulaire obtenue en détourant l'image, elle est définie par les coordonnées $(i_{min}, j_{min}), (i_{max}, j_{max})$ (voir figure 5.9).

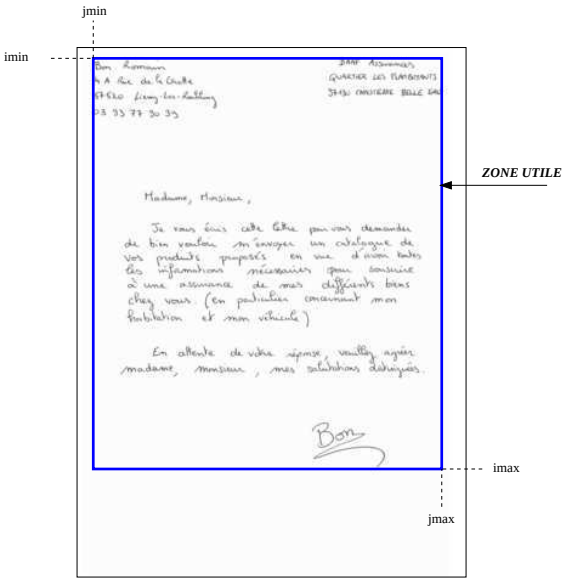


Fig. 5.9. "Zone utile" d'un courrier manuscrit

Le calcul du modèle de position est ainsi réalisé dans chacune des zones utiles des

images de la base d'apprentissage. Lors de la phase de structuration proprement dite, les sites situés à l'extérieur de la zone utile sont étiquetés comme "fond". Les sites situés à l'intérieur de la zone utile sont quant à eux étiquetés par l'intermédiaire de la programmation dynamique $2D$. Dans ce contexte l'utilisation des probabilités de position se fait en normalisant par rapport à la zone utile, c'est-à-dire que l'on calcule $P^P(\omega_{i,j}|i',j')$ où $i' = \frac{i-i_{min}}{i_{max}}$ et $j' = \frac{j-j_{min}}{j_{max}}$ avec $i_{min} \leq i \leq i_{max}$ et $j_{min} \leq j \leq j_{max}$.

5.3.4 Métrique RIMES

La métrique utilisée est celle proposée dans le cadre du projet RIMES (cf. chapitre 6). Elle tient compte de la valeur du niveau de gris des pixels. En effet, les erreurs commises sur des pixels blancs ne sont pas significatives pour cette tâche. La métrique compare les étiquettes de l'hypothèse renvoyée par le système avec les étiquettes de la vérité terrain et compte les pixels mal classés en les pondérant par leur niveau de gris. Le taux d'erreur utilisé est donc T_{err}, avec :

$$T_{err} = \frac{\sum_{pixels\ mal\ classes} (255 - I(i,j))}{\sum_{tous\ les\ pixels} (255 - I(i,j))},$$

où $I(i,j)$ est la valeur du niveau de gris de l'image I aux coordonnées (i,j) $(0 \leq I(i,j) \leq 255)$.

5.4 Base de données

Nous utilisons la base de données créée dans le cadre du projet RIMES. Les images de courriers manuscrits fournies par cette base sont associées à des vérités terrain qui donnent les coordonnées des boîtes englobantes des différents champs par l'intermédiaire d'un fichier XML.

Pour l'étude du choix des paramètres ainsi que des diverses expérimentations, la base de développement et de validation ont été utilisées. La base de développement se compose de 800 courriers manuscrits associés à leurs vérités terrain et la base de validation en compte 100.

5.5 Expérimentations

5.5.1 Étude des performances suivant le nombre de sites

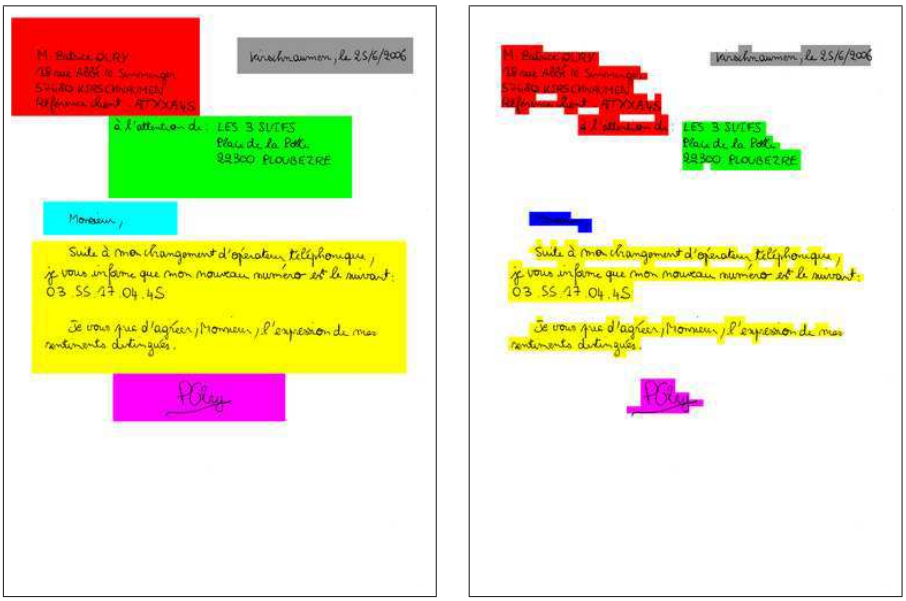
Le tableau 5.1 synthétise les taux d'erreur obtenus en fonction du nombre de sites N_s. La figure 5.7 montre que l'on peut se limiter à une résolution $N_s \leq 100 \times 141$. En effet, elle est suffisamment précise pour une analyse au niveau mot et de plus, une trop grande résolution engendrerait un temps de calcul trop important.

L'analyse du tableau montre que les performances décroissent rapidement avec la résolution à partir de $N_s < 50 \times 70$. Pour N_s compris entre 60×85 et 100×141, les

Tab. 5.1. Taux d'erreur en fonction du nombre de sites

Nombre de sites	100×141	75×106	60×85	50×70	25×35
T_{err}	15.6%	15.5%	15%	16.5%	27.5%

résultats sont assez stables. Un bon compromis entre temps de calcul et performance est de considérer $N_s = 60 \times 85$. Un taux d'erreur égal à **15**% est ainsi atteint. La figure 5.10 montre un exemple de structuration obtenue avec notre algorithme pour une image de la base de test (un taux d'erreur de 6.3% a été obtenu sur cette image).



Vérité terrain — Structuration obtenue avec notre algorithme

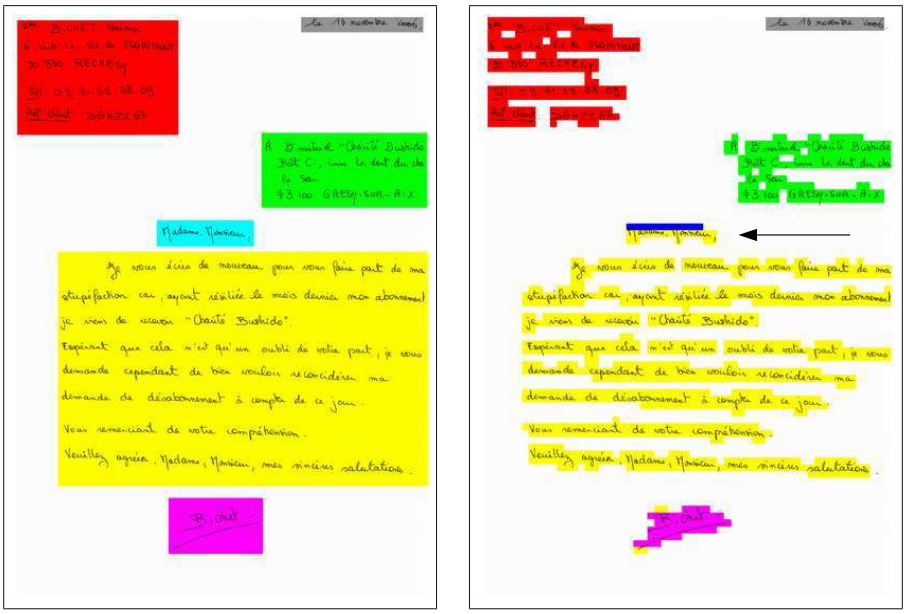
Fig. 5.10. Exemple de structuration obtenue avec $N_s = 60 \times 85$

Pour se faire une idée sur la qualité des performances obtenues avec notre algorithme, on peut tout d'abord comparer le taux d'erreur obtenu avec celui atteint en attribuant à tous les pixels l'étiquette du champ le plus probable c'est-à-dire le champ "corps de texte" : le taux d'erreur obtenu est de 49.8%. De plus on peut également le comparer avec le taux d'erreur obtenu en considérant seulement l'information de position spatiale (et plus l'information de texture), celui-ci est égal à 26%.

5.5.2 Analyse des erreurs

En analysant les erreurs commises par notre système, on peut tout d'abord remarquer que certains champs sont moins bien reconnus que d'autres. Cela est illustré figure 5.10 et figure 5.11. On peut constater que tous les pixels ont correctement été étiquetés exceptés ceux correspondant au champ "ouverture".

Cela est lié à la faible représentation de ce champ au sein de la base de développement comparée à d'autres champs (le corps de texte par exemple est fortement représenté comme on peut le constater figure 5.12).



Vérité terrain — Structuration obtenue avec notre algorithme

Fig. 5.11. Exemple de structuration obtenue montrant que certains champs sont moins bien reconnus que d'autres (ici le champ "Ouverture")

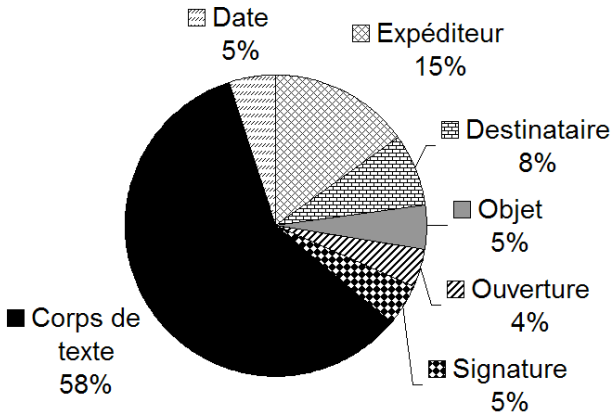
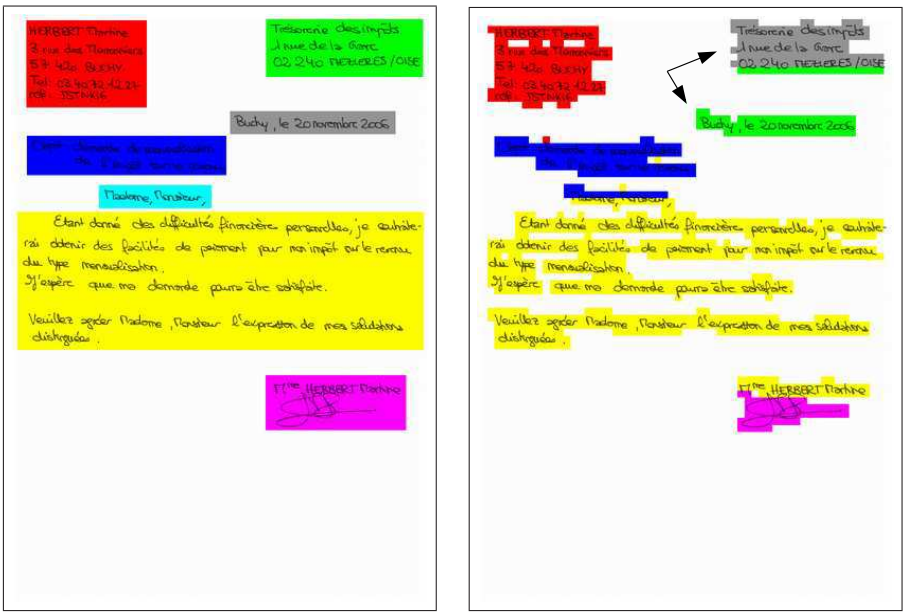


Fig. 5.12. Probabilité d'apparition des différents champs dans la base de développpement RIMES

Pour diminuer l'impact de la faible probabilité d'apparition de certains champs dans la base d'apprentissage, on peut envisager d'ajouter une étape supplémentaire préalable de reconnaissance de mots clés tels que "Objet", "Madame, Monsieur", etc. Cette perspective d'amélioration sera détaillée par la suite.

Ensuite, on constate que de nombreuses erreurs sont commises au niveau de la détection des champs "date, lieu" et "coordonnées destinataire". En effet, ces champs sont spatialement très proches et on trouve soit des confusions entre les deux, soit les deux champs sont étiquetés comme "coordonnées destinataire". Par exemple dans le cas de la figure 5.13, on constate que les deux champs ont été confondus l'un avec l'autre.



Vérité terrain — Structuration obtenue avec notre algorithme

Fig. 5.13. Exemple de structuration obtenue montrant une confusion entre les champs "coordonnées destinataire" et "date, lieu"

Le troisième type d'erreur observé correspond aux cas des figures 5.14 et 5.15. On remarque que différentes étiquettes comme "corps de texte" et "signature" ou "objet" et "ouverture" apparaissent dans une même zone d'écriture. Cela peut s'expliquer d'une part par la prépondérence du modèle de position par rapport aux probabilités de transition (cas de la figure 5.14) et d'autre part par la présence de blocs de pixels blancs correspondant au fond entre deux blocs d'écriture (cas de la figure 5.15).

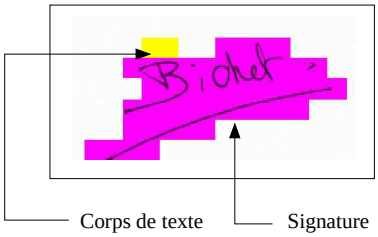


Fig. 5.14. Exemple d'erreurs de segmentation dues à la prépondérance du modèle de position

Enfin, un dernier facteur influant sur le taux d'erreur est lié à la métrique RIMES. En effet, dans notre approche seuls les sites correspondant à de l'écriture sont étiquetés. Or, pour s'adapter au mieux à cette métrique, il faudrait renvoyer des grands rectangles englobant plusieurs lignes d'écriture. En effet, si deux pixels non purement blancs ne sont pas étiquetés et qu'ils le sont dans l'hypothèse, ils seront comptabilisés comme faux.

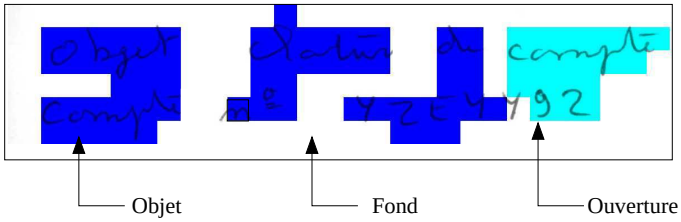


Fig. 5.15. Exemple d'erreurs de segmentation dues à la présence de blocs de pixels blancs

5.6 Augmentation du nombre d'états

Pour tenter de résoudre le problème dû à la présence de blocs de pixels blancs générant différentes étiquettes au sein d'une même zone d'écriture, nous proposons d'introduire de nouveaux états dans notre modèle pour distinguer les pixels blancs appartenant au fond de ceux appartenant à une zone de texte. Trois configurations pour le choix du nombre d'états ont ainsi été étudiées :

- 8 états : 7 états correspondant aux 7 champs et 1 état modélisant le fond,
- 9 états : on ajoute un état pour distinguer les pixels blancs appartenant aux différents champs de ceux appartenant au fond,
- 15 états : par rapport au premier cas, on ajoute 7 états pour distinguer les pixels blancs appartenant à chacun des 7 champs de ceux appartenant au fond.

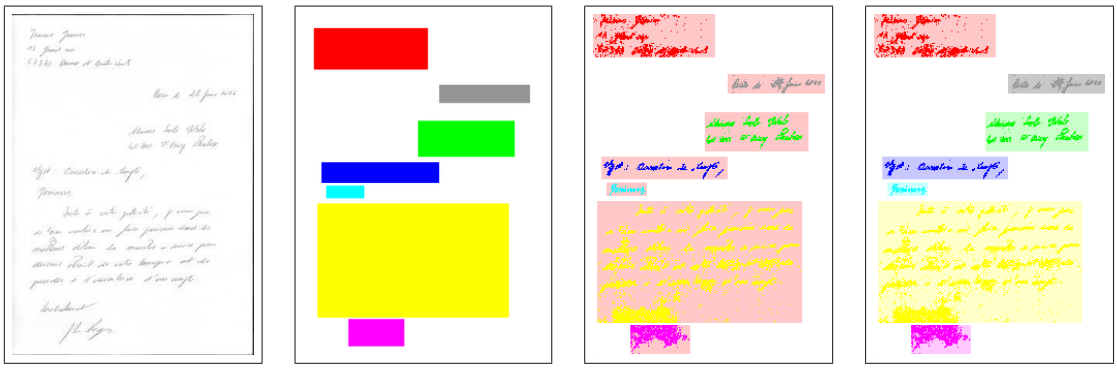


Fig. 5.16. Vérités terrain d'une image pour différents nombres d'états utilisés

Les taux d'erreur obtenus pour chacune de ces 3 configurations sont donnés dans le tableau 5.2. Avec 15 états, on remarque qu'un plus grand nombre de confusions se produit du fait du trop grand nombre d'états. Avec 9 états, la plupart des problèmes de segmentation explicités précédemment sont résolus (figure 5.17), mais les erreurs se propagent plus facilement. Ainsi, le nombre de 8 états semble être le mieux adapté.

Tab. 5.2. Resultats obtenus avec différents nombres d'état

Nombre d'états	8	9	15
T_{err}	15%	17%	27%

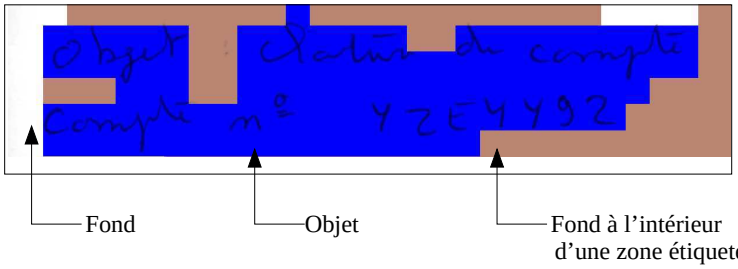


Fig. 5.17. Résolution de l'erreur présentée figure 5.15 avec 9 états

5.7 Phase de post-traitement

Une solution pour résoudre une partie des erreurs commises est de réaliser une phase de post-traitement. Celle-ci va se décomposer en plusieurs étapes, chacune d'elles correspondant à la résolution d'une erreur citée précédemment.

1. La première étape consiste à résoudre le problème de présence de plusieurs étiquettes au sein d'une même zone d'écriture. Pour ce faire, nous détectons les zones connexes correspondant aux sites auxquels on a attribué une étiquette différente de celle du fond. Puis, on détermine l'étiquette la plus courante apparaissant dans chacune de ces zones connexes et on attribue à toute la zone cette étiquette (figure 5.18).

Avant Après

Fig. 5.18. Étape 1 du post-traitement

2. La seconde étape consiste à résoudre le problème d'erreur de détection des champs "date, lieu" et "coordonnées destinataire". On a en effet constaté deux phénomènes à ce sujet : soit les deux champs sont confondus l'un avec l'autre, soit les deux

champs sont étiquetés comme “coordonnées destinataire”. On distingue donc deux cas :

- Si on a une seule zone connexe étiquetée “date, lieu” et une seule zone connexe étiquetée “coordonnées destinataire”, on calcule leur surface. Si la surface de la zone étiquetée “date, lieu” est plus grande que celle étiquetée “coordonnées destinataire”, alors on intervertit les deux étiquettes (figure 5.19).

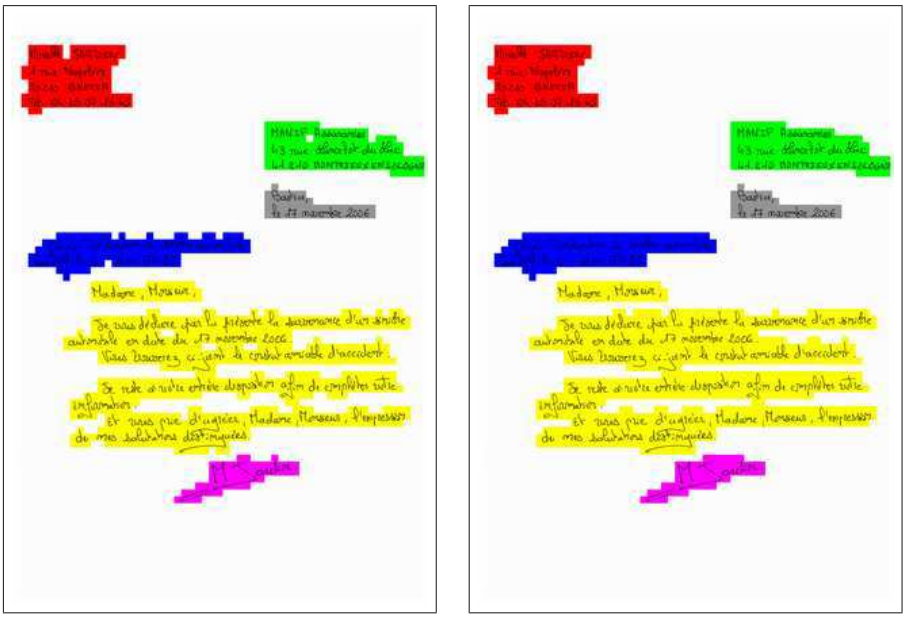
- Si on a exactement deux zones étiquetées “coordonnées destinataire” et pas de zone étiquetée “date, lieu”, alors on attribue à la zone dont la surface est la plus petite l’étiquette “date, lieu”.

Avant Après

Fig. 5.19. Étape 2 du post-traitement

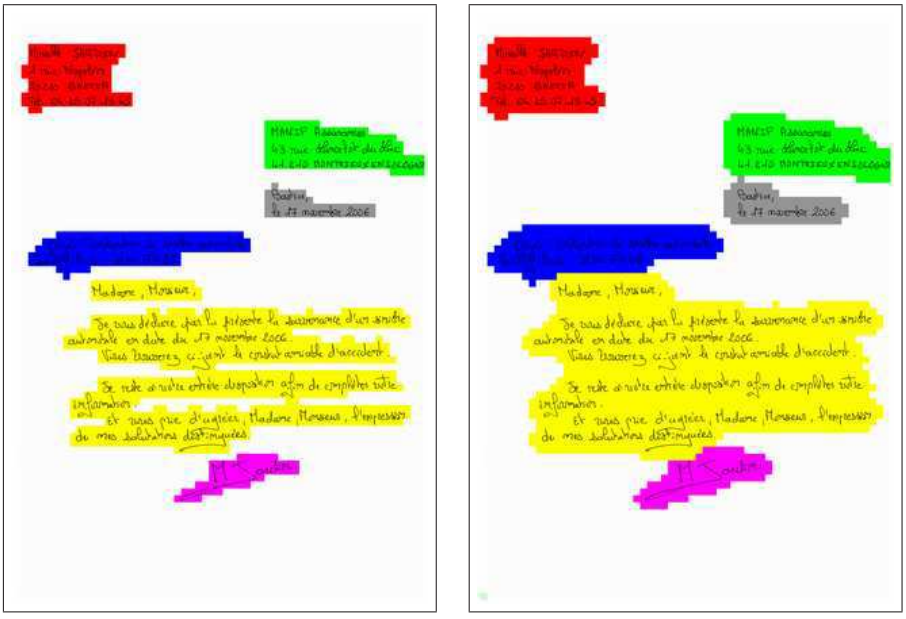
3. Enfin, la dernière étape consiste à s’adapter le mieux possible à la métrique proposée par le projet RIMES. Pour cela, nous procédons en deux temps :
   - On considère un site étiqueté “fond”. Si celui-ci est situé, horizontalement parlant, entre deux sites ayant la même étiquette E (E est différente de “fond”) et à une distance inférieure à un certain seuil de chacun de ces deux sites, alors on lui attribue cette étiquette E. Le seuil est introduit pour ne pas créer d’erreurs en fusionnant deux blocs ayant une même étiquette mais très éloignés l’un de l’autre. L’opération n’est pas réalisée verticalement puisque les champs sont très proches les uns des autres dans cette direction (figure 5.20).

   - Pour chaque site étiqueté “fond” dont le niveau de gris moyen est différent de 255 (qui correspond au blanc) et situé à côté d’un site dont l’étiquette E est différente de “fond”, on lui attribue cette même étiquette E (figure 5.21).



Avant Après

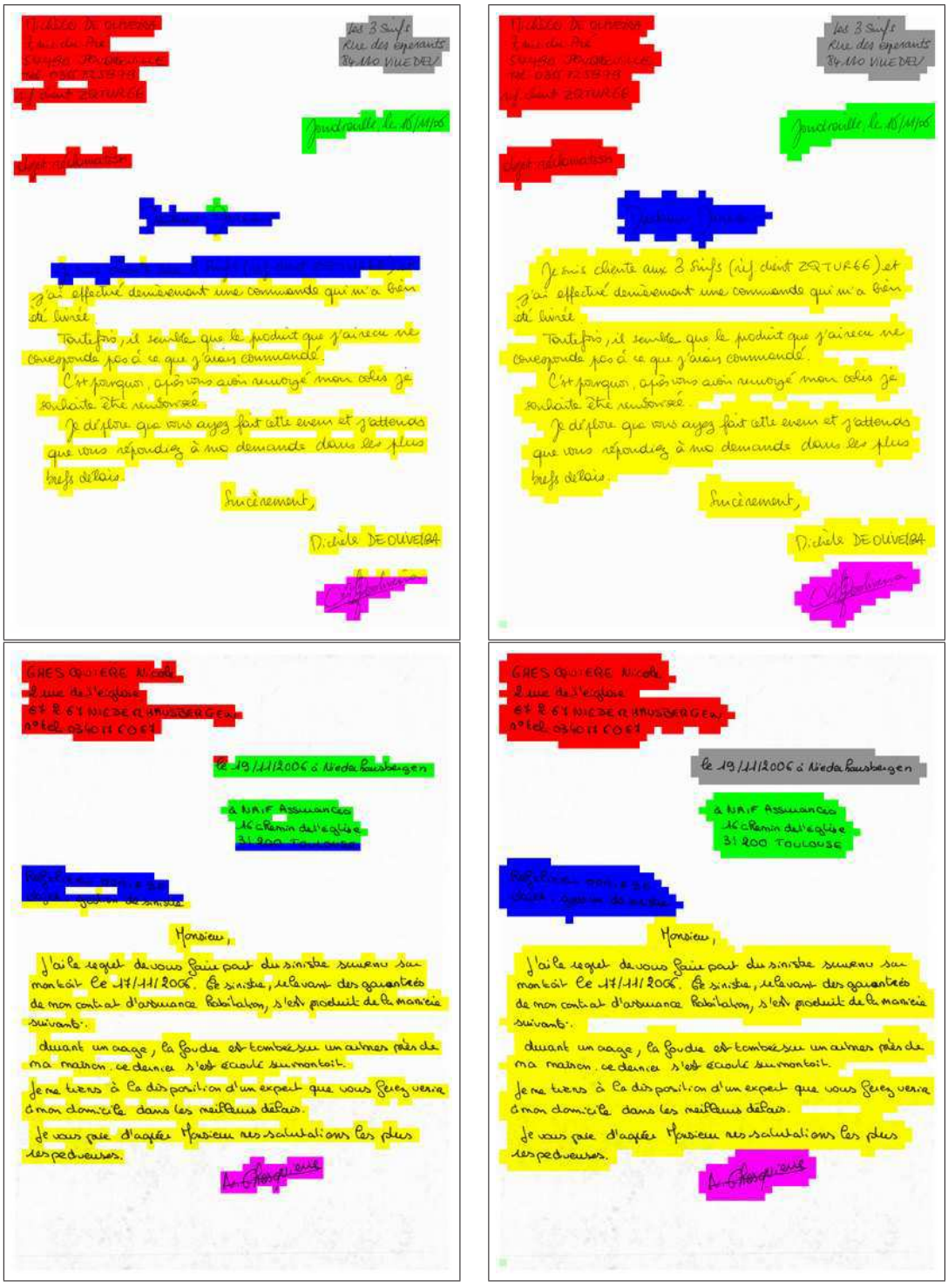
Fig. 5.20. Étape 3-1 du post-traitement



Avant Après

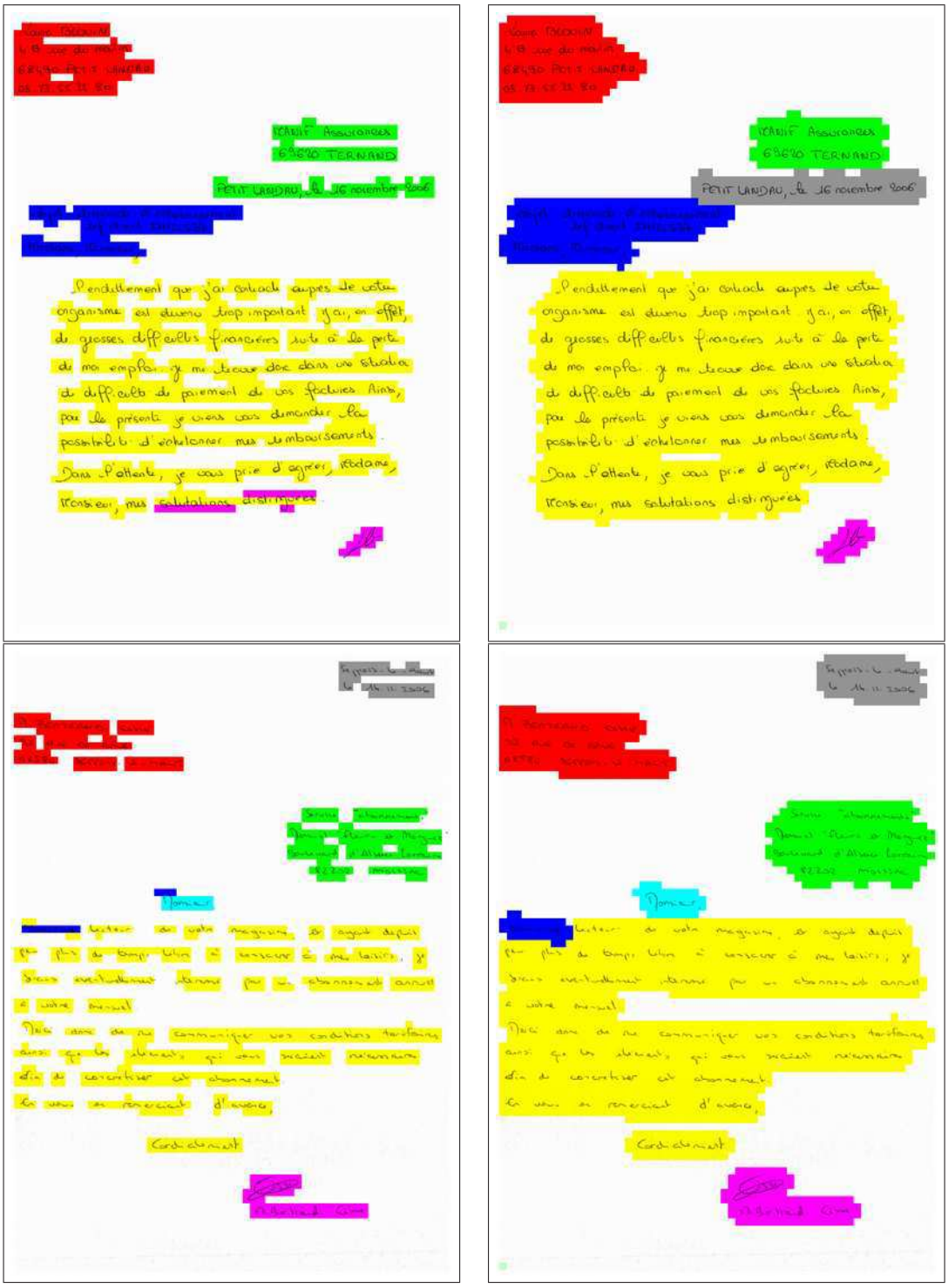
Fig. 5.21. Étape 3-2 du post-traitement

Cette stratégie de post-traitement permet de diminuer de 28% le taux d'erreur, puisque l'on obtient un taux d'erreur de 10.73%. Quelques résultats sont présentés figures 5.22 et 5.23. Le tableau 5.3 consigne les performances obtenues à l'issue de chacune des phases du post-traitement.



Avant Après

Fig. 5.22. Exemples de résultats obtenus après le post-traitement-1



Avant Après

Fig. 5.23. Exemples de résultats obtenus après le post-traitement-2

Tab. 5.3. Performances obtenues à l'issue de chacunes des phases du post-traitement

Étape	Taux d'erreur	Amélioration relative par rapport à l'étape précédente
Initiale	15%	
1	13.5%	10%
2	12.7%	6%
3	10.7%	15.7%

5.8 Proposition d'amélioration

Afin d'améliorer les performances du système de structuration de courriers manuscrits, une perspective serait d'ajouter une étape préalable de reconnaissance de mots clés correspondant chacun à un champ en particulier. Celle-ci permettrait de faire une première passe pour étiqueter les mots clés reconnus et ainsi améliorer la structuration puisque l'incertitude sur la configuration d'un certain nombre de pixels se trouverait diminuée.

Pour mettre en application ce nouveau système décrit figure 5.24, il faut :

- mettre en place un système de segmentation de documents en mots,
- faire un choix sur les mots clés à utiliser,
- mettre en place un système de reconnaissance de mots clés.

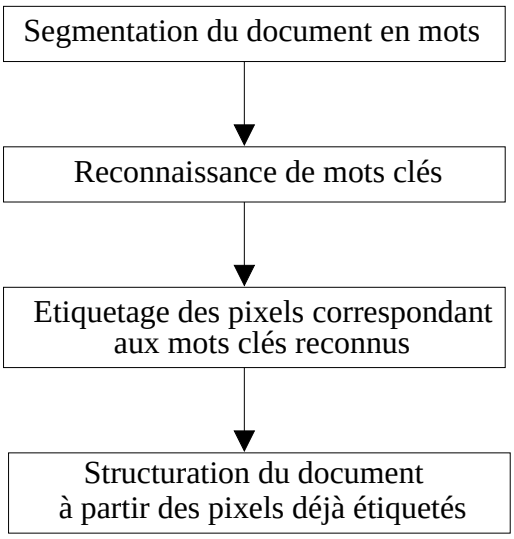


Fig. 5.24. Proposition d'amélioration du système de structuration de courriers manuscrits grâce à une étape supplémentaire de reconnaissance de mots clés

5.9 Conclusion

Nous avons présenté un système complet de structuration de documents manuscrits fondé sur l'approche AMBRES également décrit dans [52]. Le modèle de position défini par AMBRES a permis de tenir compte de la position spatiale des différents champs. Une zone utile a été définie pour gérer toutes les longueurs de lettre. Ainsi, en tenant compte uniquement de l'information de texture (localisation de l'écriture) et de position des différents champs on obtient de bonnes performances sur la base de validation RIMES : un taux d'erreur de 15% a en effet été obtenu.

L'étude des erreurs a conduit à mettre au point une phase de post-traitement. Celle-ci est réalisée en plusieurs étapes dont l'objectif est de résoudre un type d'erreur en particulier. Cela a permis d'améliorer de 28% le taux d'erreur qui est alors de 10.7%. Ce post-traitement pourrait être remplacé à l'avenir par des modifications effectuées directement sur le modèle.

Enfin, en perspective, nous proposons un système introduisant une étape supplémentaire de reconnaissance de mots clés qui laisse entrevoir de meilleures performances.

Notre système de structuration de courriers manuscrits a été évalué dans le cadre de la campagne d'évaluation RIMES : les résultats sont donnés chapitre 6.

Chapitre 6

AMBRES dans la campagne d'évaluation RIMES

6.1 Introduction

Dans nos travaux, nous nous sommes attachée à utiliser des bases de données publiques pour l'évaluation de nos algorithmes. Même si ces bases sont assez classiquement exploitées dans la littérature, la comparaison des résultats n'est pas toujours évidente. En effet, les laboratoires n'utilisent pas tous les mêmes protocoles pour évaluer leurs systèmes. De plus il n'est pas rare de constater une mauvaise utilisation de la base de test qui ne devrait être exploitée que pour l'évaluation finale du système, l'optimisation étant réalisée sur la base de validation. La participation à des campagnes d'évaluation apparaît ainsi nécessaire pour évaluer les algorithmes. Dans ce cadre, les bases de données sont communes et les protocoles d'évaluation identiques pour tous.

Le programme Techno-Vision, initiative des ministères en charge de la recherche et de la défense, a pour but d'augmenter le nombre d'applications opérationnelles des techniques d'analyse d'images par ordinateur grâce à la mise en place de moyens d'évaluation quantitative de leurs performances. Ces moyens comprennent des bases de données d'images annotées, des protocoles et des outils d'évaluation ainsi que des métriques de calcul des performances. Dix projets ont ainsi été mis en place et ont débuté en 2004 pour s'achever pour la plupart en 2007. Parmi ces dix projets, le projet RIMES [4, 75] (Reconnaissance et Indexation de données Manuscrites et de fac similÉS) est dédié à la reconnaissance et à l'indexation de données manuscrites et de fac-similés.

Dans ce chapitre, nous décrivons tout d'abord la campagne d'évaluation RIMES, puis nous présentons les résultats obtenus par AMBRES dans le cadre de l'évaluation. Soulignons que la généralité de l'approche AMBRES a permis de proposer des systèmes pour 5 des 6 tâches évaluées lors de la campagne RIMES couvrant des domaines aussi variés que la reconnaissance de caractères manuscrits, la structuration de courriers ou encore la reconnaissance de logos.

6.2 Le projet RIMES

6.2.1 Objectifs

Le projet RIMES organisé par le Centre d'Expertise Parisien de la DGA, le département ARTEMIS de l'INT et la société A2iA, vise à permettre l'évaluation de systèmes de reconnaissance et d'indexation de documents manuscrits adressés par des individus à des entreprises par courrier ou par fax. Les principaux objectifs de RIMES sont :

- la création d'une grande base de données (collecte SCRIBEO [74]),
- la mise en place de protocoles d'évaluation,
- la conduite de la campagne,
- le dépouillement, l'analyse et la synthèse des résultats,
- la mise à disposition de la base pour la communauté scientifique à l'issue de la campagne.

6.2.2 Tâches envisagées par le projet RIMES

Le projet RIMES couvre l'essentiel des problématiques de l'analyse automatique de document, tant dans le domaine de la reconnaissance de l'écriture manuscrite que dans celui de l'analyse de la structure des documents. Ces domaines sont combinés autour d'une tâche complète et réaliste visant à déterminer les informations essentielles (thème, identité de l'expéditeur, ...) de documents tels que lettres manuscrites, fax et questionnaires. Les courriers considérés sont représentatifs de ceux envoyés par des particuliers à des entreprises et administrations.

Différentes tâches sont ainsi envisagées dans le cadre du projet RIMES :

- l'analyse de la structure du document,
- la reconnaissance du contenu des champs manuscrits, des plus élémentaires (chiffre, mot isolé) aux plus complexes (paragraphe, bloc) et plus généralement la reconnaissance de l'écriture connaissant ou non la structuration du document,
- la reconnaissance des logos,
- l'identification du scripteur,
- l'extraction d'information de haut niveau comme l'identité du destinataire ou l'objet de la lettre.

6.2.3 Base de données

La collecte SCRIBEO des documents manuscrits constituant la base de données a été réalisée auprès de scripteurs volontaires rémunérés par chèque cadeaux via Internet [74]. Chaque scripteur volontaire rédige 5 courriers comportant chacun une lettre manuscrite, une page de garde de fax (optionnelle) et un formulaire structuré (figure 6.1). Le scripteur écrit ces documents d'après une identité et un scénario fictifs qui lui ont été attribués. La numérisation est effectuée par un scanner professionnel (300dpi) en niveaux de gris, mais il est également demandé au scripteur, dans la mesure du possible, d'effectuer une numérisation avec son scanner personnel et/ou avec un fax, cela afin d'étudier l'impact de la qualité de la numérisation sur les performances des différents

systèmes.

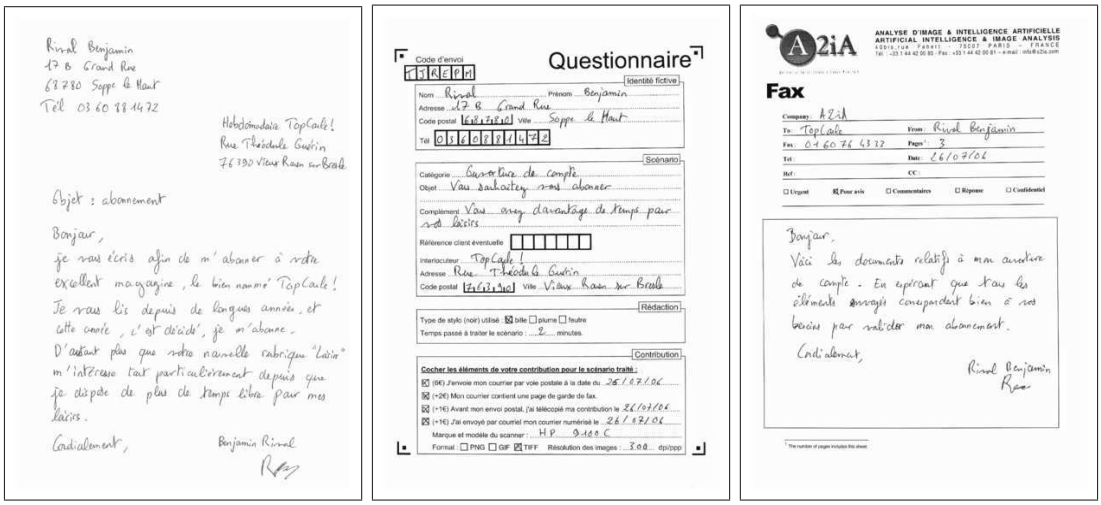


Fig. 6.1. Composition d'un courrier SCRIBEO : lettre + questionnaire + fax (facultatif)

Les différents scénarii réalistes proposés aux scripteurs concernent les sujets suivants :

- changement de données personnelles (par exemple l'adresse),
- demande d'information,
- difficulté de paiement,
- ouverture (de compte par exemple),
- fermeture,
- modification de contrat ou de commande,
- réclamation,
- relance de courrier sans réponse,
- sinistre.

Les caractéristiques principales de la base de documents ainsi récoltés sont les suivants :

- 12600 pages, soit 5600 courriers,
- 44% de ces pages sont des lettres,
- 44% de ces pages sont des questionnaires,
- 12% de ces pages sont des pages de garde de fax,
- il y a 1300 scripteurs différents dont 1000 ont envoyé 5 courriers.

De plus les caractéristiques concernant les scripteurs sont les suivantes :

- 54% des scripteurs sont des étudiants, lycéens et inactifs,
- 71% des scripteurs sont âgés entre 19 et 30 ans,
- 76% des scripteurs ont un niveau d'étude supérieur au bac,
- 11% des scripteurs sont gauchers,
- 4% des scripteurs ont appris à écrire via une autre langue que le français,

- 58% des scripteurs sont des femmes et 42% sont des hommes,
- 34% des scripteurs habitent en Ile de France.

Chaque image de la base de données est associée à une vérité terrain. Celle-ci est donnée par un fichier XML (figure 6.2) pouvant être créé à partir du logiciel d'annotation PARADI fourni par le projet RIMES (figure 6.3).

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE annotation SYSTEM "annotation.dtd">
<annotation date="14/11/2005" author="rimes">

  <source width="2444" height="3457">00786_L.png</source>
  <document>Lettre manuscrite</document>
  <scenario>Modification de contrat/Commande</scenario>
  <writer>NIQUET Jean-Pierre</writer>
  <sender_person>Jean-Pierre Niquet</sender_person>
  <sender_company></sender_company>
  <sender_tel>+33(0)3.56.26.57.49</sender_tel>
  <sender_fax></sender_fax>
  <receiver_person>¤{M. Mabur/Monsieur Mabur}¤
  </receiver_person>
  <receiver_company>Débit Lyonnais</receiver_company>
  <subject></subject>
  <date>28/10/05</date>
  <place>Vermo¤{n/u}¤dans</place>

  <box top_left_x="0" top_left_y="24" bottom_right_x="788"
  bottom_right_y="548">
    <type>Coordonnées expéditeur</type>
    <text>Jean-Pierre Niquet\n1 rue du Chemin Noir\n25150
    Vermo¤{n/u}¤dans\n03.56.26.57.49</text>
  </box>

  <box top_left_x="1164" top_left_y="591"
  bottom_right_x="2052" bottom_right_y="767">
    <type>Date, Lieu</type>
    <text>Vermo¤{n/u}¤dans, le 28/10/05</text>
  </box>
  ...


</annotation>
```

Fig. 6.2. Fichier XML

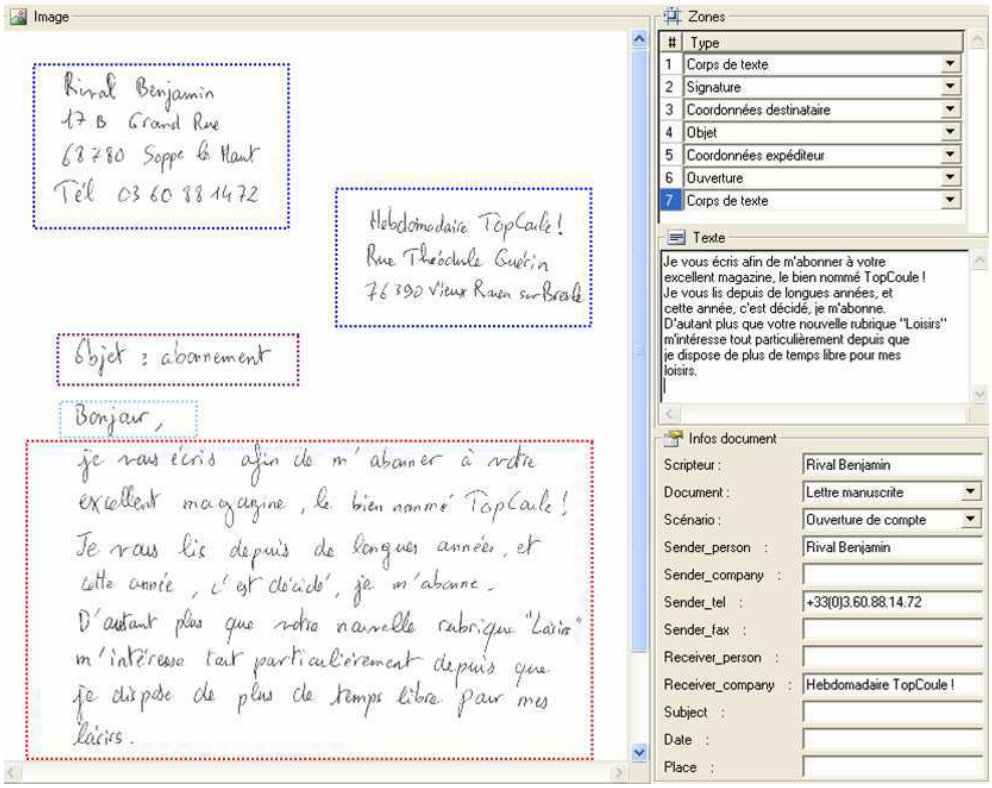


Fig. 6.3. Outil d'annotation "Paradi" proposé par le projet RIMES

6.2.4 Première phase d'évaluation : tâches évaluées

Dans le cadre de la première évaluation de la campagne RIMES qui s'est déroulée du 4 au 18 juin 2007, 4 thèmes comportant 6 tâches ont été évalués : la reconnaissance de caractères isolés, la structuration de courriers, la reconnaissance de logos avec rejet et l'identification de scripteur avec rejet. Grâce à la généralité de notre approche, nous avons pu participer aux 3 premiers thèmes correspondant à 5 tâches. Pour chacune des tâches, les données disponibles (bases de développement, validation et test) ainsi que les métriques utilisées sont décrites dans la suite.

6.2.4.1 Tâche de reconnaissance de caractères isolés

Elle se décompose en trois sous-tâches : la reconnaissance de chiffres, la reconnaissance de lettres et la reconnaissance de caractères alphanumériques. Les imagettes utilisées pour ces trois tâches ont été extraites automatiquement des peignes des questionnaires par la société A2iA l'un des organisateurs du projet RIMES.

La répartition des imagettes en développement, validation et test est donnée dans le tableau 6.1.

Tab. 6.1. Répartition des imagettes de caractères en développement, validation et test

	Chiffres	Lettres	Caractères alphanumériques
Développement	4773	1502	6275
Validation	2881	1182	4063
Test	5937	2098	8035

La métrique utilisée est celle classique du calcul du taux d'erreur $T_{err_{RC}}$:

$$T_{err_{RC}} = \frac{\text{Nombre d'images mal reconnues}}{\text{Nombre total d'images}}.$$

Remarque : Dans le cadre de la reconnaissance de caractères alphanumériques une confusion entre un "O" et un "0" n'est pas comptée comme une erreur.

Fig. 6.4. Exemple d'imagettes de caractères de la base RIMES

6.2.4.2 Tâche de structuration de courriers manuscrits

Il s'agit de détecter les champs suivants : "coordonnées expéditeur", "coordonnées destinataire", "objet", "ouverture", "signature", "corps de texte", "date, lieu", "PS / PJ".

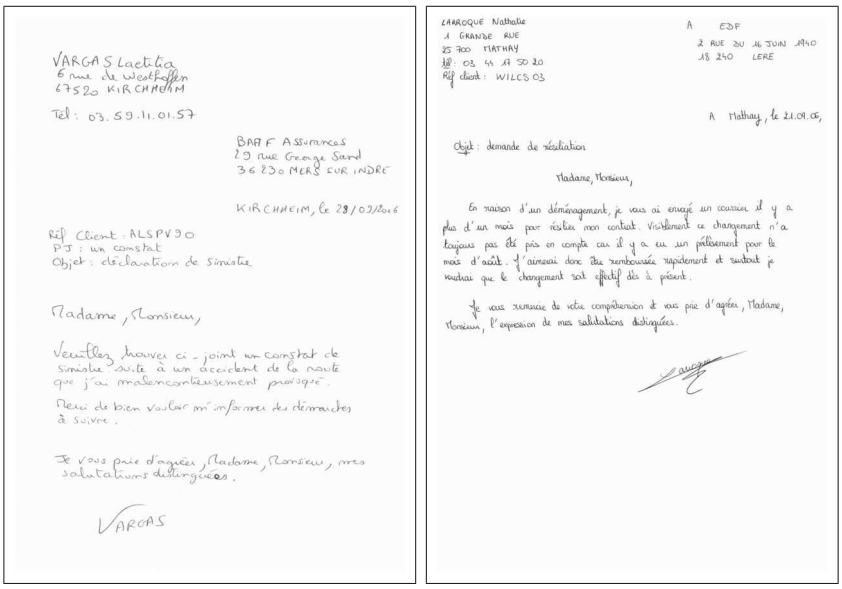


Fig. 6.5. Exemple d'images de lettres de la base RIMES

La répartition des imagettes disponibles en développement, validation et test est donnée dans le tableau 6.2.

Tab. 6.2. Répartition des images de courriers en développement, validation et test

Développement	Validation	Test
950	100	100

La métrique mise en place dans le cadre du projet RIMES tient compte de la valeur des niveaux de gris des pixels mal classés. En effet, les erreurs commises sur des pixels blancs ne sont pas significatives pour cette tâche. Elle compare les champs de chaque pixel de la solution renvoyée par le système à ceux de la vérité terrain et compte les pixels mal classés en les pondérant par leur niveau de gris. En cas de recouvrement de boîtes englobantes, les pixels correspondant sont considérés comme mal classés. La métrique s'exprime donc par :

$$T_{err_S} = \frac{\sum_{pixels\ mal\ classes} (255 - I(i,j))}{\sum_{tous\ les\ pixels} (255 - I(i,j))},$$

où $I(i,j)$ est la valeur du niveau de gris de l'image I aux coordonnées (i,j).

6.2.4.3 Tâche de reconnaissance de logos avec rejet

Une autre tâche évaluée est la reconnaissance de logos avec rejet, c'est-à-dire que tous les logos de la base de test ne sont pas représentés dans la base d'apprentissage.

Fig. 6.6. Exemples d'images de logos de la base RIMES

La répartition des images de logos en développement, validation et test est donnée dans le tableau 6.3.

Tab. 6.3. Répartition des images de logos en développement, validation et test

Développement	Validation	Test
236	100	100

La métrique utilisée est celle classique du calcul du taux d'erreur $T_{err_{RL}}$:

$$T_{err_{RL}} = \frac{\text{Nombre d'images mal reconnues}}{\text{Nombre total d'images}}.$$

6.3 Résultats de AMBRES dans la première campagne d'évaluation RIMES

Les résultats obtenus par AMBRES sont décrits ci-dessous.

Pour la participation à cette première phase d'évaluation, 5 laboratoires étaient représentés : le département TSI de l'ENST, l'équipe IMADOC de l'IRISA, le LITIS de l'Université de Rouen, l'équipe SIP du CRIP5 de l'Université Paris 5 et le Centre d'Expertise Parisien de la DGA. De par le caractère expérimental de cette première campagne, les résultats obtenus par les différents participants sont anonymes.

6.3.1 Tâche de reconnaissance de caractères isolés

Pour chacune des trois tâches de reconnaissance de caractères isolés, l'approche utilisée est identique à celle décrite dans le troisième chapitre (cf. page 76). Une étape préalable de prétraitement est réalisée afin de redimensionner les images à une taille

28×28 qui est la taille des imagettes de la base MNIST sur lesquelles les paramètres de nos algorithmes ont été optimisés. Ce redimensionnement est plus particulièrement important pour le calcul des primitives afin de conserver une fenêtre d'analyse de taille 7×7. Cette phase de prétraitement a été décrite dans le troisième chapitre dans le cadre de la reconnaissance des lettres isolées.

6.3.1.1 Tâche de reconnaissance de chiffres

La base d'apprentissage de la base RIMES ne contient que 7654 images de chiffres manuscrits isolés. Or, on dispose de la base publique MNIST qu'il serait intéressant d'exploiter du fait du grand nombre d'échantillons disponibles pour chaque chiffre.

Un point bloquant est que la base MNIST est constituée de chiffres écrits par des scripteurs américains alors que la base RIMES a été constituée par des scripteurs français. En effet, les Américains écrivent certains chiffres différemment des Français : par exemple le "7" n'est pas barré. Aussi tous les modèles obtenus sur la base MNIST ne sont pas directement exploitables dans le cas de chiffres écrits par des Français.

Nous proposons donc d'utiliser à la fois les 10 modèles appris sur la base MNIST et les 10 modèles appris sur la base RIMES : on teste donc 20 modèles. Ainsi pour des chiffres écrits de la même façon par un Américain et un Français (par exemple le "0") les modèles appris sur MNIST seront plus précis et dans le cas contraire les modèles appris sur RIMES seront plus pertinents puisqu'appris sur des chiffres "français".

Pour montrer l'efficacité de cette méthode trois systèmes ont été soumis à l'évaluation RIMES, chacun correspondant à une base d'apprentissage différente :

Système 1 20 modèles testés : 10 modèles appris sur MNIST et 10 modèles appris sur RIMES,

Système 2 10 modèles testés : 10 modèles appris sur RIMES,

Système 3 10 modèles testés : 10 modèles appris sur MNIST.

Les résultats obtenus pour les trois systèmes à l'issue de l'évaluation et renvoyés par les organisateurs du projet RIMES sont donnés tableau 6.4. Nous vérifions bien que l'utilisation seule de la base MNIST n'est pas judicieuse puisqu'un taux d'erreur très élevé est obtenu (13.67%). En n'utilisant que la base RIMES le taux d'erreur est de 2.71%. Si l'on adopte la stratégie explicitée précédemment consistant à tester les modèles appris sur RIMES et les modèles appris sur MNIST, ce taux d'erreur baisse de 16%.

Nous obtenons finalement un taux d'erreur égal à 2.26% ce qui est cohérent avec les résultats obtenus sur la base MNIST, puisqu'un taux d'erreur de 2.32% avait été atteint sur la base de test.

Les résultats des participants à cette tâche se situent entre 2.26% et 2.37%.

Tab. 6.4. Taux d'erreur obtenus par les 3 systèmes soumis pour la tâche de reconnaissance de chiffres

Syst. 1	Syst. 2	Syst. 3
2.26%	2.71%	13.67%

6.3.1.2 Tâche de reconnaissance de lettres

Pour l'apprentissage des modèles de lettres manuscrites isolées, on ne dispose que de 2684 échantillons ce qui est très peu pour avoir des modèles précis de chacune des 26 classes. Tout comme pour la reconnaissance de chiffres on exploite donc une autre base : la base ENST-FAX-CHAR de lettres majuscules extraites d'images de pages de fax pour laquelle on dispose de plus de 5800 échantillons.

On procéde de la même façon que pour la reconnaissance de chiffres : on apprend indépendamment les modèles sur la base RIMES et sur la base ENST-FAX-CHAR. Cette stratégie est adoptée car si on mélangeait les images des deux bases, les images de la base RIMES se trouveraient largement minoritaires et influeraient donc peu sur les modèles finaux.

Pour montrer l'efficacité de cette méthode trois systèmes ont ainsi été soumis à l'évaluation RIMES, ces trois systèmes correspondant à différentes bases d'apprentissage :

Système 1 52 modèles testés : 26 modèles appris sur RIMES et 26 modèles appris sur ENST-FAX-CHAR,

Système 2 26 modèles testés : 26 modèles appris sur ENST-FAX-CHAR,

Système 3 26 modèles testés : 26 modèles appris sur RIMES.

Les résultats obtenus pour les trois systèmes à l'issue de l'évaluation et renvoyés par les organisateurs du projet RIMES sont donnés tableau 6.5. On peut constater dans un premier temps que les résultats obtenus avec la base RIMES seule sont bien moins bons que ceux obtenus avec la base ENST-FAX-CHAR seule. Cela s'explique par la précision des modèles qui dépend de la quantité de données d'apprentissage. Finalement, la stratégie adoptée consistant à exploiter à la fois la base RIMES et la base ENST-FAX-CHAR s'avère concluante : un taux d'erreur de 5% est obtenu contre 11.96% avec la base RIMES seule et 7.39% avec la base ENST-FAX-CHAR seule.

Les résultats obtenus par les participants à cette tâche se situent entre 5% et 6.39%.

Tab. 6.5. Taux d'erreur obtenus par les 3 systèmes soumis pour la tâche de reconnaissance de lettres

Syst. 1	Syst. 2	Syst. 3
5%	7.39%	11.96%

6.3.1.3 Tâche de reconnaissance de caractères alphanumériques

Pour la reconnaissance de caractères alphanumériques, on exploite ainsi les bases RIMES, MNIST et ENST-FAX-CHAR. On soumet deux systèmes différents à l'évaluation RIMES :

Système 1 72 modèles testés : 36 modèles appris sur RIMES, 26 modèles appris sur ENST-FAX CHAR et 10 modèles appris sur MNIST,

Système 2 36 modèles testés : 36 modèles appris sur RIMES.

Les résultats obtenus pour les 2 systèmes à l'issue de l'évaluation et renvoyés par les organisateurs du projet RIMES sont donnés tableau 6.6. On constate qu'avec la base

RIMES seule on obtient un taux d'erreur de 8.69%. En exploitant d'autres bases de données on diminue de 18% ce taux d'erreur puisqu'on atteint 7.12%.

Les résultats obtenus par les participants à cette tâche se situent entre 6.97% et 7.27%.

Tab. 6.6. Taux d'erreur obtenus par les 2 systèmes soumis pour la tâche de reconnaissance de caractères alphanumériques

Syst. 1	Syst. 2
7.12%	8.69%

6.3.2 Tâche de structuration de courriers manuscrits

Pour la tâche de stucturation de courriers manuscrits, l'approche utilisée est identique à celle décrite dans le chapitre 5. Deux systèmes ont été soumis afin de confirmer l'efficacité du post-traitement :

Système 1 après le post-traitement,

Système 2 avant le post-traitement.

Les résultats obtenus pour chacun de ces deux systèmes sont donnés dans le tableau 6.7. On observe que le post-traitement permet bien d'améliorer de 30% le taux d'erreur comme on l'avait constaté sur la base de validation. On obtient au final un taux d'erreur de 11.26%.

L'autre participant à cette tâche obtient un taux d'erreur de 9.13%. La comparaison des résultats montre que les erreurs des deux systèmes ne sont pas corrélées. De plus si le taux d'erreur du second participant est plus faible, on constate un taux d'erreur très élevé sur un certain nombre de lettres. Notre système semble obtenir des performances plus stables sur l'ensemble des courriers.

Tab. 6.7. Taux d'erreur obtenus par les 2 systèmes soumis pour la tâche de structuration de courriers manuscrits avec la première métrique

Syst. 1	Syst. 2
11.26%	16.18%

6.3.3 Tâche de reconnaissance de logos avec rejet

Nous avons montré dans les précédents chapitres que l'approche AMBRES est une approche générale. En effet, elle a été appliquée à la reconnaissance de caractères isolés, à la reconnaissance de mots ainsi qu'à la structuration de courriers manuscrits. C'est donc naturellement que nous l'avons appliquée pour la reconnaissance de logos dans le contexte de la campagne d'évaluation RIMES. La méthode adoptée est la même que celle mise au point pour la reconnaissance de caractères manuscrits : algorithme d'apprentissage itératif de type EM et vecteur de primitives spectrales locales. Concernant le nombre d'états et la topologie on a choisi une grille 5×5. Ce choix s'est effectué de façon intuitive et n'a pas été optimisé du fait du court délai entre l'acquisition des images de logos et la date de l'évaluation.

Une phase de prétraitement est réalisée afin de redimensionner les images à la taille 50×50 et ainsi de limiter le temps de calcul.

Du fait de l'assez grand nombre de modèles de logos à tester sur chaque image de la base de test (51 modèles de logos différents présents dans la base d'apprentissage), une phase de réduction dynamique de la taille du lexique est utilisée. Cela est réalisé classiquement à partir de caractéristiques globales extraites de l'image : hauteur et largeur de la forme (déterminée après le prétraitement afin de s'affranchir des changements d'échelle).

Sur les 100 images de la base de test, 3 images ont été mal reconnues. On indique dans la figure 6.7 les erreurs commises et les confusions réalisées. La première erreur est commise sur le logo nommé "Veolia" : l'erreur est due au fait que dans la base d'apprentissage le logo ne contenait pas l'inscription *"AGENCE DU LUC EN PCE"*. La seconde erreur est commise sur le logo "Christian-BERTHOU" qui n'est pas connu dans la base d'apprentissage : l'erreur est due à une confusion avec le logo "BERTHOU-Christian" qui est le même logo mais avec une interversion des mots "BERTHOU" et "Christian". Enfin la troisième erreur est commise sur le logo "Loria" qui a une très faible représentation dans la base d'apprentissage (2 exemplaires).

Logo reconnu : inconnu Logo réel : Veolia	Logo reconnu : BERTHOU-Christian Logo réel : Inconnu	Logo reconnu : inconnu Logo réel : Loria

Fig. 6.7. Erreurs commises par AMBRES pour la reconnaissance de logos

L'autre participant à cette tâche obtient un taux d'erreur égal à 1%. Nous pouvons noter que les résultats obtenus sur la base de validation par notre système conduisent également à ce même taux d'erreur.

Du fait de la petite taille de la base de logos, cette tâche n'est pas encore très significative, mais le deviendra dans les prochaines campagnes d'évaluation RIMES. Elle a permis néanmoins de montrer que l'approche AMBRES est une approche générale, elle peut s'appliquer assez rapidement à d'autres tâches que celles décrites en détail dans cette thèse et ce avec de bonnes performances.

6.4 Conclusion

La participation au projet RIMES a permis d'évaluer AMBRES sur une nouvelle base de données avec des protocoles d'évaluation bien définis. La généralité de AMBRES ressort directement puisque c'est la seule approche proposée dans le cadre de cette évaluation qui ait permis de participer à 5 des 6 tâches évaluées couvrant des domaines

aussi variés que la reconnaissance de l'écriture manuscrite, la structuration de documents manuscrits ou la reconnaissance de logos.

La participation à la tâche de reconnaissance de caractères manuscrits conduit aux observations suivantes :

- les résultats obtenus sur la base RIMES et sur la base MNIST sont cohérents puisque les taux d'erreur sont respectivement de 2.26% et de 2.32%,
- l'exploitation de plusieurs bases de données permet d'améliorer considérablement les résultats.

Enfin, la participation de AMBRES à la reconnaissance de logos confirme la généralité de AMBRES et laisse penser que d'autres tâches pourraient encore être traitées par notre approche (reconnaissance de scripteurs, ...).

Conclusion générale et perspectives

Dans le cadre de l'analyse de documents manuscrits, nous avons présenté une approche générale bidimensionnelle markovienne appelée AMBRES. Elle a été appliquée avec succès à des tâches aussi diverses et variées que la reconnaissance de l'écriture manuscrite, la structuration de documents manuscrits ou encore la reconnaissance de logos, et a été évaluée lors de la campagne d'évaluation RIMES. AMBRES est fondée sur les champs de Markov, la programmation dynamique 2D et une analyse fenêtrée de l'image. De plus, un modèle de position est introduit pour relier la configuration d'un site à sa configuration spatiale.

Après avoir détaillé l'approche AMBRES nous avons abordé dans un premier temps la reconnaissance de caractères manuscrits. Les performances obtenues avant la thèse ont été significativement améliorées. En effet, les diverses optimisations et améliorations ont permis de diminuer le taux d'erreur de 35% sur la base de validation de MNIST et passe ainsi de 2.34% à 1.52%.

AMBRES fait intervenir des paramètres liés à la topologie du champ de Markov, au terme d'attache aux données, aux termes de position et de transition, à la programmation dynamique 2D et à l'extraction des primitives. Un gros effort a donc été conduit pour l'étude et l'optimisation de chacun de ces paramètres. En particulier nous avons prêté une attention particulière aux primitives, dont la pertinence est tout aussi importante que celle du modèle. Vue comme un ensemble de traits de différentes positions et directions, l'écriture manuscrite est modélisée par une grille d'états cachés chacun représentant une certaine position et direction de traits. L'information de direction est extraite à partir des primitives spectrales locales. L'étude approfondie des paramètres liés à leur extraction était nécessaire et a ainsi permis d'obtenir un taux d'erreur égal à 2.04%.

De par la grande complexité du modèle étudié, une étude des performances du système après optimisation des paramètres était incontournable. Nous avons donc adopté une démarche consistant dans un premier temps à étudier les erreurs, puis à analyser les modèles et enfin à suggérer des pistes d'amélioration. Ainsi dans le cadre de la reconnaissance de chiffres, nous avons proposé une approche originale de combinaison de systèmes utilisant différents vecteurs de primitives spectrales locales. En utilisant la méthode Borda Count, on atteint un taux d'erreur de 1.52% qui est proche des meilleures performances obtenues dans l'état de l'art. Ensuite, une stratégie de division et fusion d'états a été introduite afin d'affiner la segmentation en états de certaines classes de chiffre. Si elle ne donne pas encore tout à fait les résultats escomptés, cette méthode

apparaît comme une perspective intéressante à explorer à l'avenir.

Enfin, nous avons montré que le système de reconnaissance de caractère présenté et optimisé sur la base MNIST pouvait être appliqué à n'importe quelle base de caractères. Pour cela une étape préalable de prétraitement est introduite afin de dimensionner les images à la même taille que les images de la base MNIST. En effet certains paramètres tels que la taille de la fenêtre d'extraction des primitives sont liés à la taille des images.

L'approche a été ensuite étendue à la reconnaissance de mots manuscrits avec un grand vocabulaire. La méthode proposée consiste à construire des modèles de mots par concaténation de modèles de lettres appris sur des imagettes automatiquement extraites des images de mots.

La concaténation des modèles de lettres est rendue possible par l'introduction d'un nouveau mode de segmentation des lettres en états. Celui-ci varie selon si la lettre contient une hampe, un jambage ou s'il s'agit d'une majuscule ou d'une minuscule. En effet la concaténation doit permettre d'aligner correctement les parties centrales de chaque lettre. La stratégie de concaténation décrite dans nos travaux a été validée sur une tâche de reconnaissance de nombres obtenus en collant des images de chiffres de la base MNIST : le taux d'erreur obtenu au niveau chiffres est le même que celui obtenu sur la base MNIST.

Une difficulté de l'approche concerne l'étape de découpage automatique des images de mots en images de lettres. Nécessaire pour avoir des modèles de lettres précis, cette étape souffre de quelques lacunes. Une manière de contourner le problème serait d'utiliser une grande base de lettres isolées.

Les résultats obtenus sur la base publique Senior et Robinson donnent un taux de reconnaissance de 73.7% et n'atteignent pas encore les meilleures performances à l'état de l'art. Néanmoins des perspectives d'amélioration sont possibles comme par exemple l'introduction de modèles de lettre en contexte.

La généralisation au niveau document a été abordée dans le cadre de la structuration de courriers manuscrits visant à détecter des champs utiles à leur traitement automatique tels que les coordonnées de l'expéditeur, les coordonnées du destinataire ou l'objet de la lettre. La méthode proposée exploite l'information de texture, c'est-à-dire la présence ou non de l'écriture, ainsi que l'information de position spatiale des différents champs. Cette dernière est en effet pertinente dans le cas des courriers manuscrits dont la rédaction est régie par des conventions culturelles qui peuvent varier plus ou moins d'un scripteur à l'autre. Ces conventions sont pour la plupart des règles de positionnement : par exemple l'adresse de l'expéditeur est en haut à gauche, la signature en bas de la page.

La méthode mise en œuvre permet déjà à ce stade d'obtenir de bonnes performances. Une analyse approfondie des performances est réalisée et permet de faire ressortir cinq principaux types d'erreurs. La modification des modèles par l'ajout d'états supplémentaires est étudiée mais ne donne pas d'amélioration. Une phase de post-traitement est donc envisagée en plusieurs étapes afin de résoudre chacun des cinq types d'erreurs. Celle-ci diminue de 28% le taux d'erreur qui est alors égal à 10.7% sur la base de validation de RIMES.

Une perspective d'amélioration importante est la mise en place d'une étape préalable de reconnaissance de mots clés qui permettrait d'affiner la segmentation.

Dans cette thèse, nous nous sommes attachée à utiliser des protocoles rigoureux pour l'évaluation de nos systèmes. Les bases de données utilisées sont des bases publiques permettant ainsi la comparaison de nos résultats avec ceux d'autres systèmes. De plus chaque système est optimisé sur une base de validation, la base de test n'étant utilisé que pour l'évaluation finale du système. Enfin, la participation à la campagne d'évaluation RIMES a permis d'évaluer notre approche et de montrer sa généralité. En effet, nous avons pu participer à 5 des 6 tâches évaluées couvrant des domaines aussi variés que la reconnaissance de l'écriture manuscrite, la reconnaissance de logos et la structuration de courriers manuscrits et obtenons pour chaque tâche des résultats proches de ceux obtenus par des systèmes plus spécialisés.

Bibliographie

[1] *Traitement automatique des documents*, vol. 22. Traitement du Signal, 2005.

[2] Africa, N., and Yarman-Vural, F. An overview of character recognition focused on off-line handwriting. *IEEE Tansactions on SMC 31*, 2 (2001), 216–233.

[3] Agazi, O., and Kuo, S. Hidden Markov model based optical character recognition in the presence of deterministic transformations. *Pattern Recognition 26*, 12 (1993), 1813–1826.

[4] Augustin, E., Carré, M., Grosicki, E., Brodin, J., Geoffrois, E., and Prêteux, F. RIMES evaluation campaign for handwritten mail processing. *International Workshop on Frontiers in Handwriting Recognition* (2006), 231–235.

[5] Belaid, A. Reconnaissance automatique de l'écriture et du document. *Pour la science* (2001).

[6] Belaid, A., and Saon, G. Utilisation des processus markoviens en reconnaissance de l'écriture. *Traitement du signal 14*, 2 (1997), 161–177.

[7] Bellman, R. *Dynamic Programming*. Princetown Univ. Press, 1957.

[8] Benouareth, A., Ennaji, A., and Sellami, M. Semi-continuous HMMs with explicit state duration applied to arabic handwritten word recognition. *International Workshop on Frontiers in Handwriting Recognition* (2006).

[9] Besag, J. E. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society B 48*, 3 (1986), 259–302.

[10] Bippus, R. 1-dimensional and pseudo 2-dimensional HMMs for the recognition of german literal amounts. *International Conference on Document Analysis and Recognition 2* (1997), 487–490.

[11] Brakensiek, A., Rottland, J., Kosmala, A., and Rigoll, G. Off-line handwritting recognition using various hybrid modeling techniques and character n-grams. *International Workshop on Frontiers in Handwriting Recognition* (2000), 343–352.

[12] Bunke, H., and Caelli, T., Eds. *Hidden Markov Models Applications in computer vision*, vol. 45 of *Machine perception artificial intelligence*. World Scientific, 2001.

[13] Chatelain, C. *Extraction de séquences numériques dans des documents manuscrits quelconques*. PhD thesis, Université de Rouen, 2006.

[14] Chatelain, C., Heutte, L., and Paquet, T. Discrimination between digits and outliers in handwritten documents applied to the extraction of numerical fields. *International Workshop on Frontiers in Handwriting Recognition* (2006), 475–480.

[15] Chen, S., and Goodman, F. An empirical study of smoothing techniques for language modeling. *Thirty-Fourth Annual Meeting of the Association for Computational Linguistics* (1996).

[16] Chevalier, S. *Reconnaissance d'écriture manuscrite par des techniques markoviennes : une approche bidimensionnelle et générique.* PhD thesis, Université René Descartes - Paris 5, 2004.

[17] Chevalier, S., Geoffrois, E., and Prêteux, F. A 2D dynamic programming approach for Markov random field-based handwritten character recognition. *International Conference on Image and Signal Processing* (2003), 617–630.

[18] Chevalier, S., Geoffrois, E., and Prêteux, F. Programmation dynamique 2D pour la reconnaissance de caractères manuscrits par champs de Markov. *Reconnaissance des Formes et Intelligence Artificielle* (2004), 1143–1152.

[19] Chevalier, S., Geoffrois, E., Prêteux, F., and Lemaitre, M. A generic 2D approach of handwriting recognition. *International Conference on Document Analysis and Recognition 1* (2005), 489–493.

[20] Choisy, C., and Belaïd, A. Analytic word without segmentation based on Markov random fields. *International workshop on frontiers in Handwriting recognition* (2000).

[21] Consortium-MADONNE. Madonne : Masse de données issues de la numérisation du patrimoinne. *Atelier sur la Numérisation de l'Ecrit Ancien et des GRAndes Masses de données (ANAGRAM)* (2006).

[22] Coüasnon, B. DMOS, a generic document recognition method : application to table structure analysis in a general and in a specific way. *International Journal on Document Analysis and Recognition 8*, 2 (2006), 111–122.

[23] Dargenton, P. *Contribution à la segmentation et à la reconnaissance de l'écriture manuscrite.* PhD thesis, INSA Lyon, 1994.

[24] Delalandre, M., Trupin, E., and Ogier, J.-M. Analyse structurelle en interprétation de documents : un bref survol. *International Conference on Image and Signal Processing* (2003), 640–649.

[25] Devijver, P., and Kittler, J. *Pattern recognition, a statistical approach.* Prentice Hall, 1982.

[26] Dupré, X. *Contributions à la reconnaissance de l'écriture cursive à l'aide de modèles de Markov cachés.* PhD thesis, Université René Descartes - Paris 5, 2003.

[27] El-Yacoubi, A., Gilloux, M., and Bertille, J.-M. A statistical approach for phrase location and recognition within a text line : An application to street name recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence 24*, 2 (2002), 172–188.

[28] Feldbach, M., and Tönnies, K. D. Line detection and segmentation in historical church registers. *International Conference on Document Analysis and Recognition* (2001), 743–747.

[29] Fiscus, J. A post-processing system to yield reduced word error rates : recognizer output voting error reduction (ROVER). *IEEE Workshop on Automatic Speech Recognition and Understanding* (1997), 347–354.

[30] Gatos, B., Pratikakis, I., Kesidis, A., and Perantonis, S. Efficient off-line cursive handwriting word recognition. *International Workshop on Frontiers in Handwriting Recognition* (2006).

[31] Geman, S., and Geman, D. Stochastic relaxation, gibbs distributions and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence 6*, 6 (1984), 721–741.

[32] Geoffrois, E. Multi-dimensional dynamic programming for statistical image segmentation and recognition. *International Conference on Image and Signal Processing* (2003), 617–630.

[33] Geoffrois, E., Jullian, A., and Debaert, C. Programmation dynamique 2D pour la reconnaissance d'images par modèle de Markov cachés. Tech. rep., DGA/DCE/CEP/GIP, 1998.

[34] Heutte, L., Barbosa-Perreira, P., Bourgeois, O., Moreau, J., Plessis, B., Courtellement, P., and Lecourtier, Y. Multi-bank check recognition system ; consideration on the numeral amount recognition module. *International Journal of Pattern Recognition and Artificial Intelligence 11*, 4 (1997), 595–618.

[35] Huang, X., Acero, A., and Hon, H. *Spoken language processing.* Prentice Hall, 2001.

[36] Jain, A., and Bhattacharjee, S. Address block location on envelopes using Gabor filters. *Pattern recognition 25*, 12 (1992), 1459–1477.

[37] Jain, A., Duin, P., and Mao, J. Statistical pattern recognition : A review. *Pattern Analysis and Machine Intelligence* (2000).

[38] Jelinek, F. *Statistical methods for speech recognition.* MIT Press, 1997.

[39] Kalcheva, E., and Gluchev, G. Segmentation and analysis of handwritten scripts from patients with neurological diseases. *International Conference Computer Systems and Technologies* (2003), 272–277.

[40] Kapoor, R., Bagai, D., and Kamal, T. Representation and extraction of nodal features of DevNagri letters. *ICVGIP* (2003).

[41] Karacs, K., and Roska, T. Holistic feature extraction from handwritten words on wave computers. *Cellular Neural Networks and their Applications* (2004), 364–369.

[42] Kavallieratou, E., Sgarbas, K., Fakotatis, N., and Kokkinakis, G. Handwritten word recognition based on structural characteristics and lexical support. *International Conference on Document Analysis Recognition* (2003), 562–566.

[43] Kim, G., and Govindaraju, V. Bank check recognition using cross validation between legal and courtesy amounts. *International Journal of Pattern Recognition and Artificial Intelligence 11*, 4 (1997), 657–674.

[44] Kirkpatrick, S., Gellat, C. D., and Vecchi, M. P. Optimisation by simulated annealing. *Science*, 220 (1983), 671–680.

[45] Knerr, S., and Augustin, E. A neural network-hidden Markov model hybrid for cursive word recognition. *International Conference on Pattern Recognition 2* (1998), 1518.

[46] Koerich, A. *Large vocabulary off-line handwritten word recognition.* PhD thesis, École de technologie supérieure - Université du Québec, 2002.

[47] Kullback, S., and Leibler, R. A. *On information and sufficiency*, vol. 2. Annals of Mathematical Statistics, 1951, pp. 79–86.

[48] Kuo, S., and Agazi, O. Keyword spotting in poorly printed documents using pseudo 2-D hidden Markov models. *IEEE Transactions on Pattern Analysis and Machine Intelligence 16*, 8 (1994), 842–848.

[49] LeCun, Y. The MNIST database. http ://yann.lecun.com/exdb/mnist/index.html, 1998.

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

[51] Lemaitre, M., Chevalier, S., Geoffrois, E., Grosicki, E., and Prêteux, F. Etude de primitives spectrales pour la reconnaissance de caractères manuscrits dans le cadre d'une approche markovienne 2D. *Reconnaissance des Formes et Intelligence Artificielle* (2006).

[52] Lemaitre, M., Grosicki, E., Geoffrois, E., and Prêteux, F. Preliminary experiments in layout analysis of handwritten letters based on textural and spatial information and a 2D Markovian approach. *International Conference on Document Analysis and Recognition* (2007).

[53] Levin, E., and Pieraccini, R. Dynamic planar warping for optical character recognition. *IEEE International Conference on Acoustics, Speech and Signal Processing* (1992).

[54] Li, Y., Zheng, Y., Doermann, D., and Jaeger, S. A new algorithm for detecting text line in handwritten documents. *International Workshop on Frontiers in Handwriting Recognition* (2006).

[55] Likforman, L., Godeau, J., and Sigel, M. ENST fax character database. CD-ROM.

[56] Likforman-Sulem, L., and Faure, C. Extracting lines on handwritten documents by perceptual grouping. *dvances in Handwriting and Drawing : a multidisciplinary approach* (1994), 21–39.

[57] Likforman-Sulem, L., and Faure, C. A Hough based algorithm for extracting text lines in handwritten documents. *International Conference On Document Analysis and Recognition* (1995), 774–777.

[58] Liu, C. L., Nakashima, K., Sako, H., and Fujisawa, H. Handwritting digit recognition using state-of-the-art techniques. *International Workshop on Frontiers in Handwriting Recognition* (2002), 320–325.

[59] Madhvanath, S., Govindaraju, V., Ramanaprasad, V., Lee, D. S., and Srihari, S. N. Reading handwritten US census forms. *International Conference on Document Analysis and recognition 1* (1995), 82–87.

[60] Marti, U., and Bunke, H. A full english sentence database for off-line handwriting recognition. *International Conference on Document Analysis and Recognition* (1999), 705–708.

[61] Marti, U., and Bunke, H. Handwritten sentence recognition. *International Conference on Pattern Recognition 3* (2000), 467–470.

[62] Marti, U., and Bunke, H. Unconstrained handwriting recognition : Language models, perplexity, and system performance. *International Workshop on Frontiers in Handwriting Recognition* (2000).

[63] Marti, U., and Bunke, H. Text line segmentation and word recognition in a system for general writer independent handwriting recognition. *International Conference on Document Analysis and Recognition* (2001), 159–163.

[64] Miled, H., and Amara, N. B. Planar Markov modeling for arabic writing recognition : Advancement state. *International Conference on Document Analysis and Recognition 1* (2001), 19–73.

[65] Milewski, R. J. *Automatic Recognition of Handwritten Medical Forms for Search Engines.* PhD thesis, Université de New York, 2006.

[66] Nagy, G. Twenty years of document image analysis in PAMI. *IEEE Trans. on Pattern Analysis and Machine Intelligence 22*, 1 (2000), 38–62.

[67] Nicolas, S. *Segmentation par champs aléatoires pour l'indexation d'images de documents.* PhD thesis, Université de Rouen, 2006.

[68] Nicolas, S., Kessentini, Y., Paquet, T., and Heutte, L. Handwritten document segmentation using Hidden Markov Random Fields. *International Conference on Document Analysis and Recognition 1* (2005), 212–216.

[69] Nicolas, S., Paquet, T., and Heutte, L. Markov random field models to extract the layout of complex handwritten documents. *International Workshop on Frontiers in Handwriting Recognition* (2006), 563–568.

[70] Pal, U., Belaïd, A., and Choisy, C. Water reservoir based approach for touching numeral segmentation. *International Conference on Document Analysis and Recognition* (2001), 892.

[71] Park, H., and Lee, S. An HMMRF-based statistical approach for off-line handwritten character recognition. *International Conference on Pattern Recognition 2* (1996), 320–324.

[72] Rabiner, L., and Juang, B. *Fundamentals of speech recognition.* Prentice Hall, 1993.

[73] Rath, T., Lavrenko, V., and Manmatha, R. A statistical approach to retrieving historical manuscript images without recognition. Tech. rep., Center for Intelligent Information Retrieval technical, 2003.

[74] RIMES. Collecte de courriers manuscits SCRIBEO pour le projet RIMES. http ://www.scribeo.org.

[75] RIMES. Le projet RIMES, 2007. http ://www.int-evry.fr/rimes/.

[76] Rousseau, L., Anquetil, E., and Camillerapp, J. Recovery of a drawing order from off-line isolated letters dedicated to on-line recognition. *International Conference on Document Analysis and Recognition 2* (2005), 1121–1125.

[77] Saon, G. *Modèles markoviens uni- et bidimensionnels pour la reconnaissance de l'écriture manuscrite hors-ligne.* PhD thesis, Université Henri Poincaré - Nancy 1, 1997.

[78] Sayre, K. M. Machine recognition of handwritten words : a project report. *Pattern recognition 5* (1973), 213–228.

[79] Schmid, C. *Appariement d'images par invarients locaux de niveaux de gris.* PhD thesis, INPG, 1996.

[80] Senior, A. *Offline handwriting recognition using recurrent neural networks.* PhD thesis, Cambridge University, 1994.

[81] Senior, A. W., and Fallside, F. Off-line handwriting recognition : A review and experiments. *Technical Report 105 Cambridge University Engineering Department* (1992).

[82] Senior, A. W., and Robinson, A. J. An off-line cursive handwriting recognition system. *Pattern Analysis and Machine Intelligence 20* (1998), 309–321.

[83] Simon, J., and Baret, O. Regularities and singularities in line pictures. *International Journal of Pattern Recognition and Artificial Intelligence 5*, 1-2 (1991), 57–77.

[84] Stolcke, A., and Omohundro, S. Hidden Markov model induction by bayesian model merging. *Advances in Neural Information Processing Systems* (1993).

[85] Taconet, B., Zahour, A., Ramdane, S., and Boussellaa, W. Classification des k-ppv par sous-voisinages emboîtés. *Colloque International Francophone sur l'Écrit et le Document* (2006), 145–150.

[86] Tanaka, H., Onizuka, K., and Asai, K. Classification of proteins via successive state splitting of hidden Markov network. *International Joint Conference on Artificial Inteligence, Artificial Inteligence and Genome* (1993).

[87] Tappert, C. C., Suen, C., and Wakahara, T. On-line handwriting recognition - A survey. *In Proceedings og 9th International Conference on Pattern Recognition* (1988), 1123–1127.

[88] Thomsen, R. Evolving the topology of hidden Markov models using evolutionary algorithms. *Parallel Problem Solving from Nature VII* (2002), 861–870.

[89] Touj, S., Amara, N. B., and Amiri, H. Generalized Hough transform for arabic optical character recognition. *Internation Conference on Document Analysis Recognition* (2003), 1242–1246.

[90] Trier, I., Jain, A., and Taxt, T. Feature extraction methods for character recognition - A survey. *Pattern Recogntion 29*, 4 (1995), 641–662.

[91] Trupin, E. La reconnaissance d'images de documents : Un panorama. *Traitement du signal 22*, 3 (2005), 159–189.

[92] Tulyakov, S., and Govindaraju, V. Postal address block location by contour clustering. *International Conference on Document Analysis and recognition* (2003), 429–432.

[93] Vapnik, V. *The nature of statistical learning theory*. Springer, 1995.

[94] Verma, B. A contour code feature based segmentation for handwriting recognition. *International Conference on Document Analysis and Recognition* (2003), 1038–1042.

[95] Vincent, P., and Bengio, Y. K-local hyperplane and convex distance. *Technical Report 1197* (2001).

[96] Vinciarelli, A. A survey on off-line cursive word recognition. *Pattern Recognition 35*, 07 (2002), 1433–1446.

[97] Vinciarelli, A., and Luettin, J. Off-line cursive script recognition based on continuous density HMM. *International Workshop on Frontiers in Handwriting Recognition* (2000), 493–498.

[98] Vintsyuk, T. K. *Speech discrimination by dynamic programming*. Kibenetika, 1968, ch. 4, pp. 81–88.

[99] Wienecke, M., Fink, G. A., and Sagerer, G. Experiments in unconstrained offline handwritten text recognition. *International Workshop on Frontiers in Handwriting Recognition* (2002).

[100] Xu, Q., Lam, L., and Suen, C. Y. Automatic segmentation and recognition system for handwritten dates on canadian bank cheques. *International Conference on Document Analysis and recognition* (2003), 704–708.

[101] Yada, T., Ishikawa, M., Tanaka, H., and Asai, K. Extraction of hidden Markov model representations of signal patterns in DNA sequences. *Pacific Symposium on Biocomputing* (1996), 686–696.

[102] Zhang, D., and Lu, G. Review of shape representation and descriptions techniques. *Pattern recognition 37*, 1 (2004), 1–19.

[103] Zimmermann, M. The IAM database. http ://www.iam.unibe.ch/ zimmerma/iamdb/iamdb.html.

[104] Zimmermann, M., and Bunke, H. Automatic segmentation of the IAM offline database for handwritten english text. *International Conference on Pattern Recognition 4* (2000), 35–39.

[105] Zouari, H., Heutte, L., Lecourtier, Y., and Alimi, A. Un simulateur de classifieur pour évaluer les méthodes de combinaison. *Reconnaissance des formes et Intelligence Artificielle* (2002).

Résumé

Dans cette thèse, nous présentons une approche bidimensionnelle markovienne générale pour l'analyse et la reconnaissance de documents manuscrits appelée AMBRES (Approche Markovienne Bidimensionnelle pour la Reconnaissance et la Segmentation d'images). Elle est fondée sur les champs de Markov, la programmation dynamique 2D et une analyse bidimensionnelle de l'image.
AMBRES a été appliquée avec succès à des tâches aussi diverses que la reconnaissance de caractères et de mots manuscrits isolés, la structuration de documents manuscrits et la reconnaissance de logos et pourrait être étendue à d'autres problématiques du domaine de la vision.
Des protocoles rigoureux ont été utilisés pour l'étude du système et de ses paramètres ainsi que pour l'évaluation des performances. En particulier, AMBRES a pu être validée au sein de la campagne d'évaluation RIMES (Reconnaissance et Indexation de données Manuscrites et de fac similÉS).

Mots-clés : Reconnaissance de l'écriture manuscrite, structuration de documents manuscrits, champs de Markov, programmation dynamique 2D, approche générale.

Abstract

In this thesis, we present a general bidimensional markovian approach in the framework of handwritten document analysis and recognition. This approach called AMBRES (Bidimensional Markovian Approach for image Recognition and Segmentation) is based on Markov random fields, 2D dynamic programming and a bidimensional analysis of images.
AMBRES has been successfully applied to a wide variety of tasks such as handwritten character recognition, handwritten word recognition, layout analysis of handwritten documents and logo recognition. It could be also used for other tasks in computer vision.
Rigorous protocols have been considered in order to evaluate the system and its performances. In particular, AMBRES has been tested in the framework of the RIMES evaluation campaign.

Keywords: Handwriting recognition, layout analysis of handwritten documents, Markov random fields, 2D dynamic programming, general approach.