Reconnaissance d'écriture manuscrite par des techniques markoviennes : une approche bidimensionnelle et générique

Sylvain Chevalier

Université René Descartes - Paris 5
Centre Universitaire des Saints - Pères
UFR de mathématiques et informatique

Thèse présentée en vue de l'obtention du grade de Docteur
de l'Université René Descartes - Paris 5

Reconnaissance d'écriture manuscrite par des techniques markoviennes : une approche bidimensionnelle et générique

Sylvain Chevalier

Soutenue le 3 décembre 2004, devant le jury composé de :

Georges Stamon	Président du jury
Abdel Belaïd	Rapporteur
Hubert Emptoz	Rapporteur
Emmanuel Augustin	Examinateur
Edouard Geoffrois	Examinateur
Christine Graffigne	Examinatrice
Françoise Prêteux	Directrice de thèse

Résumé

Nous présentons une approche de reconnaissance d'écriture manuscrite à partir de champs de Markov cachés et fondée sur une analyse entièrement bidimensionnelle de l'écriture. Son originalité réside dans la combinaison d'une analyse fenêtrée de l'image, d'une modélisation markovienne et dans la mise en œuvre de la programmation dynamique 2D qui permet un décodage rapide et optimal des champs de Markov. Un aspect important de ces travaux est la méthodologie de développement employée qui est centrée sur l'évaluation systématique des apports algorithmiques et des paramètres utilisés. Ces algorithmes sont en partie empruntés aux techniques utilisées dans le domaine de la reconnaissance de la parole et sont très génériques.

L'approche proposée est validée sur deux applications correspondant à des bases de données standard et librement disponibles. L'application de cette méthode extrêmement générique à une tâche de reconnaissance de chiffres manuscrits a permis d'obtenir des résultats comparables à ceux de l'état de l'art. L'application à une tâche de reconnaissance de mots manuscrits a permis de confirmer que l'extension de cette approche à des tâches plus complexes était naturelle.

L'ensemble de cette recherche a démontré la validité de l'approche développée qui apparaît comme candidate au statut d'approche standard pour plusieurs problèmes de vision. En outre, elle ouvre la voie à de très nombreux développements concernant la tâche de traitement de l'écriture manuscrite et des améliorations significatives pourraient encore être apportées en recourant à d'autres principes issus du traitement de la parole et du langage. D'autres tâches comme la segmentation d'image devraient tirer avantage de la robustesse et de la faculté d'apprentissage de la modélisation que nous proposons.

Mots-clés

Reconnaissance de l'écriture manuscrite, champs de Markov cachés, programmation dynamique 2D.

Abstract

We present an approach of the problem of handwriting recognition using hidden Markov random fields and based on a trully bidimensional analysis of the handwriting. The main innovative aspect of this approah is the combination of a windowed analysis of the image, a Markovian modelisation and an implementation of the 2D dynamic programming algorithm that achieves a fast and optimal decoding of Markov fields. Another feature of this study is the development methodology that focuses on a systematic evaluation of the algorithms and parameters. These algorithms are partly taken from techniques of the domain of speech processing and are very generic.

This approach is validated on two different applications corresponding to standard public databases. The application of this generic algorithm to a handwritten digits recognition task achieved results similar to the ones of state-of-the-art methods. The application to a handwritten words recognition task showed that this approach can be extended to more complex tasks in a natural way.

This work showed that the proposed approach is valid and appears as a candidate standard method for solving various tasks in computer vision. It paves the way for further developments in handwriting recognition and other important enhancements are expected with the use of other principles commonly used in speech and language processing problems. Other tasks such as image segmentation could also benefit from the robustness and the learning ability of our approach.

Keywords

Handwriting recognition, hidden Markov random fields, 2D dynamic programming.

Remerciements

En premier lieu, je souhaite témoigner de toute ma reconnaissance à M. Edouard Geoffrois, encadrant et soutien au quotidien, et Mme Françoise Prêteux, ma directrice de thèse, pour m'avoir fait confiance pour conduire ces travaux et pour m'avoir enseigné le métier de chercheur. Je suis particulièrement touché par la disponibilité qu'ils m'ont accordée tout au long de ces recherches. Je leur suis redevable du plaisir pris pendant ces trois années et de précieuses connaissances qui m'accompagneront pendant les suivantes.

Je souhaite remercier particulièrement M. Georges Stamon pour avoir suivi avec bienveillance tout le déroulement de cette recherche et pour avoir accepté de prendre part à mon jury de thèse.

Je tiens à remercier M. Abdel Belaïd et M. Hubert Emptoz, pour avoir accepté le difficile rôle de rapporteur de ces travaux, ainsi que Mme Christine Graffigne et M. Emmanuel Augustin, témoins et acteurs de la genèse de cette thèse, pour avoir accepté de prendre part à ce jury. Je souhaite associer à ces remerciements M. Marc Sigelle et Mme Laurence Likforman pour l'intérêt qu'ils ont manifesté pour ces travaux et pour pour leurs remarques, toujours très pertinentes.

Enfin, je ne saurais trop remercier l'ensemble de mes collègues du Centre technique d'Arcueil et de l'unité de projets ARTEMIS de l'INT pour l'ambiance chaleureuse qu'ils créent au quotidien et que j'ai particulièrement appréciée.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction générale

5000 ans après son invention, 550 ans après son automatisation, l'écriture est toujours au cœur des communications entre les hommes. À l'heure des interactions entre l'humain et la machine toujours plus sophistiquées et performantes à l'aide de « boutons », de microphones ou de caméras, il est naturel de chercher à comprendre automatiquement l'écriture. Depuis les premières tentatives, les systèmes de lecture des adresses pour le traitement automatique du courrier ou de lecture des chèques ont connu d'importants développements et sont maintenant largement utilisés. Toutefois, la compréhension de l'écriture par un ordinateur est encore loin d'être pleinement satisfaisante. La raison est liée au fait que l'étude de la reconnaissance de l'écriture est un domaine très vaste tant par ses applications que par ses techniques.

On distingue traditionnellement la reconnaissance d'écriture hors-ligne de la reconnaissance en-ligne. La reconnaissance hors-ligne traite des images numérisées et produit, après analyse, un fichier texte comportant éventuellement des indications de mise en page ou de fontes. La reconnaissance en-ligne traite des signaux temporels qui sont saisis au moyen d'un stylet sur une surface sensible, comme dans les assistants personnels numériques (PDA) désormais populaires. De plus, de grandes différences de complexité existent entre les problèmes de reconnaissance de caractères, de mots ou de documents imprimés ou manuscrits, plus ou moins dégradés. Enfin, la taille du vocabulaire ainsi que le type de support (chèque, formulaire, page blanche, etc.) peuvent être largement variables.

Une grande partie des problèmes de reconnaissance des formes est aujourd'hui traitée par des méthodes statistiques, qui constituent de plus en plus le cœur de la plupart des meilleurs systèmes de reconnaissance automatique de l'écriture (RAE). L'approche statistique avait déja fait le succès des systèmes de reconnaissance automatique de la parole (RAP), et a donc été rapidement exportée vers d'autres problématiques.

Une approche commune du problème de la reconnaissance automatique de l'écriture consiste à traiter des lignes de texte comme un signal monodimensionnel en recourant aux techniques de la reconnaissance automatique de la parole. Les algorithmes restant quasiment identiques, la différence principale se situe au niveau du calcul des primitives. Après le calcul de ces primitives, le processus de RAP traite un signal temporel lorsque le processus de RAE traite un signal dépendant de la coordonnée horizontale. L'approche classique qui

permet de traiter efficacement ces problèmes, fondée sur l'hypothèse markovienne, considère que le signal observé à un instant t ne dépend des instants précédents que sur une durée limitée. Cependant, si l'utilisation d'un signal temporel pour la parole correspond à la réalité physique, considérer l'écriture comme un signal de la coordonnée horizontale est un choix plus contestable car il ignore la nature intrinsèquement bidimensionnelle des lettres et des mots. Pour cette raison des systèmes de reconnaissance de caractères et de mots utilisant une modélisation par champs de Markov ont été proposés en une sorte d'extension 2D des chaînes de Markov monodimensionnelles de la reconnaissance de la parole. Cependant, si les calculs d'optimisation dans le cadre d'une dépendance locale monodimensionnelle étaient traités par programmation dynamique, rien de tel n'existait dans le cas de la dépendance locale des champs de Markov. Cela obligeait à avoir recours à des algorithmes soit très lents, soit sous-optimaux. L'apparition de l'algorithme de programmation dynamique 2D a ouvert la voie à la modélisation entièrement 2D de l'écriture.

Pour justifier de l'intérêt d'une approche, d'une technique ou d'un modèle par rapport à un autre, il est crucial d'évaluer un système de reconnaissance pour le comparer aux autres. Cette remarque de bon sens est pourtant trop souvent négligée et on trouve rarement, à la suite de la présentation d'un système, la donnée d'une performance attachée à des images disponibles publiquement, avec une métrique ainsi que des données de test et d'apprentissage clairement définies.

Notre approche intègre totalement cette dimension et recherche une modélisation complètement 2D de l'écriture. Elle a en particulier pour apports principaux :

➪ l'utilisation de la programmation dynamique 2D,

➪ un processus d'extraction de primitives proche du cepstre de RAP et strictement 2D,

➪ une construction de modèles de mots par concaténation de modèles de lettres,

➪ une évaluation sur des bases de données publiques de caractères et de mots manuscrits.

Les apports spécifiques de ce travail peuvent être synthétisés en évoquant la situation en début et en fin de thèse :

- En début de thèse il m'a été demandé de traiter une tâche de reconnaissance d'écriture manuscrite en utilisant des méthodes statistiques et en particulier le principe de la programmation dynamique 2D dont une première implantation permettait la segmentation en facettes d'images de profondeur.
- En fin de thèse, des algorithmes utilisant ces méthodes ont été utilisés avec succès sur deux tâches de reconnaissance d'écriture manuscrite et sont directement utilisables sur de nombreuses autres tâches de traitement d'image.

Nous présenterons au chapitre 2 les techniques et algorithmes classiques de la reconnaissance de l'écriture manuscrite hors-ligne, ainsi qu'un aperçu des bases de données standard utilisées pour les expérimentations de RAE et les performances des algorithmes sur ces bases. Au chapitre 3, nous détaillerons les techniques mises en œuvre au cours de notre recherche empruntant aussi bien au domaine du traitement de la parole qu'à la modélisation par champs de Markov. Le chapitre 4 décrira nos principales contributions sur l'adaptation et l'implantation de ces techniques à une tâche de reconnaissance de caractères manuscrits.

Nous y proposerons également une extension de ces techniques pour le traitement des mots manuscrits. Le dernier chapitre recensera les développements envisagés pour améliorer les performance de notre approche et étendre son champ d'application au traitement des documents manuscrits.

On trouvera en annexe des illutrations abondantes de nos données, un aperçu de l'application de notre approche à une tâche de reconnaissance de route dans des vidéos ainsi que les grandes lignes de l'implantation informatique de nos algorithmes.

Chapitre 2

État de l'art en reconnaissance de l'écriture manuscrite

Dans cette étude, nous nous intéresserons à la reconnaissance hors-ligne. La reconnaissance de l'écriture hors-ligne est un processus dont l'entrée est une image (un tableau de pixels) et la sortie un texte informatique. Cette définition exclut les systèmes de reconnaissance dits en-ligne où le texte est saisi avec un stylet sur une surface sensible, fournissant ainsi un signal d'entrée temporel.

Le champ des applications recouvert par la reconnaissance de l'écriture manuscrite est très vaste : reconnaissance de caractères, mots isolés, textes ou documents entiers, plus ou moins contraints. Actuellement, de nombreux systèmes de reconnaissance de caractères imprimés fonctionnent avec d'excellentes performances de même que des systèmes de lecture automatique de chèques ou d'adresses postales. Pour ces applications, les contraintes sur le vocabulaire et sur la structure des documents sont fortes et une modélisation imparfaite de l'écriture peut alors convenir, en particulier concernant son aspect bidimensionnel. Cependant, le problème de la reconnaissance de documents manuscrits non contraints reste loin d'être résolu.

Depuis le début des recherches dans le domaine, la plupart des techniques de reconnaissance des formes ont été expérimentées et de nombreuses analyses/synthèses de ces techniques proposées [Lorette et Crettez, 1998, Dupré, 2003].

2.1 Techniques classiques

Le processus de reconnaissance est précédé de la phase d'acquisition [Senior et Robinson, 1998], typiquement au moyen d'un scanner en noir et blanc ou en niveaux de gris. La plupart des travaux de recherche se fondent sur des images de mots déjà segmentés [Casey et Lecolinet, 1996]. Les techniques utilisées pour ce processus ont radicalement changé depuis le début des systèmes d'OCR commerciaux dans les années 50 [Govindan et Shivaprasad, 1990], et surtout depuis l'apparition de moyens de calculs puissants dans les années 80 [Hildebrandt et Liu, 1993, Belaid et Belaid, 1992]. On est passé de méthodes structurelles

où chaque caractère était comparé à un patron rigide, à des méthodes statistiques [Gilloux, 1994a] ou par réseaux de neurones [Alpaydin et Gurgen, 1996]. Ces derniers donnent de bons résultats pour certains types de documents comme les chèques ou les adresses manuscrites. Ces applications nécessitent de faibles taux de rejets et des taux d'erreurs très faibles car le coût des erreurs est extrêmement élevé.

2.1.1 Les prétraitements

Une fois l'image numérisée, une série de prétraitements est appliquée, d'abord pour extraire des lignes de texte, éventuellement segmentées en mots ou en caractères en détectant les espaces. Cette phase comprend aussi des processus de traitement d'image en général (atténuation du bruit, binarisation...) et d'autres plus spécifiques au traitement de l'écriture manuscrite. Traditionnellement, les prétraitements contribuent à diminuer la variabilité intra-classes par des traitements de normalisation. Toutefois, l'orientation actuelle vise à diminuer le nombre de prétraitements pour gérer la variabilité au niveau des modèles [Saon, 1997]. Contrairement aux autres phases de la reconnaissance, celle des prétraitements est relativement standard, la plupart des systèmes de reconnaissance utilisent aujourd'hui les mêmes prétraitements renvoyant :

1. au filtrage de l'image,
2. à la normalisation des mots,
3. à la réduction de la taille des données.

Filtrage de l'image

Les applications actuelles de reconnaissance de l'écriture utilisent souvent des supports dont le fond n'a pas une couleur unie (chèques). Il faut donc isoler le mot du motif d'arrière-plan. Les lignes droites et les cadres sont détectés par des projections. Les textures et couleurs de fond sont traitées par des techniques de filtrage [Arica et Yarman-Vural, 2001]. Des connaissances *a priori* sur le support sont parfois utilisées [Knerr *et al.*, 1997, Gilloux, 1994a] (structure conventionnelle d'un chèque barré, d'une adresse postale).

Une atténuation du bruit est ensuite effectuée. Ce bruit peut être déjà présent sur le support avant la numérisation ou être introduit par la phase d'acquisition ou les premiers prétraitements. Ce sont là encore des techniques de filtrage qui sont utilisées pour éliminer le bruit de type "poivre et sel" ou pour arrondir les contours. Un test sur la taille des motifs est aussi effectué pour éliminer les tâches trop petites pour faire partie d'un caractère.

Normalisation des mots

La normalisation tend à réduire les variations entre les styles, tailles et orientations d'écriture. La tendance est à réduire l'importance de cette phase pour prendre en compte cette variabilité au niveau des modèles. Toutefois, les algorithmes 1D sont souvent peu efficaces pour absorber certaines variations comme l'inclinaison de l'écriture. En effet, le

découpage en fenêtres verticales pour les HMM (Hidden Markov Models, modèles de Markov cachés en français) serait moins pertinent pour une écriture inclinée.

Les traitements de normalisation sont les suivants :

1. Mise à l'horizontale de la ligne d'écriture

   Un défaut d'orientation du support pendant l'acquisition, ou une écriture imprécise peuvent conduire à une inclinaison et une déformation de la ligne de base. Des traitements sont appliqués pour la rendre horizontale [Senior et Robinson, 1998]. Cette ligne de base du texte (sur laquelle reposent les caractères) est utilisée ensuite pour l'extraction des primitives.

2. Correction de l'inclinaison des caractères

   On remarque que très souvent, les caractères sont inclinés : leur direction générale forme un angle avec la verticale [Steinherz *et al.*, 1999] variable selon les scripteurs. Les systèmes de reconnaissance corrigent toujours cette inclinaison pour diminuer la variabilité intra-classe et pour rendre la segmentation en caractères plus facile, notamment dans le cas de l'utilisation des chaînes de Markov qui supposent que l'on peut segmenter en caractères par des lignes verticales (notion de fenêtre qui se déplace de gauche à droite dans le temps).

   Les techniques de correction sont en général fondées sur la détection des traits quasi-verticaux. On calcule leur inclinaison moyenne par rapport à la verticale et on corrige l'inclinaison de tous les mots avec ce même angle moyen [Vincieralli et Luettin, 2000, Arica et Yarman-Vural, 2001].

Réduction de la taille des données

La plupart des systèmes de reconnaissance utilisent des images en noir et blanc. L'écriture manuscrite étant souvent noire sur fond blanc, les systèmes actuels traitent généralement les images sans utiliser les informations de niveaux de gris ou de couleurs qui sont pourtant fournies par le procédé de numérisation. Suivant le procédé d'acquisition, cette phase peut être plus ou moins délicate. En général, le seuil de binarisation est facilement déterminé sur l'histogramme des niveaux de gris, mais il peut être nécessaire, dans certains cas, d'utiliser une méthode locale où le seuil est choisi sur chaque petite partie de l'image [Arica et Yarman-Vural, 2001].

Il est aussi fréquent d'éroder l'épaisseur des traits jusqu'à obtenir un squelette pour diminuer la taille des données [Arica et Yarman-Vural, 2001]. Ce traitement peut cependant causer la perte de données importantes.

2.1.2 Les primitives classiques

Suivant la technique utilisée pour la phase de reconnaissance proprement dite, les primitives utilisées peuvent être très différentes. Peu d'études théoriques sont faites sur ce sujet où l'intuition prévaut. On peut tout de même mentionner la thèse d'Olivier Baret [Baret, 1990, Simon et Baret, 1990] sur l'importance des irrégularités dans le pouvoir

de discrimination et une étude sur le choix des ensembles de primitives [Grandidier *et al.*, 2000].

Une première distinction peut être faite entre les primitives locales et globales. Les primitives globales cherchent à représenter au mieux la forme générale d'un caractère ou d'un mot et sont donc calculées sur des images relativement grandes. Les primitives locales sont calculées lors d'un parcours des pixels de l'image avec un pas d'analyse qui dépend de la modélisation, du type de primitive et de la taille de l'image. On ne désigne pas seulement par ce terme les primitives issues d'une analyse locale autour d'un pixel, mais aussi celles nécessitant de considérer les pixels plus éloignés (primitives cellulaires [Hildebrandt et Liu, 1993]).

Primitives locales

Les primitives locales sont principalement utilisées dans le cadre d'une modélisation markovienne qui modélise la structure générale du caractère. Suivant le type de caractère ou de mot analysé (caractère chinois, latin cursif, détaché), ces primitives sont différentes. Les primitives cellulaires s'adaptent bien à la reconnaissance de caractères isolés [Hildebrandt et Liu, 1993] alors que pour une utilisation dans une chaîne de Markov dont les états sont des graphèmes ou pseudo-lettres, on préférera analyser les singularités comme les intersections entre traits [Knerr *et al.*, 1997, Feray et de Brucq, 1996], les extrémités ou les maxima et minima de l'écriture [Gilloux, 1994a]. Pour certaines applications comme la lecture de chèques, un même système de reconnaissance doit pouvoir traiter les lettres manuscrites et imprimées, d'où la nécessité d'inclure plusieurs types de primitives locales [Knerr *et al.*, 1997].

Primitives globales

La quasi-totalité des primitives globales est décrite et analysée par Trier *et al.* dans un article de synthèse [Trier *et al.*, 1996] où les primitives sont discutées en termes de propriétés d'invariance, de reconstructibilité et de variabilité des caractères. Une première classification de ces primitives est faite par le type de données sur lesquelles elles s'appliquent (binaire, niveaux de gris ou vectorielle). Si les primitives globales cherchent d'abord à modéliser les caractères au moyen d'invariants (comme les différents moments) ou de caractéristiques plus sensibles aux déformations (projections, histogrammes), mais calculées après une normalisation. Ces primitives globales ne s'appliquent pas qu'à des classifications classiques du type k plus proches voisins, mais aussi aux approches statistiques. Les caractéristiques extraites des boucles en particulier (position, hauteur, largeur...) sont importantes pour les modélisations par des chaînes de Markov [Knerr *et al.*, 1997]. L'utilisation de ce type de primitives nécessite souvent d'avoir en entrée des images de caractères ou de mots déjà correctement segmentés.

2.1.3 Reconnaissance de mots à base de chaînes de Markov cachées

Les succès des modèles de Markov cachés dans le traitement de la parole ont incité les chercheurs à les utiliser pour traiter l'écriture manuscrite. Le principe est d'assimiler la dimension horizontale de l'écriture au temps et de calculer des vecteurs de primitives dans une fenêtre parcourant cet axe. La plupart des techniques de reconnaissance de la parole peuvent ainsi être employées en remplaçant la notion de lettre par celle de phonème. Ainsi, depuis le milieu des années 90, l'utilisation des HMM s'est généralisée pour la reconnaissance de mots, pour des applications comme la lecture de chèques [Leroux *et al.*, 1997, Knerr *et al.*, 1997] ou d'adresses postales [Grandidier *et al.*, 1999, El-Yacoubi *et al.*, 1999] pour lesquelles on dispose de bases de données importantes (cf. section 2.2).

Les techniques employées pour la reconnaissance par HMM diffèrent peu de de celles décrites dans la section 3.2 pour la parole. Leurs caractéristiques principales sont :

- **Primitives utilisées**
  C'est un point crucial pour les performances de la reconnaissance. Les systèmes les plus performants utilisent de très grands vecteurs de primitives [Knerr *et al.*, 1997] à la fois locales et globales ainsi que des informations de segmentation sur les connections entre les fenêtres consécutives.
- **Gestion des vecteurs d'observation**
  On trouve dans la littérature des HMM discrets [Procter *et al.*, 2000, Grandidier *et al.*, 2000], continus (les observations sont modélisées par des combinaisons linéaires de gaussiennes) [Gilloux, 1994a] ou dont les vecteurs d'observation sont fournis par des réseaux de neurones [Knerr et Augustin, 1998]. L'observation peut être faite de manière équivalente sur les états [Knerr *et al.*, 1997] ou sur les transitions [Gilloux, 1994a].
- **Type de discrimination**
  On choisit une approche d'un modèle discriminant pour une application de reconnaissance de mots isolés et pour de petits vocabulaires (on a un modèle pour chaque mot et la reconnaissance se fait suivant la loi de Bayes avec la probabilité *a posteriori* de l'observation), ou d'un chemin discriminant quand le vocabulaire est important ou pour la reconnaissance de mots enchaînés (on a un modèle pour plusieurs mots et c'est le décodage de Viterbi qui effectue la reconnaissance en cherchant le chemin optimal dans un treillis de mots).

Il faut noter qu'au-delà de la modélisation des lettres et des mots, les chaînes de Markov sont aussi utilisées pour la modélisation du langage, comme cela se fait en parole [Knerr *et al.*, 1997, Gilloux, 1994a].

2.1.4 Vers une approche statistique 2D

La structure 2D de l'image a incité les chercheurs à se pencher sur une modélisation markovienne à deux dimensions depuis le milieu des années 90 [Gilloux, 1994a]. La plupart des auteurs font remarquer qu'une approche entièrement bidimensionnelle du problème, avec les algorithmes actuels, conduirait à une complexité excessive [Levin et Pieraccini, 1992].

Ainsi, les modèles utilisés sont-ils des chaînes de Markov améliorées (modèles de Markov pseudo-2D) ou des champs de Markov avec hypothèses simplificatrices. Les champs de Markov sont utilisés déjà depuis longtemps en traitement d'images avec succès [Chellappa et Jain, 1993, Derras, 1993, Cai et Liu, 2001]. Plusieurs types de champs de Markov sont mis en œuvre suivant notamment les contraintes de dépendance et de causalité imposées.

On peut ainsi distinguer deux extensions des HMM au cas 2D [Belaïd et Saon, 1997] :
- les modèles de Markov pseudo-2D,
- les champs de Markov.

Le premier modèle est plutôt un modèle deux fois 1D alors que le second est intrinsèquement bidimensionnel.

Le modèle de Markov pseudo-2D ou planaire (PHMM)

La modélisation par PHMM consiste à reporter le traitement des distorsions horizontales et verticales sur deux types de chaînes de Markov. Un ensemble de chaînes de Markov détermine les motifs horizontaux les plus probables tandis qu'une chaîne orientée verticalement associe des états (appelés super-états [Belaïd et Saon, 1997]) à ces motifs pour en déduire la forme complète [Levin et Pieraccini, 1992, Gilloux, 1994b, Pessoa, 1995]. Cette approche interdit la prise en compte de la corrélation des distorsions verticales et horizontales.

Les états cachés associés à la modélisation par PHMM sont indéxes sur une matrice d'observables qui peuvent être issus de la valeur brute du pixel (sur 1, 4 ou 8 bits) ou d'une extraction de primitives de type cellulaire [Kuo et Agazzi, 1994, BenAmara, 1995] : La couleur du pixel (0 ou 1), la taille du trait courant, la distance aux traits voisins, etc. D'autre part, l'utilisation combinée de modèles monodimensionnels permet d'effectuer un décodage grâce à la programmation dynamique.

Le champ de Markov

La modélisation par champs de Markov est très performante pour le traitement de nombreux problèmes d'analyse d'image et l'hypothèse markovienne de dépendance locale a fréquemment montré sa pertinence. Les bases de la théorie des champs de Markov sont données dans la section 3.3. L'utilisation de ces modèles se heurte cependant à des problèmes algorithmiques importants en particulier concernant les temps de calcul. Ainsi, plusieurs hypothèses supplémentaires ont été proposées pour traiter efficacement certains problèmes de vision comme le traitement de l'écrit. Deux modèles principaux ont été proposés qui diffèrent selon les contraintes de voisinage et de causalité qu'ils proposent [Belaïd et Saon, 1997] :

1. les réseaux de Markov,
2. les champs de Markov unilatéraux.

1. Les réseaux de Markov
   À la suite des recherches de Gilloux [Gilloux, 1994b], le modèle des réseaux de Markov (Hidden Markov Mesh Random Field) a été développé et utilisé surtout pour

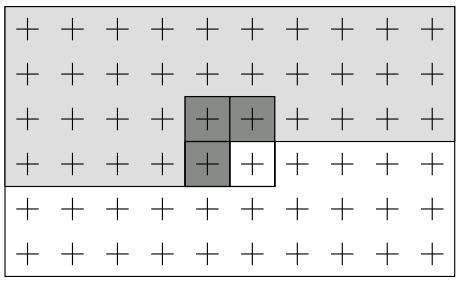


FIG. 2.1 : Dépendance locale pour un réseau de Markov : l'état du pixel courant ne dépend que des 3 pixels voisins compte-tenu du sens de parcours.

la reconnaissance de caractères chinois dont l'aspect 2D est encore plus important que pour les mots manuscrits en caractères latins. Une particularité des réseaux de Markov est qu'ils ne respectent pas la définition de voisinage des champs de Markov en introduisant une notion de causalité qui permet de réaliser le décodage par un balayage monodimensionnel de l'image (Figure 2.1).

Wang *et al.* [Wang *et al.*, 2000] ont utilisé ce modèle sur des caractères chinois en utilisant comme voisinage, le pixel supérieur et le pixel gauche. Une pré-classification a été faite avec le nombre de traits et des primitives de type cellulaire ont été utilisées. Un taux de reconnaissance de 88,1% est annoncé sur une base de données non publique.

Dans un article de synthèse, Park et Lee [Park et Lee, 1998] décrivent leur système de reconnaissance de caractères (latins) en exposant la genèse de ce modèle et en détaillant les algorithmes d'apprentissage et de reconnaissance. La dépendance locale est ici limitée aux trois pixels en haut et à gauche, et le taux de reconnaissance annoncé est de 90,8% sur une base de données de caractères de l'université de Concordia.

2. Les champs de Markov unilatéraux

   L'originalité de ce modèle réside dans l'utilisation d'un voisinage qui peut être de structure variable et de taille importante ainsi que dans de la dépendance locale qui se situe au niveau des observations plutôt qu'au niveau des états cachés. Cette modélisation s'adapte donc mieux à une situation où les observables sont quantifiés sur un petit nombre de valeurs, typiquement la valeur brute du pixel codée sur 1 bit.

   Ces modèles ont surtout été étudiés au LORIA par A. Belaïd [Belaïd et Saon, 1997, Choisy et Belaïd, 2000,Saon et Belaïd, 1997] avec d'excellents résultats sur les bases de données standard (cf. section 2.2). L'approche retenue consistait à modéliser l'écriture par des champs de Markov au niveau des lettres puis par des chaînes de Markov pour le niveau des mots.

Quelques rares tentatives d'utilisation d'un modèle de champ de Markov non causal peuvent être recensées, on peut ainsi noter le travail de Xiong *et al.* [Xiong *et al.*, 2001] qui utilisent les champs de Markov sans hypothèse de causalité et les décodent avec un

algorithme d'ICM (cf. section 3.3). Les primitives sont ici de type cellulaire et les tests sont effectués sur un vocabulaire de 50 paires de caractères chinois très semblables deux à deux.

Le tableau 2.1 résume les différences entre les principaux types de modélisations statistiques de l'écriture.

Tab. 2.1 : Modélisations statistiques en traitement de l'écriture manuscrite.

	Dépendance locale	Décodage	Références
Chaînes de Markov	1D	programmation dynamique	[Leroux *et al.*, 1997], [Grandidier *et al.*, 1999]
Champs de Markov pseudo-2D	deux fois 1D	programmation dynamique	[Levin et Pieraccini, 1992], [Gilloux, 1994b]
Réseaux de Markov	2D causale	programmation dynamique	[Gilloux, 1994b], [Park et Lee, 1998]
Champs de Markov unilatéraux	2D causale variable	programmation dynamique	[Choisy et Belaïd, 2000], [Saon et Belaïd, 1997]
Champs de Markov non causaux	2D (4 voisins)	ICM	[Xiong *et al.*, 2001]

2.1.5 Récapitulatif des techniques utilisées en reconnaissance de l'écriture manuscrite

Le tableau 2.2 résume, pour chacune des grandes approches, les principes mis en œuvre, leurs avantages et leurs limitations.

TAB. 2.2 : Récapitulatif des techniques utilisées en reconnaissance de l'écriture manuscrite

Approches	Principes	Avantages	Limitations	Références
géométriques	Moments / distributions de primitives géométriques déterministes	Simplicité et rapidité de mise en œuvre, compacité	Un unique modèle par classe, globabilité de l'approche	[Trier *et al.*, 1996]
statistiques 1D	Chaînes de Markov et programmation dynamique	Prise en compte de la dépendance locale, algorithmes performants d'analyse 1D	Structure 2D de l'écriture non modélisée	[Leroux *et al.*, 1997], [Grandidier *et al.*, 1999]
statistiques pseudo-2D	Doubles chaînes de Markov	Vers une représentation 2D de l'écriture	Hypothèse d'indépendance directionnelle	[Levin et Pieraccini, 1992], [Gilloux, 1994b], [Belaïd et Saon, 1997]
statistiques 2D causales	Champs markoviens causaux	Modélisation 2D de l'écriture	Hypothèse de causalité	[Belaïd et Saon, 1997], [Gilloux, 1994b]
statistiques 2D	Champs markoviens	Modélisation 2D de l'écriture	Utilisation de l'algorithme d'ICM	[Xiong *et al.*, 2001]

2.2 Performances des algorithmes sur les bases de données standard de texte manuscrit

Les recherches dans le domaine de la reconnaissance de l'écriture nécessitent d'évaluer les résultats sur des bases de données représentatives des problèmes réels. Si de très nombreux travaux sont appliqués sur des bases constituées pour l'occasion mais pas rendues publiques, il existe plusieurs bases disponibles et utilisées par une partie de la communauté. Ces bases sont représentatives de problèmes réels : reconnaissance de caractères isolés (caractères latins, chinois etc.), de mots isolés ou de documents.

2.2.1 Les bases de données publiques

Base du CEDAR

La base de données du CEDAR[1] contient des images de noms de villes et d'états, de codes postaux et de caractères isolés [Hull, 1993]. Elle est divisée en plusieurs parties décrites dans le tableau 2.3.

Tab. 2.3 : Composition de la base du CEDAR.

Nom	Nombre d'images	Contenu
BB	300	adresses qui présentent des problèmes particuliers (Caractères qui se touchent, mauvais formats de codes postaux...)
BC	211	images d'un même code postal
BD	2 636	adresses
BL	973	images d'un même nom de ville avec les codes postaux
BS	500	tests pour la base BD
BR	3 962	codes postaux
BU	1 072	codes postaux avec chiffres connectés

La base est décrite en détails sur le site du CEDAR [CEDAR, 1993] qui propose aussi des utilitaires pour son utilisation.

Bases du SRTP

Le SRTP[2] utilise ses propres bases de données pour ses tests, mais celles-ci ne sont pas disponibles. De nombreux travaux utilisent cependant ces bases d'images de chèques ou d'adresses pour l'évaluation de leurs algorithmes.

Base du NIST

La base de données du NIST[3] est composée d'images extraites de formulaires de recensement. Elle contient à la fois des caractères isolés et du texte écrits par 3.600 scripteurs

[1]CEDAR : Center of Excellence for Document Analysis and Recognition (State University of New York at Buffalo)

[2]SRTP : Service de Recherches Techniques de la Poste

[3]NIST : National Institute of Standards and Technology

pour un total de 800.000 images de résolution 300dpi. La dernière version, décrite sur le site [NIST, 1995], est la *NIST Special Database 19* qui fait suite aux versions 3 et 7.

➪ Base MNIST

La base de données MNIST [LeCun, 1998] est composée de chiffres manuscrits extraits de la base NIST et déjà pré-traités :

- images codées sur 8 bits,
- images de taille fixe,
- caractères normalisés,
- caractères centrés,
- ensemble d'apprentissage : 60 000 images,
- ensemble de test : 10 000 images.

Cette base est gratuite et disponible sur le site [LeCun, 1998]. Ce site donne aussi les résultats d'évaluation de plusieurs techniques classiques de reconnaissance de formes sur cette base. Quelques échantillons sont présentés Figure 2.2. Nous utiliserons cette base pour étudier le comportement de nos algorithmes sur une tâche de reconnaissance de caractères isolés (cf. chapitre 4.1).

➪ Base TNIST

Cette base contient exclusivement des caractères connectés et a été construite pour évaluer des algorithmes sur ce problème [Zhou *et al.*, 2000]. Elle a été extraite de NIST SD 19 et contient 8500 images de deux caractères connectés.

Base du CENPARMI

La base de données du CENPARMI [4] contient des images de chèques canadiens en anglais et en français [Côté, 1997]. Son contenu est indiqué dans le tableau 2.4.

Tab. 2.4 : Caractéristiques de la base du CENPARMI.

anglais	2500 chèques	800 scripteurs
français	1900 chèques	600 scripteurs

Base de Senior & Robinson

La base créée par A.W. Senior est un texte de 22 pages écrites par un seul scripteur. Une partie du texte (qui contient 2000 mots) est écrite uniquement en minuscules. Nous utiliserons cette base pour étudier le comportement de nos algorithmes sur une tâche de reconnaissance de mots (cf. chapitre 4.2).

[4] CENPARMI : Centre for Pattern Recognition and Machine Intelligence

Bases ETL

Les bases de données de caractères ETL ont été construites au laboratoire d'électrotechnique (ETL) de l'AIST[5] en collaboration avec des universités et industriels. Les bases de données ETL1 - ETL9 contiennent environ 1,2 million de caractères manuscrits et imprimés répartis en caractères japonais, chinois, latins et numériques. Elle est gratuite pour des applications de recherche. Quelques échantillons sont présentés Figure 2.3 et les détails peuvent être obtenus sur le site d'ETL [ETL, 2001].

Les contenus des différentes bases sont synthétisés dans le tableau 2.5.

2.2.2 Tableau des performances

Le tableau 2.6 donne des résultats de tests effectués sur des bases de données standard. La variabilité dans les choix des ensembles de test et d'apprentissage et dans les post-traitements utilisés incite à comparer les résultats avec prudence. La colonne *Protocole* indique si on se place au niveau du caractère (après ou avant segmentation) ou au niveau du mot (en précisant éventuellement la taille du lexique). Les performances en termes de phrases, montants ou adresses ainsi que la technique utilisée sont données en commentaires.

En conclusion, la figure 2.4 situe, par rapport à l'état de l'art, notre approche présentée dans le chapitre suivant.

[5]AIST : Advanced Industrial Science and Technology

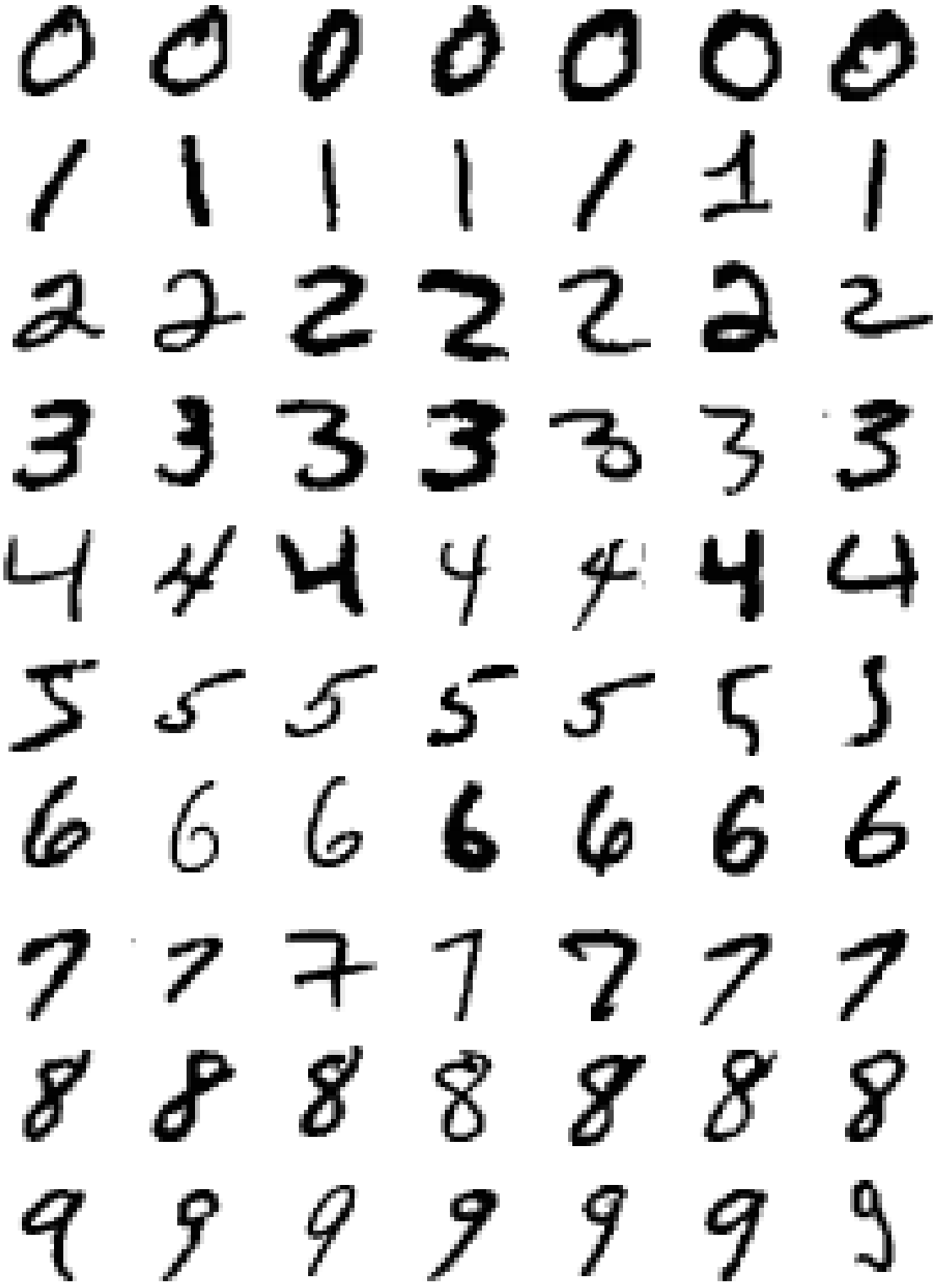


Fig. 2.2 : Échantillons de la base MNIST.

FIG. 2.3 : Échantillons de la base ETL.

Tab. 2.5 : Caractéristiques des bases ETL.

Nom	**Types de caractères**	**Nombre d'échantillons**	**Manuscrit-Imprimé**
ETL1	numériques latins spéciaux katakana	141 319	M
ETL2	kanji hiragana katakana alphanumériques spéciaux	52 796	I
ETL3	numériques latins spéciaux	9 600	M
ETL4	hiragana	6 120	M
ETL5	katakana	10 608	M
ETL6	katakana numériques latins spéciaux	157 662	M
ETL7L	hiragana	16 800	M (grands caractères)
ETL7S	hiragana	16 800	M (petits caractères)
ETL8G	kanji hiragana	152 960	M
ETL8B2	kanji hiragana	152 960	M (binaires)
ETL9G	kanji hiragana	607 200	M
ETL9B	kanji hiragana	607 200	M (binaires)

TAB. 2.6 : Bases de données et performances.

Base	Sous-base	Protocole	Perf.	Ref.	Commentaires
CEDAR (950$)	BR	C ap seg.	98,9 %	[Favata *et al.*, 1994]	
			98,8 %	[Xue et Govindaraju, 2000]	
			96,1 %	[Cha *et al.*, 1999]	
	BR + BS	C ap seg.	96,4 %	[Lecce *et al.*, 2000]	
	BD + BS	M lex : 10	88,9 %	[Grandidier *et al.*, 1999]	HMM
		M lex : 100	89,3 %	[Mohamed et Gader, 1996]	HMM+DP
			83,9 %	[Mohamed et Gader, 1996]	DP
			78,9 %	[Mohamed et Gader, 1996]	HMM
			75,8 %	[Grandidier *et al.*, 1999]	HMM
		M lex : 1000	56 %	[Grandidier *et al.*, 1999]	HMM
SRTP (non diffusée)	Adresses postales (villes)	M lex : 10	99,2 %	[El-Yacoubi *et al.*, 1999]	HMM
			98,9 %	[Grandidier *et al.*, 2000]	HMM
		M lex : 100	96,3 %	[El-Yacoubi *et al.*, 1999]	HMM
			95,8 %	[Grandidier *et al.*, 2000]	HMM
		M lex : 1000	88,9 %	[El-Yacoubi *et al.*, 1999]	HMM
			88,7 %	[Grandidier *et al.*, 2000]	HMM
	Chèques	M	90,1%	[Saon et Belaïd, 1997]	montants : 79,5%
			86,6 %	[Simon *et al.*, 1994]	montants : 74,5%
			83,7 %	[Gilloux *et al.*, 1995]	
			83,4 %	[Choisy et Belaïd, 2000]	peu de paramètres
			83 %	[Leroux *et al.*, 1997]	montants : 70 %
			82,8 %	[Saon *et al.*, 1995]	
NIST (90$)	NIST SD 19	chif av seg.	96,3 %	[Garris, 1996]	
		maj av seg.	86,2 %	[Garris, 1996]	
		min av seg.	76,6 %	[Garris, 1996]	
	NIST SD 3	C ap seg.	99,6 %	[Atukorale et Suganthan, 1999]	
			97,6 %	[Alpaydin et Gurgen, 1996]	k-NN
			96,4 %	[Alpaydin et Gurgen, 1996]	LVQ
		C ap seg.	95.0 %	[Casey et Takahashi, 1992]	
	TNIST - ND	C av seg.	88,6 %	[Zhou *et al.*, 2000]	
	MNIST gratuite	ap seg.	99,3 %	[LeCun, 1998]	LeNet 4.
			98,9 %	[LeCun, 1998]	K-NN
			98,4 %	[LeCun, 1998]	2 layer NN
ETL (gratuite)	ETL8	956 classes	97,1 %	[Tsukumo et Tanaka, 1988]	80 train/80 test
		952 classes	96,1 %	[Oka, 1982]	10 train/10 test
		881 classes	94,8 %	[Yamashita *et al.*, 1983]	40 train/20 test
		881 classes	91,1 %	[Kobayashi *et al.*, 1983]	80 train/20 test
		956 classes	89,1 %	[Hyman *et al.*, 1991]	40 train/40 test
	ETL9	3036 classes	98,5 %	[Yamamoto *et al.*, 1986]	10 train/1 test
		2965 classes	94,4 %	[Tsukumo et Tanaka, 1988]	100 train/1000 test
CENPARMI (non diffusée)		M lex : 32	73,6 %	[Côté, 1997]	
			72,6 %	[Guillevic, 1995]	
	IRIS-Bell'98	C av seg.	65,2 %	[Zhou *et al.*, 2000]	
Senior & Robinson (gratuite)		M lex : 30 000	87 %	[Senior et Robinson, 1998]	
		M	85 %	[Marti et Bunke, 1999]	bi-grames
		M lex : 1334	83,6 %	[Vincieralli et Luettin, 2000]	

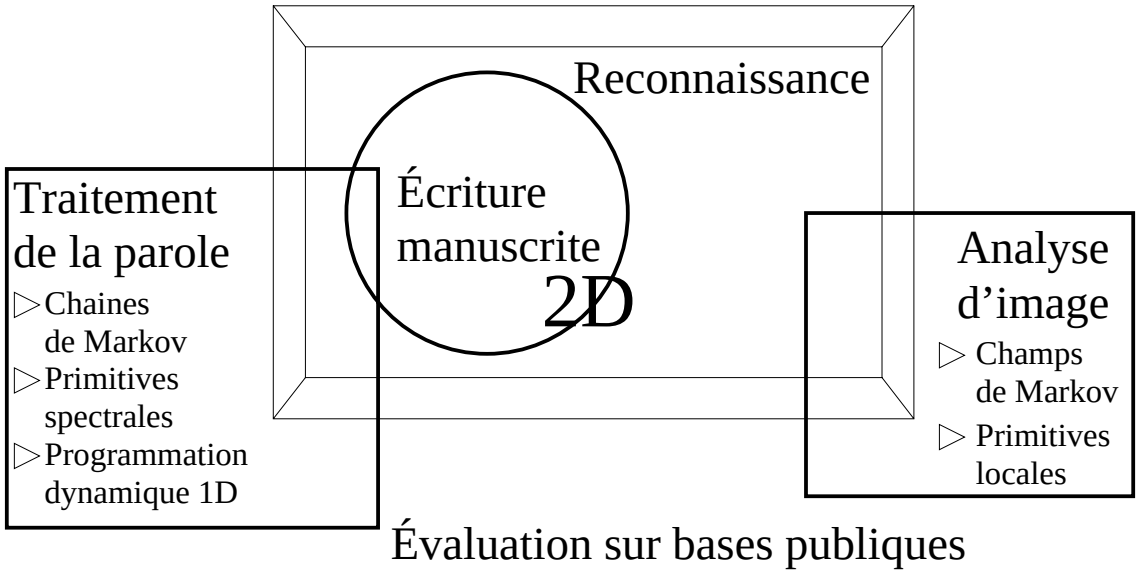


FIG. 2.4 : Positionnement de notre approche dans l'état de l'art.

Chapitre 3

Approche proposée

Une observation parallèle des techniques et méthodologies couramment utilisées pour l'analyse de signaux temporels, notamment de parole, et de leur équivalent pour l'analyse des images montre que la plupart de ces techniques existent et sont, plus ou moins fréquemment, utilisées (cf. tableau 3.1). La seule exception est l'algorithme de décodage du modèle de Markov caché, la programmation dynamique, appelée également algorithme de Viterbi ou algorithme $A*$. Jusqu'à récemment, il n'existait pas d'équivalent bidimensionnel de la programmation dynamique.

Tab. 3.1 : Parallèle entre les méthodologies 1D et 2D.

	1D	2D
Méthodologie	Évaluations régulières, bases publiques avec vérité terrain.	Peu de bases publiques (caractères, visages, suivi de véhicules...).
Analyse du signal brut	Fenêtres glissantes douces et recouvrantes (hamming...)	Fenêtres glissantes 2D
Extraction de primitives	Analyse spectrale (banc de filtres, cepstres...)	Analyse spectrale 2D (Gabor, cepstre 2D...)
Approche de la reconnaissance	Stratégie bayésienne	Stratégie bayésienne
Modélisation	Chaîne de Markov cachée	Champ de Markov caché
Attache aux données	Densités multigaussiennes	Densités multigaussiennes
Algorithme de décodage	Programmation dynamique	?

Dans ce chapitre, nous exposerons en premier lieu le principe général de notre stratégie. Cette approche du problème étant proche de la méthodologie standard du traitement de la parole, nous proposons ensuite un bref aperçu de ces techniques en insistant sur les points

communs. Nous évoquerons ensuite la modélisation par champs de Markov en traitement d'image ainsi que les algorithmes classiques qui lui sont associés. Enfin, nous expliquerons en détails le principe de fonctionnement de l'algorithme de programmation dynamique 2D, récemment proposé, et qui est au cœur de notre système.

3.1 Principe général

L'approche du problème de la reconnaissance d'écriture que nous proposons s'appuie sur les principes résumés dans le tableau 3.1, c'est à dire :

- une méthodologie de développement fondée sur l'évaluation systématique de nos propositions,
- une analyse fenêtrée du signal,
- une stratégie bayésienne pour la reconnaissance,
- une modélisation de l'écriture par champs de Markov cachés,
- une modélisation des observables par densités multigaussiennes,
- un décodage des champs de Markov par programmation dynamique bidimensionnelle.

La finalité de notre algorithme est la reconnaissance, effectuée de façon bayésienne : si on observe O, on cherche à savoir par lequel des modèles m_k cette observation a été émise. Il nous faut pour cela maximiser sa probabilité conditionnelle :

$$k_{max} = \arg\max_k P(m_k|O) = \arg\max_k \frac{P(O|m_k)P(m_k)}{P(O)} = \arg\max_k P(O|m_k)P(m_k). \quad (3.1)$$

Les problèmes principaux à résoudre sont donc :

1. faire le choix de l'observable O, à partir des images,
2. disposer d'un algorithme permettant de calculer les probabilités $P(O|m_k)$ à partir d'un modèle et d'une observation,
3. établir une stratégie pour réaliser l'apprentissage des modèles m_k.

Les choix effectués pour résoudre ces trois problèmes de façon robuste sont les suivants :

1. réaliser une analyse fenêtrée, qui à partir d'une image de taille $M \times N$ donne une matrice $m \times n$ de vecteurs d'observables (avec $m < M$ et $n < N$).
2. spécifier une modélisation par champs de Markov,
3. mettre en œuvre une approche EM (Espectation Maximisation, cf. section 3.2.4) avec un critère de maximum de la probabilité *a posteriori*.

La modélisation markovienne permet d'utiliser ce principe également dans un but de segmentation. L'annexe C montre comment ce principe a été adapté, avec succès, à une tâche de reconnaissance de route lors de cette thèse.

3.2 Un aperçu des techniques de reconnaissance de la parole

Le domaine de la reconnaissance de la parole a connu et connaît encore de grands progrès qui ont permis la création de systèmes opérationnels et performants pour de nombreuses applications comme la commande vocale, l'identification de la langue ou du locuteur ou la transcription automatique. Cette dernière regroupe des tâches de complexité très variable comme la dictée vocale mono-locuteur ou la transcription automatique d'émissions de radio et télévision. Le cœur de ces systèmes aux performances au niveau de l'état de l'art utilisent des modélisations (statistiques) et des algorithmes (à base de programmation dynamique) très similaires. Ainsi, les approches retenues en traitement de la parole ont-elles été adaptées au traitement de l'écriture manuscrite, en particulier en assimilant la dimension temporelle du signal de parole à la dimension horizontale de l'écriture. Les seules réelles différences entre ces algorithmes sont liées aux processus d'extraction des primitives. Dans cette partie, on décrira ces techniques, de la phase d'extraction de primitives jusqu'à la reconnaissance proprement dite. On trouvera une description plus complète de ces techniques dans les ouvrages [Boite *et al.*, 1999] et [Huang *et al.*, 2001].

3.2.1 Systèmes de reconnaissance

Modèles acoustiques

Un choix important pour un système de reconnaissance est la sélection du type de modèle acoustique. Pour une petite taille de vocabulaire à reconnaître, on peut choisir simplement de modéliser l'ensemble des mots du vocabulaire. Faire des modèles de mots différents pour chaque contexte (mot précédent et mot suivant) est aussi possible pour tenir compte de la coarticulation, mais le nombre de modèles augmente exponentiellement avec la taille du vocabulaire et l'ajout de nouveaux mots à un vocabulaire existant est problématique.

Pour modéliser tous les mots avec un lexique réduit, on utilise les phones, sons unitaires proches de la notion de phonème, avec lesquels on peut prononcer l'ensemble du vocabulaire d'une langue. Typiquement, pour l'anglais, on utilise une cinquantaine de phones différents (cf tableau 3.2).

L'effet de coarticulation entraînant une grande variabilité dans la prononciation des phonèmes, l'utilisation d'un seul modèle par phone limite donc rapidement les performances. La solution est d'utiliser des phones en contexte (PIC) qui tiennent compte du phone précédent et du phone suivant. On parle alors de triphones. L'ensemble des triphones possibles est immense (supérieur à 100.000), mais plusieurs contextes différents peuvent induire des prononciations très semblables du phone. Il est donc possible de regrouper (clusteriser) ces triphones suivant la distance entre les phones pris dans différents contextes.

En pratique, on modélise les phones par des HMM à trois états, l'état initial et l'état final portant respectivement l'influence du phone précédent et du phone suivant. On peut

Tab. 3.2 : Alphabet phonétique ARPABET.

n	Ab	Examples	n	Ab	Examples	n	Ab	Examples
1	iy	b**ea**t	22	r	**r**ed	43	zh	mea**s**ure
2	ih	b**i**t	23	y	**y**et	44	sh	**sh**oe
3	eh	b**e**t	24	w	**w**et	45	v	**v**ery
4	ae	b**a**t	25	m	**m**om	46	f	**f**ief
5	ix	ros**e**s	26	em	butt**om**	47	dh	**th**ey
6	ax	th**e**	27	n	**n**on	48	th	**th**ief
7	ah	b**u**tt	28	nx	(flapped) n	49	hh	**h**ay
8	uw	b**oo**t	29	en	butto**n**	50	hv	Le**h**eigh
9	uh	b**oo**k	30	ng	si**ng**	51	dcl	(d closure)
10	ao	ab**ou**t	31	eng	Washi**ng**ton	52	bcl	(b closure)
11	aa	c**o**t	32	ch	**ch**urch	53	gcl	(g closure)
12	er	b**ir**d	33	jh	**j**udge	54	tcl	(t closure)
13	axr	din**er**	34	b	**b**ob	55	pcl	(p closure)
14	ey	b**ai**t	35	p	**p**op	56	kcl	(k closure)
15	ay	b**i**te	36	d	**d**ad	57	q	(glottal stop)
16	oy	b**oy**	37	dx	bu**tt**er	58	epi	(epi closure)
17	aw	b**ou**ght	38	t	**t**ot	59	qcl	(d closure)
18	ow	b**oa**t	39	g	**g**ag	60	h#	beg. sil.
19	ux	b**eau**ty	40	k	**k**ick	61	#h	end sil.
20	l	**l**ed	41	z	**z**oo	62	pau	betwe. sil.
21	el	bott**le**	42	s	**s**is			

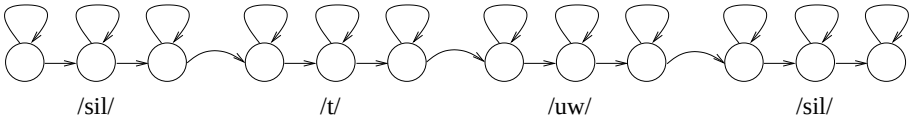


FIG. 3.1 : HMM composé pour le mot *two*.

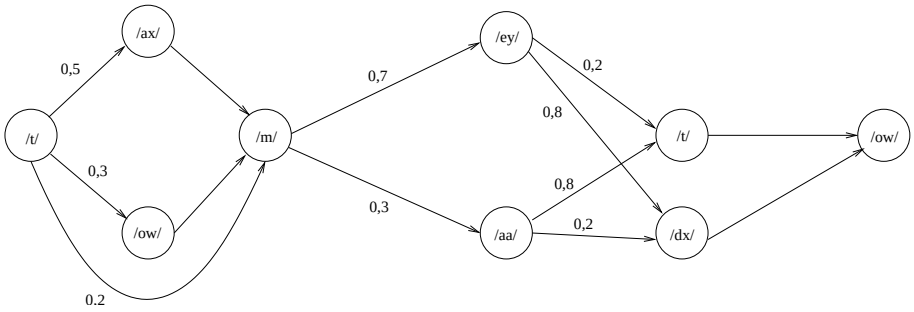


FIG. 3.2 : Prononciation du mot *tomato*.

ainsi clusteriser les états et non plus les triphones entiers, on obtiendra ainsi une meilleure précision car deux triphones ne différant que d'un état seront distingués et non plus regroupés en un unique modèle.

La modélisation acoustique construit donc des HMM à trois états pour chaque phone qui, par concaténation, donneront les HMM des mots complets grâce au dictionnaire (cf. Figure 3.1). L'apprentissage est réalisé par l'algorithme de Baum-Welch ou l'algorithme de Viterbi (cf section 3.2.4) qui effectuent l'alignement des phonèmes.

Lexique de prononciation

Un lexique de prononciation associe à chaque mot du vocabulaire sa prononciation sous la forme de suite de phonèmes avec éventuellement des variantes de prononciation. Il faut donc en tenir compte lors dès la réalisation du dictionnaire. La définition de la prononciation peut aussi se faire grâce à des machines à états finis, très proches des chaînes de Markov. On peut par exemple modéliser les prononciations anglaise et américaine du mot *tomato* par la machine à états finis de la figure 3.2.

Le phénomène de coarticulation (la prononciation d'un mot peut dépendre des mots précédents et suivants) et les variations suivant le contexte ou le locuteur (accents régionaux ou nationaux) sont surtout pris en compte au niveau de la modélisation acoustique décrite au paragraphe 3.2.1.

Comme pour le HMM, l'apprentissage doit se faire sur un important corpus de données pour s'adapter à toutes les prononciations possibles et les poids des transitions sont estimés à partir de la fréquence d'occurrence de ces prononciations.

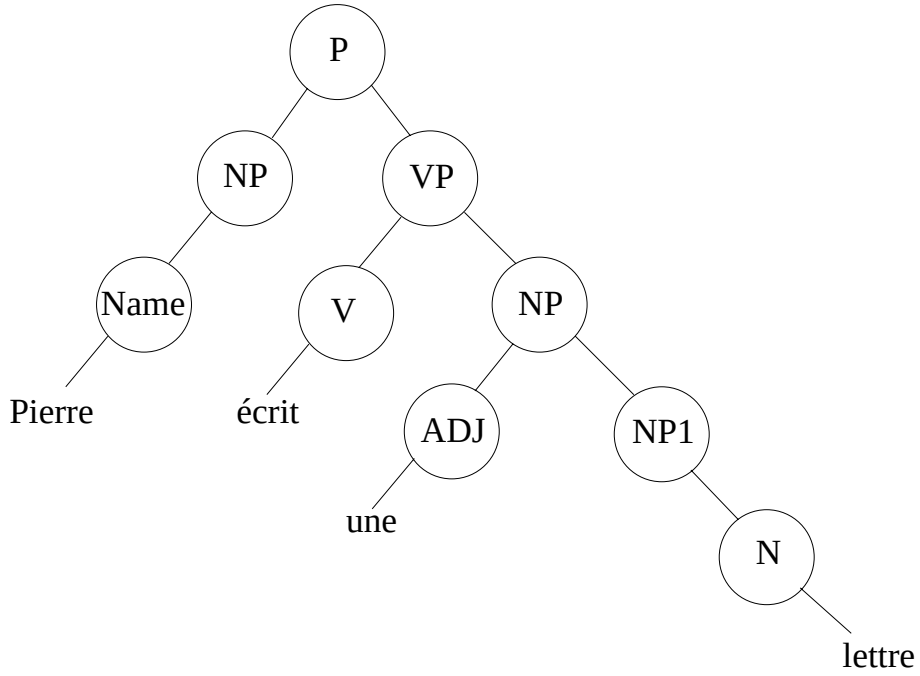


FIG. 3.3 : Représentation en arbre d'une phrase.

Modèles de langage

Une tâche de reconnaissance ne peut être conduite avec des données acoustiques seules. Un minimum de connaissance *a priori* sur le résultat est nécessaire. L'utilisation d'un lexique prédéfini permet d'obtenir uniquement des mots de la langue utilisée mais l'ordonnancement peut être quelconque. Il est donc important de modéliser les règles du langage. En pratique les modèles de langages et acoustiques sont très liés et les deux informations sont utilisées simultanément par les algorithmes de recherche. Cette modélisation peut se faire :

- en construisant une grammaire formelle qui contient les règles usuelles et dispose d'algorithmes pour vérifier sa compatibilité avec une phrase,
- ou en calculant un modèle statistique d'apparition des mots.

Typiquement, la première approche utilise souvent la grammaire formelle de Chomsky, la deuxième est généralement mise en œuvre à partir de N-grammes.

- Modélisation du langage par grammaire formelle
  La construction d'une grammaire formelle permet de déterminer si une phrase est compatible ou non avec cette grammaire, sans prendre en compte la fréquence d'apparition de cette phrase en pratique et sans permettre de légères entorses à cette grammaire comme c'est couramment le cas dans le langage parlé.
  Cette modélisation s'exprime sous la forme d'un arbre dont les feuilles sont les mots de la phrase et les noeuds modélisent la fonction grammaticale et les associations que la grammaire autorise (voir fig 3.3).
  Les règles sont définies de la façon suivante : on donne pour une séquence gramma-

ticale les décompositions possibles en séquences plus petites. Dans l'exemple de la figure 3.3, on a $VP \rightarrow V\ NP$ ou $NP \rightarrow Name$ mais pas $NP \rightarrow NP\ V$ [Young et Bloothooft, 1997].
Deux approches sont possibles selon que l'on parte de la phrase pour en déduire sa validité non ou que l'on parte de la racine de l'arbre pour tenter d'en déduire la phrase via les règles de la grammaire. Les redondances dans l'exploration de l'arbre sont typiquement celles que l'on évite grâce à la programmation dynamique. On a donc pour cela des algorithmes similaires à ceux des HMM décrits dans la section 3.2.4 [Huang *et al.*, 2001, p 549].

– Modélisation stochastique du langage
L'approche formelle de la modélisation du langage est trop rigide pour être utilisée pour du langage parlé qui peut être en revanche efficacement décrit par une approche statistique. Le principe est d'évaluer les probabilités d'apparition des séquences de mots $P(w_1 w_2 ... w_n)$ dans le langage :

$$P(w_1 w_2 ... w_n) = P(w_1)P(w_2|w_1)P(w_3|w_1 w_2)P(w_4|w_1 w_2 w_3)...P(w_n|w_1 w_2 ... w_{n-1}). \tag{3.2}$$

Un traitement exhaustif de toutes ces chaînes dans tous les contextes serait prohibitif. On utilise donc en pratique l'évaluation des N-grammes qui explicitent l'approximation markovienne d'ordre $N-1$ en réduisant la dépendance contextuelle aux $N-1$ mots précédents. Ainsi pour le cas des trigrammes, on fait l'hypothèse :

$$P(w_n|w_1 w_2 ... w_{n-1}) = P(w_n|w_{n-2} w_{n-1}), \tag{3.3}$$

et on a alors :

$$P(w_1 w_2 ... w_n) = P(w_1)P(w_2|w_1)P(w_3|w_1 w_2)P(w_4|w_2 w_3)...P(w_n|w_{n-2} w_{n-1}). \tag{3.4}$$

L'apprentissage de tous les trigrammes n'est pas non plus possible en pratique pour des applications nécessitant un vocabulaire important (V^3 trigrammes pour un vocabulaire de taille V). On effectue donc les simplifications suivantes :
  – on ne calcule que les probabilités des trigrammes les plus courants et on assigne le reste des probabilités aux autres,
  – on calcule les trigrammes peu fréquents en multipliant le bigramme correspondant par un coefficient ne dépendant que du contexte :

$$\hat{P}(w_k|w_{k-2} w_{k-1}) = B(w_{k-2}, w_{k-1})P(w_k|w_{k-1}). \tag{3.5}$$

– Modélisation formelle et statistique du langage
Chacune des deux approches a ses propres avantages et inconvénients et est plus ou moins adaptée aux différentes applications. La grammaire formelle nécessite peu de données d'apprentissage et permet d'obtenir des phrases correctes grammaticalement, mais engendre de nombreuses erreurs de transcription pour des applications peu contraintes. La modélisation statistique nécessitera un très grand nombre de données d'apprentissage pour faire peu d'erreurs mais sera beaucoup plus tolérante.

Il est possible d'adopter une approche médiane en introduisant l'idée de probabilité d'une phrase ou d'une séquence de mots dans la grammaire formelle : une phrase n'est ainsi plus simplement acceptée ou rejetée.

Décodage

Le décodage est l'opération qui recherche la séquence de mots la plus probable compte tenu de l'observation. Sa complexité dépend directement de la taille de l'espace de recherche qui est déterminée principalement par le modèle de langage. Les techniques de décodage sont fondées sur l'algorithme de Viterbi et sur des techniques d'élagage que l'on retrouve dans la recherche en faisceau de Viterbi (cf. section 3.2.4).

On définit un coût de la transcription qui diminue quand la probabilité de la transcription W connaissant l'observation acoustique X, $P(W|X)$, augmente :

$$C(W|X) = -\log[P(W)P(X|W)]. \tag{3.6}$$

Ce choix permet :
- d'éviter de manipuler des probabilités qui deviennent rapidement inférieures à la précision de la machine,
- de calculer des additions plutôt que des multiplications.

On obtient ainsi un coût que l'on peut interpréter en terme d'énergie ou de distance.

Le modèle acoustique donne la probabilité $P(X|W)$ et le modèle de langage la probabilité $P(W)$. On pourra éventuellement pondérer l'influence de ces deux probabilités (la partie acoustique étant souvent sous-estimée) en élevant $P(W)$ à une puissance donnée déterminée à partir des données de développement.

Le cas de la reconnaissance de mots isolés se résout facilement en évaluant la probabilité de l'observation pour chaque HMM (les HMM de mots étant éventuellement construits par concaténation de HMM de phonèmes ou de syllabes) et en choisissant le modèle qui maximise cette probabilité.

En ce qui concerne le cas plus complexe de la reconnaissance de la parole continue, on évalue les performances d'une reconnaissance sur toutes les segmentations possibles pour ne retenir que la meilleure, qui donne le modèle le plus probable et la séquence de mots reconnue.

3.2.2 L'extraction des primitives pour le traitement de la parole

Le processus d'extraction des primitives dans un signal de parole répond aux trois objectifs suivants :
- extraire les caractéristiques pertinentes du signal,
- réduire l'influence du bruit,
- décorréler les données,
- diminuer le volume des données.

Ces caractéristiques sont en grande partie dans le domaine fréquentiel, c'est pourquoi une analyse spectrale est nécessaire. Si on garde autant de coefficients que d'échantillons

dans la fenêtre, ces représentations sont équivalentes, et on peut passer de l'une à l'autre par des transformations simples. L'intérêt étant de ne garder que les coefficients les plus discriminants, ces traitements ne sont en pratique pas équivalents. Le signal de parole n'étant pas stationnaire, on réalise une analyse fenêtrée du signal (analyse à court terme). On détermine les vecteurs de primitives en découpant le signal en fenêtres avant d'extraire les coefficients sur chacune d'elles. En général, on utilise des fenêtre de Hamming de 20 à 30 ms toutes les 10ms de signal.

Les bancs de filtres

Une première approche d'extraction de caractéristiques dans un signal de parole consiste à réaliser un banc de filtres pour extraire les coefficients de Fourier. L'analyse de Fourier à court-terme ainsi effectuée permet d'estimer la densité spectrale de puissance à chaque instant.

Le codage linéaire prédictif

Une autre méthode simple et efficace pour déterminer les caractéristiques d'un signal comme un signal de parole est le codage linéaire prédictif (LPC en anglais ou modélisation auto-régressive AR) où l'on cherche à estimer la valeur d'un signal x à un instant n en fonction de sa valeur aux instants précédents. Une prédiction d'ordre p détermine les p coefficients a_k tels que le signal

$$\tilde{x}[n] = \sum_{k=1}^{p} a_k x[n-k], \tag{3.7}$$

soit le plus proche possible de x pour un certain critère (comme l'écart quadratique moyen). Plusieurs algorithmes, notamment fondés sur la covariance ou l'autocorrélation du signal, existent pour déterminer ces coefficients.

On peut utiliser une échelle de fréquence non linéaire adaptée à la perception de l'oreille humaine, on calcule alors la prédiction linéaire perceptuelle (PLP).

Le cepstre

Le calcul du cepstre réalise une analyse fréquentielle du signal avant de repasser dans le domaine temporel : Le principe est de prendre le logarithme du module de la transformée de Fourier du signal (le *spectre*) et d'en faire la transformée de Fourier inverse (d'où le nom *cepstre*). L'intérêt de ce traitement est de déconvoluer la fonction de transfert du conduit vocal de l'excitation, qui porte l'information utile. Les coefficients du *cepstre* sont donc définis par

$$c[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} \ln |X(e^{j\omega})| e^{j\omega n} d\omega, \tag{3.8}$$

où $X(e^{j\omega})$ est la transformée de Fourier du signal de parole. La variable n, homogène à un temps, peut être de la même façon que le cepstre désignée par *quéfrence*.

Il est plus efficace d'associer les coefficients cepstraux à une échelle perceptuelle des fréquences, la plus courante aujourd'hui étant l'échelle Mel. Les fréquences de l'échelle Mel sont déduites de la fréquence f par la relation

$$B(f) = 1125 \ln(1 + f/700). \tag{3.9}$$

Ces coefficients sont les Mel-Frequency Cepstral Coefficients (MFCC).

Les vecteurs de primitives pour la reconnaissance de la parole sont typiquement composés des coefficients c_k du traitement MFCC jusqu'au 13^{e} ordre, le coefficient c_0 étant généralement remplacé par l'énergie de la trame de signal.

Afin de rendre mieux compte de la dynamique du signal de parole, on utilise généralement aussi les dérivées des coefficients MFCC. On calcule ces dérivées sur les deux ou quatre échantillons autour de l'échantillon courant, soit

$$\Delta c_k = c_{k+1} - c_{k-1}, \tag{3.10}$$

ou

$$\Delta c_k = c_{k+2} - c_{k-2}, \tag{3.11}$$

ou bien encore

$$\Delta c_k = 2c_{k+2} + c_{k+1} - c_{k-1} - 2c_{k-2}. \tag{3.12}$$

On utilise ces coefficients aussi jusqu'au 13^{e} ordre, et on ajoute enfin la notion d'accélération, elle aussi jusqu'au 13^{e} ordre :

$$\Delta\Delta c_k = \Delta c_{k+1} - \Delta c_{k-1}. \tag{3.13}$$

On obtient finalement un vecteur de primitives de 39 composantes toutes les 10 ms. La plupart des systèmes aux performances du niveau de celles de l'état de l'art utilisent ce vecteur de primitives dont la pertinence est donc déjà largement démontrée.

Le calcul du cepstre ne comporte que des opérations aisément généralisables au cas 2D : les transformations de Fourier et logarithme du module. On peut aussi découper l'image en imagettes grâce à une fenêtre de Hamming 2D. On pourrait ainsi modéliser la présence de traits dans la fenêtre et leur direction. Ce traitement peut être rapproché des classiques filtres de Gabor qui font une analyse fréquentielle sur une fenêtre gaussienne de l'image [Hamamoto *et al.*, 2001, Huo *et al.*, 2001].

3.2.3 Les chaînes de Markov cachées

À la base de la plupart des modélisations statistiques actuelles des signaux de parole, les modèles de Markov interviennent à tous les niveaux d'observation (phonème, mot, langage).

Un modèle de Markov caché est un processus doublement stochastique dans le sens où un premier processus aléatoire modélise l'état du système, lui même observé au moyen

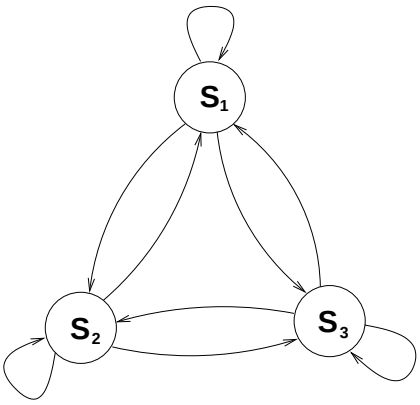


FIG. 3.4 : Chaîne de Markov.

d'une émission aléatoire. La succession des états du système est modélisée par une chaîne de Markov.

Les chaînes de Markov

Une chaîne de Markov est un ensemble d'états reliés par des transitions. Dans une chaîne de Markov d'ordre 1, la probabilité de se trouver en un état ne dépend que de l'état précédent :

$$P(s_i|s_1 s_2 ... s_{i-1}) = P(s_i|s_{i-1}). \tag{3.14}$$

Ainsi, une chaîne de Markov est-elle totalement déterminée par (Figure 3.4) :

1. son ensemble d'états,
2. les probabilités de transitions entre états,
3. la densité de probabilité d'être à l'état initial.

Les chaînes de Markov cachées

Un signal modélisé par une chaîne de Markov cachée est considéré comme une séquence d'observations émises par des états sous-jacents appartenant à une chaîne de Markov. Si l'observation est discrète, les observations prennent leurs valeurs dans un ensemble $V = \{v_1, v_1, ..., v_M\}$ à chaque instant n. Dans le cas d'observations continues, la densité de probabilité d'émission est continue, en pratique c'est un mélange de fonctions gaussiennes :

$$b(x) = \sum_{k=1}^{M} c_k G(x, \mu_k, \Sigma_k), \tag{3.15}$$

où $G(x, \mu, \Sigma)$ est la valeur en x d'une gaussienne de moyenne μ et de matrice de covariance Σ (souvent diagonale) et où

$$\sum_{k=1}^{M} c_k = 1. \tag{3.16}$$

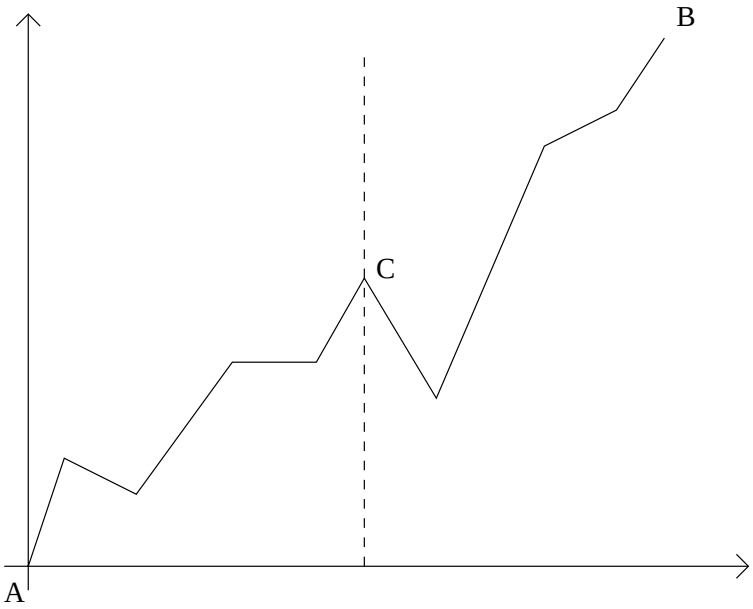


FIG. 3.5 : Programmation dynamique : trajectoire et sous-trajectoires optimales.

Le choix de ce type de densité de probabilité permet d'approcher la plupart des comportements avec un nombre de paramètres raisonnable (M ne dépasse pas quelques dizaines), tout en étant facile à apprendre par des formules de réestimation de type Estimation-Maximisation (cf. section 3.2.4).

La programmation dynamique

Pour résoudre les problèmes associés aux modèles de Markov, on utilise des algorithmes d'optimisation dont l'approche repose sur le principe de programmation dynamique qui est une stratégie d'optimisation de trajectoire. La programmation dynamique se fonde sur le principe d'optimalité de Bellmann [Bellman, 1957] : si une trajectoire entre deux points A et B est optimale, et si C est un point de cette trajectoire, alors les sous-trajectoires de A à C et de C à B sont elles aussi optimales (Figure 3.5).

Ainsi plutôt que d'explorer toutes les trajectoires possibles (p^n trajectoires de longueur n prenant leurs valeurs dans un ensemble à p éléments), pour chaque point C, on explore les demi trajectoires ($\sum_{y_C} p^{x_C - x_A - 1} + \sum_{y_C} p^{x_B - x_C - 1}$ trajectoires). Ce processus est itéré n fois. On obtient alors un algorithme en np^2.

Soit un exemple très simple d'optimisation de trajectoire illustré figure 3.6 : supposons que l'on veuille déterminer la trajectoire optimale entre deux points A et B respectivement aux instants 0 et T. Le passage par un point M a un coût $C_1(M)$ et une portion de trajectoire entre les points N et N' a un coût $C_2(N, N')$. À l'instant $t = 1$, on a p positions possibles, chacune de coût $C(t = 1, M_p) = C_1(A) + C_2(A, M_p) + C_1(M_p)$. À l'instant $t = 2$, on a à nouveau p positions possibles, chacune de coût $C(t = 2, M_p) = \min_q C(t = 1, M_q) + C_2(M_q, M_p) + C_1(M_p)$. Et ainsi de suite jusqu'à $t = T$, où seule la position B est possible,

le coût minimum est alors : $C(t = T, B) = \min_q C(t = T - 1, M_q) + C_2(M_q, B) + C_1(B)$.

C'est sur ce principe que sont bâtis les algorithmes couramment utilisés avec les modèles de chaînes de Markov décrits dans la section 3.2.4.

3.2.4 Les algorithmes de décodage des chaînes de Markov cachées

Les algorithmes associés aux HMM sont fondés sur la programmation dynamique qui tire pleinement parti de l'hypothèse markovienne. La tâche de reconnaissance confiée au modèle se fait suivant une approche bayésienne qui permet de transformer la probabilité *a posteriori* de la configuration (probabilité de la configuration connaissant l'observation) en probabilité *a priori* de l'observation (probabilité d'une observation connaissant la configuration) :

$$\hat{\omega} = \arg\max_{\omega \in \Omega} P(\omega|O) = \arg\max_{\omega \in \Omega} \frac{P(O|\omega).P(\omega)}{P(O)}. \tag{3.17}$$

La probabilité de l'observation étant indépendante de la configuration, la maximisation se fait de la façon suivante :

$$\hat{\omega} = \arg\max_{\omega \in \Omega} P(O|\omega).P(\omega). \tag{3.18}$$

Traditionnellement, on décompose l'utilisation des HMM en trois problèmes de base [Rabiner et Juang, 1993] :

- **Problème 1** : Calcul de la probabilité d'une observation.
- **Problème 2** : Calcul de la séquence d'états la plus probable compte tenu d'une observation.
- **Problème 3** : Calcul des paramètres du modèle compte tenu des observations d'apprentissage.

Calcul de la probabilité d'une émission : algorithme Forward

Un calcul direct de la probabilité d'apparition d'une observation par sommation des probabilités pour chaque séquence d'états serait d'un coût prohibitif. On utilise donc le principe de la programmation dynamique en mémorisant les optimisations partielles dans une variable α :

Si X_1^t est l'observation de l'instant 1 à t, s_t l'état courant à l'instant t et Φ le modèle, on définit

$$\alpha_t(i) = P(X_1^t, s_t = i|\Phi), \tag{3.19}$$

variable Forward.

L'algorithme Forward est alors le suivant :

1. **Initialisation**

$$\alpha_1(i) = \pi_i b_i(X_1) \quad \text{pour } 1 \leq i \leq N. \tag{3.20}$$

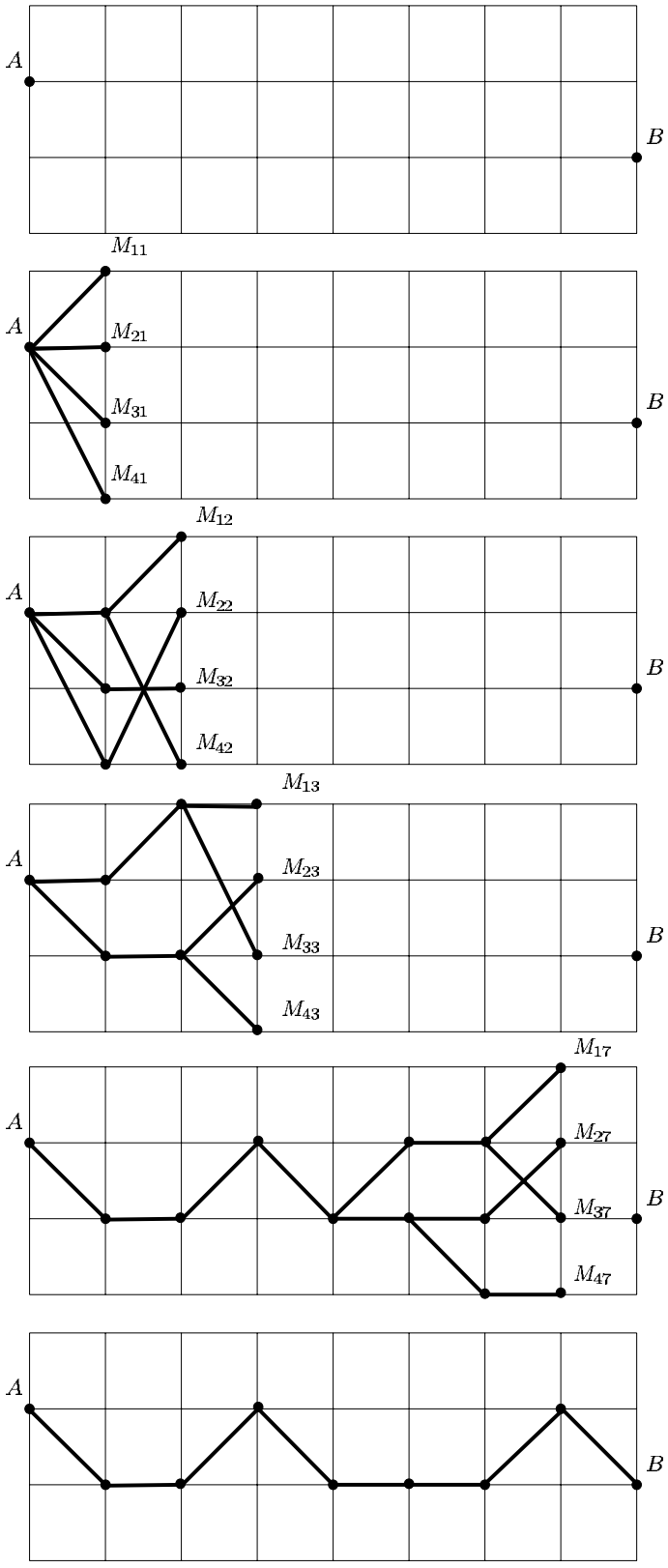


FIG. 3.6 : Recherche d'une trajectoire optimale par l'algorithme de programmation dynamique.

2. **Itération**

$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij}\right] b_j(X_t) \quad \text{pour } 2 \leq t \leq T \text{ et } 1 \leq j \leq N. \tag{3.21}$$

3. **Terminaison**

$$P(X|\Phi) = \sum_{i=1}^{N} \alpha_T(i). \tag{3.22}$$

La complexité de l'algorithme Forward est en $O(N^2T)$ contre $O(N^T)$ pour une exploration systématique de toutes les séquences d'états. Le calcul de la probabilité d'une observation permet, par utilisation directe de la loi de Bayes de reconnaître une observation comme appartenant à une classe décrite par un modèle.

Décodage d'une séquence d'états : algorithme de Viterbi

En réponse au second problème des HMM, l'algorithme de Viterbi permet de trouver la séquence d'états la plus probable compte tenu d'une observation. En pratique, on peut utiliser l'algorithme de Viterbi plutôt que l'algorithme forward si on effectue la reconnaissance sans calculer la probabilité d'une observation. On ignore alors les observations issues de séquences d'états moins probables que le meilleur chemin [Huang *et al.*, 2001, p 388]. On introduit la probabilité d'une séquence X_1^t correspondant aux états S_1^{t-1} se terminant à l'instant t à l'état $s_t = i$:

$$V_t(i) = P(X_1^t, S_1^{t-1}, s_t = i|\Phi). \tag{3.23}$$

L'algorithme de Viterbi est alors le suivant :

1. **Initialisation**

$$\text{pour } 1 \leq i \leq N \quad V_1(i) = \pi_i b_i(X_1) \tag{3.24}$$

$$B_1(i) = 0. \tag{3.25}$$

2. **Itération**

$$\text{pour } 2 \leq t \leq T \text{ et } 1 \leq j \leq N \quad V_t(j) = \max_{1 \leq i \leq N}\left[V_{t-1}(i) a_{ij}\right] b_j(X_t) \tag{3.26}$$

$$\text{pour } 2 \leq t \leq T \text{ et } 1 \leq j \leq N \quad B_t(j) = \arg\max_{1 \leq i \leq N}\left[V_{t-1}(i) a_{ij}\right]. \tag{3.27}$$

3. **Terminaison**

$$V_T = \max_{1 \leq i \leq N}\left[V_T(i)\right], \tag{3.28}$$

$$s_T^* = \arg\max_{1 \leq i \leq N}\left[B_T(i)\right]. \tag{3.29}$$

4. **Calcul du meilleur chemin**

$$\text{pour } t = T-1, T-2, ..., 1 \quad s_t^* = B_{t+1}(s_{t+1}^*) \tag{3.30}$$
$$S^* = (s_1^*, s_2^*, ..., s_T^*). \tag{3.31}$$

Apprentissage de paramètres : l'algorithme EM

L'apprentissage des paramètres d'un modèle consiste à maximiser la probabilité d'émission d'une séquence : On cherche le modèle Φ qui maximise la probabilité $P(Y = y|\Phi)$, y étant une séquence d'apprentissage. Si l'observation dépend d'un paramètre inconnu (comme c'est le cas pour les HMM), cette maximisation ne peut pas se faire directement, il faut tenir compte de toutes les valeurs possibles de cette séquence cachée X.

On introduit donc la log-vraisemblance définie par :

$$Q(\Phi, \bar{\Phi}) = E_\Phi[\log P(X, Y = y|\bar{\Phi})]_{X|Y=y} \tag{3.32}$$
$$= \sum_x P(X, Y = y|\Phi) \log P(X, Y = y|\bar{\Phi}). \tag{3.33}$$

L'algorithme EM consiste à calculer cette espérance (phase E), puis à la maximiser (phase M) et ce de façon itérative jusqu'à convergence.

On peut donc décrire l'algorithme EM de la façon suivante :

1. **Initialisation :** On choisit un modèle initial Φ.
2. **Phase E :** On calcule la fonction auxiliaire $Q(\Phi, \bar{\Phi})$.
3. **Phase M :** On calcule $\hat{\Phi} = \arg\max_\Phi Q(\Phi, \bar{\Phi})$ en maximisant Q.
4. **Itération :** On itère le processus avec $\Phi = \hat{\Phi}$, jusqu'à la convergence de l'algorithme.

Apprentissage des HMM : l'algorithme de Baum-Welch

L'apprentissage des HMM est un exemple typique d'apprentissage semi-supervisé (la séquence d'états est inconnue) tel qu'on le traite par l'algorithme EM. On utilise donc la fonction log-vraisemblance définie pour l'algorithme EM, dont l'expression sera ici :

$$Q(\Phi, \hat{\Phi}) = \sum_S \frac{P(X, S|\Phi)}{P(X|\Phi)} \log P(X, S|\hat{\Phi}). \tag{3.34}$$

La sommation se fait sur toutes les séquences d'états possibles, on a donc besoin de la variable de prédiction α, qu'on utilise conjointement avec la variable de prédiction Backward β. L'algorithme de Baum-Welch s'appelle aussi algorithme Forward-Backward.

$$\beta_t(i) = P(X_{t+1}^T|s_t = i, \Phi). \tag{3.35}$$

Les variables de prédiction permettent le calcul de la probabilité d'une transition de l'état i vers l'état j à l'instant t :

$$\gamma_t(i, j) = P(s_{t-1} = i, s_t = j|X_1^T, \Phi) = \frac{\alpha_{t-1}(i) a_{ij} b_j(X_t) \beta_t(j)}{\sum_{k=1}^N \alpha_T(k)}. \tag{3.36}$$

La maximisation de la fonction log-vraisemblance donne les formules de réestimation suivantes :

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T} \gamma_t(i,j)}{\sum_{t=1}^{T} \sum_{k=1}^{N} \gamma_t(i,k)}, \tag{3.37}$$

$$\hat{b}_j(k) = \frac{\sum_{t,X_t \in O_k} \sum_i \gamma_t(i,j)}{\sum_{t=1}^{T} \sum_i \gamma_t(i,j)}. \tag{3.38}$$

L'algorithme de Baum-Welch peut donc être décrit de la façon suivante :

1. **Initialisation :** On choisit un modèle initial Φ
2. **Phase E :** On calcule la fonction auxiliaire $Q(\Phi, \bar{\Phi})$
3. **Phase M :** On calcule $\hat{\Phi}$ grâce aux formules de réestimation.
4. **Itération :** On itère le processus avec $\Phi = \hat{\Phi}$, jusqu'à la convergence de l'algorithme.

En pratique, on ne somme pas forcément sur tous les chemins possibles, on recherche d'abord le chemin optimal de Viterbi avant de maximiser la log-vraisemblance.

3.3 Les champs de Markov

Les champs de Markov sont utilisés déjà depuis longtemps en traitement d'images [Chellappa et Jain, 1993, Descombes, 1993, Pérez, 1999]. Les applications sont très diverses et concernent en particulier les problèmes d'analyse de textures [Derras, 1993], de reconnaissance de formes [Prêteux, 1991, Cai et Liu, 2001, Elyoubi, 1995], de débruitage [Geman et Geman, 1993] ou de segmentation [Wang, 1994, Yu *et al.*, 2001].

3.3.1 Définition des champs de Markov

Structure d'un champ de Markov

Les champs markoviens sont des champs aléatoires eux-mêmes fondés sur la structure de graphe. Les noeuds du graphe sont appelés **sites** et en pratique correspondent en général aux pixels de l'image. On note S l'ensemble des sites du champ de Markov.

Voisinages

Les arêtes du graphe définissent le système de voisinages du champ : deux sites s_1 et s_2 sont dits **voisins** s'il existe une arête qui les lie. On note alors $V(s)$ l'ensemble des sites voisins de s. La notion de voisinage dans le contexte markovien est souvent exprimée en termes de **cliques** : Les **cliques** sont les singletons et les parties de S dont les éléments sont voisins deux à deux. Si les sites du champ de Markov sont les pixels de l'image, on prendra souvent pour système de voisinage les 4-voisins ou les 8-voisins. Les cliques associées sont représentées sur la figure 3.7.

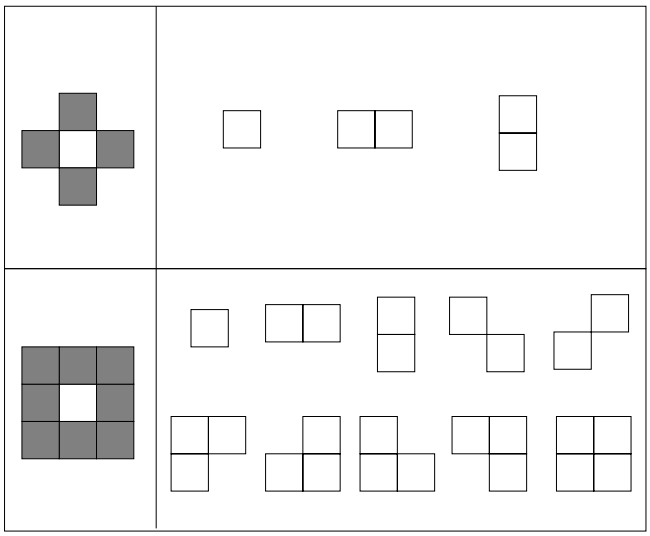


FIG. 3.7 : Systèmes de voisinage et cliques associées.

Champ aléatoire

Les modèles des champs de Markov sont des champs aléatoires, c'est-à-dire qu'un ensemble de variables aléatoires X_s fait correspondre à chaque site s une configuration x_s dans un ensemble Ω fini, dénombrable ou non dénombrable [Pérez, 1993].

L'hypothèse markovienne

Un champ aléatoire est un champ de Markov si et seulement si :

$$P(X_s = x_s | X_r = x_r, r \in S - \{s\}) = P(X_s = x_s | X_r = x_r, r \in V(s)). \tag{3.39}$$

Le champ de Gibbs

Un champ aléatoire est un champ de Gibbs si la probabilité d'une configuration est de la forme :

$$p(x) = \frac{1}{Z} \exp\left[- \sum_{c \in C} U_c(x) \right] \tag{3.40}$$

où Z est la constante de normalisation et U est un potentiel lié au système de voisinage, donc aux cliques $c \in C$.

En général, la relation 3.40 ne permet pas le calcul de $p(x)$ car la constante Z est inconnue, mais la probabilité conditionnelle à la restriction de X au voisinage V_s est calculable par :

$$p(x_s | x_{V_s}) = \frac{\exp\left[- \sum_{c \in C, s \in c} U_c(x_s, x_{\bar{c}}) \right]}{\sum_{\omega \in \Omega} \exp\left[- \sum_{c \in C, s \in c} U_c(\omega, x_{\bar{c}}) \right]} \tag{3.41}$$

où $\bar{c} = c - \{s\}$ [Prêteux, 1991].

On peut citer deux algorithmes classiques pour simuler une loi de Gibbs :

➪ La procédure de l'échantillonneur de Gibbs [Geman et Geman, 1984] :

L'échantillonneur de Gibbs construit une suite d'images qui converge vers une observation du champ de Gibbs correspondant [Sigelle et Tupin, 1999].

- On part d'une configuration quelconque $x^0 \in \Omega^N$ (Ω étant l'ensemble des états possibles et N le nombre de sites).
- On parcourt les sites s (de façon aléatoire ou déterministe, chaque site devant avoir été visité un grand nombre de fois) et on fait des tirages aléatoires des configurations suivant la loi 3.41 en tenant compte des nouvelles valeurs des pixels voisins.
- On réitère le processus avec la nouvelle configuration obtenue, jusqu'à la convergence.

➪ L'algorithme de Métropolis

L'algorithme de Métropolis construit aussi une suite d'images qui converge vers une observation, mais la nouvelle configuration d'un site est ici tirée aléatoirement suivant une loi uniforme, la nouvelle configuration étant acceptée si elle entraîne une diminution de l'énergie, ou sinon, on soumet son acceptation à un tirage au sort :

- On part d'une configuration quelconque $x^0 \in \Omega^N$ (Ω étant l'ensemble des états possibles et N le nombre de sites).
- On parcourt les sites s (de façon aléatoire ou déterministe, chaque site devant avoir été visité un grand nombre de fois), et à chaque étape :
  - une configuration ω est tirée au sort selon une loi uniforme,
  - la variation d'énergie $\Delta U = U_s(\omega|V_s^{n-1}) - U_s(x_s^{n-1}|V_s^{n-1})$ est calculée,
  - la nouvelle configuration est acceptée si $\Delta U < 0$ ou si $\Delta U \geq 0$ et après un tirage aléatoire de probabilité $\exp(-\Delta U)$.
- On stoppe la convergence

Le théorème de Hammersley-Clifford établit l'équivalence entre les champs de Markov (assortis de l'hypothèse de positivité $P(X = x) > 0$) et les champs de Gibbs.

3.3.2 Décodage d'un champ de Markov

On appellera ici décodage l'action de déterminer la configuration la plus probable d'un champ de Markov. Deux algorithmes principaux existent dans la littérature : le recuit simulé et l'ICM (Iterated Conditional Modes).

➪ Le recuit simulé

Le principe du recuit simulé est d'introduire une notion de température qui sera diminuée graduellement. Entre chaque changement de température, une configuration x^n est simulée (grâce à l'échantillonneur de Gibbs ou l'algorithme de Métropolis) avec la loi d'énergie $\frac{U(x)}{T^n}$ où T^n est la température à l'étape n. La température initiale T^0 doit être grande et la décroissance doit être suffisamment lente (idéalement logarithmique) pour converger vers un minimum global d'énergie. On cesse les itérations lorsque le taux de changements des configurations des sites entre deux étapes est suffisamment faible. La nécessité de choisir une décroissance de la température lente, rend l'algorithme de recuit simulé très lourd en temps de calcul.

➪ L'ICM

C'est un algorithme itératif déterministe qui consiste à diminuer l'énergie du champ à chaque étape. Il est rapide mais converge vers un minimum local de l'énergie. On construit là encore une suite d'images x^n, entre chacune de ces images, on visite une fois chaque site de l'image de façon déterministe. Le principe est de choisir pour chaque site, la configuration maximisant la probabilité conditionnelle locale : $x_s^{n+1} = \arg\max_\omega P(X_s = \omega | x_r^n, r \in V_s)$. La convergence de l'algorithme est ainsi très rapide mais s'arrête à un minimum local de l'énergie.

On voit ainsi l'intérêt de l'algorithme de programmation dynamique 2D décrit dans la section 3.4 qui permet de déterminer rapidement le minimum global de l'énergie.

3.4 La programmation dynamique 2D

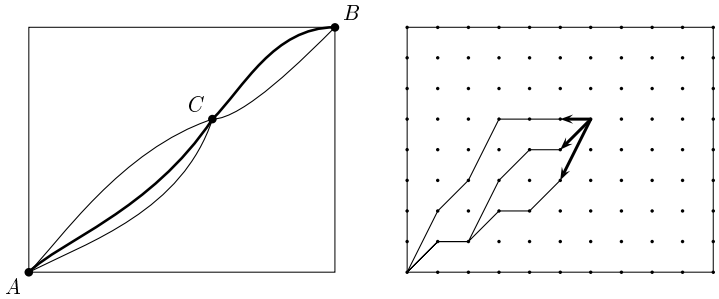
Think globally, act locally.
principe de management

Le concept de programmation dynamique a été proposé dès 1957 pour une tâche de contrôle optimal [Bellman, 1957]. Il a été introduit en reconnaissance de la parole en 1968 [Vintsyuk, 1968] et est toujours resté au cœur des meilleurs systèmes depuis.

Le principe de l'algorithme, à la fois simple et puissant (Figure 3.8), permet la recherche d'un chemin optimal dans un espace combinatoire avec une complexité linéaire et non exponentielle pour une recherche directe. D'abord employé pour déterminer la distance minimum entre deux séquences acoustiques, l'algorithme fut ensuite adapté à la modélisation par modèles de Markov cachés (HMM) pour rechercher la séquence d'états la plus probable étant donnée une séquence acoustique (algorithme de Viterbi [Forney, 1973]). C'est grâce à l'hypothèse commune aux HMM et à la programmation dynamique de dépendance à court terme que les deux concepts fonctionnent bien ensemble.

La modélisation markovienne est devenue très populaire en vision et les champs de Markov sont très utilisés en traitement d'images pour la segmentation, la restauration et la reconnaissance [Dubes et Jain, 1989, Li, 1994]. Il est donc tentant d'utiliser la programmation dynamique pour calculer efficacement l'alignement optimal entre l'image observée et les étiquettes ou états du modèle. Pourtant, les méthodes classiquement utilisées pour le décodage sont soit sous-optimales, soit très lentes. Par exemple l'ICM [Besag, 1986] est rapide mais largement sous-optimal, et le recuit simulé [Geman et Geman, 1984] doit converger extrêmement lentement pour assurer l'optimalité de la solution. Cela est dû au fait que ces techniques n'exploitent pas la structure de l'espace de recherche.

La programmation dynamique optimise le coût d'une trajectoire entre deux points. Si une trajectoire peut, à chaque instant prendre une valeur parmi N, pour chaque point intermédiaire chaque sous-trajectoire est aussi optimale. Ainsi, le calcul direct d'une trajectoire de longueur L qui peut prendre N valeurs aurait une complexité en N^L mais le calcul d'une demi-trajectoire est en $N^{L/2}$. En itérant L fois ce processus, on obtient une



(a) (b)

$$\forall t, \forall j, \begin{cases} D(t,j) = \min_{j'}[d(j') + D(t-1,j')] \\ b(t,j) = \arg\min_{j'}[d(j') + D(t-1,j')] \end{cases}$$

(c)

FIG. 3.8 : Principe de la programmation dynamique (a) et algorithme (b,c). Le principe est que chaque sous-trajectoire de la trajectoire optimale de A à B passant par C (en gras sur la figure) est elle-même optimale. L'algorithme consiste à calculer récursivement la trajectoire optimale de A au point courant. D est la distance accumulée et b mémorise les chemins jusqu'au point courant.

approche "diviser pour régner" dont le coût est en LN^2.

Ce principe a toujours été considéré comme intrinsèquement monodimensionnel et les différentes tentatives d'utilisation dans le cadre du traitement d'images ont toutes été élaborées à partir d'un parcours monodimensionnel de la surface de l'image. Pourtant, ce principe se généralise de façon simple et canonique à un cadre 2D et même à une dimension n quelconque. Cette extension, proposée dans [Geoffrois *et al.*, 1998] et décrite plus en détails dans [Geoffrois, 2003], se place dans un contexte markovien d'une dépendance locale.

Si une région est un ensemble connexe de pixels d'une image, une région R_1 ne dépend des pixels d'une région R_2 que dans une zone qu'on appellera frontière entre ces deux régions. La frontière d'une région R étant l'ensemble des frontières avec les autres régions, pour chaque configuration de cette frontière, il n'existe qu'une seule configuration optimale de l'intérieur de la région R soit les pixels de R qui n'appartiennent pas à sa frontière. On peut alors déterminer la configuration optimale de $R_1 \cup R_2$ en explorant seulement les différentes configurations de la frontière entre R_1 et R_2.

3.4.1 Cadre général

Considérons une approche bayésienne classique de reconnaissance d'images où l'image observée correspond à un état sous-jacent, typiquement une configuration d'étiquettes associées à chaque pixel ou site, qui doit être retrouvée. Si on prend une décision par maximum *a posteriori* (MAP), la loi de Bayes permet d'écrire que la configuration optimale est

donnée par :

$$\hat{\omega} = \arg\max_{\omega\in\Omega} P(\omega|o) = \arg\max_{\omega\in\Omega} P(o|\omega)P(\omega),$$

où o est l'observation et Ω est l'ensemble de toutes les configurations d'étiquettes possibles. Plus précisément,

$$o = \{o_{(i,j)}, 1 \leqslant i, j \leqslant n\}$$

où les observations $o_{(i,j)}$ peuvent être scalaires ou vectorielles, et

$$\omega = \{\omega_{(i,j)}, 1 \leqslant i, j \leqslant n\}$$

où $\omega_{(i,j)} \in \{1, \ldots, L\}$ est l'étiquette du site (i, j).

Pour simplifier les notations, on considérera une image carrée $(n \times n)$, et l'ensemble des étiquettes de cardinal L. On a alors un nombre de configurations possibles $|\Omega| = L^{n^2}$.

Le modèle de champ de Markov caché, ou de façon équivalente le modèle de distribution de Gibbs [Besag, 1974], fait l'hypothèse d'une dépendance locale contextuelle :

$$P(o|\omega) = \prod_{(i,j)} P(o_{(i,j)}|\omega_{(i,j)}) \tag{3.42}$$

$$\text{et} \quad P(\omega) = \frac{1}{Z}\exp(-\sum_{c\in C} V_c(\omega)),$$

où C est l'ensemble des cliques associé au type de voisinage, V_c est le potentiel de la clique c et Z est la constante de normalisation telle que $\sum_\omega P(\omega) = 1$.

On peut alors introduire la fonction de potentiel :

$$U(\omega) = \sum_{(i,j)} -\log(P(o_{(i,j)}|\omega_{(i,j)})) + \sum_{c\in C} V_c(\omega), \tag{3.43}$$

qui ramène le problème de maximisation de la probabilité *a posteriori* au problème de minimisation de la fonction de potentiel :

$$\hat{\omega} = \arg\min_{\omega\in\Omega} U(\omega).$$

Une propriété importante de la fonction de potentiel est qu'elle ne comprend que des termes d'interaction locale. On considérera dans la suite un voisinage du premier ordre dont les cliques sont illustrées par la figure 3.9 mais la méthode est valable pour tout type de voisinage.

3.4.2 Programmation dynamique multidimensionnelle

La structure du problème peut être exploitée pour calculer efficacement la solution optimale. Dans cette partie, on montrera d'abord comment le problème peut être décomposé en deux problèmes moins complexes et comment l'optimum global peut être déterminé à partir des solutions optimales partielles. Par récursion, on obtient le déroulement complet

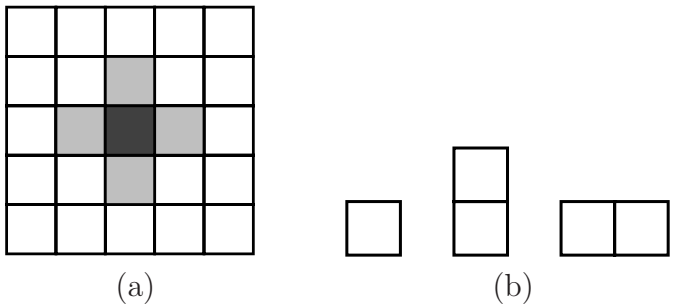


FIG. 3.9 : Voisinage du premier ordre (a) et cliques associées (b).

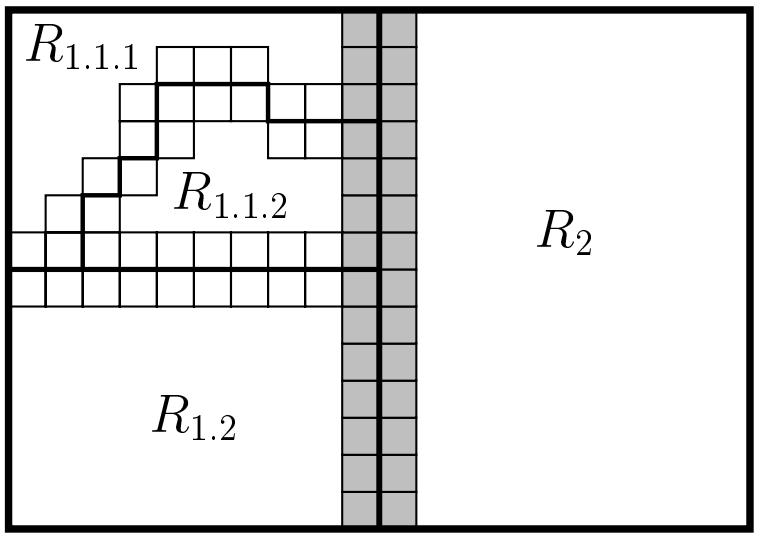


FIG. 3.10 : Partition de l'image en deux régions R_1 et R_2, qui peuvent à leur tour être divisées en sous-régions. La partition peut être régulière ($R_1 = R_{1.1} \cup R_{1.2}$) ou pas ($R_{1.1} = R_{1.1.1} \cup R_{1.1.2}$). Seuls les sites appartenant aux frontières sont représentés. La frontière de R_1 et R_2 est en gris.

de l'algorithme. Les illustrations sont données pour le cas 2D des images mais le principe s'applique aussi aux dimensions plus élevées.

Considérons une partition de l'image en deux régions R_1 et R_2, une région étant un ensemble connexe de pixels (Figure 3.10). Ces régions peuvent être de forme quelconque et ne sont pas attachées au contenu de l'image. Soient ∂R_1 et ∂R_2 les *frontières* de ces régions définies comme l'ensemble des pixels appartenant à des cliques qui contiennent à la fois des pixels de R_1 et de R_2 (ensemble des sites qui interagissent avec des voisins dans l'autre région, en gris sur la figure 3.10). Les sites qui n'appartiennent pas à la frontière sont dits *intérieurs*.

Pour une configuration donnée ω, soient ω_1, ω_2, $\partial\omega_1$ et $\partial\omega_2$ les restrictions de cette configuration à R_1, R_2, ∂R_1 et ∂R_2 respectivement. La fonction à minimiser $U(\omega)$ peut être réécrite en distinguant les termes associés à chaque région et les termes d'interaction

associés aux sites de la frontière :

$$U(\omega) = U(\omega_1) + I(\partial\omega_1, \partial\omega_2) + U(\omega_2).$$

Les notations $U(\omega_1)$ et $U(\omega_2)$ sont des simplifications pour $U_{R_1}(\omega_1)$ et $U_{R_2}(\omega_2)$ et correspondent aux termes de $U(\omega)$ qui ne dépendent que d'une seule région. De même, $I(\partial\omega_1, \partial\omega_2)$ est une simplification de $I_{\partial R_1, \partial R_2}(\partial\omega_1, \partial\omega_2)$ et correspond aux termes restants, associés aux cliques traversant la frontière.

Considérons ensuite deux configurations ω et ω' qui sont égales pour les pixels frontière (c'est-à-dire que $(\partial\omega_1, \partial\omega_2) = (\partial\omega'_1, \partial\omega'_2)$) et diffèrent seulement pour les pixels intérieurs. On voit alors que, comme $I(\partial\omega_1, \partial\omega_2) = I(\partial\omega'_1, \partial\omega'_2)$,

$$\left.\begin{array}{l} U(\omega_1) < U(\omega'_1) \\ U(\omega_2) < U(\omega'_2) \end{array}\right\} \Rightarrow U(\omega) < U(\omega').$$

Par conséquent, pour une configuration donnée $(\partial\omega_1, \partial\omega_2)$ des frontières, on obtient

$$\left.\begin{array}{l} \hat{\omega}_1 = \arg\min U(\omega_1) \\ \hat{\omega}_2 = \arg\min U(\omega_2) \end{array}\right\} \Rightarrow \hat{\omega}_1 \cup \hat{\omega}_2 = \arg\min U(\omega_1 \cup \omega_2),$$

c'est-à-dire,

$$\hat{\omega} = \hat{\omega}_1 \cup \hat{\omega}_2.$$

Il n'est donc pas nécessaire de calculer les sommes $U(\omega_1) + I(\partial\omega_1, \partial\omega_2) + U(\omega_2)$ pour tous les ω_1 et ω_2 pour trouver l'optimum. Il n'est pas non plus nécessaire de stocker toutes les configurations. Il faut simplement stocker la configuration optimale $\hat{\omega}_1$ pour chaque configuration de sa frontière $\partial\hat{\omega}_1$ possible et de même pour $\hat{\omega}_2$.

Résumons ce qui doit être stocké pour chaque région de façon à obtenir la configuration optimale globale $\hat{\omega}$. Soit $\partial\Omega_r$ $(r = 1, 2)$ l'ensemble de toutes les configurations possibles des frontières de la région R_r, et soit $\hat{\Omega}_r = \{\hat{\omega}_r / \partial\omega_r \in \partial\Omega_r\}$ l'ensemble des configurations optimales des pixels intérieurs pour chaque configuration de la frontière. L'optimum global peut alors être atteint en combinant les configurations de $\hat{\Omega}_1$ et $\hat{\Omega}_2$ et en sélectionnant le minimum :

$$\hat{\omega} = \arg \min_{(\hat{\omega}_1, \hat{\omega}_2) \in \hat{\Omega}_1 \times \hat{\Omega}_2} U(\hat{\omega}_1) + I(\partial\omega_1, \partial\omega_2) + U(\hat{\omega}_2).$$

Ce processus est récursif. De même que $\hat{\Omega} = \{\hat{\omega}\}$ peut être calculée facilement à partir de $\hat{\Omega}_1$ et $\hat{\Omega}_2$, $\hat{\Omega}_1$ peut elle-même être calculée à partir de $\hat{\Omega}_{1.1}$ et $\hat{\Omega}_{1.2}$ de la même manière. Seule une partie des sites frontière de $R_{1.1}$ et $R_{1.2}$ forme la frontière de la nouvelle région R_1 (en gris sur la figure 3.10), l'autre partie devenant intérieure. Pour les mêmes raisons que précédemment, il est possible de minimiser la configuration de l'intérieur de R_1 pour chaque configuration de sa frontière $\partial\omega_1$. De façon générale, pour une région R_r, $\hat{\Omega}_r$ peut être calculée à partir des configurations optimales de deux sous-régions $\hat{\Omega}_{r.1}$ et $\hat{\Omega}_{r.2}$, et ainsi de suite jusqu'aux régions élémentaires d'un seul pixel, dont l'initialisation est triviale.

Ainsi, appliquer l'algorithme de programmation dynamique 2D sur une image consiste à initialiser d'abord n^2 régions d'un pixel, chacune ayant L configurations possibles, puis à

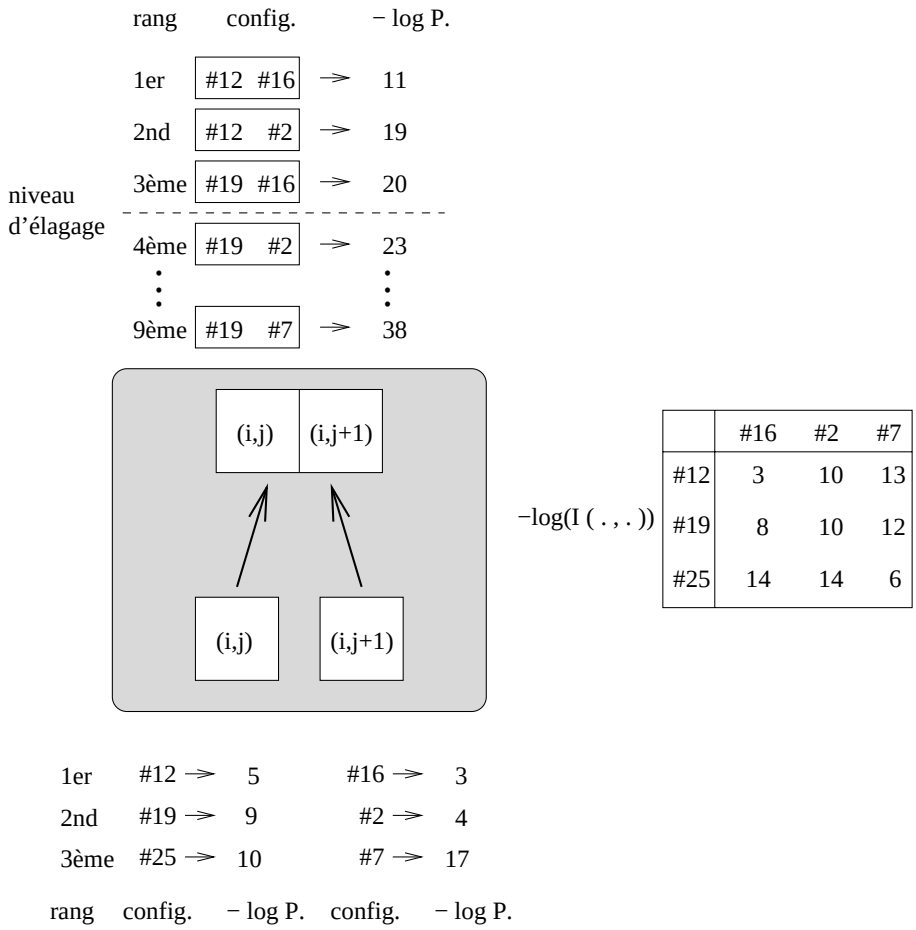


FIG. 3.11 : Fusion de deux pixels : La région résultant de la fusion est associée à une nouvelle liste de configurations possibles et à une vraisemblance qui est la somme des vraisemblances des deux pixels et du terme d'interaction I.

fusionner ces régions deux à deux en ne gardant que la configuration optimale pour chaque configuration de la frontière de chaque région. Pour résumer, lors de la fusion de deux pixels, chaque pixel est associé à une liste de configurations possibles et à la vraisemblance correspondante. La région résultant de la fusion est associée à une nouvelle liste de configurations possibles et à une vraisemblance qui est la somme des vraisemblances des deux pixels et du terme d'interaction I (Figure 3.11). Dans le cas général de la fusion de deux régions, chacune des régions R_1 et R_2 est associée à une liste de configurations possibles et à la vraisemblance correspondante. La région résultant de la fusion $R_1 \cup R_2$ est associée à une nouvelle liste de configurations possibles et une vraisemblance correspondante qui est la somme des vraisemblances des deux régions R_1 et R_2 et des termes d'interaction I (Figure 3.12). Ces termes d'interaction sont calculés sur la frontière et chaque configuration de la frontière est associée à une unique configuration des pixels intérieurs.

L'ordre dans lequel on effectue ces fusions n'a pas d'influence sur la configuration obtenue (qui est la solution optimale), mais elle détermine le temps de calcul et l'espace mémoire

utilisé. Après avoir effectué toutes les fusions, il ne reste plus qu'une seule grande région qui recouvre toute l'image ainsi que la configuration optimale correspondante (Figure 3.13).

Un exemple du déroulement de l'algorithme est donné par la figure 3.14. À chaque étape, pour chaque configuration de la frontière, on n'a qu'une seule configuration optimale de l'intérieur de la région (sites en noir). Un site peut être à l'état 0 ou 1, les sites en blanc sont des régions d'un seul site qui n'ont pas encore été fusionnées. Le système de cliques est associé aux 4-voisins (Figure 3.9).

En conclusion, la programmation dynamique, qui n'est qu'un cas particulier de l'approche "diviser pour régner", peut être généralisée naturellement aux données bidimensionnelles. Si on considère le point courant de la programmation dynamique monodimensionnelle (C sur la figure 3.8) comme une frontière de dimension 0 entre le passé et le futur sur la trajectoire monodimensionnelle, on voit alors que la généralisation au cas 2D est canonique. Les frontières sont alors de dimension 1 et sont situées entre l'intérieur et l'extérieur de régions 2D. La programmation dynamique peut en fait être généralisée au cas 3D, avec pour frontières des surfaces séparant des volumes, ou même au cas d'une dimension plus élevée avec pour frontières des hypersurfaces séparant des régions multidimensionnelles.

3.4.3 Implantation des champs de Markov et de la programmation dynamique 2D

Une fois défini le cadre théorique, l'implantation du modèle nécessite un choix de fonction de potentiel et donc de termes d'interaction. Cette section présente l'algorithme de programmation dynamique 2D développé avec un champ de Markov caché.

➜ **Les probabilités de transition**

On définit les fonctions d'interaction I_0, I_1, I_2 et I_3 :

$$\begin{aligned} I_0(\omega_k,\omega_l) &= \frac{P\left(\begin{array}{|c|}\hline \omega_l \\ \hline \omega_k \\ \hline \end{array}\right)}{P(\omega_k)P(\omega_l)},\ I_1(\omega_k,\omega_l) = \frac{P\left(\begin{array}{|c|}\hline \omega_k \\ \hline \omega_l \\ \hline \end{array}\right)}{P(\omega_k)P(\omega_l)}, \\ I_2(\omega_k,\omega_l) &= \frac{P\left(\begin{array}{|c|c|}\hline \omega_l & \omega_k \\ \hline \end{array}\right)}{P(\omega_k)P(\omega_l)},\ I_3(\omega_k,\omega_l) = \frac{P\left(\begin{array}{|c|c|}\hline \omega_k & \omega_l \\ \hline \end{array}\right)}{P(\omega_k)P(\omega_l)}, \end{aligned} \tag{3.44}$$

où

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

On peut interpréter ces termes comme une information mutuelle [Duda *et al.*, 2001] qui tient compte de l'orientation. L'ensemble des cliques C est défini par :

$$C = C_0 \cup C_1 \cup C_2$$

où,

$$C_0 = \{(i,j),\ 1 \leqslant i,j \leqslant n\},$$
$$C_1 = \{((i,j),(i+1,j)),\ 1 \leqslant i \leqslant n-1,\ 1 \leqslant j \leqslant n\},$$
$$C_2 = \{((i,j),(i,j+1)),\ 1 \leqslant i \leqslant n,\ 1 \leqslant j \leqslant n-1\}.$$

Les potentiels V_c sont alors définis par :

$$V_c(\omega) = \begin{cases} -\log(P(\omega_k)) & \text{si } c \in C_0 \text{ avec } c=(i,j) \text{ et } \omega_k = \omega_{i,j} \\ -\log(I_0(\omega_k,\omega_l)) & \text{si } c \in C_1 \text{ avec } \begin{cases} c = ((i,j),(i+1,j)) \\ \omega_l = \omega_{i,j} \text{ et} \\ \omega_k = \omega_{i+1,j}. \end{cases} \\ -\log(I_2(\omega_k,\omega_l)) & \text{si } c \in C_2 \text{ avec } \begin{cases} c = ((i,j),(i,j+1)) \\ \omega_l = \omega_{i,j} \text{ et} \\ \omega_k = \omega_{i,j+1}. \end{cases} \end{cases}$$

Ces potentiels sont stockés dans une matrice T de dimension $L \times L$ où L est le nombre d'étiquettes du modèle et les coefficients associés sont :

$$T(i,j)(k) = I_k(\omega_i,\omega_j),\ 1 \leqslant i,j \leqslant n,\ 0 \leqslant k \leqslant 3. \tag{3.45}$$

➜ Les probabilités d'émission

On calcule ces fonctions de probabilité à partir d'images déjà étiquetées. Il faut faire un choix sur le type de primitives utilisées (pour notre application, cf. section 4.1.2), et sur le type de modélisation de ces lois de probabilité (cf. section 4.1.2). Dans le cas d'observations continues, on utilise en général une modélisation par des mélanges de gaussiennes, de la forme :

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

où $G(o,\mu,\Sigma)$ 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 couramment traité par une approche EM et très employé en reconnaissance de la parole [Huang *et al.*, 2001]. Si on observe des échantillons $o =$

$\{o_1, ..., o_N\}$, que l'on suppose 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 la convergence.

L'ajout d'une nouvelle composante gaussienne se fait après l'étape d'estimation. On perturbe la composante gaussienne principale (correspondant à la composante c_k maximale),

$$\begin{aligned} &P(k)G(o, \mu_k, \Sigma_k) \longrightarrow \\ &(1-\alpha_1)P(k)G(o, \mu_k, \Sigma_k) + \alpha_1 P(k)G(o, \mu_k + \alpha_2\Sigma_k, \Sigma_k), \end{aligned}$$

puis on réestime la nouvelle distribution multigaussienne par l'algorithme EM.

➜ **L'élagage**

L'algorithme de programmation dynamique 2D assigne d'abord à chaque pixel une valeur de vraisemblance pour chaque étiquette possible grâce aux densités de probabilité d'observation avant de fusionner les pixels en additionnant les log-vraisemblance d'observation et les termes d'observation (équation 3.43). Le principe de la programmation dynamique réduit considérablement l'espace de recherche, mais celui-ci est encore beaucoup trop grand pour être totalement exploré. On adopte donc une stratégie d'élagage pour ne garder, à chaque fusion, que les configurations les plus probables.

En dressant à nouveau le tableau de comparaison 1D/2D (tableau 3.3) on remarque que l'on peut utiliser à toutes les étapes les mêmes méthodologies pour analyser une image ou un signal temporel.

TAB. 3.3 : Parallèle entre les méthodologies 1D et 2D.

	1D	2D
Méthodologie	Évaluations régulières, bases publiques avec vérité terrain.	Peu de bases publiques (caractères, visages, suivi de véhicules...).
Analyse du signal brut	Fenêtres glissantes douces et recouvrantes (hamming...)	Fenêtres glissantes 2D
Extraction de primitives	Analyse spectrale (banc de filtres, cepstres...)	Analyse spectrale 2D (Gabor, cepstre 2D...)
Approche de la reconnaissance	Stratégie bayésienne	Stratégie bayésienne
Modélisation	Chaîne de Markov cachée	Champ de Markov caché
Attache aux données	Densités multigaussiennes	Densités multigaussiennes
Algorithme de décodage	Programmation dynamique	Programmation dynamique 2D

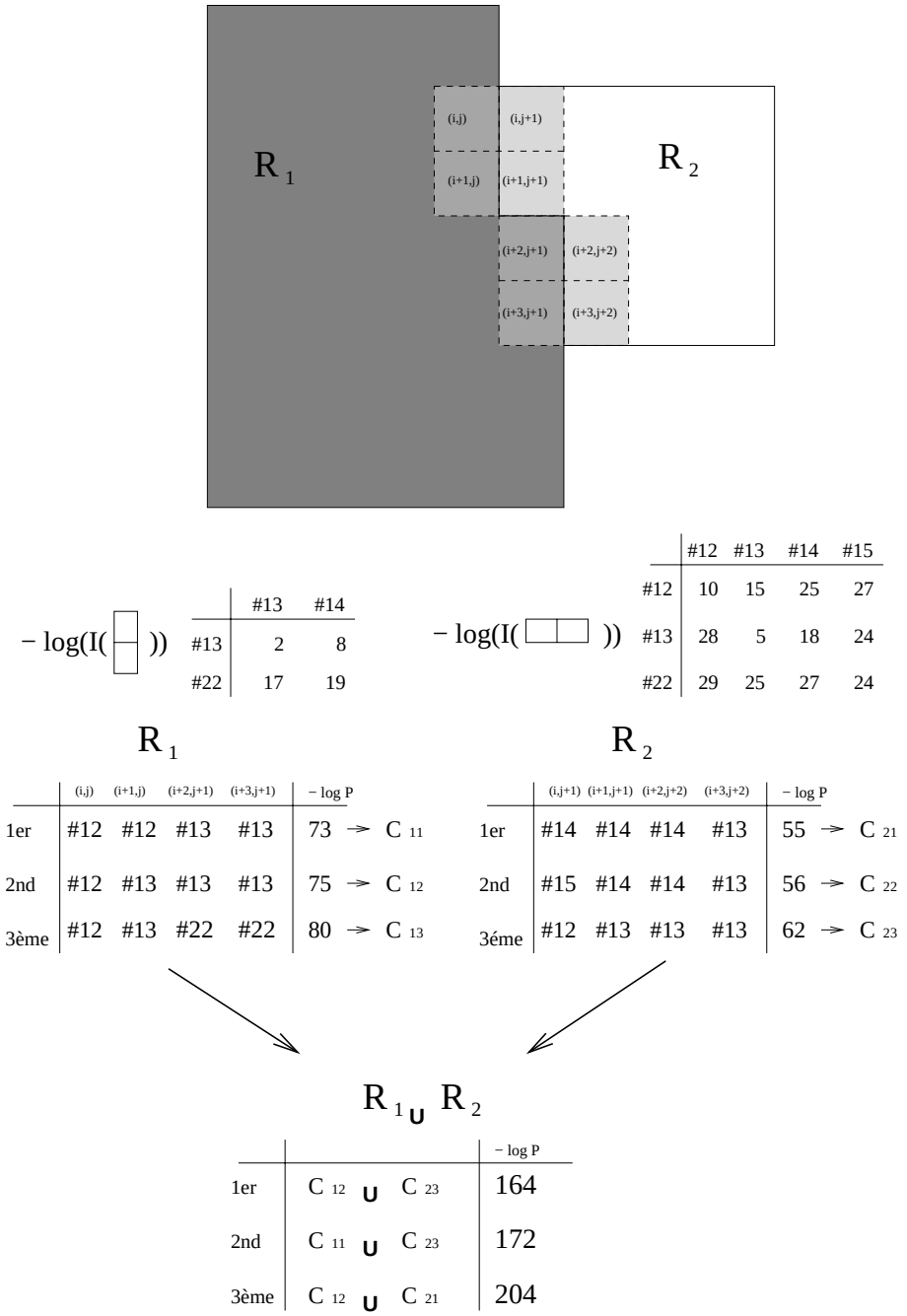


FIG. 3.12 : Fusion de deux régions : La région résultant de la fusion $R_1 \cup R_2$ est associée à une nouvelle liste de configurations possibles et une vraisemblance correspondante qui est la somme des vraisemblances des deux régions R_1 et R_2 et des termes d'interaction I.

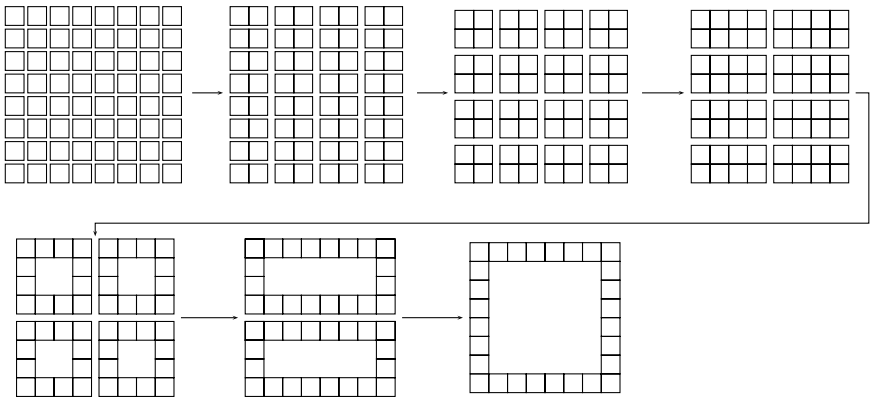


FIG. 3.13 : Une séquence de fusion canonique débutant par n^2 régions d'un seul pixel jusqu'à une seule région recouvrant l'image entière. À chaque étape, les régions sont fusionnées deux par deux. Seuls les pixels frontière sont représentés.

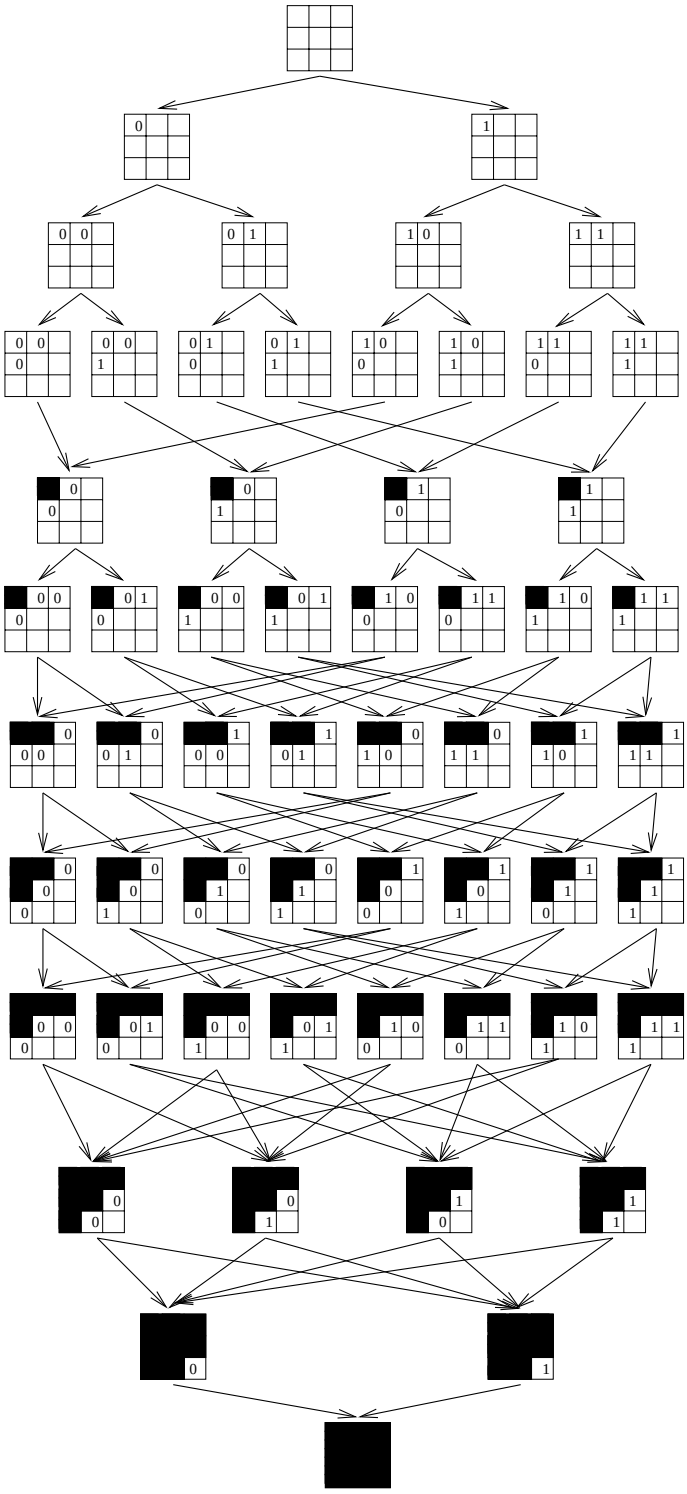


FIG. 3.14 : Déroulement de l'algorithme de programmation dynamique 2D sur une image 3×3 à deux états.

Chapitre 4

Application à la reconnaissance d'écriture manuscrite

Dans ce chapitre, nous exposerons notre approche du problème de la reconnaissance de caractères manuscrits ainsi que son extension au cas de la reconnaissance des mots cursifs. Ces expériences sont effectuées sur des bases de données publiques que nous décrirons.

4.1 Reconnaissance de caractères manuscrits

Dans un premier temps, nous avons traité le problème de la reconnaissance de chiffres manuscrits. Une base de données est disponible publiquement, ce qui permet de comparer les performances obtenues avec d'autres systèmes. Ces expériences ont été en partie décrites dans [Chevalier *et al.*, 2003] et [Chevalier *et al.*, 2004].

Nous décrirons tout d'abord la base de données utilisée pour ces expériences, puis la mise en œuvre de l'approche présentée. Nous exposerons ensuite nos expériences sur la base de développement qui déterminent les paramètres de nos algorithmes. Nous donnerons enfin les résultats obtenus sur la base de test ainsi que leur analyse.

4.1.1 Base de données utilisée

Les premières expériences de l'algorithme de programmation dynamique 2D pour la reconnaissance d'écriture manuscrite ont été conduites sur la base de données MNIST (cf. section 2.2.1) qui est une base standard composée de 70.000 images de chiffres extraits de la base NIST (Figure 2.2) et divisées en une base d'apprentissage et une base de test. 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 est gardé autour des caractères,
- des niveaux de gris résultent des prétraitements de normalisation en taille,
- les bases d'apprentissage et de test on été écrites par des scripteurs distincts.

Pour garantir la validité d'une évaluation, il est nécessaire de n'utiliser la base de test qu'au cours de tests finaux, les algorithmes et paramètres ne doivent en aucun cas être appris sur cette base de test. Ainsi, une base de données est idéalement découpée en trois parties :

- une base d'apprentissage,
- une base de validation,
- une base de test.

Le développement de l'algorithme se fait alors en testant les modèles appris sur la base d'apprentissage sur la base de validation. Seuls quelques tests finaux sont effectués sur la base de test, en utilisant des modèles appris sur l'ensemble des données des bases d'apprentissage et de validation.

La base MNIST est découpée en une base d'apprentissage et une base de test. Pour notre développement, nous avons donc prélevé une partie de la base d'apprentissage pour en faire une base de validation, le reste de la base d'apprentissage étant appelé base de développement. Le test final a été effectué sur la base de test, en utilisant l'ensemble des données de la base d'apprentissage pour le calcul des modèles. Cette partition est illustrée figure 4.1.

Il est intéressant d'observer les résultats obtenus avec un même système en choisissant différentes bases de validation pour observer la stabilité de la qualité des données dans la base. La courbe 4.2 montre le comportement du taux d'erreur de notre système calculé pour différentes bases de validation dont la position est illustrée figure 4.3. Pour chaque classe, les images 1 à 6000 correspondent à la base d'apprentissage, les images suivantes correspondent à la base de test. Les deux courbes étant quasiment identiques, on peut en déduire que les performances de notre modélisation ne sont pas améliorées par l'ajout d'images d'apprentissage supplémentaires et qu'elle est donc probablement perfectible par exemple par l'augmentation du nombre de paramètres. La courbe est de plus quasiment stable, i.e. tous les sous-ensembles de la base sont représentatifs de la base entière, et la base de test est légèrement plus facile que la moyenne (3,1% d'erreurs contre 3.4%). Cela valide notre choix de partitionnement en développement, validation et test.

4.1.2 Mise en œuvre

Approche générale

On cherche à étiqueter les pixels de l'image de manière à obtenir des zones homogènes en termes de traits, informations obtenues grâce à une extraction de vecteurs de primitives adéquates sur les images (cf. section 4.1.2), et dont la vraisemblance serait maximale compte tenu d'un modèle. Ces zones correspondent aux états possibles du champ de Markov et on cherchera à obtenir une structure de grille qui segmentera les différents types de traits (Figure 4.4).

L'algorithme de programmation dynamique 2D est utilisé à la fois au niveau de l'apprentissage, et au niveau de la reconnaissance. On associe à chacune des classes (10 classes au moins pour une tâche de reconnaissance de chiffres) un modèle de champ de Markov

caché appris par un processus itératif de type apprentissage de Viterbi. Durant la phase de reconnaissance, on calcule les vraisemblances des images pour chaque modèle, puis on sélectionne le meilleur score.

La structure des algorithmes d'apprentissage et de reconnaissance peut être ainsi résumée :

- **Apprentissage**
  1. Initialisation : Pour chaque classe, les images sont segmentées en zones suivant un maillage régulier 5×7 (plus petit maillage supposé pouvoir modéliser le caractère le plus complexe, le 8, et choix validé par l'expérience sur d'autres topologies). Chaque zone est associée à une étiquette et les densités d'observation sont calculées pour tous les pixels d'une même étiquette. La matrice de transition initiale définie dans l'équation 3.45 est aussi calculée sur ce maillage en comptant les transitions observées sur cette segmentation initiale et en en déduisant les termes d'interaction définis dans l'équation 3.44 (Figure 4.5). Une probabilité faible mais non nulle est cependant affectée aux transitions non observées.
  2. Récursion : Chaque image de la base d'apprentissage est segmentée grâce à l'algorithme de programmation dynamique 2D pour maximiser la vraisemblance. Les nouveaux paramètres sont alors appris sur ces segmentations (modèles d'émission et de transition) et ce jusqu'à convergence de la vraisemblance (Figure 4.6). À chaque itération la structure de treillis est préservée car les transitions très peu probables ne sont quasiment jamais observées et restent donc improbables.
- **Reconnaissance**
  La configuration optimale de chaque image de test est calculée pour chacun des modèles avec la programmation dynamique 2D. Le modèle donnant la plus grande vraisemblance est ensuite sélectionné.

La stratégie d'apprentissage est une approche EM (voir section 3.2.4) avec un critère MAP. La reconnaissance est de type bayésien, chaque classe ayant la même probabilité d'apparition *a priori*.

Extraction des vecteurs de primitives

Le choix du type de primitives utilisées en reconnaissance de l'écriture manuscrite dépend directement du type de modélisation utilisé. Plusieurs études ont été menées pour déterminer le pouvoir discriminant des primitives [Baret, 1990, Simon et Baret, 1990] et sur le choix du type de primitives [Grandidier *et al.*, 2000]. Comme déjà mentionné dans la section 2.1.2, on peut distinguer deux classes dans les types de primitives utilisées : les primitives globales et locales.

Les primitives globales, décrites en détails dans [Trier *et al.*, 1996], recherchent des invariants dans l'image entière, c'est donc surtout au niveau de l'extraction de primitives qu'on modélisera la variabilité intra-classe. Les primitives locales sont calculées dans des fenêtres autour des pixels de l'image. Pour la reconnaissance de caractères isolés, les primitives

cellulaires [Hildebrandt et Liu, 1993] sont très employées et visent à extraire la structure en termes de traits. Les algorithmes à base de chaînes de Markov cachées extraient des primitives dans des graphèmes ou pseudo-lettres comme les boucles, les croisements de traits ou les extrémités [Feray et de Brucq, 1996, Kuo et Agazzi, 1994].

Nous utilisons des primitives spectrales locales. Une FFT est calculée dans une fenêtre gaussienne autour des pixels. Le logarithme du module de la transformée est calculé et les premiers coefficients sont gardés (Figure 4.7). Le premier coefficient contient l'information de l'énergie de la fenêtre et les quatre suivants donnent les intensités dans les quatre directions principales (Figure 4.8). Le premier coefficient n'est pas pertinent pour notre modélisation car il dépend trop de la largeur du trait, on utilise donc comme vecteur de paramètres les quatre coefficients suivants. Ces primitives, transformées de Fourier fenêtrées, sont connues sous le nom de primitives de Gabor, fondées sur les filtres de Gabor [Jain et Bhattacharjee, 1992, Lampinen *et al.*, 2001]. L'ajout des deux premières composantes de phase permet d'améliorer les performances obtenues.

En pratique, il n'est pas nécessaire de calculer un vecteur de paramètres pour chaque pixel, on peut centrer la fenêtre de calcul des primitives sur une partie des pixels seulement. Le nombre de sites à étiqueter étant déterminant pour le temps de calcul, il est très intéressant de le diminuer. Diviser le nombre de vecteurs de primitives par 4, nous a permis de réduire d'autant le temps de calcul sans affecter le taux de reconnaissance.

Modélisation des densités d'observation

Les vecteurs d'observation, issus du calcul des primitives spectrales locales (cf. section 4.1.2), sont modélisés par une fonction multigaussienne (cf. section 3.4.3). À chaque itération de la phase d'apprentissage, après le décodage des images dont on déduit un ensemble de valeurs des observations pour chaque état, ces densités d'observation sont calculées. En pratique, les paramètres à fixer sont :

- le nombre maximal de composantes gaussiennes,
- le seuil d'arrêt des itérations de l'algorithme EM,
- une valeur minimale de l'écart-type des composantes gaussiennes.

La figure 4.9 montre que les fonctions multigaussiennes modélisent bien les distributions réelles (8 composantes gaussiennes, 8 itérations, écart-type minimal fixé à 50).

Il est intéressant d'observer les histogrammes des états des modèles correspondant aux 10 classes dans la configuration dans laquelle ils apparaissent dans l'image. Les figures 4.10 et B.1 à B.9 illustrent les allures des histogrammes des distributions associées aux 35 états des modèles pour chacun des 6 coefficients des vecteurs d'observation. La forme de chacune des classes apparaît nettement et chacune des 6 composantes semble apporter des informations pertinentes.

Élagage

Le principe de l'élagage est décrit dans la section 3.4.3. Notre stratégie, à chaque fusion de deux régions est d'agir en deux étapes :

1. Les configurations dont la différence de vraisemblance avant et après la fusion est au dessus d'un seuil sont élaguées.
2. On ne garde ensuite qu'un nombre maximal de configurations à chaque étape.

Seules les configurations les moins prometteuses, loin de la configuration optimale sont élaguées, la configuration optimale globale est donc généralement préservée. En pratique, dans nos expériences, le seuil d'élagage (niveau 1) agit seulement sur les configurations qui comportent une transition improbable qui ne préserve pas la structure en treillis. Le nombre maximal de configurations de la frontière (niveau 2) est un compromis entre la qualité de la segmentation et le temps de calcul.

Stratégie de fusion

La stratégie de fusion est l'ordre dans lequel on fusionne les pixels jusqu'à obtenir l'image entière. En théorie, sans élagage, cet ordre n'a aucune influence sur la solution trouvée. Il est cependant déterminant pour le temps de calcul et l'espace mémoire utilisé. En pratique, il interagit avec l'élagage. Dans le cas des images de la base MNIST, les pixels du bord de l'image sont toujours blancs, on peut donc fixer leur étiquette à la même valeur que dans la phase d'initialisation illustrée par la figure 4.5. On fusionne ensuite les pixels, en commençant par ceux situés au bord de l'image pour lesquels l'incertitude est la moins grande, puis les pixels de plus en plus proches du centre (Figure 4.11).

4.1.3 Expérimentations et optimisation des paramètres

Bien qu'un petit nombre de paramètres soit requis pour le fonctionnement de l'algorithme, il est nécessaire de déterminer leur valeur à partir d'expérimentations effectuées uniquement sur la base d'apprentissage pour utiliser un algorithme et des modèles complètement indépendants de la base de test. On a donc défini une base de développement issue de la base d'apprentissage de MNIST (cf. 4.1.1). Dans cette partie, on donnera les taux d'erreur obtenus en faisant varier différents paramètres de façon à obtenir les paramètres optimaux. Les conditions expérimentales pouvant varier, les taux d'erreurs obtenus pour ces différents test ne se recoupent pas toujours.

Paramètres du calcul des vecteurs de primitives

Le calcul des primitives est décrit dans la section 4.1.2 et illustré plus en détails dans l'annexe A. Si ces illustrations permettent de déterminer quels paramètres seraient optimaux en théorie, la figure 4.12 et le tableau 4.1 montrent que le choix de fenêtres de taille 7×7 est bien judicieux en pratique. Une diminution de la fenêtre d'analyse est pénalisante mais son augmentation serait inutile et coûteuse en temps de calcul.

De plus, pour observer la pertinence des composantes des vecteurs de primitives, on peut observer les performances en reconnaissance de notre système en utilisant seulement certaines des composantes. Le tableau 4.2 donne les résultats de reconnaissance.

TAB. 4.1 : Taux d'erreurs en fonction de la taille des fenêtres gaussiennes.

Taille des fenêtres	Taux d'erreurs
3×3	17.2%
5×5	5.2%
7×7	5.1%
9×9	5.1%

TAB. 4.2 : Taux d'erreurs du processus de reconnaissance en fonction du type de primitives utilisé.

primitives	nombre de coefficients	Taux d'erreurs
module	4	5.5%
phase	2	7.8%
module+phase	6	2,9%

Pour démontrer la pertinence du choix de ce type de primitives et déterminer les paramètre optimaux, il est utile d'observer la faculté de reconstruction des images à partir des paramètres extraits. Cette étude est illustrée dans l'annexe A.

Paramètres de la modélisation des densités d'observation

1. **Nombre de composantes gaussiennes**
   L'observation émise par chacun des états est modélisée par une densité de probabilité multigaussienne. La complexité du calcul de ces probabilités est un élément clé dans le coût total de l'algorithme. En particulier, si M est le nombre maximal de composantes gaussiennes, le temps de décodage d'une image est presque linéaire en M. Il est donc nécessaire de limiter ce nombre. Cependant on observe que le résultat final du taux de reconnaissance est sensible à ce nombre. Le tableau 4.3 montre l'influence du nombre de gaussiennes sur le taux de reconnaissance.

TAB. 4.3 : Influence du nombre de composantes gaussiennes.

Nombre de composantes gaussiennes maximum par état	Taux d'erreurs associé
4	2,96%
8	2,79%
12	2,51%
16	2,39%
20	2,36%

Il est nécessaire d'effectuer un compromis entre performance et temps de calcul. Avec $M = 8$, l'algorithme de décodage traite environ deux images de la base MNIST par

seconde. Au dessus de 16 composantes le taux d'erreurs semble atteindre un pallier (Figure 4.13).

2. **Nombre de distributions**
   Chacun des états de chacun des modèles est associé à une distribution mais en pratique, de nombreuses distributions sont très proches, comme celles associés aux états émettant du blanc (Figures 4.10 et B.1 à B.9). Une approche possible est de disposer d'un "pool" de distributions multigaussiennes, chacun des états pointant sur l'une des distributions de ce *pool*. Ainsi, on a moins de probabilités d'émission à évaluer. Pour constituer ce *pool* de distributions, on commence par associer une distribution par état, puis on regroupe deux à deux les distributions dont la distance est au-dessous d'un certain seuil. C'est en faisant varier ce seuil qu'on ajuste le nombre de distributions différentes.
   La figure 4.14 donne le taux d'erreur en fonction du nombre total de distributions (au maximum 350 distributions pour 10 modèles à 7×5 états). On remarque que l'on peut réduire le nombre de distributions de plus d'un tiers sans affecter significativement les performances. La difficulté réside dans le fait qu'il est difficile de fixer *a priori* le seuil de regroupement des distributions, en particulier parce que la distance entre deux distributions se calcule sur une modélisation monogaussienne.
   L'allure des densités d'observations (Figures 4.10 et B.1 à B.9) incite à adopter une stratégie légèrement différente pour le partage des états : on peut forcer dès le départ les 20 états situés sur le tour de l'image (états 0 à 4, 30 à 34 et 5, 9, 10, 14 etc.) à partager la même distribution. Le tableau 4.4, établi sur notre base de validation, montre qu'on a certes une dégradation du taux d'erreurs mais très petite comparée à ce qui est obtenu par la stratégie de fusion automatique des distributions présentée précédemment (qui revient à comparer les résultats pour 160 et 350 distributions). Ce résultat sera utile pour justifier notre choix de fusion des distributions dans le cas du traitement des mots (chapitre 4.2) pour lequel on définit une distribution modélisant les espaces entre les mots et éventuellement entre les lettres.

Tab. 4.4 : Partage forcé des gaussiennes des états du tour de l'image.

Partage	Taux d'erreur associé
N	**2,79%**
O	3,03%

Élagage et liberté de déformation

La stratégie d'élagage au cours des fusions est déterminante pour la qualité des résultats obtenus (section 3.4.3). Le nombre maximal de configurations gardées à chaque fusion est fixe, et on peut observer son influence sur le taux d'erreur sur la figure 4.15. On remarque que garder moins de 10 configurations à chaque fusion entraîne des dégradations importantes en termes de taux de reconnaissance, et à l'inverse, en garder plus est inutile.

Il faut cependant considérer le fait que pour limiter l'espace de recherche, chaque site de l'image ne peut être étiqueté que par une partie des états dont la position dans la grille d'initialisation n'est pas trop éloignée (Figure 4.5). L'indice de liberté détermine le nombre d'états (dans chaque direction) dont l'étiquette d'un site peut varier par rapport à l'initialisation. L'indice de liberté détermine donc la liberté de déformation de l'alignement. Le tableau 4.5 montre l'évolution du taux d'erreurs en fonction de l'indice de liberté et de l'élagage. On remarque que pour tirer partie d'une liberté de déformation plus grande (coûteuse en temps de calcul), la modélisation doit être accompagnée d'une stratégie d'élagage plus lâche (elle aussi consommatrice de temps de calcul). Ce résultat est logique dans la mesure où l'augmentation du nombre de configurations initialisées à chaque étape accroît le risque d'éliminer des configurations utiles par un élagage trop brutal.

Indice de liberté	Élagage	Taux d'erreur associé
2	30	**2,79%**
2	50	2,85%
3	30	2,82%
3	50	2,73%

Tab. 4.5 : Influence de la liberté de déformation.

Évolution du taux d'erreurs

Lors du développement notre système de reconnaissance, chaque proposition d'évolution et chaque réglage de paramètre a donné lieu à une évaluation pour tester sa pertinence. Les propositions retenues sont celles ayant conduit à une diminution du taux d'erreurs sur la base de validation. Le tableau 4.6 résume l'évolution du taux d'erreurs sur différentes parties de la base MNIST. L'évolution est due principalement à l'amélioration :

- du type de primitives utilisé (valeur du pixel, puis FFT avec différents choix de coefficients),
- de la modélisation des densités de probabilité des observations (constante par morceaux, puis multigaussiennes),
- de la topologie des modèles.

Dans le même temps, la complexité des modèles a été réduite par :

- la diminution de la taille du vecteur de primitives,
- la simplification de la topologie des modèles,
- la diminution du nombre de sites dans les images.

Il faut aussi noter que les bases d'apprentissage et de test utilisées pour le développement ont évolué pour permettre le calcul de modèles plus performants, jusqu'aux expérimentations sur l'ensemble de la base MNIST.

Les premiers test ont été effectués avec les modules des 5 premiers coefficients de la FFT (FFT 5 coeffs), puis les 9 premiers (FFT 9 coeffs). L'élimination du premier coefficient, qui était trop dépendant de la taille du trait a permis d'améliorer les résultats (FFT 4 coeffs).

Par la suite, l'ajout des phases des deux premiers coefficient (FFT 6 coeffs) a encore permis d'améliorer les résultats.

TAB. 4.6 : Évolution du taux d'erreur. Seuls les tests en gras ont été effectués sur la base de test de MNIST.

Primitives	ddp des observations	Topologie	Appr	Test	Tx err.
Pixel	cte par morceaux 2n	5×7	100	100	26,7%
Pixel	cte par morceaux 4n	5×7	100	100	13,6%
FFT 5 coeffs	multigaussienne	5×7	1000	100	5,4%
FFT 5 coeffs	multigaussienne	5×7	1000	200	6,2%
FFT 9 coeffs	multigaussienne	10×14	1000	100	3,5%
FFT 9 coeffs	multigaussienne	10×14	1000	200	5,1%
FFT 9 coeffs	multigaussienne	10×14	1000	300	5,6%
FFT 5 coeffs	**multigaussienne**	5×7	**6000**	**1000**	**5,5%**
FFT 4 coeffs	**multigaussienne**	5×7	**6000**	**1000**	**3,7%**
FFT 6 coeffs	multigaussienne	5×7	5000	1000	2,8%
FFT 6 coeffs	**multigaussienne**	5×7	**6000**	**1000**	**2,7%**

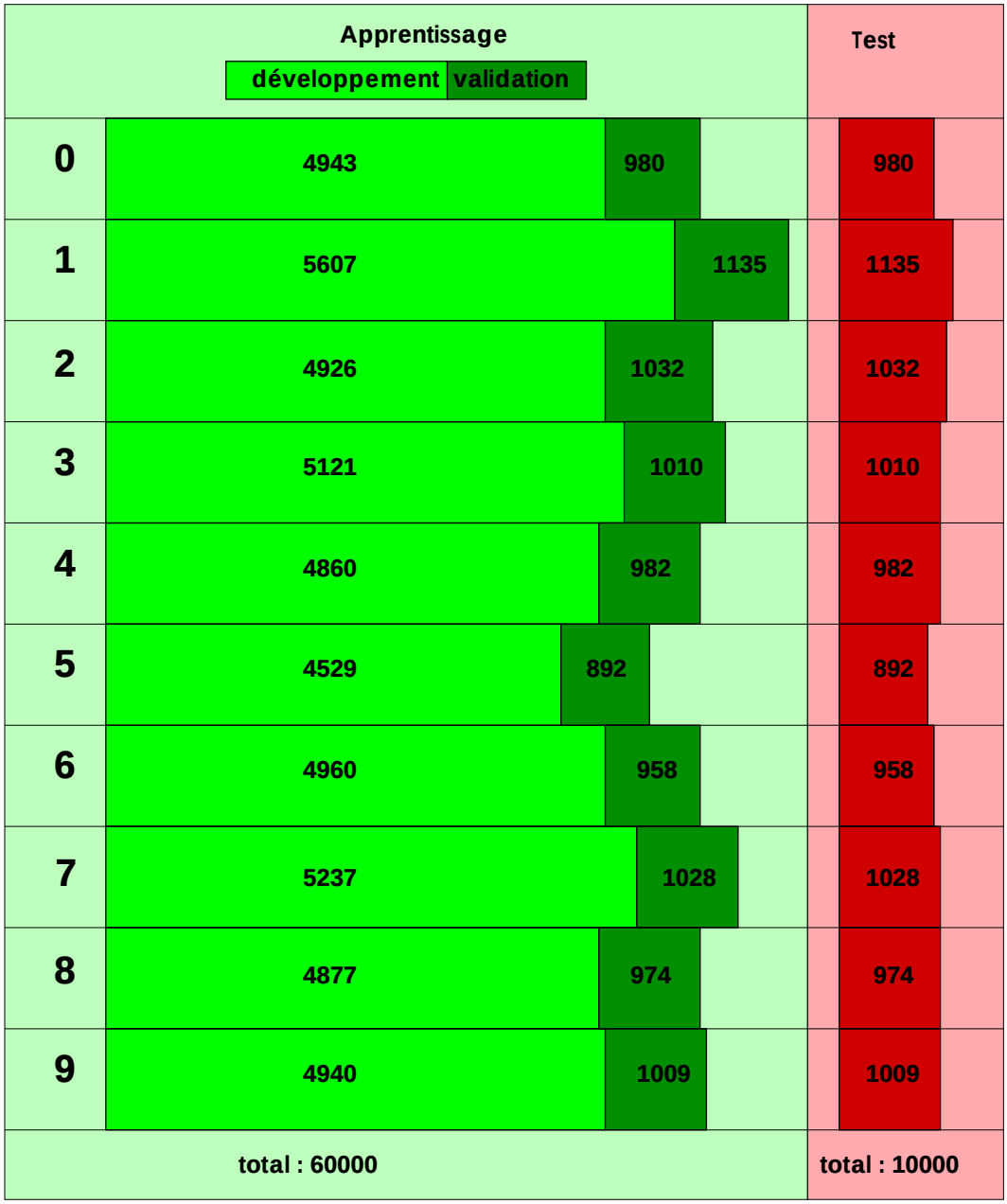


FIG. 4.1 : Composition de la base MNIST.

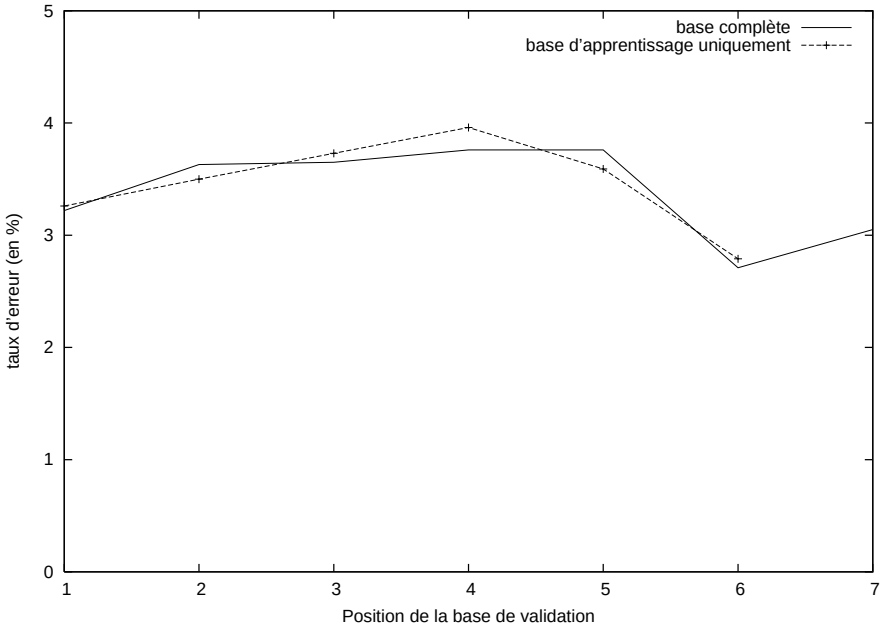


FIG. 4.2 : Stabilité de la base MNIST.

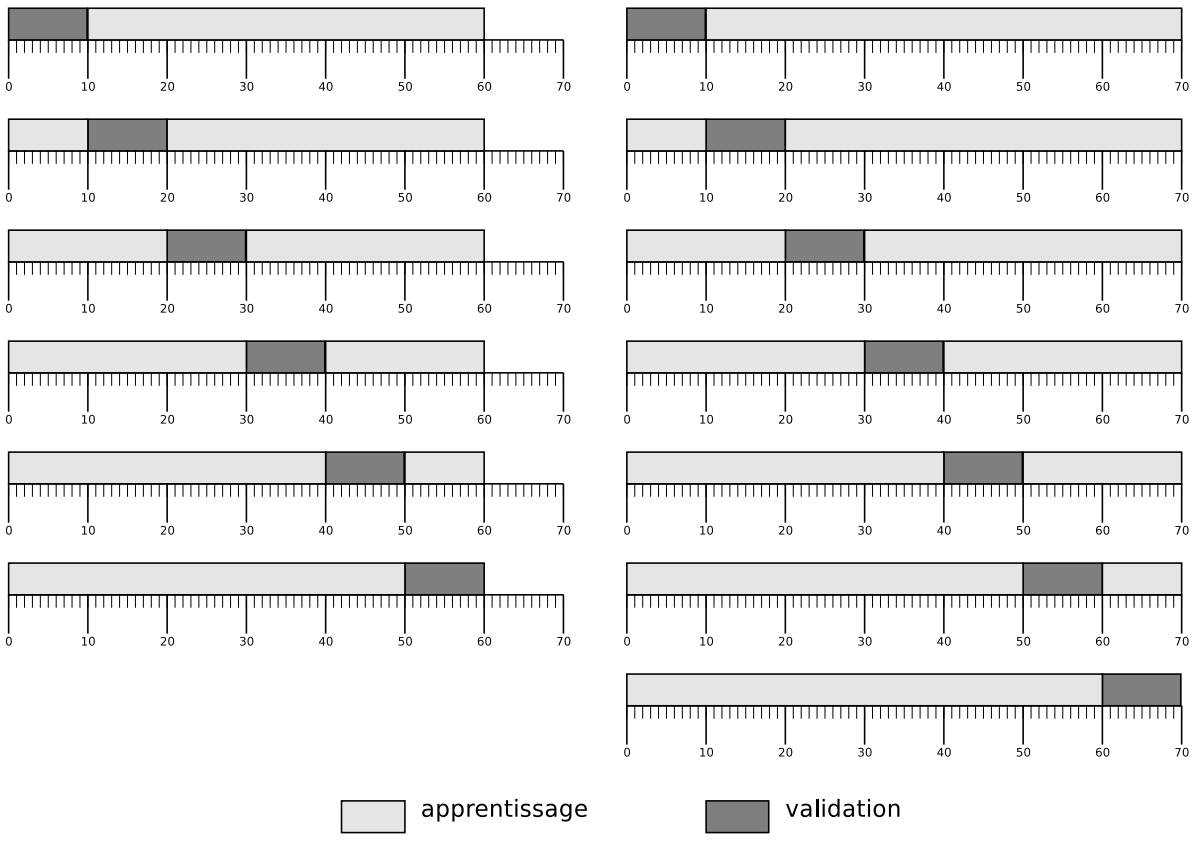


FIG. 4.3 : Positions de la base de validation dans la base MNIST.

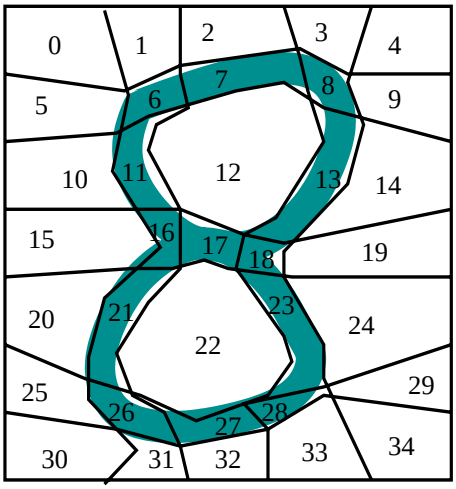


FIG. 4.4 : Configuration attendue d'une image.

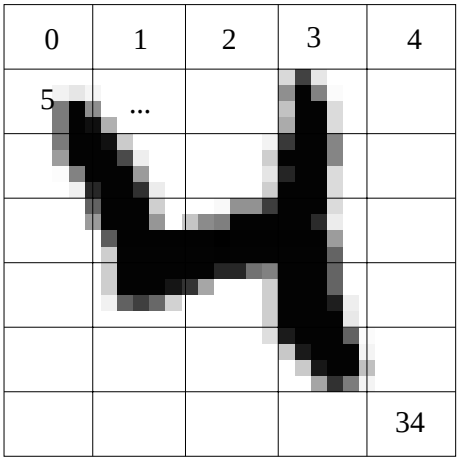


FIG. 4.5 : Initialisation : segmentation initiale utilisée pour le calcul des paramètres du modèle initial.

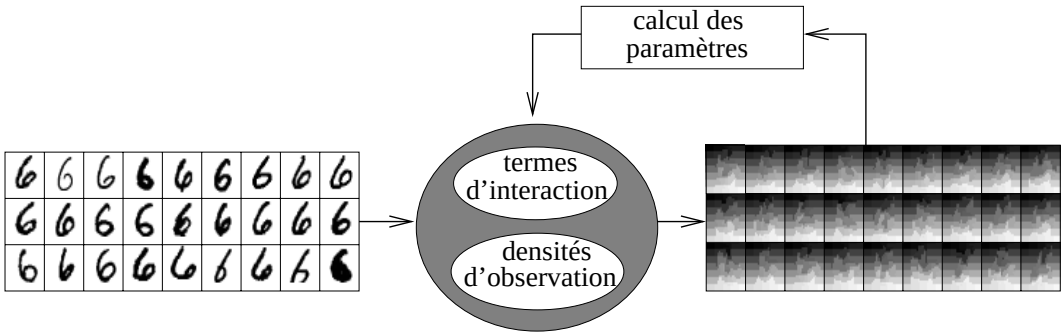


FIG. 4.6 : Apprentissage : À chaque itération, les paramètres du modèle sont calculés sur les segmentation issues de l'itération précédente.

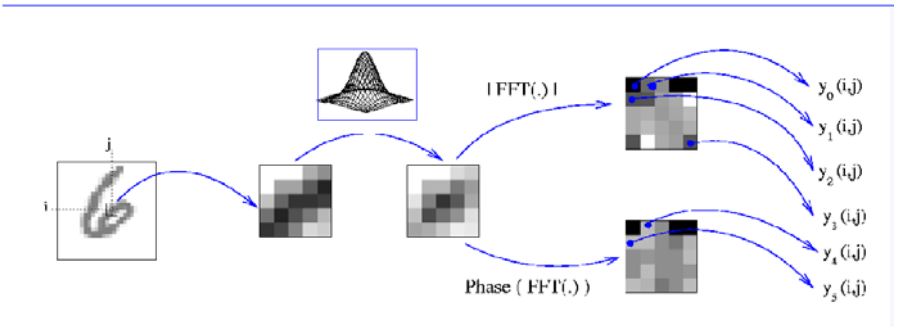


FIG. 4.7 : Processus d'extraction des primitives.

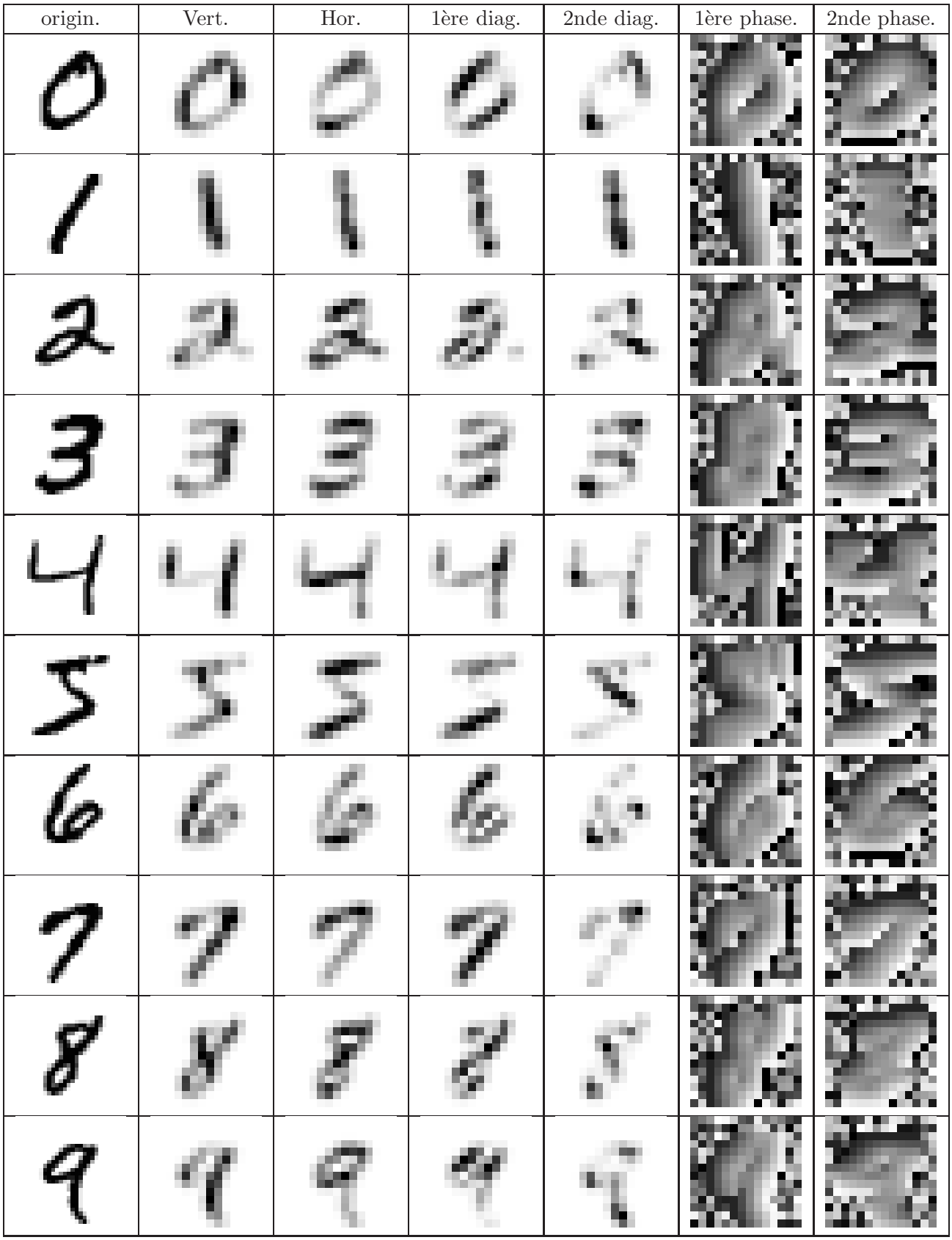


FIG. 4.8 : Les primitives spectrales locales : à droite de l'image originale sont représentées les 6 composantes des vecteurs d'observation.

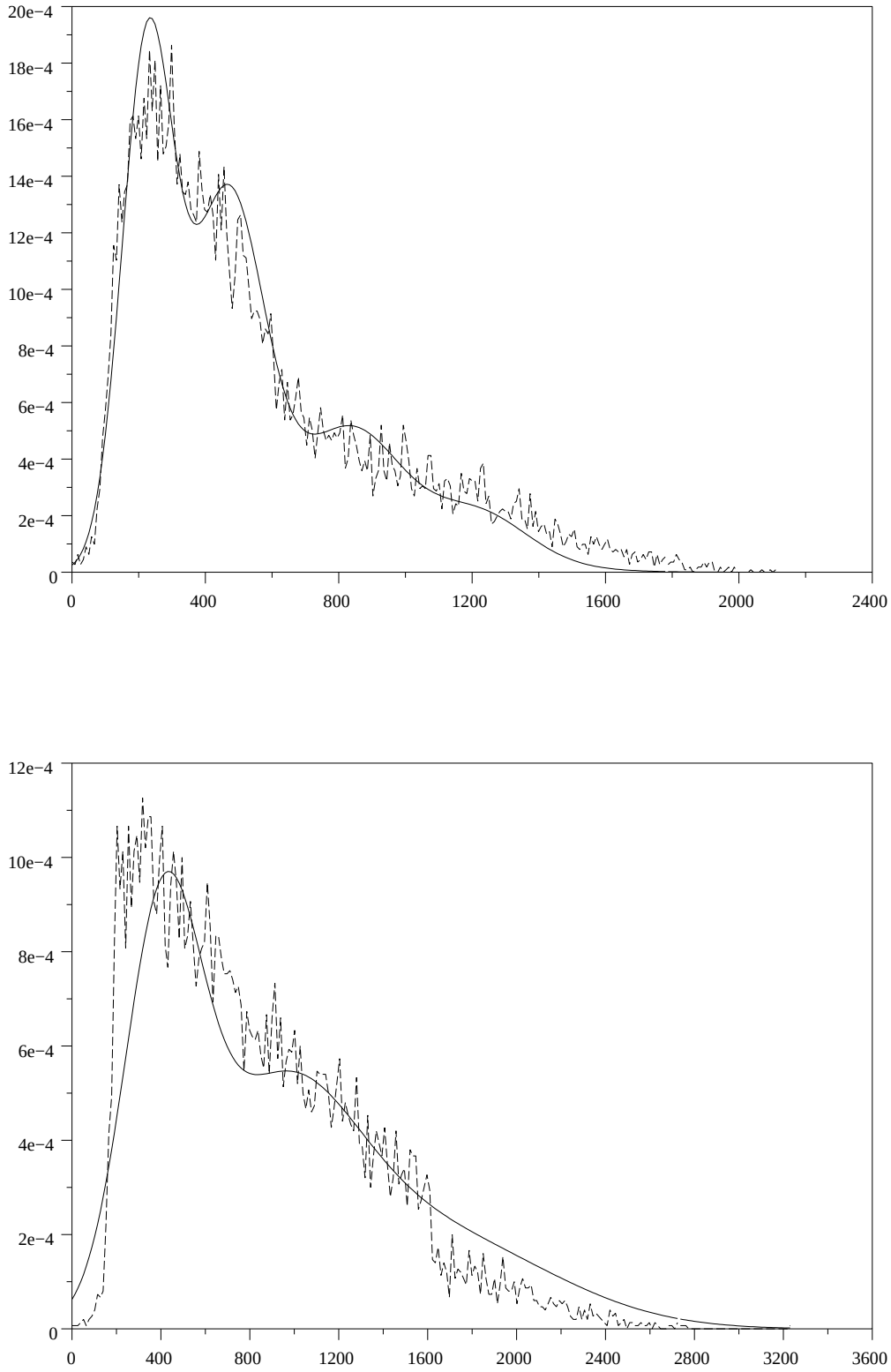


FIG. 4.9 : Exemples d'histogrammes des données d'apprentissage et fonctions multigaussiennes associées (2 premières composantes des sites affectés à l'étiquette 8 (Figure 4.4) des données d'apprentissage de la classe du chiffre 0).

FIG. 4.10 : Histogrammes du modèle 0 : pour les 6 composantes des vecteurs d'observations, les histogrammes des 35 états sont représentés.

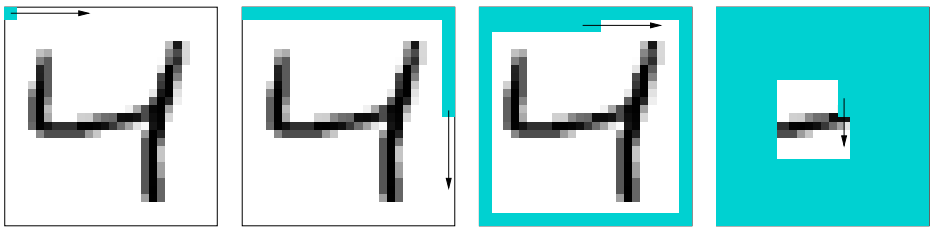


FIG. 4.11 : Stratégie de fusion : les pixels du tour de l'image sont fusionnés en premier, avant ceux du centre de l'image.

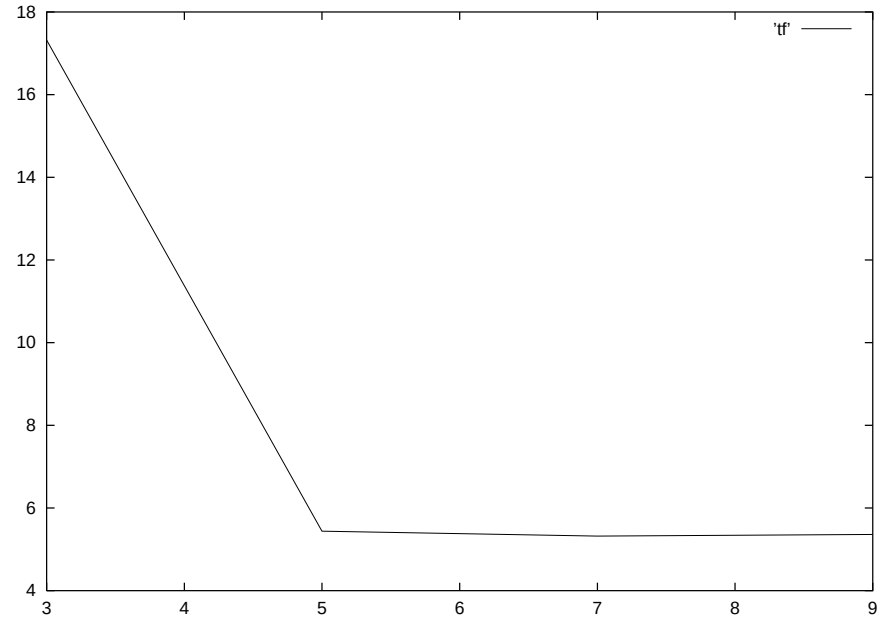


FIG. 4.12 : Taux d'erreurs en fonction de la taille des fenêtres gaussiennes pour le calcul des primitives.

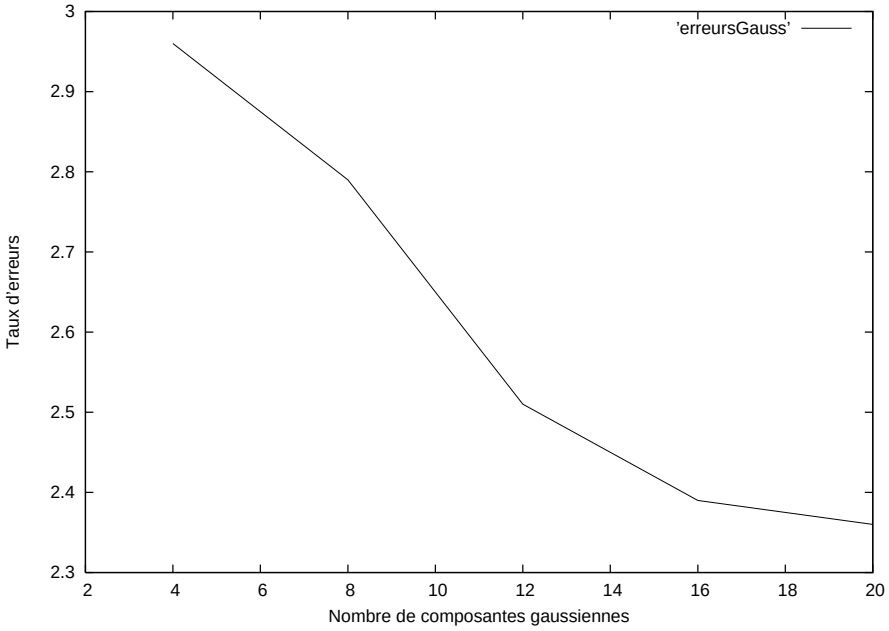


FIG. 4.13 : Taux d'erreurs en fonction du nombre maximal de composantes des multigaussiennes.

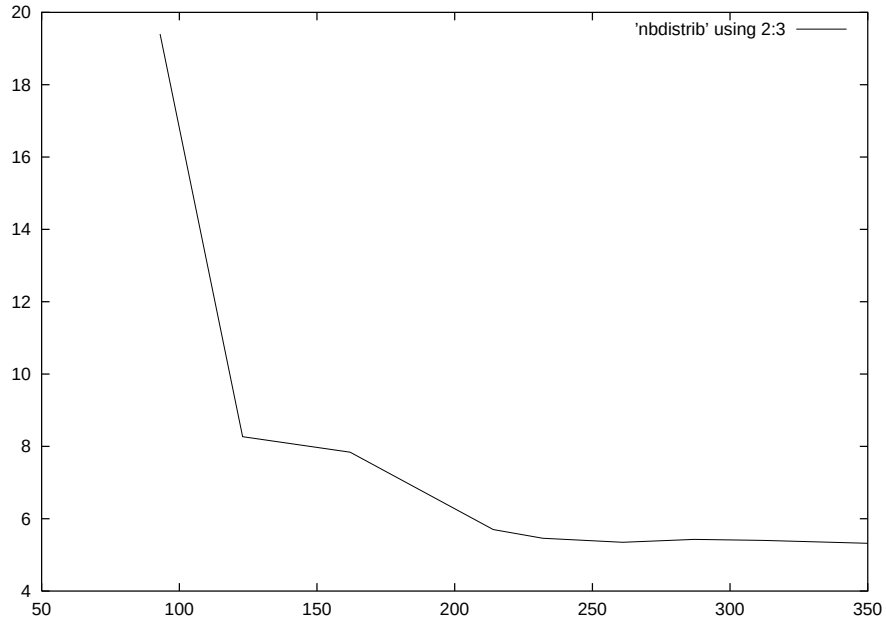


FIG. 4.14 : Taux d'erreur en fonction du nombre de distributions multigaussiennes différentes (maximum 350).

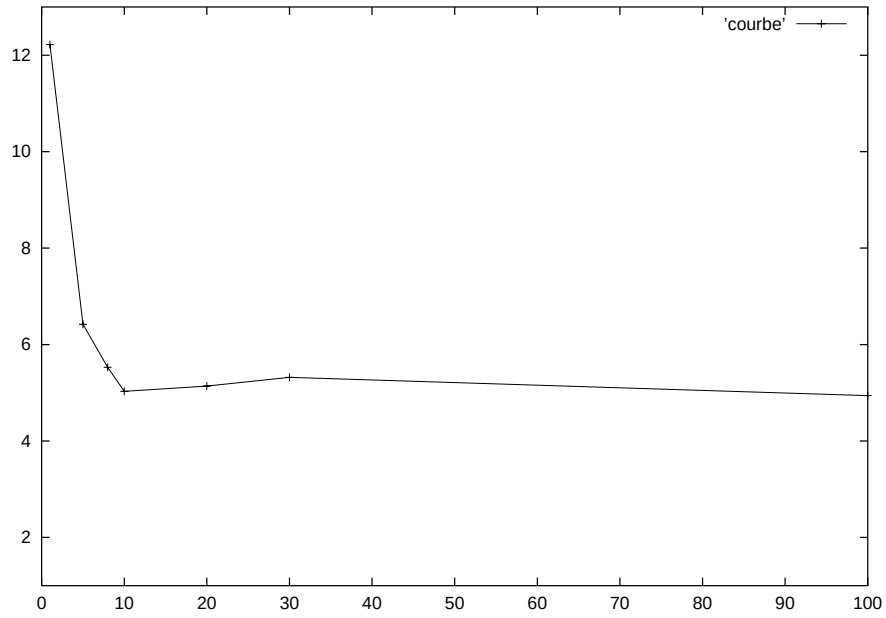


FIG. 4.15 : Taux d'erreur en fonction du nombre de configurations gardées dans l'élagage.

TAB. 4.7 : Paramètres de la meilleure modélisation et taux d'erreurs associé.

nombre d'étiquettes	gaussiennes par état	fenêtre gaussienne	élagage	nombre d'observables	nombre d'itérations	taux d'erreurs
5×7	20	7×7	30	14×14	6	2,74%

4.1.4 Résultats sur la base de test

Les tests ont été effectués en gardant un nombre d'itérations fixe dans le calcul du modèle. On peut observer la convergence des modèles en traçant la vraisemblance des données d'apprentissage en fonction du nombre d'itérations (Figures 4.16 et 4.17).

Une caractéristique importante de notre approche est que le développement a été effectué uniquement sur la base d'apprentissage de MNIST. Une partie de cette base est utilisée pour l'apprentissage des modèles, l'autre pour l'évaluation des taux de reconnaissance. La base de test de MNIST n'est donc pas utilisée pour l'optimisation des paramètres mais uniquement pour l'évaluation finale des modèles, qui sont alors appris sur la totalité de la base d'apprentissage. Cette méthode donne des résultats plus réalistes qu'une optimisation des paramètres sur la base de test. On a retenu les meilleures valeurs des paramètres conduisant à un temps de calcul raisonnable, ils sont recensés dans le tableau 4.7.

On peut tracer les limites des zones de même étiquette (voir figure 4.18), on s'aperçoit alors que les images ont bien été segmentées suivant les traits et leur direction et que la structure de grille est bien préservée.

Le taux d'erreur atteint est de 2,7% sur l'ensemble des données de test de la base MNIST, ce qui est proche des meilleurs résultats publiés dans la littérature. Les détails des résultats des différentes expériences sont donnés dans la section 4.1.3.

Influence du rejet

Dans les applications pratiques, on utilise une politique de rejet qui évite de prendre une décision quand celle-ci serait trop incertaine. Plusieurs stratégies peuvent être envisagées :

- une stratégie de rejet par seuil absolu où seules les probabilités supérieures à un seuil sont retenues,
- une stratégie par seuil relatif où la décision n'est prise que si la différence entre les deux plus grandes probabilités est supérieure à un seuil.

La figure 4.19 montre les résultats obtenus pour les deux stratégies. Le rejet relatif est nettement plus efficace et permet d'obtenir un taux d'erreurs inférieur à 1% pour un rejet de 10% des images d'entrée. Le score de confiance ainsi obtenu est un indicateur utile à l'analyse des résultats du processus de reconnaissance car il permet d'évaluer la difficulté de classification d'un échantillon.

Analyse des meilleures reconnaissances

Le score de confiance décrit dans la section 4.1.4 est utile à l'analyse des modèles. On peut, pour chaque classe, classer les échantillons suivant leur score de confiance donné par le modèle et déterminer ainsi la forme-type des caractères du point de vue du modèle. La figure 4.20 recense les 15 premières images reconnues avec la plus grande confiance par chacun des modèles. On remarque que, pour chaque classe, la forme de ces caractères est très homogène et est donc probablement représentative du prototype défini par le modèle. Le cas de la classe 7 est un peu différent car deux modèles ont été utilisés pour la modéliser, l'un pour le 7 sans barre, l'autre pour le 7 barré.

Analyse des confusions

Une grande partie des confusions peut être évitée par la stratégie de rejet décrite dans la section 4.1.4. Une autre partie de ces échantillons mal classifiés a un score de confiance trop important pour être rejeté. Il est intéressant d'observer ces échantillons et de déterminer quel serait le comportement d'un opérateur humain sur la même tâche. La figure 4.21 recense les confusions du processus de reconnaissance par ordre de confiance (les premières images sont les premières rejetées si on introduit un seuil de rejet). La légende de chacune de ces images est de la forme :

`classe réelle` → `classe reconnue`

On remarque que la classification de la plupart de ces échantillons serait problématique aussi pour un opérateur humain, en particulier pour les images ayant les taux de confiance les plus élevés.

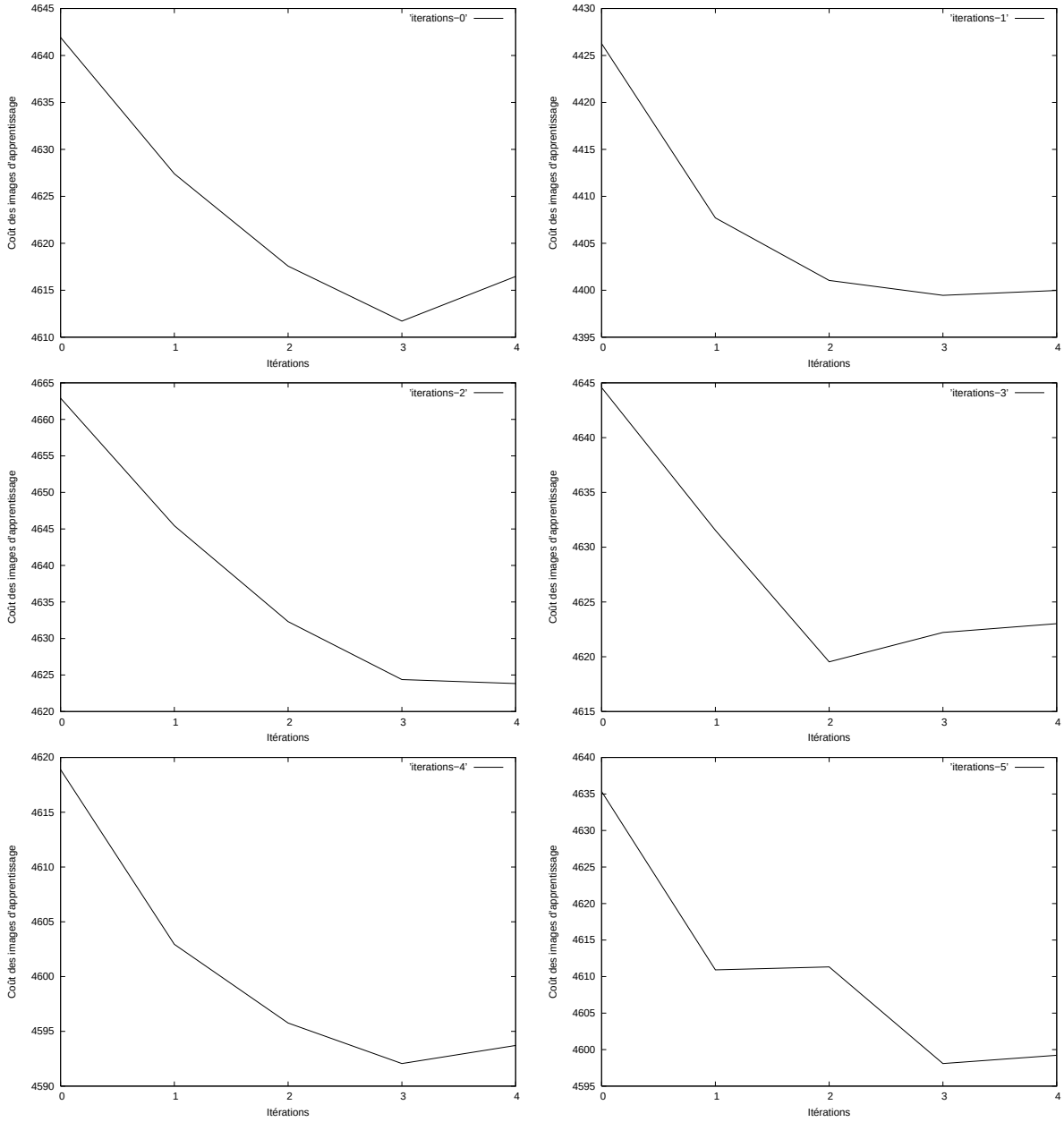


FIG. 4.16 : Convergence des modèles (classes 0 à 5) : Le coût moyen des données d'apprentissage ($-\log(P)$) est tracé en fonction du nombre d'itérations dans le calcul du modèle.

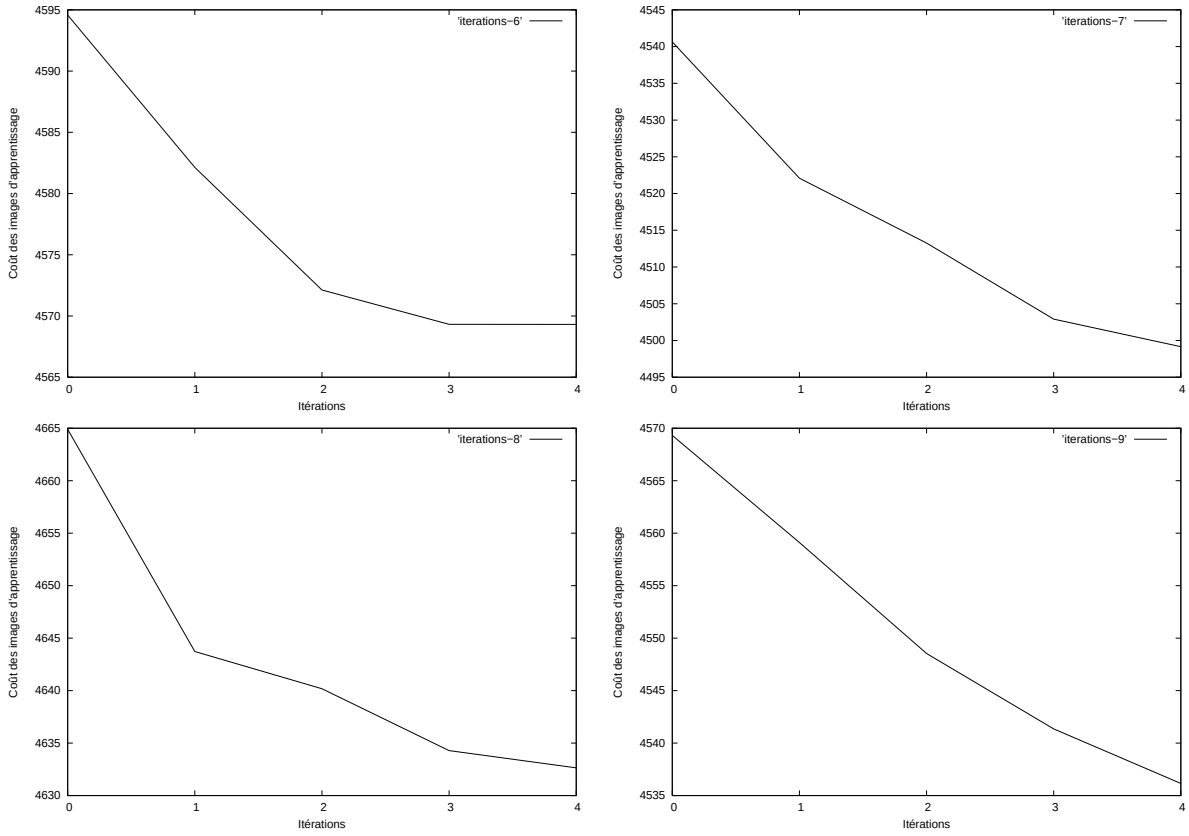


FIG. 4.17 : Convergence des modèles (classes 6 à 9) : Le coût moyen des données d'apprentissage ($-\log(P)$) est tracé en fonction du nombre d'itérations dans le calcul du modèle.

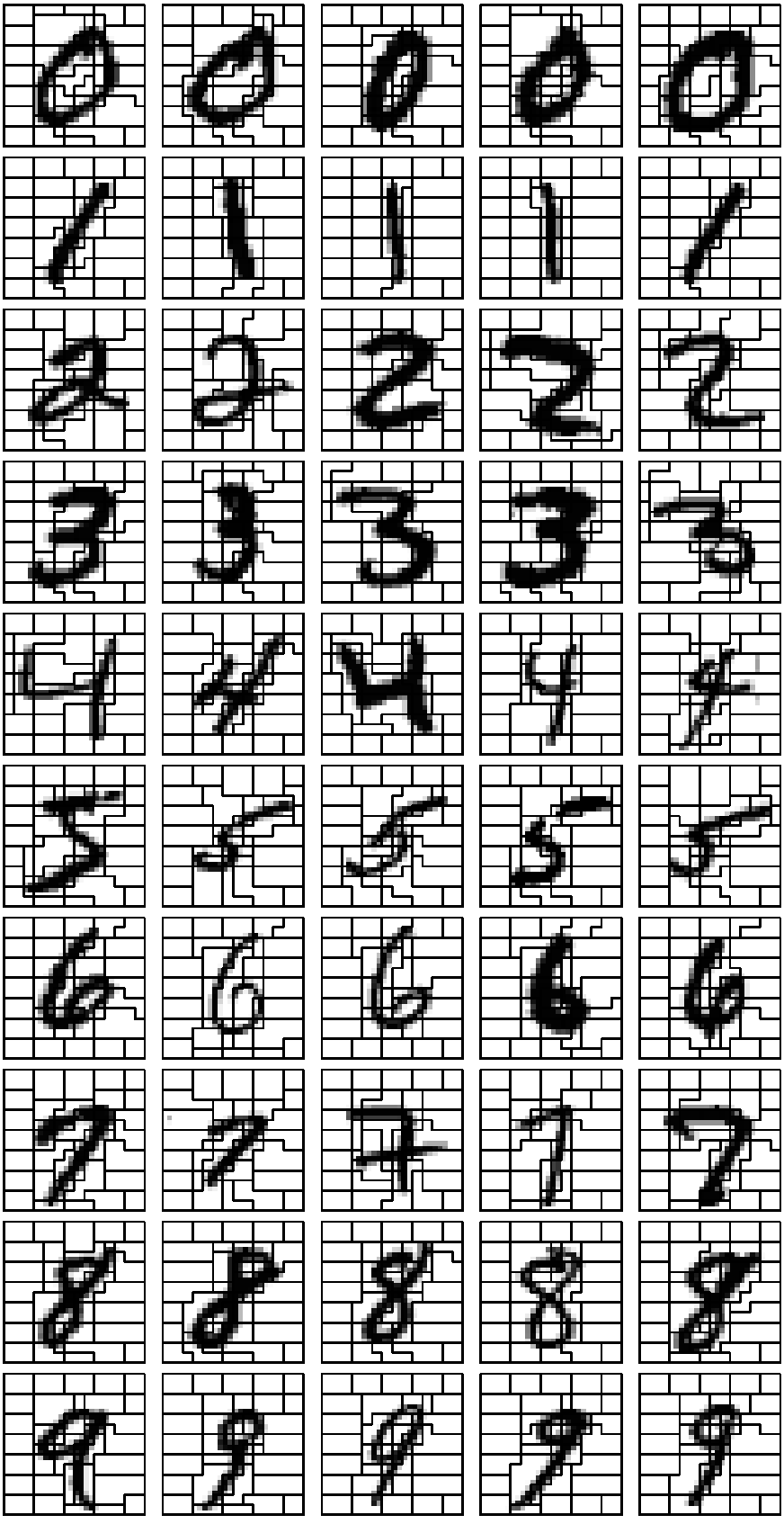


FIG. 4.18 : Résultats de la segmentation des images.

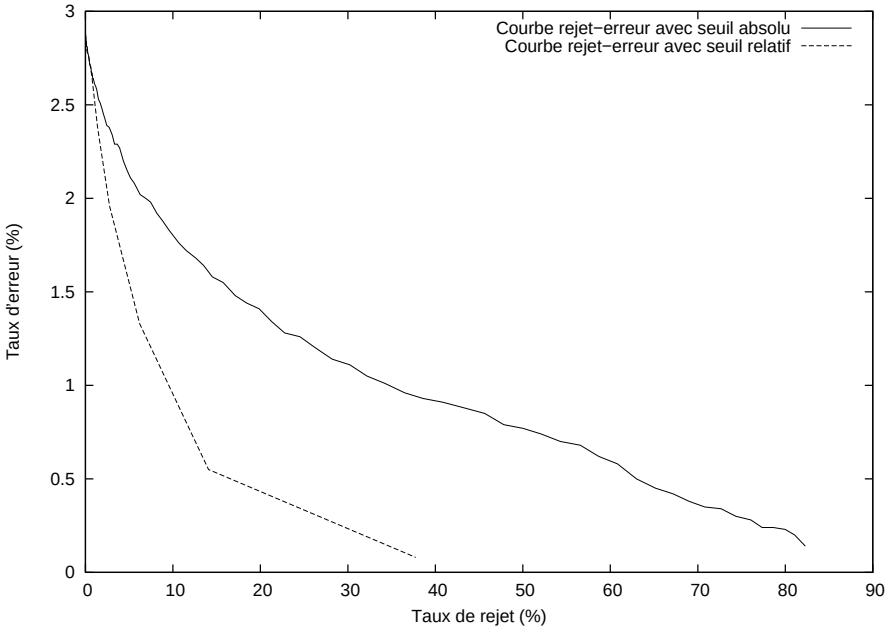


FIG. 4.19 : Taux d'erreur en fonction du pourcentage d'images rejetées pour les deux types de stratégie de rejet.

FIG. 4.20 : Images correspondant aux meilleurs scores de confiance.


8→5 8→0 3→7 5→6 7→2 9→7 7→2 0→2 4→9 6→1 4→6 4→8 9→2 9→7 1→2 9→1
3→7 2→6 7→9 2→7 2→1 3→2 2→6 3→7 6→5 2→8 2→6 0→6 4→6 0→6 7→8 1→8
1→8 3→7 5→3 2→7 3→7 7→2 8→0 7→1 0→7 0→2 8→9 7→2 7→9 9→2 1→4 8→2
7→9 6→5 7→2 0→2 3→2 9→2 4→9 3→7 6→5 8→9 5→0 8→4 9→8 4→9 5→8 6→5
9→8 8→0 7→9 7→2 3→2 2→7 3→8 2→8 8→7 1→4 9→1 9→3 9→5 7→9 6→5 9→1
0→8 3→7 8→9 3→2 4→9 6→5 7→9 8→0 3→8 7→2 5→3 1→2 9→7 2→3 6→5 8→0
9→8 2→1 8→3 0→6 3→8 0→6 3→5 6→1 2→7 4→9 9→1 8→0 9→3 8→0 6→4 3→8
3→7 7→9 6→0 4→6 2→7 4→9 9→2 9→1 7→9 1→6 9→6 2→8 8→9 9→7 7→9 0→6
6→0 7→2 7→9 9→3 8→9 8→9 6→4 5→8 8→0 7→2 2→6 4→8 3→5 9→7 7→1 4→9
5→9 7→1 6→8 1→4 9→3 5→3 3→5 5→9 9→7 9→8 7→9 9→8 5→3 4→9 5→3 5→3
8→0 3→8 5→3 3→5 1→6 6→5 0→6 7→9 9→7 7→1 7→1 5→8 7→0 2→7 7→2 4→2
4→9 0→6 3→2 0→2 9→7 6→5 5→6 9→3 5→2 8→5 3→5 5→3 1→8 8→9 9→8 3→1
4→9 2→3 3→7 8→7 9→7 9→4 8→6 7→1 6→0 8→3 9→3 0→2 4→2 5→7 5→9 1→8
6→5 3→8 2→4 8→5 9→7 9→7 9→3 1→6 9→3 8→0 3→7 3→2 3→7 7→1 4→9 7→2
4→9 9→5 8→9 9→7 7→9 4→9 9→2 7→9 5→3 3→2 7→1 9→8 4→9 8→9 0→2 4→1
9→0 6→8 3→8 9→2 3→9 9→1 5→3 7→9 7→9 7→1 3→2 9→4 2→7 6→0 0→2 3→5

4.2 Vers l'application à la reconnaissance de mots cursifs

L'approche proposée pour la reconnaissance de caractères manuscrits est extrèmement générale et ses principes restent valides à différents niveaux d'observation des documents. Nous proposons dans cette section une extension de cette méthodologie pour la reconnaissance de mots manuscrits et nous donnons quelques résultats préliminaires de son implantation. L'idée est de construire des modèles de mots à partir de modèles de lettres qui sont appris sur des images de mots. Cette approche est donc adaptée au traitement de grands vocabulaires et la difficulté réside dans l'identification de modèles de lettres sans information sur la segmentation en lettres des images d'apprentissage.

Le traitement des caractères manuscrits par des champs de Markov décrit dans la partie précédente repose sur les principes suivants :

- Même nombre d'états pour tous les modèles (un par classe).
- Décodage du champ de Markov par programmation dynamique 2D.
- Choix du meilleur modèle suivant la formule de Bayes.

L'extension la plus directe de ces principes pour le traitement des mots consiste à :

- utiliser un modèle de champ de Markov par mot lors de la reconnaissance,
- calculer un modèle de champ de Markov par lettre lors du processus d'apprentissage,

les modèles de mots étant construits par concaténation des modèles de lettres.

Pour des raisons de rapidité d'implantation des modèles de mots sont construits pour l'ensemble du dictionnaire avant la phase de reconnaissance, mais cette approche n'est cependaant pas à considérer comme holistique (un modèle est caculé pour chaque mot du dictionnaire) puisque les modèles calculés sont bien des modèles de lettres et les modèles de mots pourraient aussi bien être constrits dynamiquement.

Ainsi, le traitement des mots par cette approche nécessite simplement la création de deux nouvelles procédures :

1. la concaténation de deux modèles,
2. le découpage des couples image/segmentation.

4.2.1 Base de données

La base de données utilisée pour cette tâche est la base Senior & Robinson qui est une base publique et très simple (monoscripteur, petite taille). Un exemple de page de cette base est présenté figure 4.22. Sur les 25 pages qui composent la base, 3 sont gardées comme base de test. Le dictionnaire comportant tous les mots de l'ensemble de la base, il n'y a pas de problème de mots hors vocabulaire.

Le principal défaut de cette base est sa taille : certaines lettres ne sont présentes qu'en quelques exemplaires. La figure 4.23 illustre la répartition des lettres dans la base.

Il est intéressant d'observer la répartition des mots suivant leur nombre de lettres (Figure 4.24) et la répartition des lettres suivant la taille des mots dans lesquels elles se situent (Figure 4.25). On remarque que la plupart des mots ont 2, 3 ou 4 lettres mais que

la plupart des lettres se situent dans des mots de longueur 3 à 8. Il est donc nécessaire de pouvoir traiter efficacement ces mots lors du processus d'apprentissage pour obtenir des modèles robustes.

4.2.2 Approche de la construction des modèles

Comme pour le cas du traitement des caractères isolés, chaque modèle de lettre est de topologie 5×7, les états du tour étant forcés à partager la même densité d'observation (cf. section 2) pour obtenir un modèle de blanc inter-lettres ou inter-mots. Les modèles de mots sont de topologie $N \times 7$, où N est le nombre d'états sur la composante horizontale après concaténation des modèles de lettres. Les états du tour partagent aussi la même densité d'observation.

Plusieurs solutions peuvent être envisagées pour la création de modèles de mots à partir de modèles de lettres. Lors de la concaténation des deux modèles de lettres x et y pour former le mot xy, on peut

1. supprimer les états du bord droit de x et ceux du bord gauche de y et ajuster les probabilités de transition en conséquence (Figure 4.26),
2. supprimer les états du bord droit de x et ceux du bord gauche de y, ajouter un ensemble d'états blancs inter-lettres et ajuster les probabilités de transition en conséquence.

Les topologies des modèles de mots ainsi obtenus sont données dans le tableau 4.8.

Tab. 4.8 : Topologie des modèles.

Stratégie de concaténation	Nombre de lettres	États X	États Y
1 ou 2	1	5	7
1	n	$2 + 3 \times n$	7
2	n	$1 + 4 \times n$	7

Lors de notre étude, c'est la stratégie 1 qui a donné les meilleurs résultats, mais il est possible qu'avec un plus grand nombre de données d'apprentissage et des modèles plus précis (comme des modèles en contexte), la stratégie 2 devienne la plus performante. Une fois la segmentation effectuée par l'algorithme de programmation dynamique 2D, les images d'apprentissage sont découpées en lettres. Cette collection d'images de lettres est ensuite envoyée au programme d'apprentissage qui donne un modèle par lettre.

4.2.3 Nouvelles procédures

L'adaptation des algorithmes au traitement des mots nécessite essentiellement la création de deux nouvelles procédures : une procédure de concaténation de modèles et une procédure de découpage d'images de mots en images de lettres.

Procédure de concaténation de modèles

Un modèle de lettre isolée est un ensemble d'états définis par des fonctions multigaussiennes décrivant leurs caractéristiques d'émission et des potentiels de cliques qui capturent les positions relatives de ces états dans l'image. La création d'un modèle de mot nécessite de définir un nouvel ensemble d'états et leurs caractéristiques associées. Les modèles de lettres étant connus, il est raisonnable de supposer dans un premier temps que les probabilités d'émission des états de cette lettre au sein du mot sont les mêmes que pour la lettre isolée. De même, les potentiels de cliques à l'intérieur de chaque lettre peuvent être conservés. Seules restent à estimer les probabilités de transition entre états de lettres consécutives. Si les états associés à l'extérieur de l'image sont supprimés pour établir la connexion entre les lettres, cette estimation peut être faite directement à partir des cliques associées à ces états supprimés. Ainsi, la procédure de concaténation en entrée deux modèles de topologie $X_1 \times Y$ et $X_2 \times Y$ et retourne un modèle de topologie $(X_1 + X_2 - 2) \times Y$.

➠ Les états du bord droit du premier modèle et du bord gauche du second modèle sont supprimés.

➠ Les états du tour partagent la même densité d'observation.

➠ Les probabilités des états sont ajustées et normées.

➠ Les probabilités de cliques à l'intérieur des deux modèles sont ajustées et normées.

➠ Les probabilités de cliques dont un des états a été supprimé par la concaténation sont reportées sur les états de l'autre modèle.

Cette procédure est ainsi appelée $n - 1$ fois pour construire un modèle de mot de n lettres.

Procédure de découpage des images de mots en images de lettres

L'algorithme de programmation dynamique 2D, après avoir traité une image de mot avec le modèle correspondant à ce mot fournit une segmentation de l'image en états. La séquence de lettres étant connue, de même que la topologie des modèles élémentaires, l'identification des états appartenant à chacun des modèles de lettres est facile. Il est alors possible de découper l'image de mot en un ensemble d'images de lettres. Ces images ne sont pas nécessairement rectangulaires car la frontière entre deux lettres n'a pas de contrainte de verticalité. On peut donc compléter ces images par des pixels blancs (ou plus précisément par des valeurs d'observables correspondant à un blanc uniforme) associés aux états éliminés lors de la concaténation.

À partir de l'ensemble d'images de lettres obtenu, avec leurs segmentations associées, le processus d'apprentissage de chacun des modèles élémentaires peut se faire de la même façon que pour les chiffres manuscrits. Il est possible également d'effectuer l'apprentissage des fonction multigaussiennes et des potentiels de cliques directement sur les images de mots segmentées en états.

Pour résumer, à partir de la segmentation d'une image de mot, on construit des images de lettres correspondantes avec leur segmentation en états associée. On peut alors effectuer un apprentissage du modèle de chaque lettre. Cette procédure considère donc en entrée

une image (après extraction des primitives) et sa segmentation associée et retourne deux autres images avec leurs segmentations, extraites de l'image d'entrée suivant les paramètres d'entrée (Figure 4.27).

Le principe est détaillé ci-après :

➠ On donne en entrée le nombre d'états à extraire en X. L'extraction se faisant lettre par lettre, avec la stratégie de concaténation retenue, on extrait, à chaque appel, 4 états sur la composante horizontale, correspondant à la dernière lettre.

➠ Les sites étiquetés comme faisant partie des états à extraire sont placés dans une nouvelle image.

➠ Les bords sont complétés en ajoutant des pixels affectés aux états manquants avec la couleur du fond. De cette façon, on obtient bien une image rectangulaire et sa segmentation associée qui peuvent être traitées normalement par le processus d'apprentissage avec l'ensemble des échantillons de la même classe.

4.2.4 Synthèse des processus d'apprentissage et de reconnaissance pour l'analyse de mots

La procédure de concaténation permet de créer un modèle pour chaque mot en utilisant des modèles de lettres. L'alignement forcé d'un mot avec son modèle permet d'obtenir, outre sa probabilité, une segmentation en lettres et, pour chaque lettre, une segmentation en états.

La procédure de découpage des mots en lettres permet d'obtenir un jeu d'images de lettres isolées que l'on peut fournir au programme d'apprentissage pour obtenir des modèles de lettres.

L'apprentissage peut se faire en effectuant alternativement un alignement des mots et un apprentissage des modèles de lettres obtenues en découpant les mots grâce à leur alignement.

L'ensemble des procédures présentées permet d'effectuer un apprentissage par stratégie EM avec un critère MAP et une reconnaissance par stratégie bayésienne. Cette partie synthétise l'utilisation pour cette tâche de reconnaissance de mots de ces 4 procédures :

➪ `apprendre`	À partir d'images et de segmentations associées, calcule les paramètres d'un modèle (attache aux données et termes d'interaction).
➪ `reconnaitre`	À partir d'une image et d'un modèle, effectue le décodage optimal et fournit la segmentation en états et la vraisemblance associée.
➪ `decouper`	À partir d'une image et d'une segmentation associée, découpe ce couple en deux conformément aux paramètres fournis.
➪ `concatener`	À partir de deux modèles, calcule les paramètres d'un modèle concaténé.

Apprentissage

■ Initialisation

➪ Les images monolettres et bilettres (coupées en deux par une limite verticale fixée manuellement) associées à des segmentations régulières sont utilisées pour créer les premiers modèles. `decouper + apprendre`

➪ Ces modèles sont utilisés pour segmenter ces images monolettres. `reconnaitre`

➪ Une nouvelle série de modèles est calculée. `apprendre`

■ Itérations

Le processus suivant est effectué successivement sur les mots de 2 lettres et moins, puis 3 lettres et moins puis 4 lettres et moins etc. La taille de la base d'apprentissage augmente donc au fur et à mesure.

➪ Les modèles de mots sont construits par concaténation des modèles de lettres. `concatener`

➪ Un alignement forcé est effectué sur les images de la base d'apprentissage. `reconnaitre`

➪ Les images sont découpées en lettres. `decouper`

➪ Des nouveaux modèles sont appris. `apprendre`

➪ Ce processus est itéré plusieurs fois pour un même nombre de lettres

Reconnaissance

■ Initialisation de la reconnaissance

➪ Les modèles de tous les mots sont construits par concaténation des modèles de lettres. `concatener`

■ Reconnaissance

➪ Pour chaque image, on effectue une reconnaissance sur tous les modèles. `reconnaitre`

➪ Pour chaque image, le meilleur score est sélectionné.

4.2.5 Résultats et discussion

Cette stratégie a été évaluée sur la base *Senior & Robinson* en utilisant les deux stratégies de concaténation décrites dans la section 4.2.2. Le tableau 4.9 indique les résultats obtenus (pour les deux stratégies de concaténation) après un apprentissage effectué sur les mots de 1, 2 et 3 lettres.

Tab. 4.9 : Résultats de la reconnaissance.

Nombre de lettres	Taux de reconnaissance 1	Taux de reconnaissance 2
1	88%	77%
2	57%	53%
3	37%	25%
4	7%	9%
1 à 4	40%	36%

De plus, il est intéressant d'observer les performances obtenues en termes de taux de reconnaissance de lettres. Un processus d'alignement entre deux mots par programmation dynamique permet d'observer les phénomènes de substitution (une lettre est prise pour une autre), d'insertion ou de suppression. Les performances obtenues sont résumées tableau 4.10 (stratégie 1).

Tab. 4.10 : Résultats de la reconnaissance (taux de reconnaissance de lettres)

Nombre de lettres max.	Taux de reconnaissance de lettres
1	88%
2	65%
3	63%
4	33%

Ces performances sont relativement éloignées des meilleures performances obtenues sur cette base, mais ils ont été obtenus avec très peu de données d'apprentissage puisque seuls les mots de 3 lettres et moins ont été utilisés. Le point bloquant est que les modèles ainsi obtenus ne sont pas assez performants pour être alignés sur des mots de 4 lettres et plus et permettre ainsi l'utilisation de toutes ces données pour l'apprentissage. Il est légitime de penser qu'un plus grand nombre de données d'apprentissage pour les mots de petite taille auraient pu permettre d'obtenir de meilleurs modèles et d'accéder ainsi aux mots de plus grande taille en améliorant parallèlement les performances. De plus, de nombreuses amélioration peuvent probablement être apportées à nos algorithmes pour s'adapter au problème du traitement des mots avec plus d'efficacité.

La segmentation en états effectuée par le processus de reconnaissance permet de déduire la segmentation en lettres associée. Cette segmentation n'est pas réduite à une ligne verticale et on peut observer, pour une partie des images, que cette segmentation est pertinente (Figure 4.28).

Cependant, une grande partie des images n'est pas segmentée de façon convenable et n'est donc souvent pas reconnue par notre système (Figure 4.29). Ces échantillons sont des mots contenant des lettres rares dans l'ensemble des images d'apprentissage ou présentes en majorité dans une configuration particulière (*t* de *the*). Ces observations confirment qu'il serait probablement bénéfique d'utiliser un plus grand nombre d'échantillons pour l'apprentissage et que des évolutions de nos algorithmes sont nécessaires pour traitement des mots.

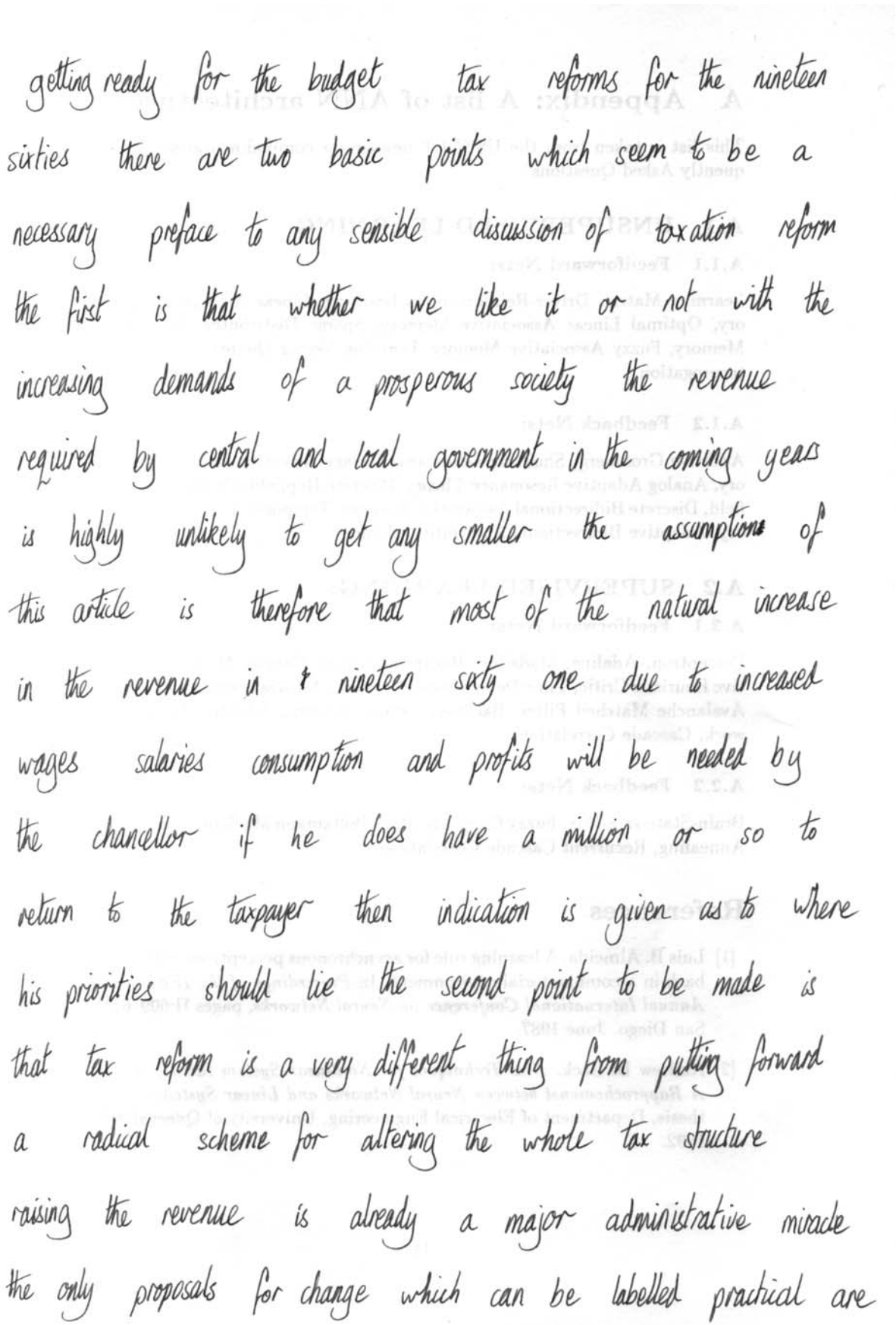


FIG. 4.22 : Une page de la base Senior & Robinson.

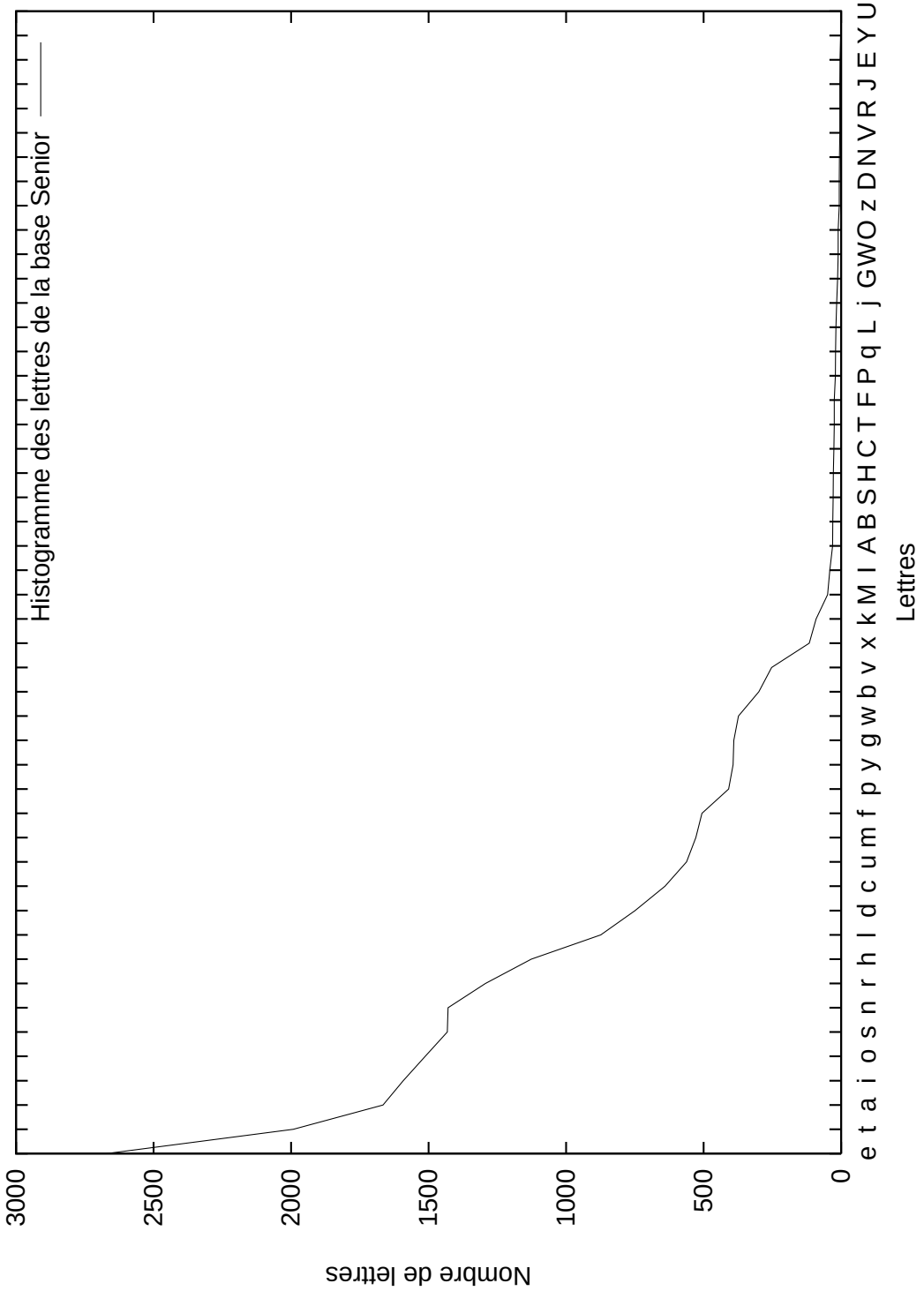


FIG. 4.23 : Histogramme des lettres dans la base Senior & Robinson.

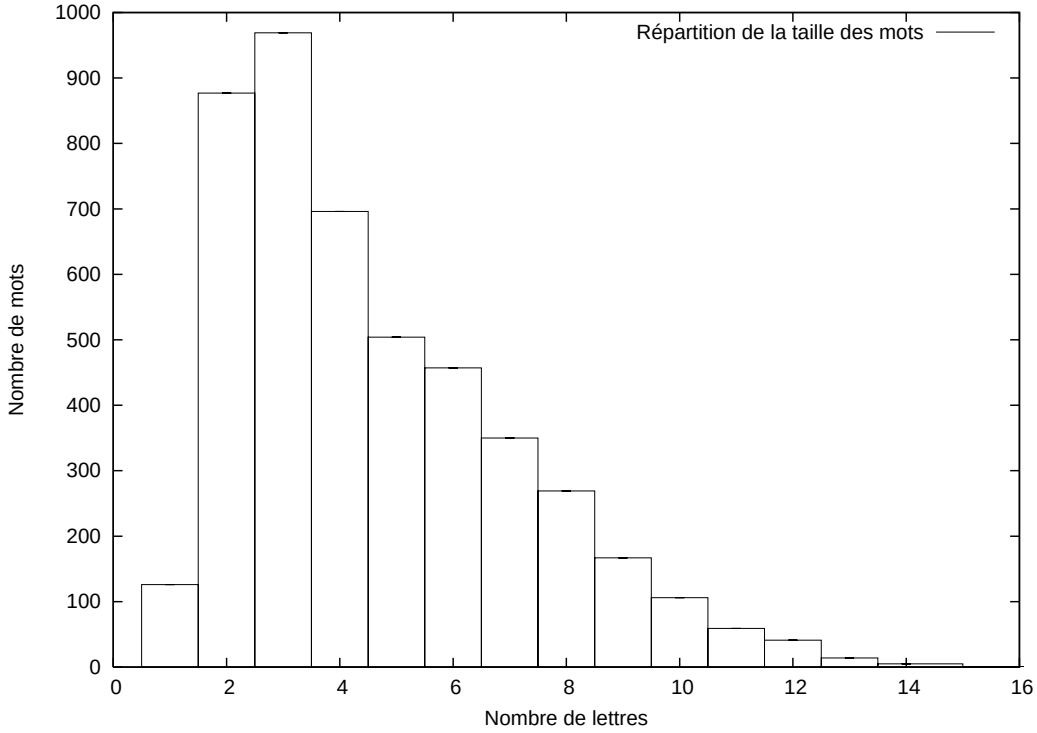


FIG. 4.24 : Histogramme des mots suivant leur nombre de lettres.

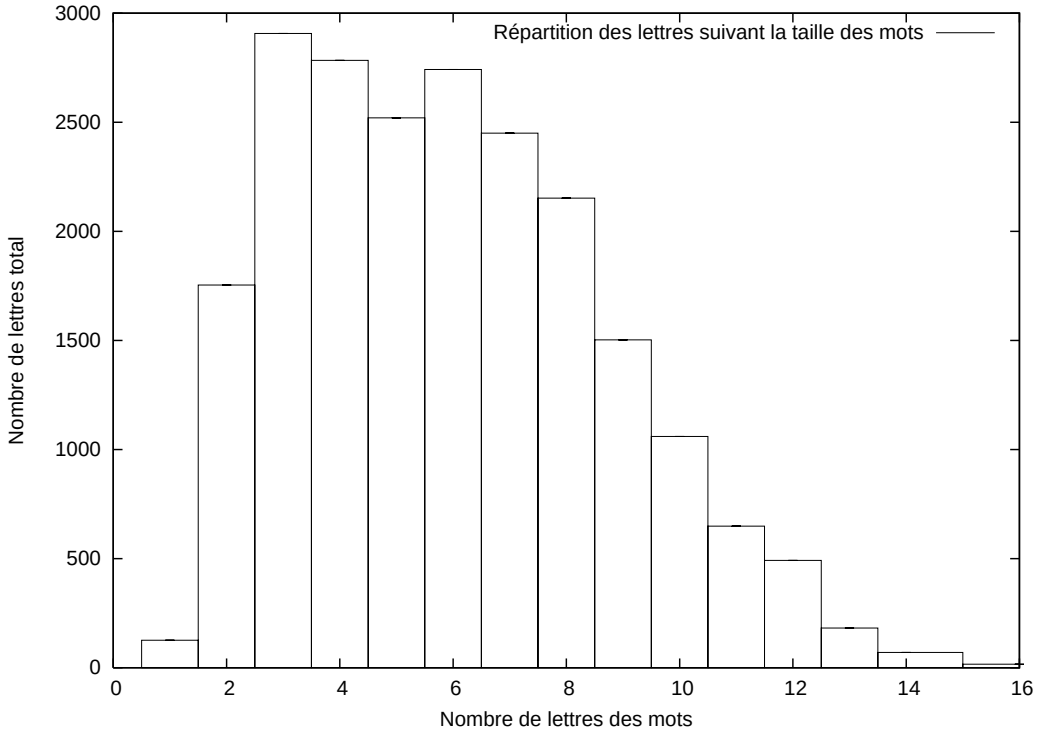


FIG. 4.25 : Histogramme des lettres suivant la taille des mots.

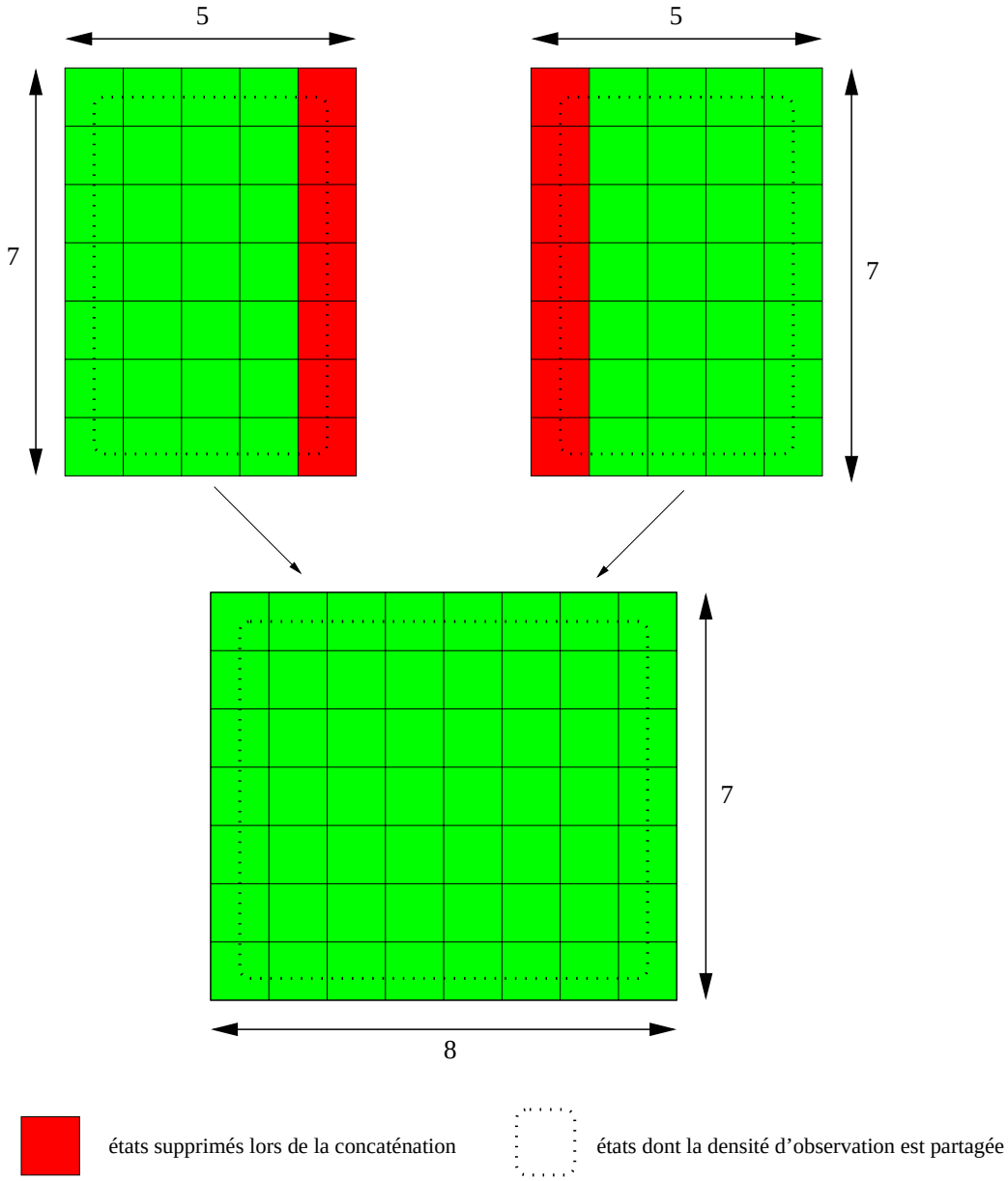


FIG. 4.26 : Concaténation de deux modèles.

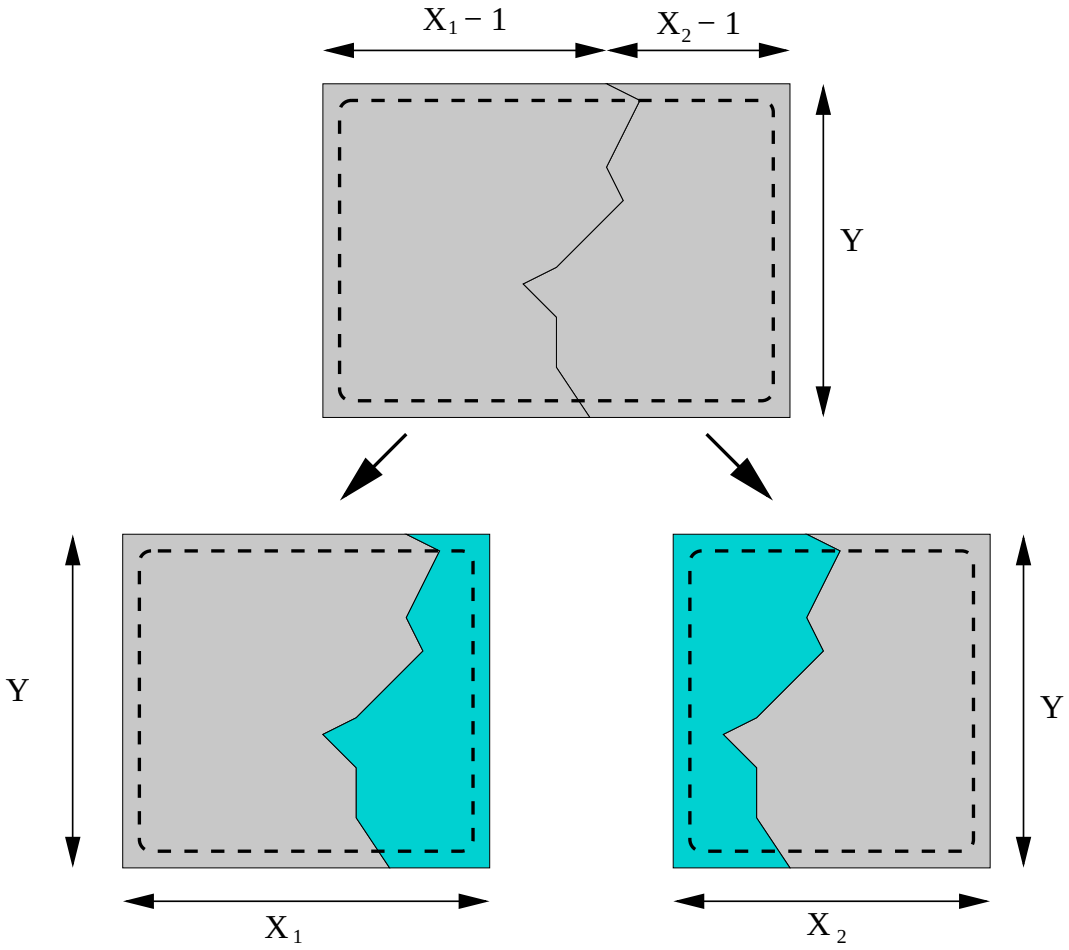


FIG. 4.27 : Découpage d'un mot en lettres.

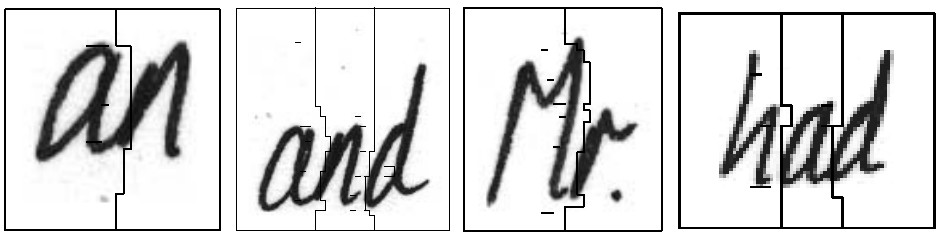


FIG. 4.28 : Exemples de segmentations en lettres convenables.

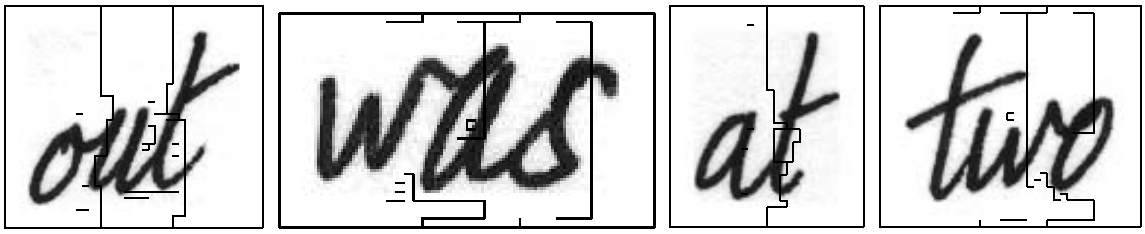


FIG. 4.29 : Exemples de segmentations en lettres aberrantes.

Chapitre 5

Conclusion et perspectives

5.1 Conclusion générale

Nous avons présenté une approche du problème de la reconnaissance d'écriture manuscrite hors-ligne et son application à une tâche de reconnaissance de caractères manuscrits ainsi qu'une proposition pour l'adaptation de cette approche à une tâche de reconnaissance de mots manuscrits. Les apports principaux de ce système sont l'utilisation d'une modélisation par champ de Markov et d'un algorithme de programmation dynamique 2D qui permettent une vraie modélisation bidimensionnelle de l'écriture avec des algorithmes rapides. L'originalité de ce travail réside aussi dans l'extraction de primitives spectrales issues d'une analyse par fenêtrage. Ces deux principes sont validés par des résultats satisfaisants : un taux d'erreurs de 2,7% a été atteint sur la tâche de reconnaissance de caractères, et des performances intéressantes ont été obtenues sur le traitement des mots malgré un nombre d'images d'apprentissage très restreint.

Une autre caractéristique importante de notre travail est la méthodologie de développement qui consiste à évaluer notre système à chaque étape de son développement pour valider les différentes techniques et paramètres : Une base de développement et une base de validation distinctes de la base de test permettent le développement des algorithmes, la base de test permettant elle d'évaluer le système obtenu sur des donnés représentatives du monde réel.

Les perspectives ouvertes par ces travaux sont très nombreuses et nécessiteront, pour être explorées, de très grandes bases de données pour l'apprentissage et le test des modèles.

5.2 Évolution de la modélisation

Les résultats de notre études suggèrent d'envisager une évolution de la modélisation suivant plusieurs axes :

➪ Utiliser un système de vote

Lors de nos travaux, nous avons étudié l'influence des différents paramètres sur le taux de reconnaissance. Cependant, un paramétrage peut être plus efficace pour une partie

des échantillons de la base : par exemple une fenêtre d'analyse plus grande pourrait permettre de s'affranchir mieux de traits épais. Une solution est donc de mettre en place une stratégie à plusieurs classifieurs, dont les paramètres sont différents, utilisés en parallèle puis déterminer la classe en combinant les résultats obtenus.

➪ Construire des modèles de lettres en contexte :

Les expériences menées sur la base *Senior& Robinson* ont confirmé qu'il était impératif de construire plusieurs modèles pour chaque lettre suivant son contexte à droite et à gauche, pour distinguer par exemple le *t* de *the*, de *at* et de *two*. Suivant la taille de la base disponible, on pourra envisager de traiter tout ou partie des contextes possibles (27×27 contextes possibles, mais beaucoup moins sont réellement observés). Les contextes pourraient même se faire sur plus d'une lettre avant ou après.

➪ Employer une stratégie d'adaptation au scripteur

Sur une tâche de reconnaissance de mots, il est possible d'effectuer une reconnaissance en plusieurs passes : un première passe, rapide, permet d'adapter les modèles au type d'écriture analysée. Ces nouveaux modèles seront alors plus efficaces en reconnaissance lors de la seconde passe.

➪ Organiser le dictionnaire en arbre et construire les modèles de mots à la volée :

Dans notre étude, les modèles de tous les mots sont construits au début de la reconnaissance, ce qui rend ce processus extrêmement long puisque tous doivent être évalués. Le temps de traitement pourrait être optimisé en organisant le dictionnaire en arbre, les configurations les moins probables étant élaguées au fur et à mesure. Une illustration de cette organisation du dictionnaire est donnée Figure 5.1

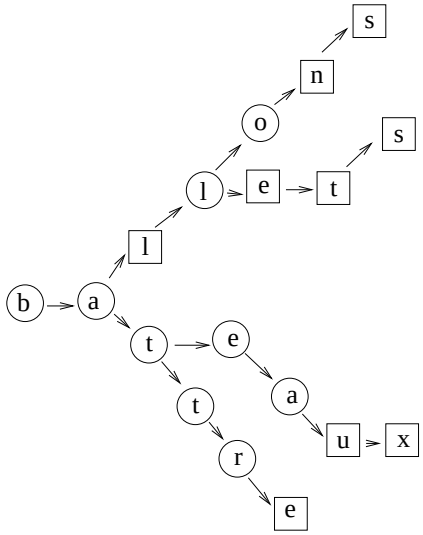


FIG. 5.1 : Dictionnaire organisé en arbre.

➪ Traiter un document entier :

Notre recherche, comme la plupart des travaux en reconnaissance de l'écriture manuscrite, s'attache à reconnaître des mots déjà segmentés. Le problème du traitement du

document dans son ensemble pourrait tirer partie du principe de programmation dynamique 2D : Une première passe sur l'ensemble du document avec deux modèles basiques `mots` / `non mots` pourrait déterminer les segmentations en mots les plus probables avant d'effectuer, à partir de ces segmentations, un processus de reconnaissance de mots comme nous l'avons proposé.

5.3 Évolution de la partie d'analyse d'images

Le calcul des observables peut probablement être amélioré. On peut par exemple tenter de repasser dans le domaine spatial après la première transformation de Fourier pour ne garder que l'information utile.

5.4 Évolution de l'implantation

Pour accélérer les traitements de traitement de l'écriture, une évolution de l'implantation de l'algorithme est nécessaire, notamment pour favoriser un véritable partage d'états pour n'effectuer qu'une seule fois des calculs identiques pour des états différents mais de densités d'observation égales.

L'utilisation de la programmation dynamique dans le cas particulier du traitement de l'écriture manuscrite pourrait être implantée de façon moins coûteuse en temps de calcul, en tirant partie du caractère déterministe de la stratégie de fusion ou de l'utilisation en pratique, d'une seule région à laquelle on fusionne les sites un à un.

Enfin une partie importante des calculs peut être menée en parallèle sur plusieurs processeurs pour réduire considérablement les temps de calcul.

5.5 Construire une base de données et organiser des évaluations

Il est apparu dans notre étude que la qualité et la taille de la base de données étaient cruciales pour le développement des algorithmes. La base *Senior& Robinson* que nous avons utilisée est trop petite et certaines lettres ne sont présentes qu'en une dizaine d'échantillons.

Il est aussi important d'organiser des campagnes d'évaluation sur une base commune pour permettre aux différents acteurs de la communauté de se situer, et de voir quelles sont réellement les voies les plus prometteuses.

Ces deux objectifs sont très liés et ont déjà été réalisés avec succès dans plusieurs domaines lors de campagnes comme celles menées par le NIST[1] aux États-Unis ou comme le programme Technolangue[2] en France, qui permettent à la communauté de disposer de

[1] National Institute of Standrard and Technology, `www.nist.gov`

[2] `www.technolangue.net`

grandes bases de données pour leurs développements et de se réunir régulièrement pour comparer leurs développements et résultats.

Le programme Techno-vision[3] destiné à financer des campagnes d'évaluation dans les domaines de la vision pourrait être d'un grand intérêt pour le domaine du traitement de l'écriture manuscrite notamment par le projet *RIMES* qui conduira des évaluations sur des courriers manuscrits.

[3] `www.recherche.gouv.fr/appel/2004/technovision.htm`

Annexe A

Illustration des informations extraites par le calcul des vecteurs de primitives

Pour observer l'information réellement disponible après le processus d'extraction des primitives, il est possible d'observer le processus de resynthèse des images à partir des paramètres extraits. La procédure est simplement le processus inverse de l'extraction des primitives, les paramètres qui ne sont pas gardés étant fixés à 0.

A.1 Reconstruction des images quand seule une partie des coefficients de la FFT est gardée

Le résultat de la reconstruction est donné pour différents nombres de coefficients gardés par imagette, chaque tableau correspondant à un type d'imagette dont les caractéristiques sont synthétisées Tableau A.1.

Tab. A.1 : Association taille de fenêtre - pas de parcours de l'image.

Taille de la fenêtre d'analyse	Pas de parcours de l'imagette
7×7	2
9×9	3
11×11	4
15×15	5

À la fois la phase et le module des transformées de Fourier sont gardés et les reconstructions sont données pour 3 à 25 coefficients conservés, 3 étant le nombre minimal de coefficients nécessaires pour garder les informations dans les deux directions.

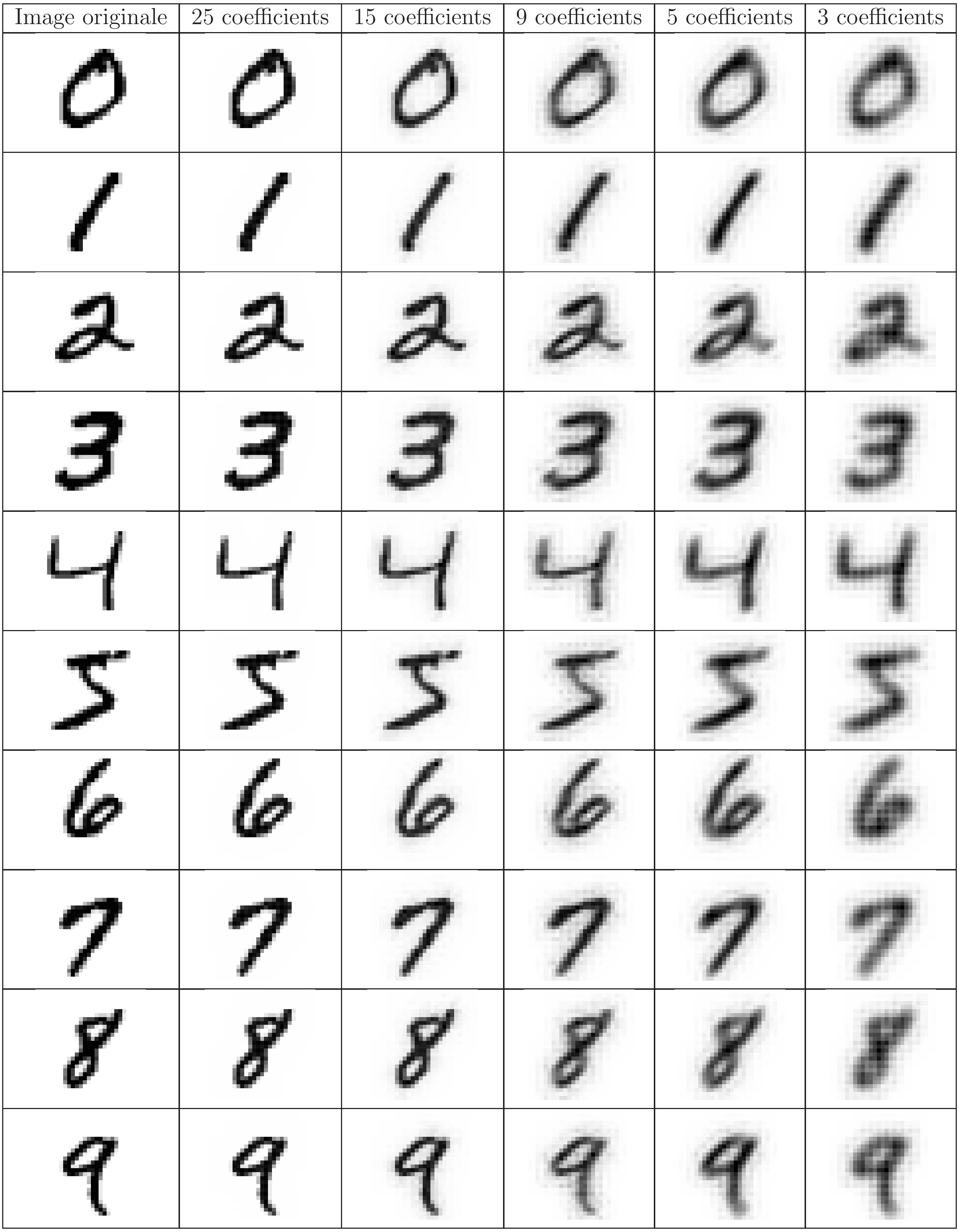


Fig. A.1 : Fenêtres 7×7.

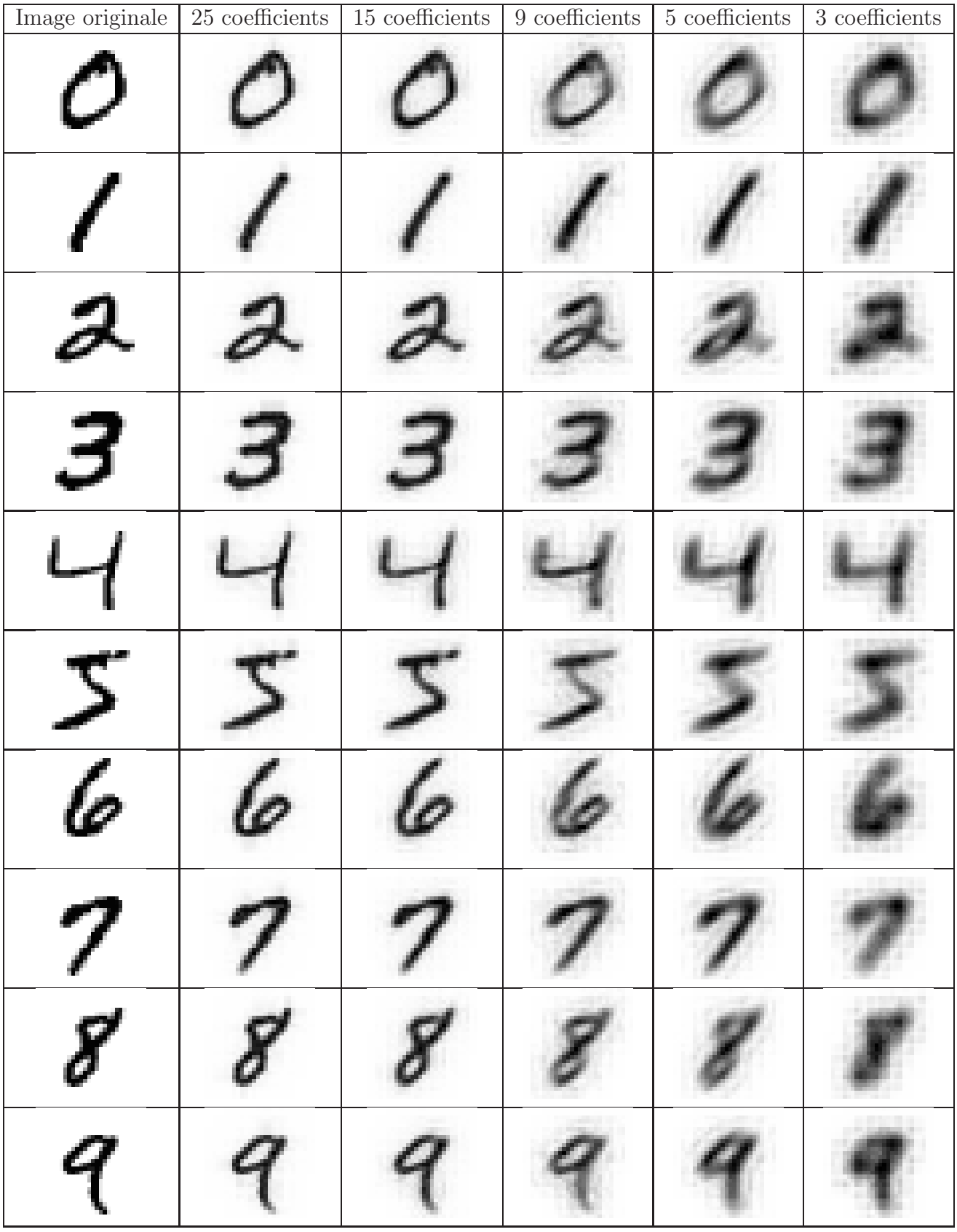


FIG. A.2 : Fenêtres 9×9.

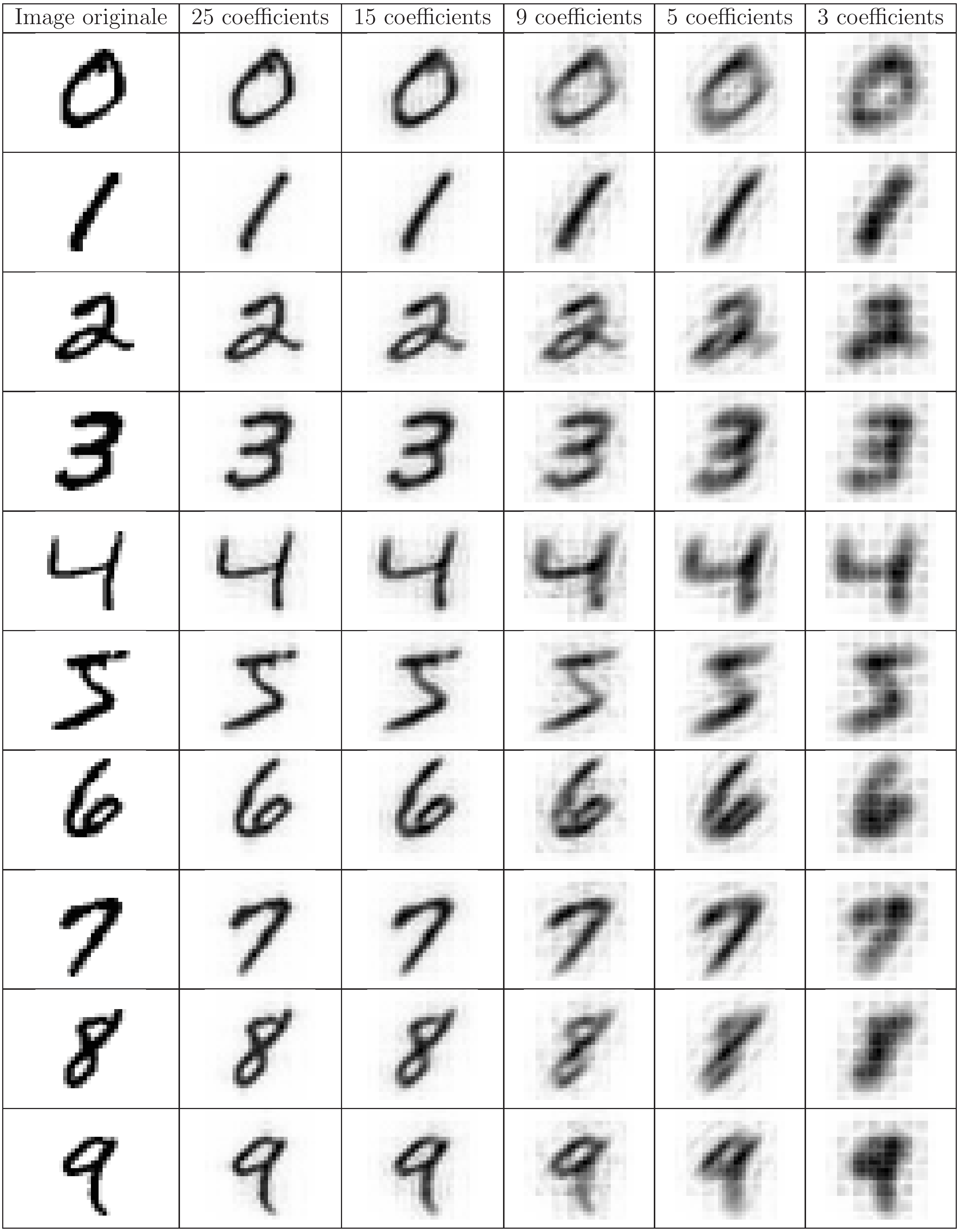


Fig. A.3 : Fenêtres 11×11.

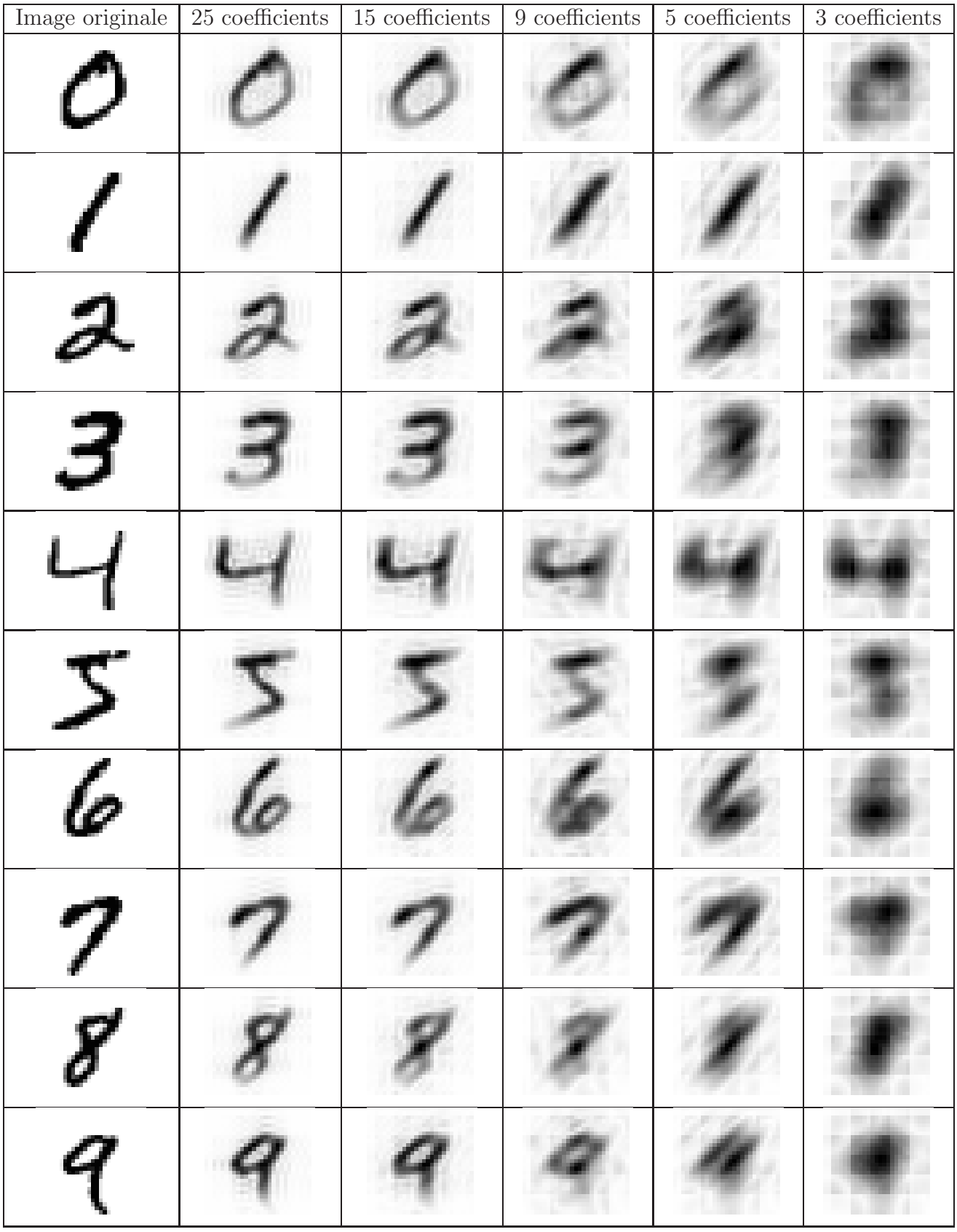
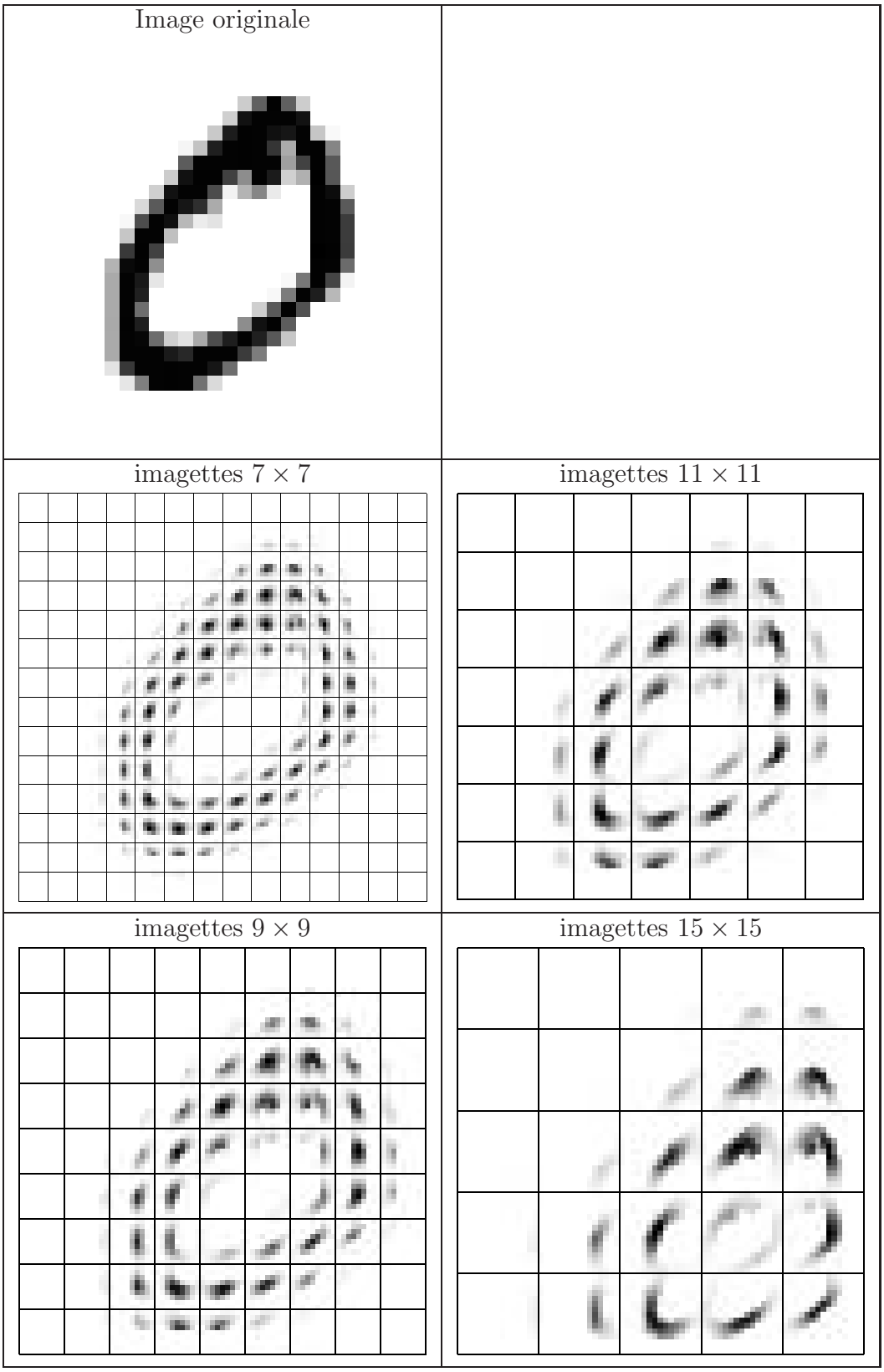


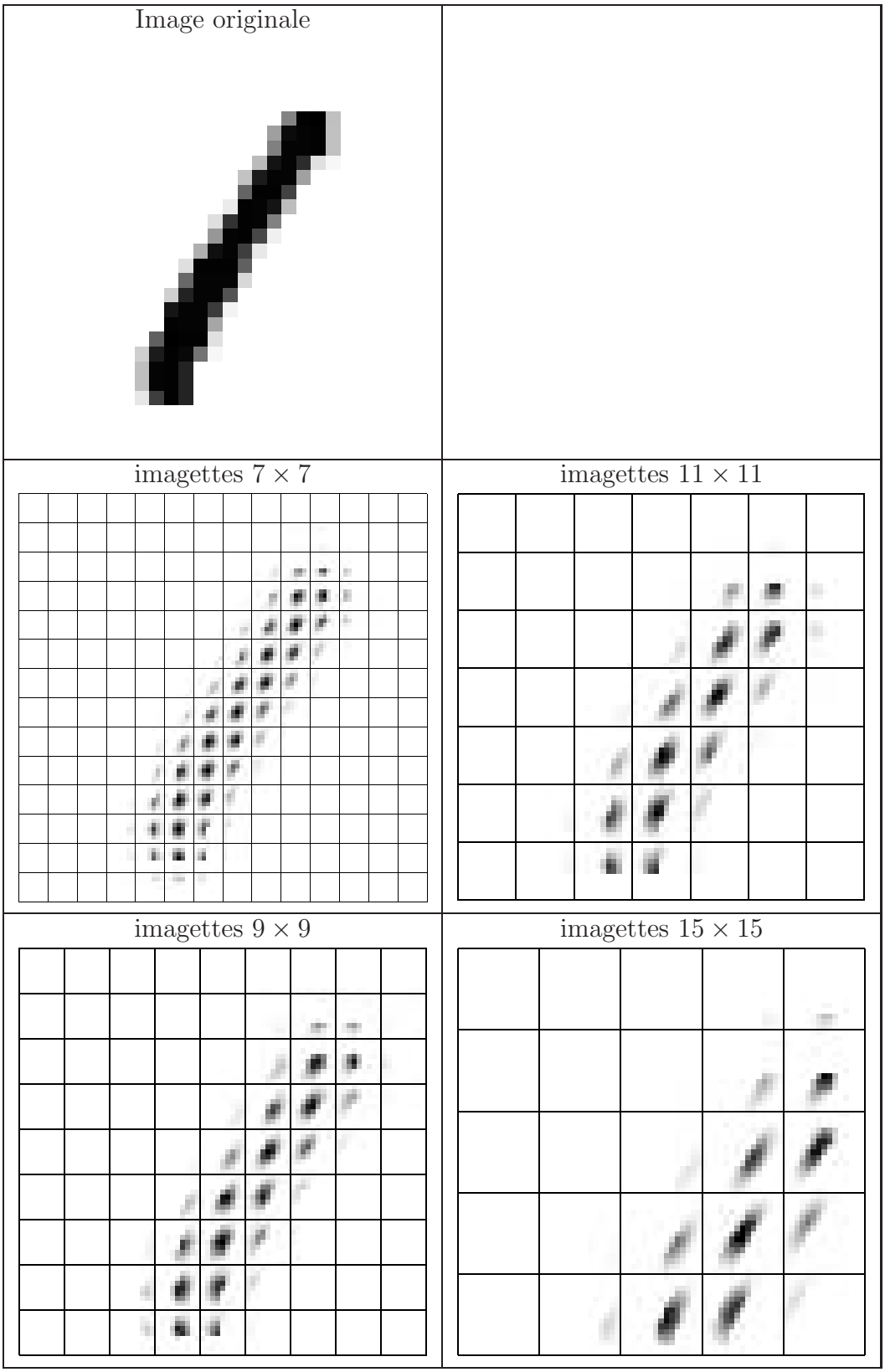
FIG. A.4 : Fenêtres 15×15.

A.2 Représentation « éclatée » des imagettes extraites

L'observation séparée de chacune des imagettes extraites permet de déterminer quelle information est contenue dans chacune de ces imagettes.

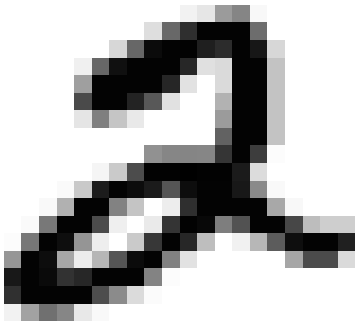


Tab. A.2 : Classe 0.

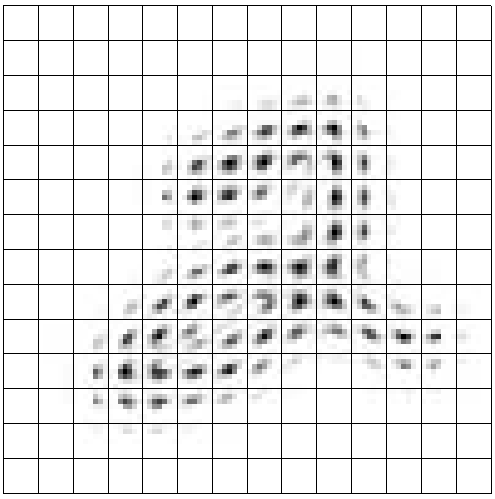


TAB. A.3 : Classe 1.

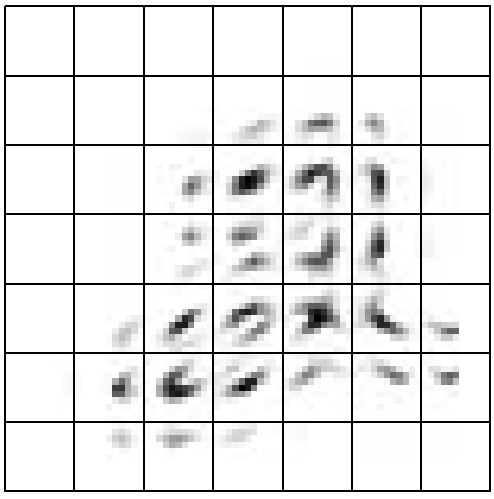
Image originale



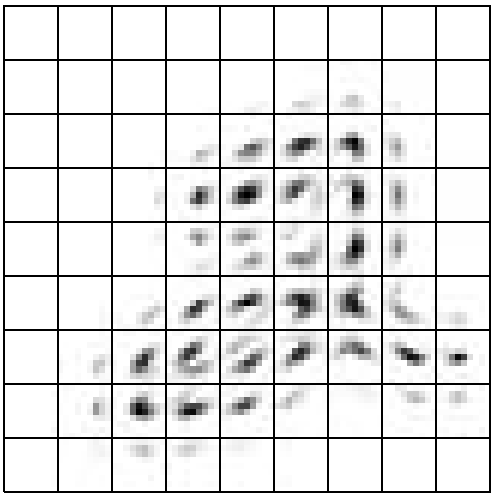
imagettes 7×7



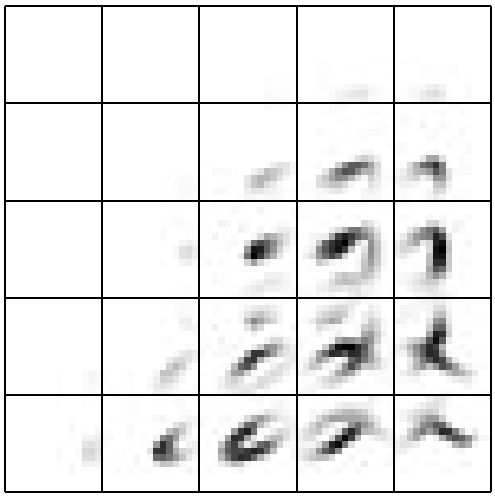
imagettes 11×11



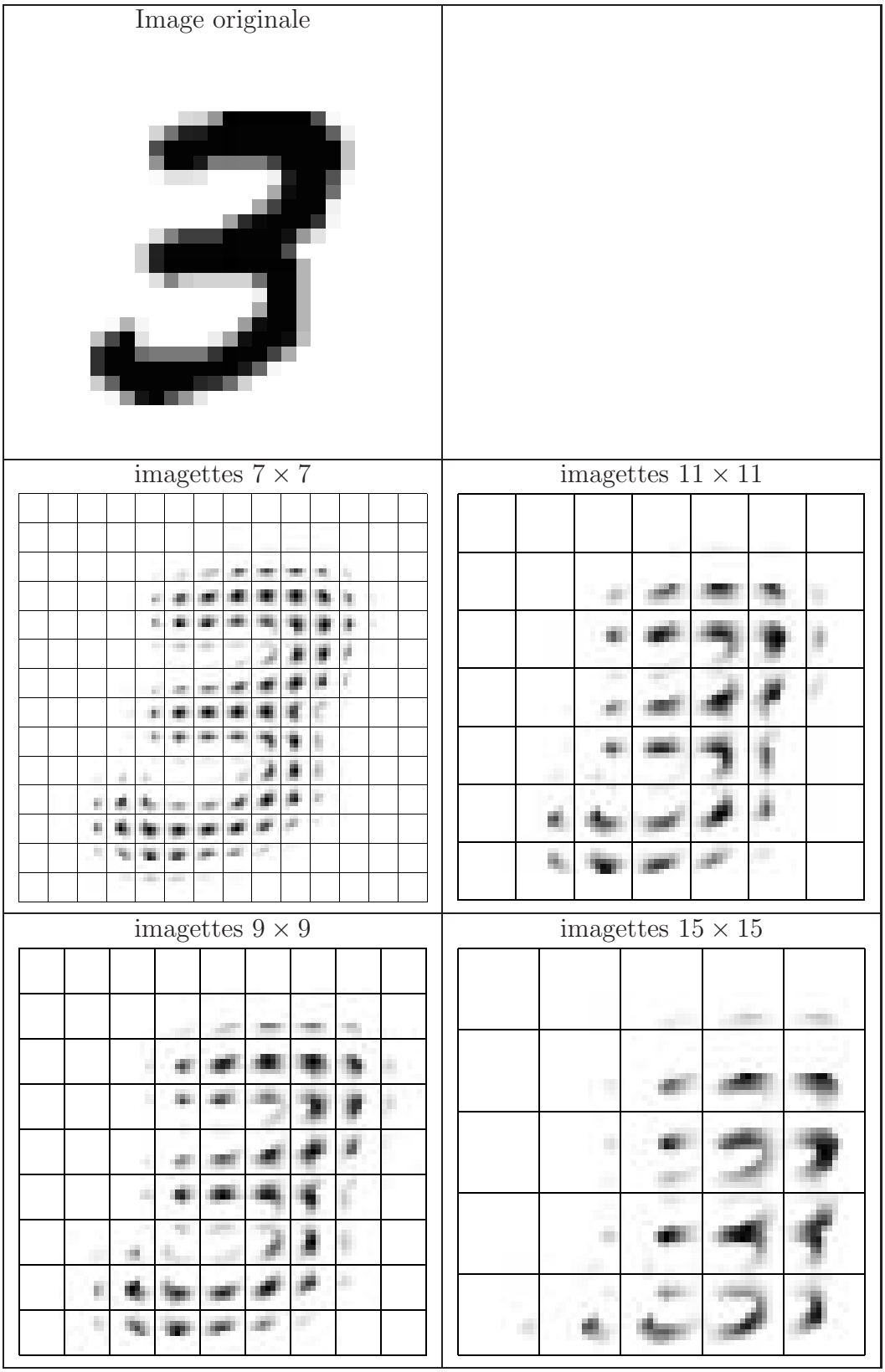
imagettes 9×9



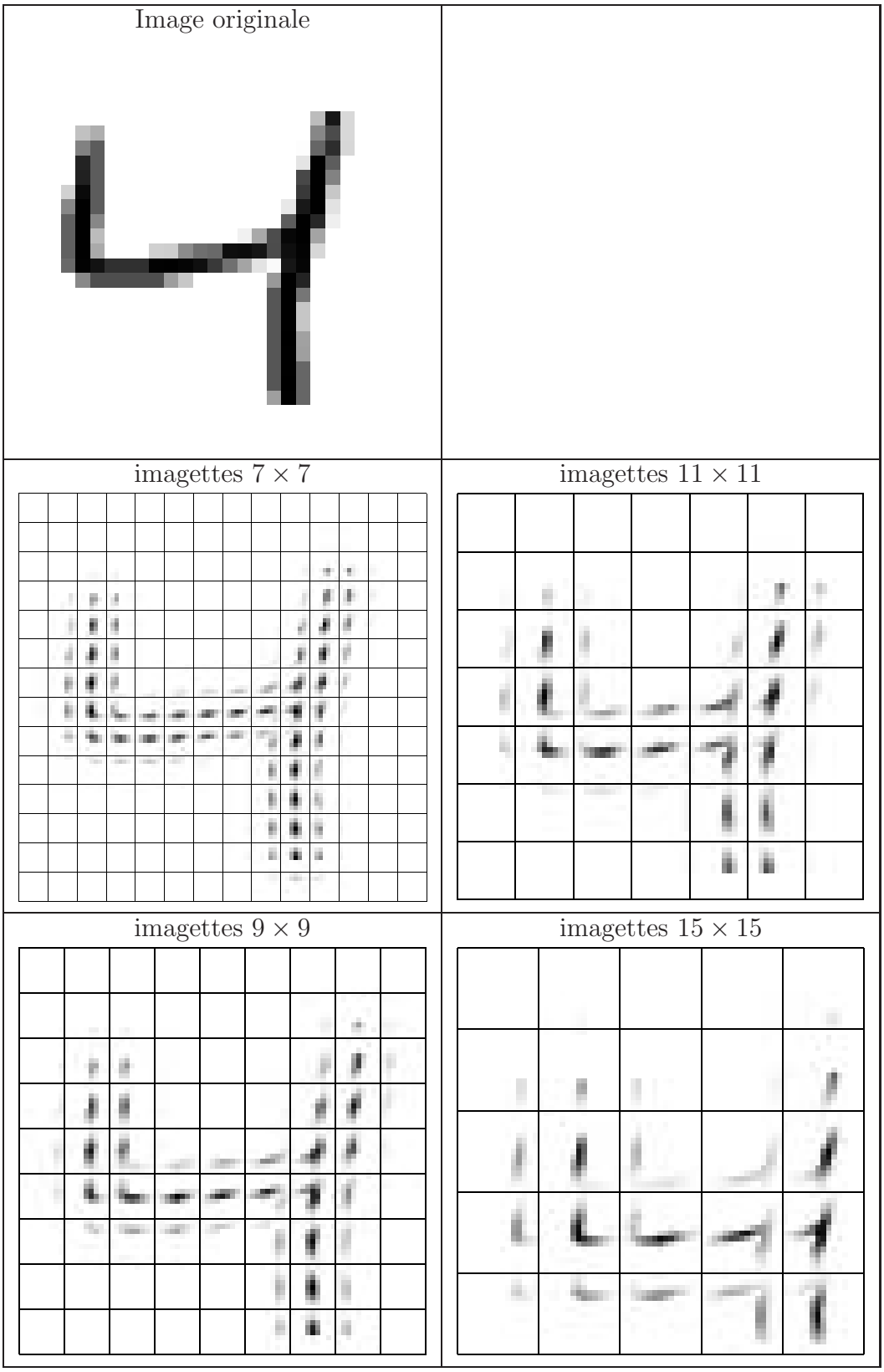
imagettes 15×15



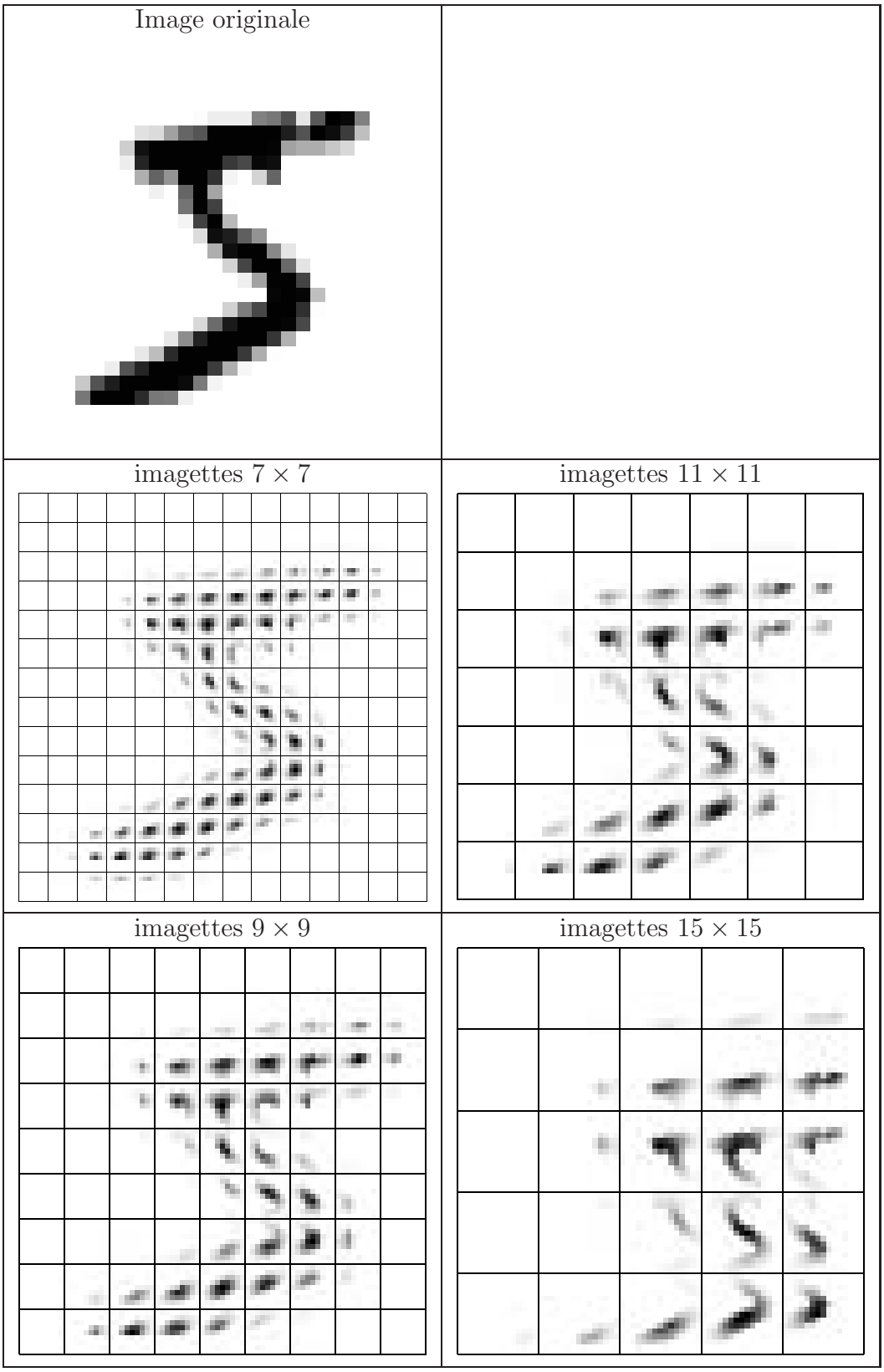
TAB. A.4 : Classe 2.



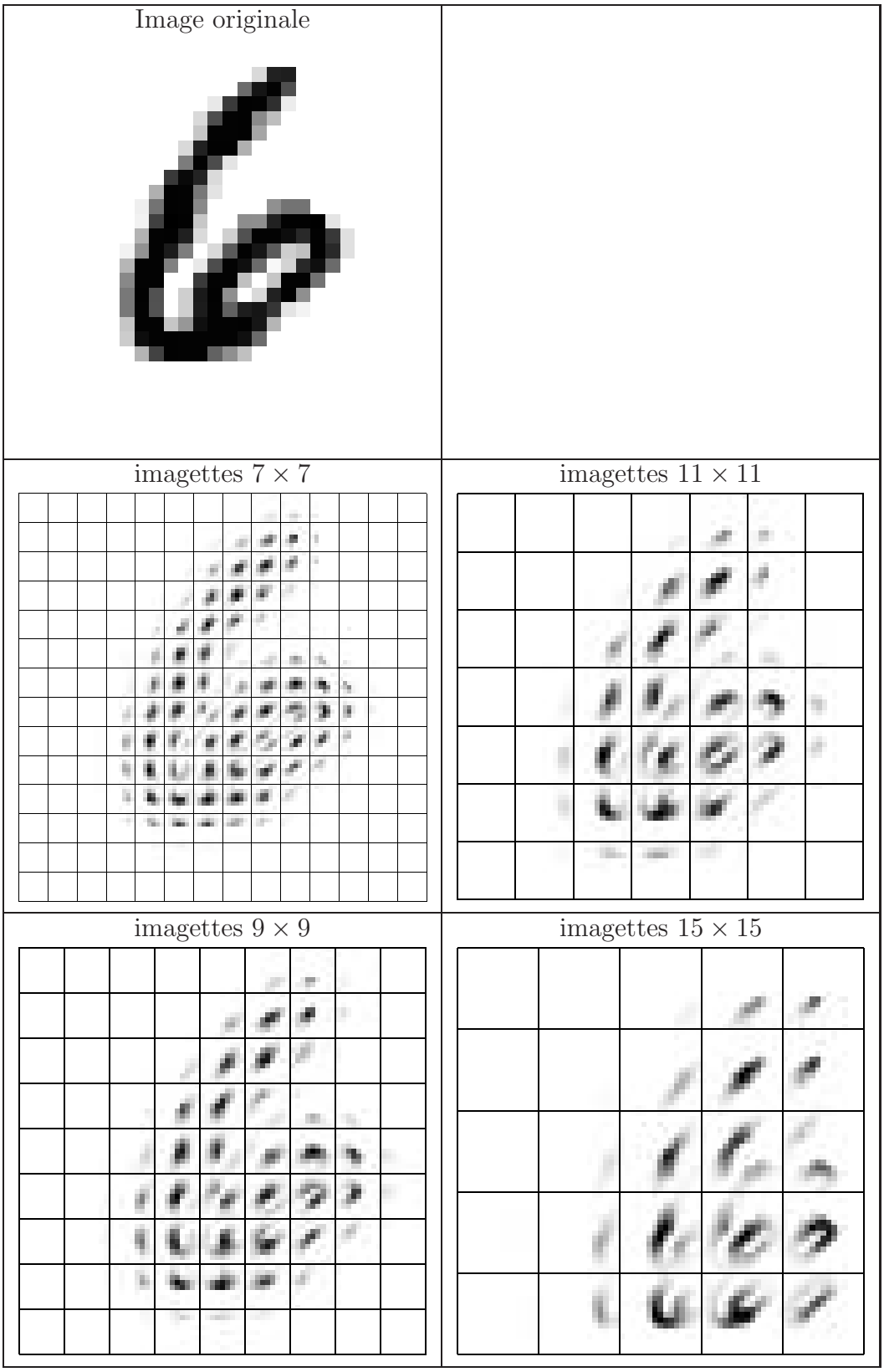
TAB. A.5 : Classe 3.



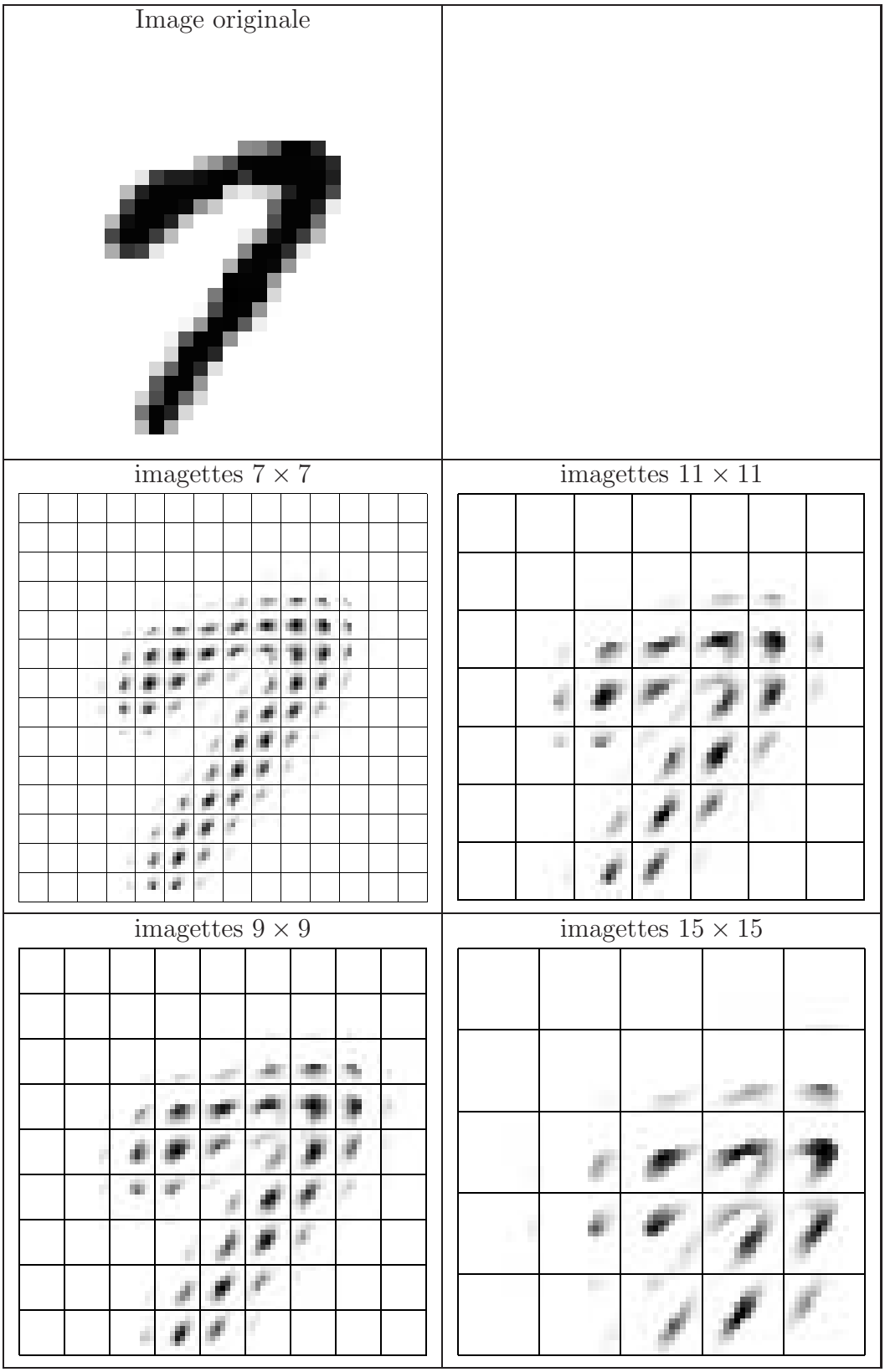
Tab. A.6 : Classe 4.



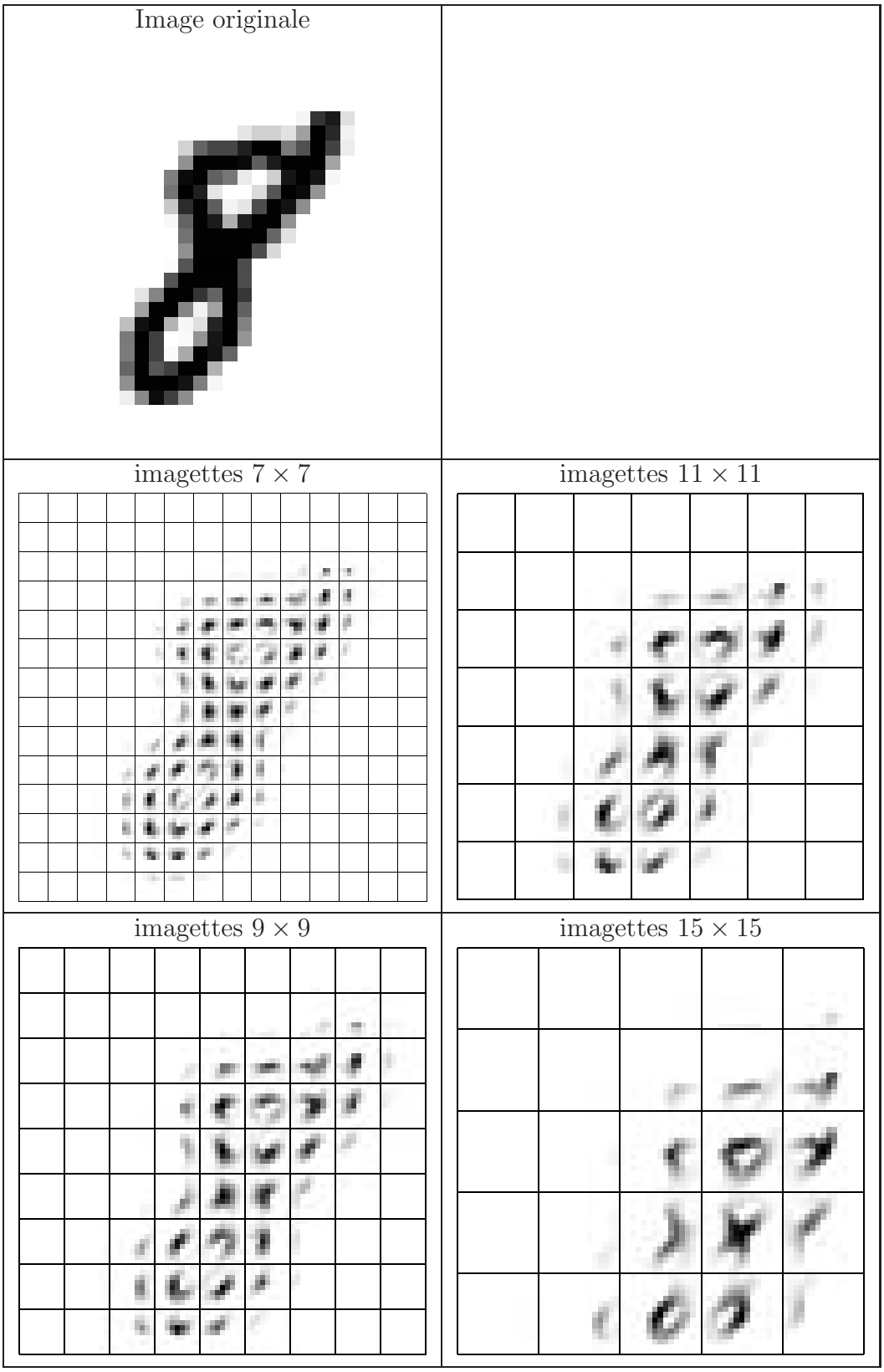
TAB. A.7 : Classe 5.



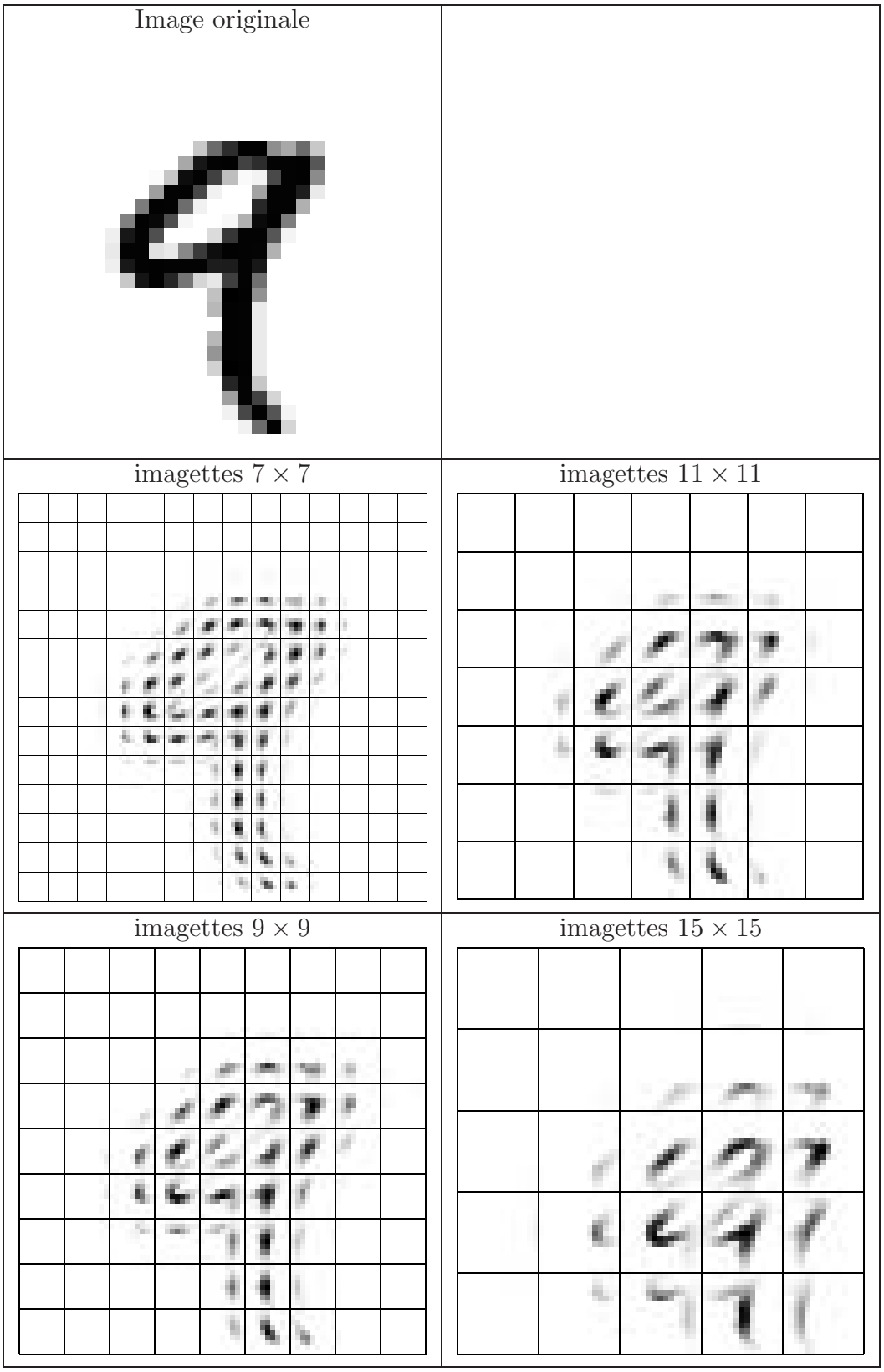
Tab. A.8 : Classe 6.



TAB. A.9 : Classe 7.



Tab. A.10 : Classe 8.



TAB. A.11 : Classe 9.

Annexe B

Histogrammes des états des modèles de chiffres manuscrits

Dans cette partie sont représentées les figures décrites dans la section 4.1.2 pour toutes les classes de la tâche de reconnaissance de chiffres manuscrits.

FIG. B.1 : Histogrammes du modèle 1.

FIG. B.2 : Histogrammes du modèle 2.

FIG. B.3 : Histogrammes du modèle 3.

FIG. B.4 : Histogrammes du modèle 4.

FIG. B.5 : Histogrammes du modèle 5.

FIG. B.6 : Histogrammes du modèle 6.

FIG. B.7 : Histogrammes du modèle 7.

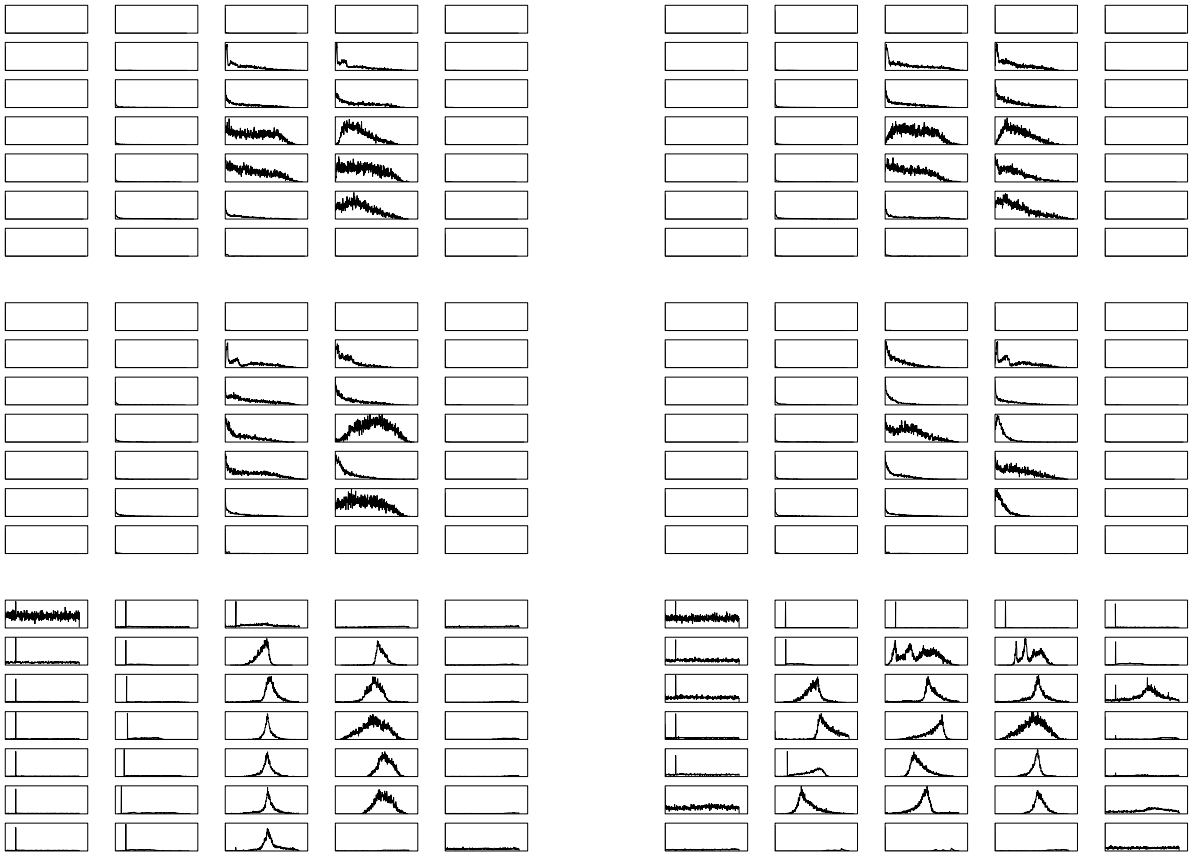


FIG. B.8 : Histogrammes du modèle 8.

FIG. B.9 : Histogrammes du modèle 9.

Annexe C

Application à une tâche de reconnaissance de route

Dans le but d'évaluer le principe de programmation dynamique sur une tâche de bas niveau, les algorithmes ont été utilisés dans le cadre d'une évalutation conduite par la DGA sur la reconnaissance de routes. Ce travail a été conduit avec deux stagiaires successifs, avec des résultats très intéressants en comparaison d'algorithmes proposés par des industriels.

C.1 La tâche de reconnaissance de route

La tâche de reconnaissance est très simple : le but est d'étiqueter les pixels d'images couleurs en *route* et *non route*. Pour cela on dispose d'une base de données d'images extraites de séquences video accompagnées de vérités terrains établies par des opérateurs humains (un exemple est donné par la figure C.1). Muni d'une telle base, on peut alors effectuer un apprentissage de modèles de route et de non route et évaluer les résultats.

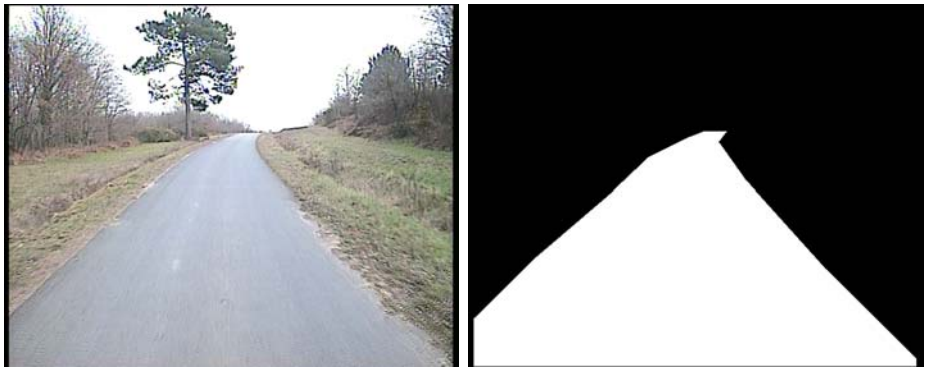


FIG. C.1 : Image de route et vérité terrain associée.

C.2 Modélisation utilisée

La réponse apportée à ce problème est une modélisation très simple :

➪ les observables sont les vecteurs des trois composantes (R, V et B) de couleurs des images sous-échantillonées,

➪ la modélisation se fait par un champ de Markov à au moins deux états (état *route* et état *non route*, mais on peut en choisir plus : état route mouillée, état ciel etc.)

➪ le décodage par programmation dynamique donne la segmentation en états la plus probable,

➪ les vérités-terrains pemettent de calculer directement les paramètres des modèles (les densités d'observation et les probabilités de transition)

C.3 Résultats obtenus

Le tableau C.1 donne les résultats finaux obtenus sur la base de test. Différents scénarios de séquences vidéo sont présents suivant le type de revêtement de la route, de conditions météo et de difficulté intrinsèque de la segmentation. Les taux d'erreurs représentés sont les taux d'erreurs de classification pour l'ensemble des pixels, pour les pixels *route* et les pixels *non route*.

Tab. C.1 : Résultats sur la base de test par scénario.

Nom du scénario	Nombre d'images	Taux d'erreur global	Taux d'erreur non route	Taux d'erreur route
goudron/couvert/difficile	52	7,73	8,58	5,02
terre/couvert/moyen	57	4,86	2,25	7,96
goudron/pluie/moyen	36	6,58	1,18	13,71
goudron/pluie/facile	100	4,51	1,26	9,03
goudron/soleil/facile	119	4,74	2,25	8,59
goudron/soleil/difficile	55	9,27	0,50	23,81
total	419	**5,82**	2,48	10,60

Un exemple de segmentation d'une image de route en *route*, *non route* et *ciel* est donné par la figure C.2.

FIG. C.2 : Segmentation d'une image de route.

Annexe D

Éléments sur l'implantation informatique

Dans cette partie, nous donnons quelques éléments sur l'implantation des algorithmes que nous avons effectuée. Classiquement, les processus d'apprentissage et de reconnaissance mettent en œuvre quelques procédures de base appelées de façon itérative par des scripts. Ces procédures s'appuient elles-mêmes sur des données dont le choix de la structure peut être déterminant.

D.1 Structure des données

À partir de la base de données MNIST, plusieurs formats d'images sont nécessaires :
- Les images sont enregistrées au format PNG, compressé sans pertes.
- Après calcul des vecteurs de primitives, chaque pixel est associé à un vecteur de flottants. À notre connaissance, il n'existe pas de format largement répandu pour les images de nombres flottants, nous avons donc utilisé notre propre format : les images de la base MNIST sont enregistrées, ligne par ligne, les unes après les autres dans un gros fichier. Chaque composante des vecteurs de primitives est cependant enregistrée dans un fichier différent pour permettre l'utilisation de différentes configurations des composantes de la FFT.
- Les segmentations en états sont représentées dans des images codées par des valeurs entières sur 1 ou plusieurs octets.

De plus, les valeurs des paramètres des modèles sont stockées de façon binaire dans plusieurs fichiers : le premier contient les valeurs des composantes gaussiennes pour l'ensemble des états, le second contient les potentiels d'interaction des cliques.

D.2 Procédures de base

L'ensemble de ces procédures, codées en C++, représentent 15 000 lignes de code. Ces procédures sont les suivantes :

- **Procédure de calcul des vecteurs de primitives :** cette procédure calcule les valeurs des observables. Le calcul des transformées de Fourier se fait au moyen de la librairie FFTW[1] qui est particulièrement rapide.
- **Procédure d'apprentissage des modèles à partir d'images segmentées :** ce programme calcule les paramètres des fonctions multigaussiennes associées à chacun des états au moyen d'un algorithme EM dont la convergence est rapide. Les potentiels de cliques sont aussi calculés par comptage.
- **Procédure de programmation dynamique 2D :** l'algorithme de programmation dynamique 2D a été implanté de façon particulièrement générale, toutes les stratégies de fusion sont possibles, et la mémoire est utilisée de façon efficace en initialisant les régions simplement au moment de la fusion et en effaçant à chaque itération l'ensemble des structures inutilisées.

En plus de ces trois procédures, deux composantes supplémentaires sont spécifiques au cas des mots manuscrits, ce sont les procédures de concaténation de modèles et de découpage d'images décrites dans la partie 4.2.3.

D.3 Scripts d'apprentissage et de reconnaissance

Un ensemble de scripts réalisés en Bash ou en Perl combinent les différentes procédures de base pour réaliser l'apprentissage des modèles ainsi que la phase de reconnaissance. Ils permettent de réaliser ces processus longs de façon robuste tout en répartissant sur plusieurs processeurs les calculs pouvant être réalisés en parallèle.

Le cas du traitement des mots manuscrits nécessite l'emploi de scripts plus complexes. En effet, ceux-ci doivent régulièrement analyser l'ensemble du dictionnaire pour construire l'ensemble des modèles de mots, et ce de façon efficace. De plus, plusieurs niveaux d'itérations sont nécessaires : calcul des modèles de lettres, construction des modèles de mots, augmentation de la taille des mots utilisés etc.

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

Bibliographie

[Alpaydin et Gurgen, 1996] Alpaydin, E. et Gurgen, F. (1996). Comparison of statistical and neural classifiers and their applications to optical character recognition and speech classification. *Neural networks systems techniques and applications*.

[Arica et Yarman-Vural, 2001] Arica, N. et Yarman-Vural, F. T. (2001). An overview of character recognition focused on off-line handwriting. *IEEE transactions on systems, man and cybernetics - part C : Applications and reviews*, 31(2) :216–233.

[Atukorale et Suganthan, 1999] Atukorale, A. S. et Suganthan, P. (1999). Combining classifiers based on confidence values. In *Proceedings of the fifth international conference on document analyis and recognition*, pages 37–40.

[Baret, 1990] Baret, O. (1990). *Régularités, singularités de représentations et leur complémentarité : Application à la reconnaissance de l'écriture manuscrite non contrainte*. PhD thesis, Université Paris 6.

[Belaid et Belaid, 1992] Belaid, A. et Belaid, Y. (1992). *Reconnaissance de formes*. Inter-Edition.

[Belaïd et Saon, 1997] Belaïd, A. et Saon, G. (1997). Utilisation des processus markoviens en reconnaissance de l'écriture. *Traitement du Signal*, 14(2) :161–177.

[Bellman, 1957] Bellman, R. E. (1957). *Dynamic Programming*. Princeton Univ. Press, Princeton, NJ.

[BenAmara, 1995] BenAmara, N. (1995). Reconnnaissance de caractères arabes imprimés par modèles de Markov pseudo-2D. In *RFIA'96*.

[Besag, 1974] Besag, J. (1974). Spatial interaction and the statistical analysis of lattice systems. *J. Roy. Stat. Soc., Ser. B*, 36 :192–236.

[Besag, 1986] Besag, J. (1986). Statistical analysis of dirty pictures. *Journal of the Royal Statitical Society*.

[Boite *et al.*, 1999] Boite, R., Boulard, H., Dutoit, T., Hancq, J., et Leich, H. (1999). *Traitement de la parole*. Presses Polytechniques et Universitaires Romandes.

[Cai et Liu, 2001] Cai, J. et Liu, Z. (2001). Pattern recognition using Markov random field models. *Pattern Recognition*, 35 :725–733.

[Casey et Takahashi, 1992] Casey, R. et Takahashi, H. (1992). Experience in segmenting and classifying the NIST database. In Impedovo, S. et Simon, J.-C., editors, *From pixels to features III*, pages 5–16. Elsevier Science.

[Casey et Lecolinet, 1996] Casey, R. G. et Lecolinet, E. (1996). A survey of methods and strategies in character segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 18(7) :690–706.

[CEDAR, 1993] CEDAR (1993). CEDAR CDROM-1. `www.cedar.buffalo.edu/Databases/CDROM1`.

[Cha *et al.*, 1999] Cha, S., Shin, Y., et S.N.Srihari (1999). Approximate stroke sequence string matching algorithm for character recognition and analysis. In *Proceedings of the ICDAR*, pages 53–56.

[Chellappa et Jain, 1993] Chellappa, R. et Jain, A., editors (1993). *Markov Random Fields - Theory and application*. Academic Press.

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

[Chevalier *et al.*, 2004] Chevalier, S., Geoffrois, E., et Prêteux, F. (2004). Programmation dynamique 2D pour la reconnaissance de caractèrs manuscrits par champs de Markov. *Actes de la conférence Reconnaissance des Formes et Intelligence Artificielle*.

[Choisy et Belaïd, 2000] Choisy, C. et Belaïd, A. (2000). Analytic word recognition without segmentation based on Markov random fields. In *Proceedings of the seventh international workshop on frontiers in handwriting recognition*, pages 487–492.

[Côté, 1997] Côté, M. (1997). *Utilisation d'un modèle d'accès lexical et de concepts perceptifs pour la reconnaissance d'images de mots cursifs*. PhD thesis, Ecole Nationale Supérieure des Télécommunications.

[Derras, 1993] Derras, M. (1993). *Segmentation non supervisée d'images texturées par champs de Markov : Application à l'automatisation de l'entretien des espaces naturels*. PhD thesis, Université Blaise Pascal de Clermont-Ferrand.

[Descombes, 1993] Descombes, X. (1993). *Champs markoviens en analyse d'images*. PhD thesis, École Nationale Supérieure des Télécommunications.

[Dubes et Jain, 1989] Dubes, R. C. et Jain, A. K. (1989). Random field models in image analysis. *Journal of Applied Statistics*, 16(2) :131–164.

[Duda *et al.*, 2001] Duda, R. O., Hart, P. E., et Stork, D. G. (2001). *Pattern Classification*. Wiley-Interscience.

[Dupré, 2003] Dupré, X. (2003). *Contibutions à la reconnaissance de l'écritutre cursive à l'aide de modèles de Markov cachés*. PhD thesis, Université Paris 5.

[El-Yacoubi *et al.*, 1999] El-Yacoubi, A., Gilloux, M., Sabourin, R., et Suen, C. (1999). An HMM-based approach for off-line unconstrained handwritten word modeling and recognition. *IEEE transactions on pattern analysis and machine intelligence*, 21(8) :752–760.

[Elyoubi, 1995] Elyoubi, A. (1995). *Statistiques exhaustives en reconnaissance de formes*. PhD thesis, Université de Rouen.

[ETL, 2001] ETL (2001). ETL character database. `http ://www.is.aist.go.jp/etlcdb/#English`.

[Favata *et al.*, 1994] Favata, J., Srikantan, G., et Srihari, S. (1994). Handprinted character/digit recognition using a multiple feature/resolution philosophy. In *Proceedings of the Fourth International Workshop on Frontiers in Handwriting Recognition*, pages 57–66.

[Feray et de Brucq, 1996] Feray, N. et de Brucq, D. (1996). Système de reconnaissance de chiffres manuscrits hors-lignes. *Traitement du signal*.

[Forney, 1973] Forney, G. D. (1973). The Viterbi algorithm. *Proc. IEEE*, 61 :268–278.

[Garris, 1996] Garris, M. D. (1996). Component-based handprint segmentation using adaptive writing style model. Technical report, NIST.

[Geman et Geman, 1984] Geman, S. et Geman, D. (1984). Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 6(6).

[Geman et Geman, 1993] Geman, S. et Geman, D. (1993). *Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images*, volume 1, chapter 4. Carfax publishing company.

[Geoffrois, 2003] Geoffrois, E. (2003). Multi-dimensional Dynamic Programming for statistical image segmentation and recognition. *International Conference on Image and Signal Processing*.

[Geoffrois *et al.*, 1998] Geoffrois, E., Jullian, A., et Debaert, C. (1998). Programmation dynamique 2D pour la reconnaissance d'images par modèles de Markov cachés. Technical report, DGA/DCE/CTA/GIP.

[Gilloux, 1994a] Gilloux, M. (1994a). Hidden Markov models in handwriting recognition. In [Impedovo, 1994], pages 264–288.

[Gilloux, 1994b] Gilloux, M. (1994b). Reconnaissance de chiffres manuscrits par modèles de Markov pseudo-2D. In *CNED'94*, pages 11–17.

[Gilloux *et al.*, 1995] Gilloux, M., Lemarié, B., et Leroux, M. (1995). A hybrid radial basis function network/hidden Markov model handwritten word recognition system. In *Proc. of the 3rd Int. Conf. on Document Analysis and Recognition (ICDAR'95)*.

[Govindan et Shivaprasad, 1990] Govindan, V. et Shivaprasad, A. (1990). Character recognition - a review. *Pattern recognition*, 23(7) :671–683.

[Grandidier *et al.*, 2000] Grandidier, F., Sabourin, R., Suen, C., et Gilloux, M. (2000). Une nouvelle stratégie pour l'amélioration des jeux de primitives d'un système de reconnaissance de l'écriture. In *Actes du deuxième Colloque International Francophone sur l'Ecrit et le Document*.

[Grandidier *et al.*, 1999] Grandidier, F., Sabourin, R., Yacoubi, A. E., Gilloux, M., et Suen, C. (1999). Influence of word length on handwriting recognition. In *Proceedings of the Fifth International Conference on Document Analysis and Recognition*, pages 777–780.

[Guillevic, 1995] Guillevic, D. (1995). *Unconstrained handwriting recognition applied to the processing of bank cheques*. PhD thesis, Concordia University, Montréal, Canada.

[Hamamoto *et al.*, 2001] Hamamoto, Y., Uchimura, S., Watanabe, M., Yasuda, T., et Tomita, S. (2001). Recognition of handwritten numerals using Gabor features. In *Proceedings of ICPR 96*.

[Hildebrandt et Liu, 1993] Hildebrandt, T. et Liu, W. (1993). Optical recognition of handwritten Chinese characters : advances since 1980. *Pattern Recognition*, 26(2).

[Huang *et al.*, 2001] Huang, X., Acero, A., et Hon, H.-W. (2001). *Spoken language processing*. Prentice Hall.

[Hull, 1993] Hull, J. J. (1993). A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16 :550–554.

[Huo *et al.*, 2001] Huo, Q., Ge, Y., et Feng, Z.-D. (2001). High performance Chinese OCR based on Gabor features, dicriminative feature extraction and model training. In *ICASSP 2001*.

[Hyman *et al.*, 1991] Hyman, S., Vogl, T., Blackwell, K., Barbour, G., Irvine, J., et Alkon, D. (1991). Classification of Japanese kanji using principal component analysis as a preprocessor to an artificial neural network. In *IJCNN91 : Proc. Int. Joint Conf. on neural networks*, volume 1, pages 233–238.

[Impedovo, 1994] Impedovo, S., editor (1994). NATO ASI Series. Springer-Verlag.

[Jain et Bhattacharjee, 1992] Jain, A. et Bhattacharjee, S. (1992). Address block location on envelopes using Gabor filters. *Pattern Recognition*, 25(12).

[Knerr *et al.*, 1997] Knerr, S., Anisinov, V., Baret, O., Gorski, N., Price, D., et Simon, J.-C. (1997). The A2iA intercheque system : courtesy amount and legal amount recognition for French checks. *International journal of pattern recognition and artificial intelligence*, 11(4) :505–548.

[Knerr et Augustin, 1998] Knerr, S. et Augustin, E. (1998). A neural network-hidden Markov model hybrid for cursive word recognition. In *Proc. of ICPR Brisbane 1998*, volume 2.

[Kobayashi *et al.*, 1983] Kobayashi, K., Yoda, F., Yamamoto, K., et Nambu, H. (1983). Recognition of handprinted kanji characters by the stroke matching method. *Pattern Recognition Letters*, 1(5–6) :481–488.

[Kuo et Agazzi, 1994] Kuo, S. et Agazzi, O. (1994). Keyword spotting in poorly printed documents using pseudo 2D hidden Markov models. *IEEE transactions on PAMI*, 16(8) :842–848.

[Lampinen *et al.*, 2001] Lampinen, J., Tamminen, T., Kostiainen, T., et Kalliomäki, I. (2001). Bayesian object matching based on MCMC sampling and Gabor filters. In *Proc. SPIE Intelligent Robots and Computer Vision XX : Algorithms, Techniques and Active Vision*, volume 4572, pages 41–50.

[Lecce *et al.*, 2000] Lecce, V., Dimauro, G., Guerriero, A., Impedovo, S., Pirlo, G., et Salzo, A. (2000). Classifiers combination : the role of a-priori knowledge. In *Proceedings of the Seventh International Workshop on Frontiers in Handwriting Recognition*, pages 143–152.

[LeCun, 1998] LeCun, Y. (1998). The MNIST database. `http ://yann.lecun.com/exdb/mnist/index.html`.

[Leroux *et al.*, 1997] Leroux, M., Lethélier, E., Gilloux, M., et Lemarié, B. (1997). Automatic reading of handwritten amounts on French checks. *International journal of pattern recognition and artificial intelligence*, 11(4) :619–638.

[Levin et Pieraccini, 1992] Levin, E. et Pieraccini, R. (1992). Dynamic planar warping for optical character recognition. In *IEEE International Conference on Accoustics, Speech and Signal Processing*.

[Li, 1994] Li, S. Z. (1994). Markov random fields models in computer vision. In *Proceedings of the European Conference on Computer Vision*, pages 361–370.

[Lorette et Crettez, 1998] Lorette, G. et Crettez, J.-P. (1998). Reconnaissance de l'écriture manuscrite. In *Traité informatique, techniques de l'ingénieur*, pages 1–15.

[Marti et Bunke, 1999] Marti, U. et Bunke, H. (1999). Towards general cursive script recognition. In *Advances in Handwriting Recognition*, pages 203–212. World Scientific.

[Mohamed et Gader, 1996] Mohamed, M. et Gader, P. (1996). Handwritten word recognition using segmentation-free hidden Markov model and segmentation based dynamic programming techniques. *IEEE transactions on pattern analysis and machine intelligence*, 18(5).

[NIST, 1995] NIST (1995). NIST special database 19. `http ://www.nist.gov/srd/nistsd19.htm`.

[Oka, 1982] Oka, R. (1982). Handwritten Chinese-Japanese characters recognition by using cellular features. In *ICPR82 : Proc. 6th Int. Conf on Pattern Recognition*, pages 783–785.

[Park et Lee, 1998] Park, H.-S. et Lee, S.-W. (1998). A truly 2D hidden Markov model for off-line handwritten character recognition. *Pattern Recognition*, 31(12) :1949–1864.

[Pessoa, 1995] Pessoa, L. F. C. (1995). Multilayer perceptrons versus hidden Markov models : comparison and applications to image analysis and visual pattern recognition. Technical report, Georgia Institute of Technology.

[Procter *et al.*, 2000] Procter, S., Illingworth, J., et Mokhtarian, F. (2000). Cursive handwriting recognition using hidden Markov models and a lexicon-driven level building algorithm. In *IEE Proc.-Vis. Image Signal Process.*, volume 147.

[Prêteux, 1991] Prêteux, F. (1991). L'approche markovienne en analyse d'image et en reconnaissance de forme. Support de cours de l'INT.

[Pérez, 1993] Pérez, P. (1993). *Champs markoviens et analyse multirésolution de l'image : application à l'analyse du mouvement*. PhD thesis, Université de Rennes I.

[Pérez, 1999] Pérez, P. (1999). Modèles markoviens en analyse d'images. notes de cours de DEA STIR - Université de Rennes I.

[Rabiner et Juang, 1993] Rabiner, L. et Juang, B.-H. (1993). *Fundamentals of speech recognition*. Prentice Hall PTR.

[Saon, 1997] Saon, G. (1997). *Modèles markoviens uni- et bidimensionnels pour la reconnaissance de l'écriture manuscrite hors-ligne*. PhD thesis, Université Henri Poincaré - Nancy 1.

[Saon et Belaïd, 1997] Saon, G. et Belaïd, A. (1997). High performance unconstrained word recognition system combining HMMs and Markov random fields. *International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI) Special Issue on Automatic Bankcheck Processing*.

[Saon *et al.*, 1995] Saon, G., Belaïd, A., et Gong, Y. (1995). Stochastic trajectory modeling for recognition of unconstrained handwritten words. In *Third International Conference on document analysis and recognition*, pages 508–511.

[Senior et Robinson, 1998] Senior, A. W. et Robinson, T. (1998). An off-line cursive handwriting recognition system. *IEEE Pattern Analysis and machine intelligence*, 20(3) :309–321.

[Sigelle et Tupin, 1999] Sigelle, M. et Tupin, F. (1999). Champs de markov en traitement d'image. École Nationale Supérieure des Télécommunications.

[Simon *et al.*, 1994] Simon, J., Baret, O., et Gorski, N. (1994). A system for the recognition of handwritten literal amounts of checks. In *Internal Association for pattern recognition workshop on document analysis system*, pages 135–155.

[Simon et Baret, 1990] Simon, J.-C. et Baret, O. (1990). Regularities and singularities in line images. In *Pre-proceedings SSPR90*, pages 423–439.

[Steinherz *et al.*, 1999] Steinherz, T., Rivlin, E., et Intrator, N. (1999). Off-line cursive script word recognition - a survey. *International journal of document analysis and recognition*, 2(2) :90–110.

[Trier *et al.*, 1996] Trier, O., Jain, A., et Taxt, T. (1996). Feature extraction methods for character recognition - a survey. *Pattern Recognition*, 29(4) :641–662.

[Tsukumo et Tanaka, 1988] Tsukumo, J. et Tanaka, H. (1988). Classification of handprinted Chinese characters using nonlinear normalization and correlation methods. In *ICPR88 : 9th Int. Conf. on Pattern Recognition*, volume 1, pages 168–171.

[Vincieralli et Luettin, 2000] Vincieralli, A. et Luettin, J. (2000). Off-line cursive script recognition based on continuous density hmm. In *Proceedings of the seventh international workshop on frontiers in handwriting recognition*.

[Vintsyuk, 1968] Vintsyuk, T. K. (1968). Speech discrimination by dynamic progamming. *Kibenetika*, 4 :81–88.

[Wang, 1994] Wang, J. (1994). *Champs Markoviens multi-échelles : applications à la segmentation d'images texturées et à la fusion multi-film*. PhD thesis, Paris Sud Orsay.

[Wang *et al.*, 2000] Wang, Q., Chi, Z., Feng, D. D., et Zhao, R. (2000). Hidden Markov random field based approach for off-line handwritten Chinese character recognition. In *Proceedings of the international conference on pattern recognition*.

[Xiong *et al.*, 2001] Xiong, Y., Huo, Q., et Chan, C. (2001). A discrete contextual stochastic model for the offline recognition of handwritten Chinese characters. *IEEE transactions on pattern analysis and machine intelligence*, 23(7).

[Xue et Govindaraju, 2000] Xue, H. et Govindaraju, V. (2000). Character recognition by matching sequences of pseudo-stroke positions and directions. In *Proceedings of the Seventh International Workshop on Frontiers in Handwriting Recognition*, pages 589–594.

[Yamamoto *et al.*, 1986] Yamamoto, K., Yamada, H., Saito, T., et Sakaga, I. (1986). Recognition of handprinted characters in the first level of JIS Chinese characters. In *ICPR86 : 8th Int. Conf. on Pattern Recognition*, pages 570–572.

[Yamashita *et al.*, 1983] Yamashita, Y., Higuchi, K., Yamada, Y., et Haga, Y. (1983). Classification of handprinted kanji characters by the structured segment matching method. *Pattern Recognition Letters*, 1(5–6) :475–479.

[Young et Bloothooft, 1997] Young, S. et Bloothooft, G., editors (1997). *Corpus-based methods in language and speech processing*. Kluwer Academic.

[Yu *et al.*, 2001] Yu, S., Lee, T. S., et Kanade, T. (2001). A hierarchical Markov random field model for figure-ground segregation. In *Third International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR'01)*.

[Zhou *et al.*, 2000] Zhou, J., Krzyzak, A., et Suen, C. Y. (2000). Recognition and verification of touching handwritten numerals. In *Proceedings of the Seventh International Workshop on Frontiers in Handwriting Recognition*, pages 179–188.