Programmation dynamique 2D pour la reconnaissance de caractères manuscrits par champs de Markov

Edouard Geoffrois[1]
Edouard.Geoffrois@etca.fr

Sylvain Chevalier[1,2]
Sylvain.Chevalier@{etca,int-evry}.fr

Françoise Prêteux[2]
Francoise.Preteux@int-evry.fr

[1] DGA/Centre Technique d'Arcueil
16 bis avenue Prieur de la Côte d'Or
94114 Arcueil cedex

[2] GET/INT, Unité de projets ARTEMIS
9, rue Charles Fourier
91011 Evry cedex

Résumé

Une extension naturelle de la programmation dynamique au cas d'une dimension quelconque dans le cadre d'une modélisation par champs de Markov est proposée. Ce principe a été adapté à une tâche de reconnaissance de caractères manuscrits et testé sur la base de données standard MNIST. Les performances obtenues sont similaires à celles rapportées dans la littérature.

Mots Clef

Programmation dynamique 2D, champs de Markov, OCR.

Abstract

A natural extension of the traditional 1D dynamic programming to a multi-dimensional case within an hidden Markov random field modeling framework is proposed. This principle has been adapted to a handwritten character recognition task and tested on the MNIST database. Preliminary results exhibit an error rate similar to the ones reported in the literature.

Keywords

2D Dynamic Programming, Markov Random Fields, OCR.

1 Introduction

Le principe de la programmation dynamique est à la base de nombreux algorithmes d'optimisation de dimension 1, en particulier pour des fonctions temporelles [22, 26]. La reconnaissance de la parole par exemple a connu de grands succès grâce à l'utilisation des chaînes de Markov cachées (HMM) optimisées par des algorithmes fondés sur la programmation dynamique comme l'algorithme de Viterbi [23, 16]. Le succès des méthodes à base de programmation dynamique pour ces problèmes a incité à l'utiliser aussi en traitement d'images [25, 31], et plus particulièrement encore en reconnaissance de caractères manuscrits.

En effet, les chaînes de Markov et la programmation dynamique 1D sont déjà couramment utilisées pour la reconnaissance d'écriture cursive [1, 6], ainsi que les champs de Markov pour la reconnaissance de l'écriture manuscrite hors ligne [13, 7], notamment pour la reconnaissance de caractères chinois [30]. Les modèles utilisés sont des champs de Markov causaux, dont le type de dépendance locale permet un parcours 1D des pixels de l'image [24] ou des modèles de Markov Pseudo-2D (combinaison de deux chaînes de Markov) [13]. Il n'est donc pas étonnant qu'une des tentatives les plus avancées pour généraliser la programmation dynamique au cas 2D soit dans le domaine de la reconnaissance de caractères [20].

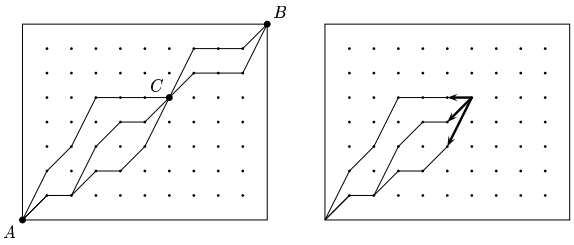
Cependant, ces diverses utilisations de la programmation dynamique au traitement d'images et à la reconnaissance de caractères ne la généralisent que de manière limitée. Certaines ne modifient pas l'algorithme de base et recourent à des artifices pour l'appliquer à la structure bidimensionnelle des images [25, 31]. D'autres étendent l'algorithme plus profondément, mais en se restreignant à des parcours particuliers [20].

Nous proposons une extension canonique et naturelle du concept de programmation dynamique au cas multidimensionnel et présentons les détails de son implantation et les premières applications du principe sur une tâche de reconnaissance de caractères manuscrits.

2 La programmation dynamique 2D

Le concept de programmation dynamique a été proposé dès 1957 pour une tâche de contrôle optimal [3], il a été introduit en reconnaissance de la parole en 1968 [29] et est toujours resté au cœur des meilleurs systèmes depuis.

Le principe de l'algorithme, à la fois simple et puissant (cf. Figure 1), permet la recherche d'un chemin optimal dans un espace combinatoire avec une complexité linéaire, au lieu d'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 [11]). 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 en-



(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. 1 – 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.

semble.

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 [8, 21]. 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 (Iterated Conditional Mode [5]) est rapide mais largement sous-optimal, et le recuit simulé [12] doit converger extrêmement lentement pour assurer l'optimalité de la solution. Ceci est dû au fait que ces techniques n'exploitent pas la structure de l'espace de recherche.

La programmation dynamique, fondée sur le principe d'optimalité de Bellman, 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 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 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 située à la frontière entre ces deux régions. La frontière d'une région R étant définie comme l'ensemble des pixels susceptibles d'interagir avec d'autres régions et son intérieur comme l'ensemble des autres pixels, pour chaque configuration de la frontière il n'existe qu'une seule configuration optimale de l'intérieur. Pour chaque configuration de la frontière de $R_1 \cup R_2$, on peut alors déterminer sa configuration optimale, en explorant seulement les différentes configurations des frontières de R_1 et R_2. C'est ce principe qui est au cœur de la programmation dynamique 2D.

2.1 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 [4], fait l'hypothèse d'une dépendance locale contextuelle :

$$P(o|\omega) = \prod_{(i,j)} P(o_{(i,j)}|\omega_{(i,j)}) \qquad (1)$$

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

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 :

$$U(\omega) = \sum_{(i,j)} -\log(P(o_{(i,j)}|\omega_{(i,j)})) + \sum_{c \in C} V_c(\omega), \qquad (2)$$

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

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

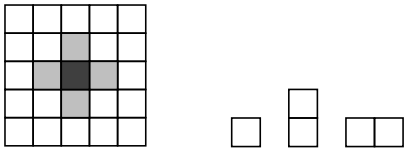


FIG. 2 – Voisinage du premier ordre (a) et cliques associées (b).

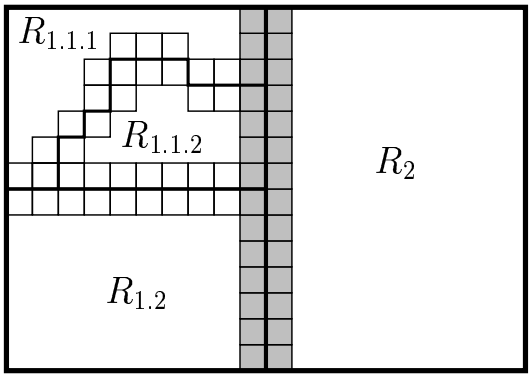


FIG. 3 – Partition de l'image en deux régions R_1 et R_2, qui peuvent à leur tour être divisées en sous-régions. La partition peut être régulière ($R_1 = R_{1.1} \cup R_{1.2}$) ou pas ($R_{1.1} = R_{1.1.1} \cup R_{1.1.2}$). Seuls les sites appartenant aux frontières sont représentés. La frontière de R_1 et R_2 est en gris.

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 2 mais la méthode est valable pour tout type de voisinage.

2.2 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 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 (cf. figure 3). 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). 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 récrite 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), 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, $\hat{\Omega}_r$

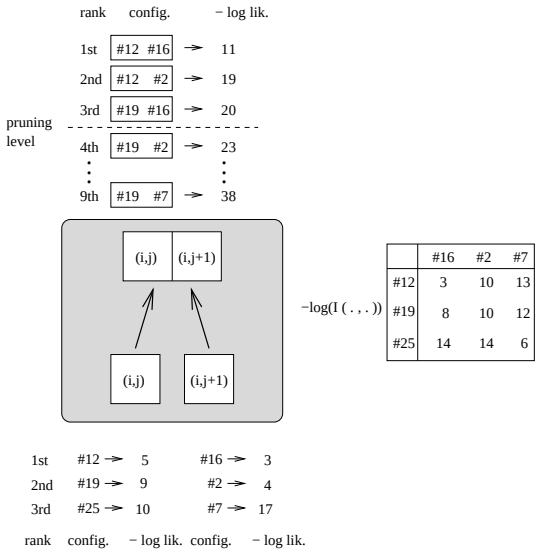


FIG. 4 – Fusion de deux pixels : La région résultant de la fusion est associée à une nouvelle liste de configurations possibles et à une vraisemblance qui est la somme des vraisemblances des deux pixels et du terme d'interaction I.

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 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 4). 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 5). 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 (voir section 3.6). 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 (voir Figure 6).

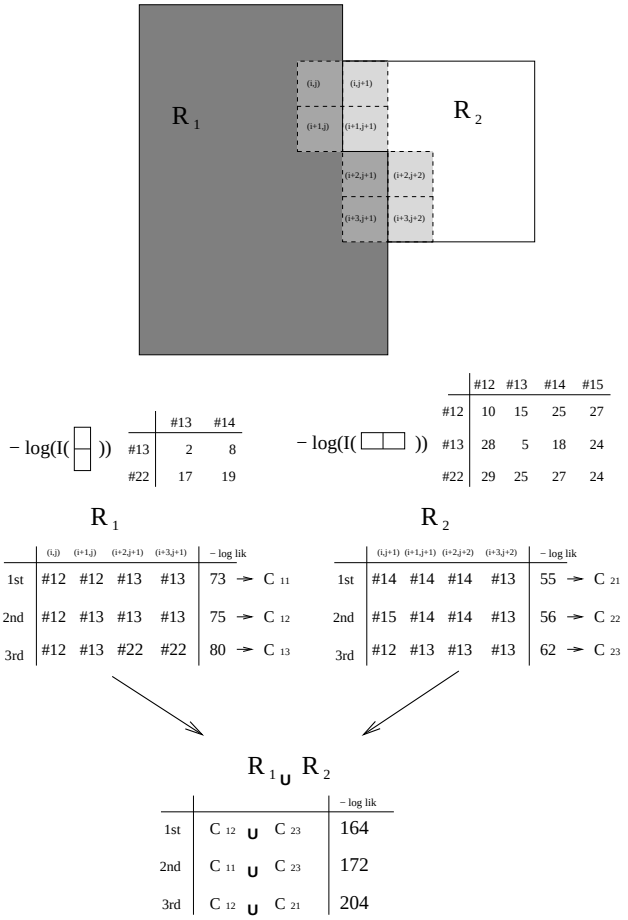


FIG. 5 – Fusion de deux régions : La région résultant de la fusion $R_1 \cup R_2$ est associée à une nouvelle liste de configurations possibles et une vraisemblance correspondante qui est la somme des vraisemblances des deux régions R_1 et R_2 et des termes d'interaction I.

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 1) 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.

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

Une fois défini le cadre théorique, l'implantation du modèle nécessite de faire des choix, en particulier pour la fonction de potentiel ou le choix du type d'observations.

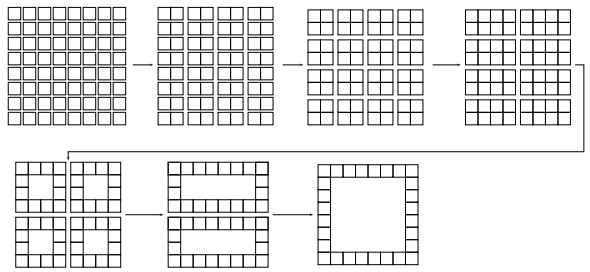


FIG. 6 – Une séquence de fusion canonique débutant par n^2 régions d'un seul pixel jusqu'à une seule région recouvrant l'image entière. À chaque étape, les régions sont fusionnées deux par deux. Seuls les pixels frontière sont représentés.

Cette section présente les détails de cette implantation.

➜ **Les probabilités de transition**

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

$$I_0(\omega_k,\omega_l)=\frac{P\left(\begin{array}{|c|}\hline\omega_l\\\hline\omega_k\\\hline\end{array}\right)}{P(\omega_k)P(\omega_l)},\ I_1(\omega_k,\omega_l)=\frac{P\left(\begin{array}{|c|}\hline\omega_k\\\hline\omega_l\\\hline\end{array}\right)}{P(\omega_k)P(\omega_l)}$$
$$I_2(\omega_k,\omega_l)=\frac{P\left(\begin{array}{|c|c|}\hline\omega_l&\omega_k\\\hline\end{array}\right)}{P(\omega_k)P(\omega_l)},\ I_3(\omega_k,\omega_l)=\frac{P\left(\begin{array}{|c|c|}\hline\omega_k&\omega_l\\\hline\end{array}\right)}{P(\omega_k)P(\omega_l)}, \quad (3)$$

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 [9] 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 :

$$T(i,j)(k)=I_k(\omega_i,\omega_j),\ 1\leqslant i,j\leqslant n,\ 0\leqslant k\leqslant 3. \quad (4)$$

➜ **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, voir section 3.3), et sur le type de modélisation de ces lois de probabilité (voir section 3.4). Dans le cas d'observations continues, on utilise en général une modélisation par des mélanges de gaussiennes, de la forme :

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

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 [16]. 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 :

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

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

$$c_k'=\frac{1}{N}\sum_{i=1}^{N}p_k^i,$$
$$\mu_k'=\frac{\sum_{i=1}^{N}p_k^io_i}{\sum_{i=1}^{N}p_k^i},$$
$$\Sigma_k'=\frac{\sum_{i=1}^{N}p_k^i(o_i-\mu_i')(o_i-\mu_i')^t}{\sum_{i=1}^{N}p_k^i}.$$

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),

$$P(k)G(o,\mu_k,\Sigma_k)\longrightarrow$$
$$(1-\alpha_1)P(k)G(o,\mu_k,\Sigma_k)+\alpha_1P(k)G(o,\mu_k+\alpha_2\Sigma_k,\Sigma_k),$$

FIG. 7 – échantillons de la base MNIST

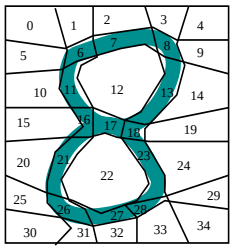


FIG. 8 – Configuration attendue d'une image.

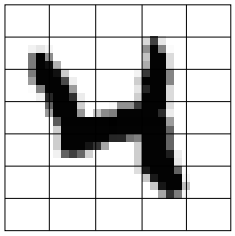


FIG. 9 – Initialisation : segmentation initiale utilisée pour le calcul des paramètres du modèle initial.

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

3 Application à la reconnaissance de caractères manuscrits

Le principe de la programmation dynamique 2D est très général et une grande variété d'applications serait susceptible d'en tirer parti. Nous travaillons à l'appliquer à diverses tâches et présentons ici son application à une tâche de reconnaissance de caractères manuscrits.

3.1 Base de données utilisée

Les premières expériences de l'algorithme de programmation dynamique 2D pour la reconnaissance d'écriture manuscrite ont été menées sur la base de données MNIST [1] qui est une base standard composée de 70.000 images de chiffres extraits de la base NIST (Figure 7) 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.

3.2 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 (voir section 3.3), 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 (cf figure 8).
L'algorithme de programmation dynamique 2D est utilisé à la fois au niveau de l'apprentissage, effectué par une approche EM, 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 EM. 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. 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 4 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 (Figure 9). 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 conver-

[1] http ://yann.lecun.com/exdb/mnist

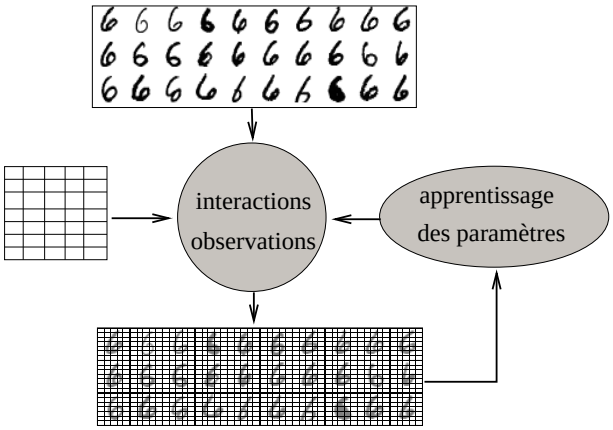


FIG. 10 – Apprentissage : À chaque itération, les paramètres du modèle sont calculés sur les segmentation issues de l'itération précédente.

gence de la vraisemblance (Figure 10). 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.

3.3 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 [2, 27] et sur le choix du type de primitives [14]. On peut distinguer deux classes dans les types de primitives utilisées : les primitives globales et locales.
Les primitives globales, décrites en détails dans [28] recherchent des invariants dans l'image entière, c'est donc surtout au niveau de l'extraction de primitives qu'on cherchera à modéliser 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 [15] sont très utilisées et cherchent 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 [10, 18].
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 premier coefficients sont gardés (voir figure 11). Le premier coefficient contient l'information de l'énergie de la fenêtre et les quatre suivants donnent les intensités dans les quatre directions principales (voir tableau 1). Le premier coefficient n'est pas pertinent pour notre modélisation car

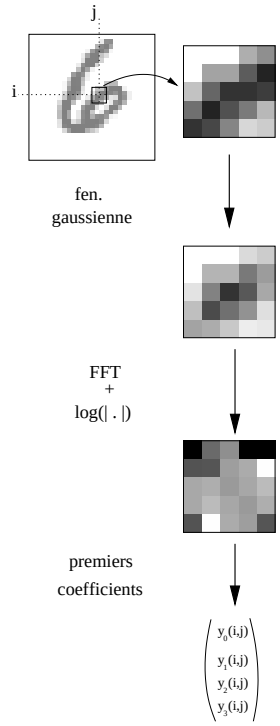


FIG. 11 – Processus d'extraction des primitives.

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 [17, 19].
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 le point 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.

3.4 Modélisation des densités d'observation

Les vecteurs d'observation, issus du calcul des primitives spectrales locales (cf section 3.3) sont modélisés par une fonction multigaussienne (voir section 2.3. En pratique, on fixe à $M = 8$ le nombre maximal de composantes gaussiennes et le seuil en log-vraisemblance est fixé à 10^{-8} pour les itérations de l'algorithme EM. Avec ces valeurs, on a des multigaussiennes qui modélisent bien les distributions réelles (cf figure 12).

3.5 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'obser-

TAB. 1 – Les primitives spectrales locales

Vert.	Hor.	1ère diag.	2nde diag.
			
			
			
			

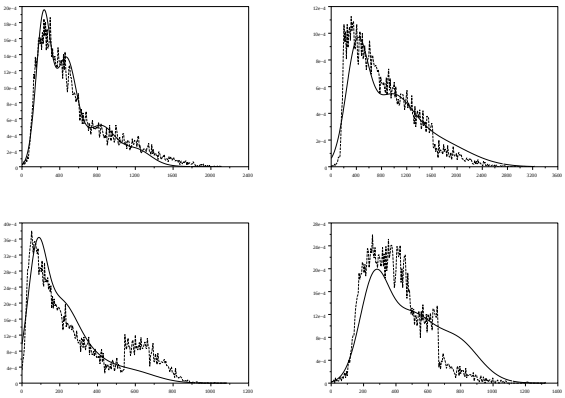


FIG. 12 – Exemples d'histogrammes des données d'apprentissage et fonctions multigaussiennes associées (4 composantes des sites affectés à l'étiquette 8 (cf figure 8) des données d'apprentissage de la classe du chiffre 0).

vation (cf équation 2). 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. Chaque fois que deux régions sont fusionnées, le processus d'élagage agit en deux étapes :

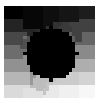
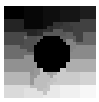
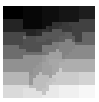
1. Les configurations dont la différence de vraisemblance avant et après la fusion est au dessus d'un seuil sont élaguées.
2. On ne garde ensuite qu'un nombre maximal de configurations à chaque étape.

Seules les configurations les moins prometteuses, loin de la configuration optimale sont élaguées, la configuration optimale globale est donc généralement préservée. En pratique, 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 fixé à 30, c'est un compromis entre la qualité de la segmentation et le temps de calcul.

3.6 Politique de fusion

La politique 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 9. On fusionne ensuite les pixels, en commençant par ceux du bord pour lesquels l'incertitude est la moins grande, puis les pixels de plus en plus proches du centre (voir tableau 2).

TAB. 2 – Politique de fusion : La configuration la plus probable de la plus grande région est représentée, chacune des 35 étiquettes est associée à un niveau de gris. Les pixels qui n'ont pas encore été fusionnés avec cette région sont laissés en noir.

nb de régions	700	600	500	400
				
nb de régions	300	200	100	1
				

3.7 Résultats sur la base de test

Les premiers 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 (voir figure 13).

Une caractéristique importante de notre approche est que le processus d'optimisation et de choix des différents paramètres 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 3.

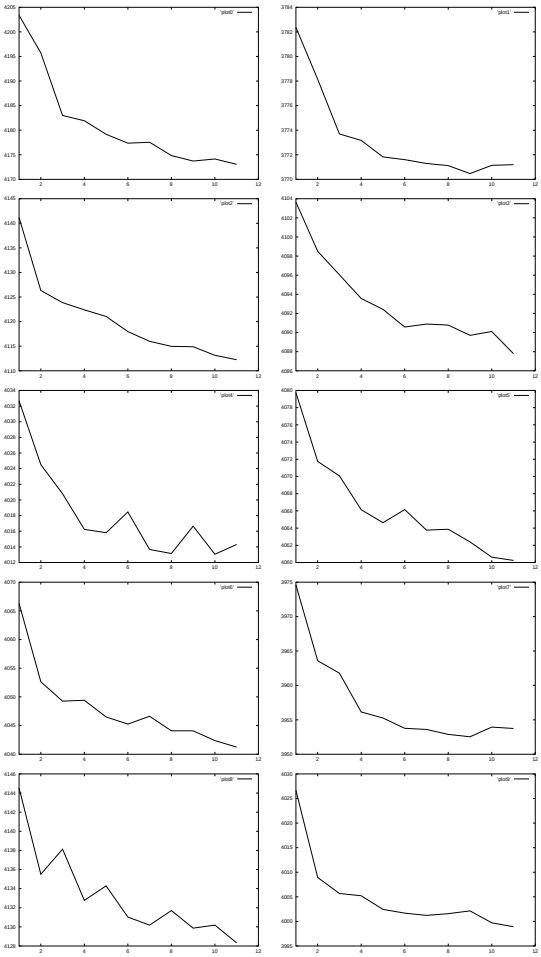


FIG. 13 – Convergence des dix modèles : Le coût moyen des données d'apprentissage ($-\log(P)$) est tracé en fonction du nombre d'itérations dans le calcul du modèle.

On peut tracer les limites des zones de même étiquette (voir figure 14), 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 3,7% sur l'ensemble des données de test de la base MNIST, ce qui est très proche des meilleurs résultats publiés dans la littérature. Des performances sensiblement accrues sont attendues avec l'amélioration des primitives utilisées et des stratégies d'apprentissage et d'élagage. Le processus de reconnaissance permet de traiter de l'ordre de 3 images par seconde sur un processeur à 800 Mhz, et ce temps de calcul devrait lui aussi être amélioré avec l'optimisation de la politique

TAB. 3 – Paramètres de la modélisation et taux d'erreur associé.

nbre d'étiq	fen gauss	élagage	nbre d'obs	nbre iter	tx err.
5×7	7×7	30	14×14	12	3,7%

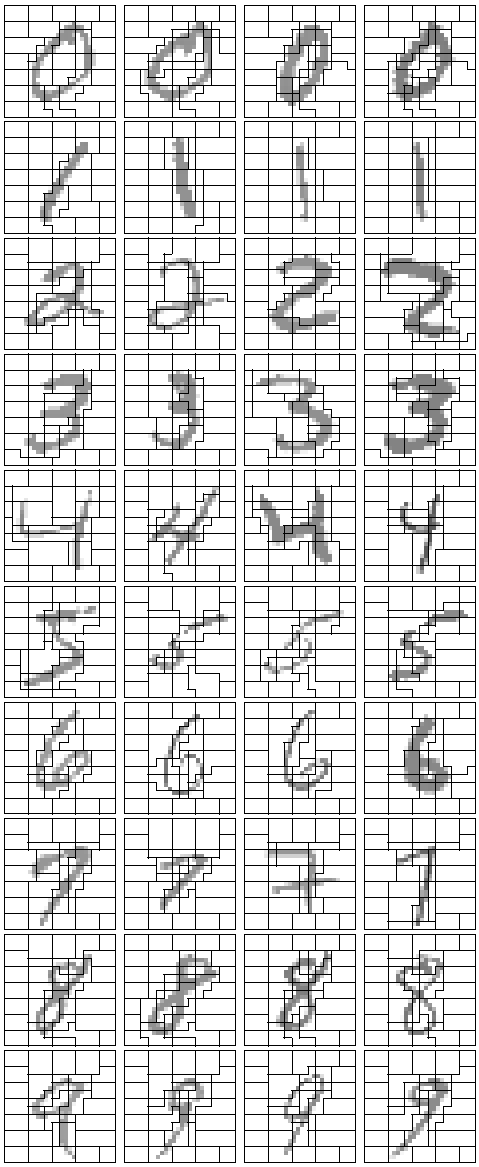


FIG. 14 – Résultats de la segmentation des images.

de fusion.

4 Conclusion et perspectives

Nous avons présenté une généralisation naturelle du principe de programmation dynamique au cas multidimensionnel ainsi que les résultats de son application à une tâche de reconnaissance de caractères manuscrits. L'approche retenue est très générale mais donne déjà de très bonnes performances sur une base de données standard bien que de nombreuses améliorations soient prévues sur la stratégie d'apprentissage des modèles, les politiques de fusion et d'élagage de l'algorithme ou l'extraction des paramètres dans l'image.

C'est cependant sur des tâches plus complexes que nous attendons les progrès les plus significatifs comme la reconnaissance de mots cursifs ou de documents entiers où le ca-

ractère entièrement multidimensionnel de l'algorithme sera le mieux exploité.

Références

[1] Nafiz Arica and Fatos T. Yarman-Vural. An overview of character recognition focused on off-line handwriting. *IEEE Transactions on Systems, Man and Cybernetics - part C : Applications and Reviews*, 31(2) :216–233, May 2001.

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

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

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

[5] Julian Besag. Statistical analysis of dirty pictures. *Journal of the Royal Statitical Society*, 1986.

[6] Richard G. Casey and Eric Lecolinet. A survey of methods and strategies in character segmentation. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 18(7) :690–706, July 1996.

[7] C. Choisy and A. Belaïd. 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, September 2000.

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

[9] Richard O. Duda, Petre E. Hart, and David G. Stork. *Pattern Classification*. Wiley-Interscience, 2001.

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

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

[12] Stuart Geman and Donald Geman. Stochastic relaxation, gibbs distributions and the Bayesian restoration of images. *IEEE transactions on Pattern Analysis and Machine Intelligence*, 6(6), 1984.

[13] Michel Gilloux. Hidden Markov models in handwriting recognition. In Sebastiano Impedovo, editor, *Fundamentals in handwriting recognition*, NATO ASI Series, pages 264–288. Springer-Verlag, January 1994.

[14] F. Grandidier, R. Sabourin, C.Y. Suen, and M. Gilloux. 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*, juillet 2000.

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

[16] Xuedong Huang, Alex Acero, and Hsiao-Wuen Hon. *Spoken language processing*. Prentice Hall, 2001.

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

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

[19] Jouko Lampinen, Toni Tamminen, Timo Kostiainen, and Ilkka Kalliomäki. 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, 2001.

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

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

[22] Magdi Mohamed and Paul Gader. 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), May 1996.

[23] Herman Ney and Stephan Ortmanns. Progress in dynamic programming search for LVCSR. *Proceedings of the IEEE*, 88(8) :1224–1240, August 2000.

[24] Hee-Seon Park and Seong-Whan Lee. A truly 2-D hidden Markov model for off-line handwritten character recognition. *Pattern Recognition*, 31(12) :1949–1864, 1998.

[25] Georges M. Quénot. The "orthogonal algorithm" for optical flow detection using dynamic programming. In *IEEE International Conference on Accoustics, Speech and Signal Processing*, march 1992.

[26] Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of speech recognition*. Prentice Hall PTR, 1993.

[27] Jean-Claude Simon and Olivier Baret. Regularities and singularities in line images. In *Pre-proceedings SSPR90*, pages 423–439, 1990.

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

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

[30] Qing Wang, Zheru Chi, David D. Feng, and Rongchun Zhao. Hidden Markov random field based approach for off-line handwritten Chinese character recognition. In *Proceedings of the International Conference on Pattern Recognition*, 2000.

[31] Hiromitsu Yamada and Kazuhiko Yamamoto. Recognition of echocardiograms by a dynamic programming method. *Pattern Recognition*, 24(2), 1991.