SEPTIEME COLLOQUE SUR LE TRAITEMENT DU
SIGNAL ET SES APPLICATIONS

NICE du 28 MAI au 2 JUIN 1979

FILTRAGE OPTIMAL POUR LA RECONNAISSANCE DE CARACTERES

Z. EL-SAYED, J. LOPEZ KRAHE, J. FLEURET

Ecole Nationale Supérieure des Télécommunications - Dpt ISST, Laboratoire Image
46 , Rue Barrault , PARIS 13è

RESUME

Le Filtre F.O.U.T. (Filtre Optimisé Unique de Transcodage) présente plusieurs avantages, pour la reconnaissance des formes : à l'aide d'une seule expérience de filtrage, on obtient un mot de code binaire pour chaque forme. Les mots de code sont choisis a-priori.

Nous présentons ici une Optimisation complémentaire du processus, conduisant à la détermination de mots de code optimaux. On considère le problème de la reconnaissance de caractères dans le cas de caractères réels bruités, et provenant de polices différentes.

Nous présentons tout d'abord l'étude de l'influence d'un bruit additif et indépendent du signal. Le choix de la meilleure affectation des mots de code conduit à un problème mathématique d'optimisation (Permutation Optimal Problem - P.O.P). Pour ce problème, la solution exhaustive étant généralement impossible à atteindre, nous avons développé un algorithme sous-optimal. Et les mots de code sous-optimaux conduisent à une amélioration très significative du taux de reconnaissance.

Dans le cas de caractères provenant de polices différentes, il faut optimiser le filtre et les mots de code. Le filtre du représentant moyen, calculé à partir de la police moyenne, constitue une réponse à la première question. Le choix de l'affectation des mots de code conduit à un problème matriciel d'optimisation, analogue au problème P.O.P.

Les résultats expérimentaux sont donnés dans le cas d'un alphabet de plusieurs polices de caractères réels.

SUMMARY

The filter F.O.U.T. (Filtre Optimisé Unique de transcodage) presents several advantages for pattern recognition : a binary code-word is obtained for each pattern, with a single filtering experiment. The binary code-words are chosen a-priori.

We present here a further optimisation process, leading to the determination of optimum code-words. The problem of character recognition is considered in the case of real noisy characters, belonging to distinct fonts.

The study of the influence of an additive signal-independent noise is first presented. The choice of the best code-word attribution is examined as a general optimisation mathematical problem (Permutation Optimal Problem - P.O.P.). For this problem, an exhaustive solution is impossible in general. Thus, we develop a sub-optimal algorithm. A significant recognition-rate improvement is obtained for the suboptimal code-words determined in this manner.

In the case of characters coming from distinct fonts, the filter and the code-words need to be optimized. An answer to the first question is given by the so-called "mean" representing filter, whose implementation depends on the mean font. As for the choice of code-word attribution, we get a matrix optimization problem, which is analogous to the problem P.O.P..

Experimental results are given in the case of real characters belonging to distinct fonts.

FILTRAGE OPTIMAL POUR LA RECONNAISSANCE DE CARACTERES

I - INTRODUCTION

Cette étude constitue un approfondissement de la méthode F.O.U.T. (Filtre Optimisé Unique de Transcodage) présentée en référence (1). Nous en rappelons donc tout d'abord le principe et les principales caractéristiques.

Il s'agit d'une méthode de reconnaissance des formes inspirée de techniques de corrélation optique. On cherche à identifier une forme

$$U_n(x,y) \qquad n = 1, 2, \ldots\ldots, N$$

parmi un alphabet de N formes (Pour cette étude, les U_n représenteront des images de caractères dactylographiés ; mais, en général, le filtre F.O.U.T. est applicable à toute classe d'images à niveaux de gris).

La réponse percutionnelle du filtre p(x,y) s'exprime de la manière suivante (cf. réf. 1) :

$$p(x,y) = \sum_{n=1}^{N} \sum_{k=1}^{K} a(n,k) \; U_n^{*} (-x + kx_o , -y) \quad (1)$$

avec $K = \log_2 N$

où les coefficients a(n,k) sont solution du système d'équations :

$$\sum_{n=1}^{N} a(n,k) \; c_{n_o,n}(o,o) = b(n_o,k) \quad (2)$$

$$k = 1,2,\ldots, K$$

$$n_o,\ n = 1,2\ldots., N$$

avec $c_{n_o,n}(o,o)$ = valeurs des intercorrélations entre $U_{n_o}(x,y)$ et $U_n(x,y)$ à x = y = o.

b(no,k) = k ième bit du mot de code affecté au caractère U_{n_o}.

En résumé, pour un alphabet donné, le calcul du filtre comprend les étapes suivantes :

- calcul de la matrice des intercorrélations $C_{n_o,n}(o,o)$
- Choix des mots de code b(no,k), k = 1,2... K affectés à chaque caractère U_{n_o}.
- Inversion du système (2)

D'après (1) et (2) la réponse du filtre à un caractère U_{n_o}, sera égale au mot de code binaire qui lui a été affecté :

$$[p(x,y) \; * \; U_{n_o}(x,y)] = b(n_o, k) \quad (3)$$

$$x = kx_o$$

$$y = o$$

(* ≡ convolution)

De par sa construction, le filtre est donc unique. Il réalise d'après (3) un transcodage de chaque caractère, la réponse étant un mot de code préalablement choisi. Enfin, le filtre est optimisé, car il délivre —théoriquement— des réponses purement binaires.

II - LE PROBLEME DU FILTRAGE DE CARACTERES REELS :

Dans la réalité, les opérations de filtrage ne se déroulent pas au sein d'un alphabet fixé de N formes. Deux problèmes réels nous en empêchent. D'une part, les caractères sont bruités, à cause de dégradations diverses (défauts d'impression, de numérisation...etc). D'autre part, il est souhaitable de pouvoir reconnaître des caractères appartenant à plusieurs polices différentes.

Par conséquent, notre alphabet fixé de N formes va maintenant être représenté par M échantillons pour chaque "classe" de caractère $\{U_n\}$, à savoir :

$$U_n^{(m)}(x,y) \qquad m = 1,2,\ldots\ldots, M$$

Nous utiliserons ce formalisme dans la suite, où $U_n^{(m)}$ pourra représenter soit une version bruitée de $\{U_n\}$ soit : le n ième caractère de la police numéro (m).

Dans les deux cas, le filtre doit être calculé à partir d'un alphabet suffisamment représentatif des classes à reconnaître. Ce problème sera examiné dans le paragraphe suivant. D'autre part, lorsque le filtre ainsi calculé est appliqué à des caractères réels, les réponses ne sont évidemment plus binaires. A ce niveau on peut optimiser davantage le filtrage au moyen d'un choix adéquat de l'affectation des mots de code. En effet, le choix de l'affectation des mots de code à chaque classe de caractères est très vaste (il y a N! permutationspossibles). Cette grande latitude peut être utilisée pour l'amélioration globale des réponses du filtre à chaque échantillon $U_n^{(m)}$ des classes $\{U_n\}$. Ce problème sera examiné au paragraphe IV.

III - FILTRE DU REPRESENTANT MOYEN :

Par "représentant moyen", nous désignons dans la suite l'alphabet obtenu par moyenne des échantillons de chaque classe :

$$\hat{U}_n(x,y) = \frac{1}{M} \sum_{m=1}^{M} U_n^{(m)}(x,y) \quad (4)$$

Appelons $b^{(m)}(n,k)$ les réponses du filtre à $U_n^{(m)}$. Ces valeurs ne sont pas binaires.

Nous allons montrer plus bas que le choix du représentant moyen (4) permet de calculer un filtre optimal, au sens où l'expression suivante :

$$\sigma^2(n,k) = \frac{1}{M} \sum_{m=1}^{M} | \, b^{(m)}(n,k) - b(n,k) \, |^2 \quad (5)$$

est minimisée.

Les coefficients $\hat{a}(n,k)$ de ce filtre vérifieront, d'après (2) :

$$\sum_{n=1}^{N} \hat{a}(n,k) \quad \hat{c}_{n_o,n}(o,o) = b(n_o, k) \quad k=1...K \quad (6)$$

où $\hat{c}$ désigne les valeurs de corrélation pour le représentant moyen.

Il en résulte la réponse percutionnelle suivante :

$$\hat{p}(x,y) = \sum_{n=1}^{N} \sum_{k=1}^{K} \hat{a}(n,k)\ \hat{U}_n^{**}(-x + kx_o, -y) \quad (7)$$

qui vérifie évidemment :

$$[\hat{p}(x,y) ** \hat{U}_n(x,y)]_{\substack{x = k\,xo \\ y = o}} = b(n,k) \quad (8)$$

D'autre part, la réponse du filtre à $U_n^{(m)}$ est $b_n^{(m)}$:

$$[\hat{p}(x,y) ** U_n^{(m)}(x,y)] = b^{(m)}_{(n,k)} \quad (9)$$
$$x = kxo \quad ; \quad y = o$$

Il est évident, d'après (9), (4), (8) que :

$$\frac{1}{M} \sum_{m=1}^{M} b^{(m)}(n,k) = b(n,k) \quad (10)$$

Le code binaire utilisé est donc le barycentre des réponses du filtre à chaque échantillon $U_n^{(m)}$. Le filtre du représentant moyen optimise donc bien l'expression (5). Ce résultat est indépendant du code employé.

Dans la suite, le filtre utilisé sera donc le filtre du représentant moyen. Ce filtre présente, en effet, l'avantage de réduire les distorsions entre échantillons d'un même caractère. Son application expérimentale au cas de caractères bruités a été faite en réf. (2).

IV - CHOIX DE LA MEILLEURE AFFECTATION DES MOTS DE CODE :

Afin de calculer le filtre, nous avons fixé le choix de l'alphabet représentant. Il nous faut maintenant choisir les mots de code. Nous allons étudier ce choix dans le cas de caractères bruités et dans le cas de caractères de polices différentes. Certains résultats ont déjà été obtenus dans le cadre d'une hypothèse statistique particuliere : à savoir, l'hypothèse d'un bruit additif indépendant du signal (réf. 3). Nous rappelons tout d'abord ces résultats. Puis, nous généraliserons l'étude au cas de M échantillons $U_n^{(m)}$ pour lesquels on ne supposera aucune hypothèse particulière.

IV.1 - CAS DE CARACTERES ENTACHES D'UN BRUIT ADDITIF INDEPENDANT DU SIGNAL

Dans ce cas, nous avons montré par une étude statistique au second ordre, que le choix optimal des mots de code passe par la résolution du Problème P.O.P. (Permutation Optimal Problem) :

> Trouver, parmi les N! permutations possibles, la permutation minimisant l'expression :
>
> $$\gamma = \left|\sum_{n=1}^{N} \alpha_n B_n\right|^2 \quad (11)$$
>
> Les coefficients α_n (réels $\geq$ o) dépendent des caractères. Les coefficients B_n représentent les poids des mots de code :
>
> $$B_n = \sum_{k=1}^{K} b(n,k)$$
>
> γ caractérise la moyenne des écarts-types des réponses $b^{(m)}(n,k)$.

Pour ce problème, la solution exhaustive est, en général impossible à atteindre, à cause des dimensions considérables , par exemple $32! = 2{,}63\ 10^{35}$. Nous avons développé un algorithme sous-optimal (réf. 4 et 5) dont nous allons décrire les grandes lignes.

Le principe consiste à chercher la valeur de :

$$g = \sum_{n=1}^{N} \alpha_n B_n$$

la plus proche de O. Les α_n et les B_n étant ordonnés, il est très facile d'obenir les valeurs extrêmes de g. Puis on engendre l'ensemble $\mathcal{G}$ obtenu par permutations des couples $n_1 \longleftrightarrow n_2$ disjoints. La différence entre les 2 valeurs de $\mathcal{G}_2$ obtenues s'exprime alors de façon très simple, en fonction des indices :

$$g_{n_1 n_2} - g_{n_1 \leftrightarrow n_2} = d_{n_1,n_2} = (\alpha_{n_1} - \alpha_{n_2})(B_{n_2} - B_{n_1}) \quad (12)$$

Et l'utilisation de ce tableau permet la mise en oeuvre aisée d'un développement ordonné de $\mathcal{G}$

On cherche alors la valeur la plus proche de o ; ce qui limite l'arborescence à une partie $\mathcal{H}$ de $\mathcal{G}$, pour laquelle les valeurs trouvées sont de même signe ($>$ o ou $<$ o).

Par rapport à la méthode exhaustive, on observe ainsi un gain de temps très appréciable.

Enfin, on trouve une amélioration considérable du taux de reconnaissance pour le bon choix des mots de code. Par exemple, pour N = 16, la meilleure solution sous-optimale trouvée correspondant à

$$\gamma_{max}/\gamma_{min} = 7 \cdot 10^{13} .$$

IV.2 - CAS GENERAL - CARACTERES PROVENANT DE POLICES DIFFERENTES

Nous ne supposons maintenant aucune hypothèse particulière sur les $U_n^{(m)}$. Ces échantillons pourront donc représenter soit : des caractères bruités de façon quelconque, soit des caractères provenant de polices différentes.

Nous nous proposons de minimiser la moyenne des écarts quadratiques de toutes les réponses. Soit :

$$\gamma = \sum_{n=1}^{N} \sum_{k=1}^{K} \sigma^2(n,k)$$

$$= \frac{1}{M} \sum_{m=1}^{M} \sum_{n=1}^{N} \sum_{k=1}^{K} |b^{(m)}(n,k) - b(n,k)|^2 \qquad (13)$$

Notre but est d'aboutir à une formulation anologue à la relation (11) .

Développons (13) :

$$\gamma = \frac{1}{M} \sum_{m,n,k} [(b^{(m)})^2 - 2b^{(m)}.b + b^2]$$

Utilisons la relation (10), et le fait que les mots de code sont binaires ($b^2 = b$), on obtient :

$$\gamma = -\gamma_o + \frac{1}{M} \sum_{m,n,k} (b^{(m)}(n,k))^2 \qquad (14)$$

avec

$$\gamma_o = \sum_{n=1}^{N} B_n \qquad (15)$$

On voit, d'après (14), que γ dépend implicitement des mots de code b(n,k), par l'intermédiaire des $b^{(m)}(n,k)$ Nous allons exprimer les $b^{(m)}(n,k)$ en fonctions des b(n,k). Soit Pk(x,y) la réponse percutionnelle non multiplexée du filtre du représentant moyen :

Soit $\hat{p}_k(x,y) = \sum_{n=1}^{N} \hat{a}(n_1,k)\ \hat{U}^{*}_{n_1}(-x + k\,x_o, -y)$ (16)

Les coefficients â sont fonction des mots de code b et de la matrice inverse de $\hat{c}$:

$$\hat{a}(n_1,k) = \sum_{n_o=1}^{N} \hat{c}^{-1}_{n_1,\,n_o}\ b(n_o,\,k) \qquad (17)$$

De par la définition des $b^{(m)}(n,k)$ et les relations (16) et (17), il résulte :

$$b^{(m)}(n,k) = \sum_{n_o} \mu^{(m)}(n,n_o)\ b(n_o,\,k) \qquad (18)$$

où $\mu^{(m)}(n,n_o) = \frac{1}{M} \sum_{n_1} \sum_{m} \hat{c}^{-1}_{n_1,n_o} [U^{(m)*}_{n_1}(-x+kx_o,-y)^{*}\ U^{(m)}_{n}(x,y)]$

$$x = k = o$$
$$y = o \qquad (19)$$

Reportons (18) dans (14) ; γ s'exprime finalement de la manière suivante :

$$\gamma = -\gamma_o + \sum_{n_1,n_2} \alpha(n_1,n_2)\ \beta(n_1,\,n_2) \qquad (20)$$

avec

$$\alpha(n_1,n_2) = \frac{1}{M} \sum_{m,n_o} \mu^{(m)}(n_o,\,n_1)\mu^{(m)}(n_o,n_2) \qquad (21)$$

$$\beta(n_1,\,n_2) = \sum_{k} b(n_1,k)\ b(n_2,\,k) \qquad (22)$$

La relation (20) est l'équivalent recherché, de la relation (11) où les α ne dépendent que descaractères, et les β des mots de code. Toutefois, il s'agit maintenant d'une relation matricielle, dont la manipulation n'est pas aussi simple que celle de (11).

En effet, d'une part, ces matrices ne sont pas quelconques, ce qui impose certaines contraintes. Par exemple, la matrice β est symétrique, ses éléments diagonaux sont les B_n...etc. D'autre part, étant donné la définition de γ, les valeurs traitées sont toujours positives. Nous ne disposons donc pas des valeurs extrêmales. En outre la simple permutation d'un couple de mots de code ($n_1 \leftrightarrow n_2$) nécessite l'échange simultané de 2 colonnes et de 2 lignes. On peut montrer aisément dans ce cas :

$$d_{n_1,\,n_2} = \gamma_{n_1,\,n_2} - \gamma_{n_1 \leftrightarrow n_2}$$

$$= 2 \sum_{i\ n_1,n_2} (\alpha_{i,n_1} - \alpha_{i,n_2})(\beta_{i,n_2} - \beta_{i,n_1})$$

$$+ (\alpha_{n_1,n_1} - \alpha_{n_2,n_2})(\beta_{n_2,n_2} - \beta_{n_1,n_1}) \qquad (23)$$

Contrairement à la relation (12), le tableau d_{n_1,n_2} doit être recalculé systématiquement à chaque permutation.

Enfin, les valeurs demeurant positives, il est nécessaire d'explorer la totalité de l'ensemble. On ne dispose plus, comme en IV.1, du passage à o.

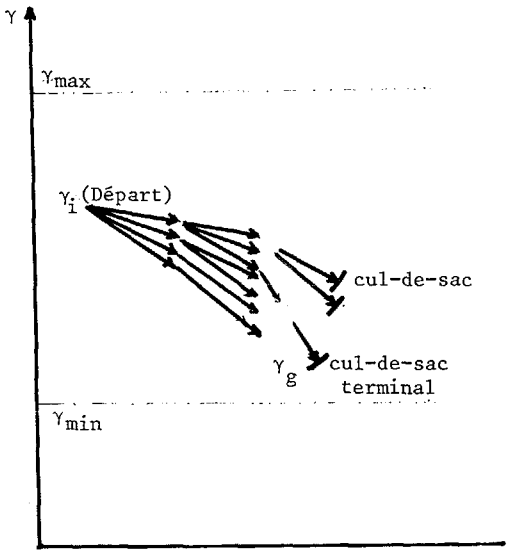
L'ensemble du traitement est donc beaucoup plus lourd que pour IV.1.

V - RESULTATS

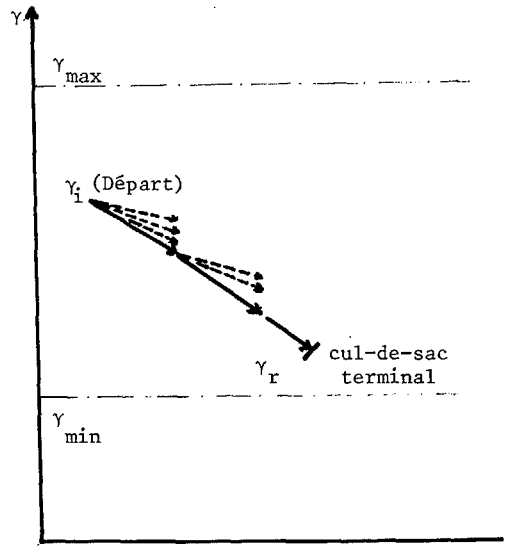
Afin d'obtenir une minimisation sous-optimale de γ, nous avons utilisé le tableau (23), recalculé à chaque itération. On part d'une valeur initiale γ_i (correspondant, par exemple, au code naturel (001, 010, 011, 100,...). On ne conserve, évidemment, que les valeurs négatives de d.

Dans une première approche, nous avons exploré systématiquement toutes les valeurs négatives de d (développement "global", cf fig.) jusqu'aux culs-de-sac. On peut, de cette manière, explorer tout l'espace correspondant, en partant d'une valeur initiale assez faible.

Dans une approche plus rapide (cf. fig.) on ne conserve que les valeurs minimales de d.



Développement global



Développement rapide

Nous avons utilisé des caractères réels numérisés provenant des polices suivantes :

P1 = IBM Elite 72
P2 = IBM Polygo Elite
P3 = IBM Courrier 12
P4 = IBM Prestige Elite 72

Les mots de code ont été optimisés et le filtre du représentant moyen a été calculé comme indiqué plus haut dans les 2 cas suivants :

- Polices P1, P2 avec 7 caractères
- Polices P1, P2, P3 et P4 avec 15 caractères.

Dans le premier cas (7! = 5040) on peut faire un développement exhaustif. Nous avons trouvé :

$$\gamma_{max}=1.2212, \qquad \gamma_{min}=0.2134 \ , \ \gamma_{max}/\gamma_{min}=5.722$$

On voit que l'amélioration du taux de reconnaissance n'est pas aussi spectaculaire que celle obtenue en IV.1.

Les résultats obtenus au moyen des développements sous-optimaux sont les suivants :

	M=3 N=7 N!=5040 K = 3	M=4 N=15 N!=1.3 10^{12} K = 4
γ_i	0.4665	8.7081
γ_g	0.2134	2.0333
Temps du calcul pour γ_g	0.36 mn	19 mn
γ_r	0.245	2.162
Temps du calcul pour γ_r	0.23 mn	0.25 mn
γ_i/γ_g	2.186	4.2827

Nous avons également calculé les réponses $b^{(m)}(n_l,k)$.

On obtient des nuages autour des valeurs nominales 0 et 1. Les écarts-types de ces nuages augmentent avec N et M (pour un code donné). Pour les valeurs de γ élevées, on observe des recouvrements entre les nuages autour du 0 et du 1. Nous avons vérifiés d'autre part que ce recouvrement disparait pour les choix adéquats des mots de code.

Il nous a été ainsi possible de détecter sans ambiguité jusqu'à15 caractères de 4 polices.

REFERENCES

1 - J. FLEURET, H. MAITRE
Optics Comm. V. 17, n°1, 1976

2 - J. LOPEZ KRAHE
Congrès AFCET-IRIA, Paris 1978, pp. 592 -600

3 - J. FLEURET, J. LOPEZ KRAHE
I.C.O. 11 conférence, Madrid 1978, pp. 343 - 346

4 - J. LOPEZ KRAHE
Etude de méthodes de reconnaissance automatiques de caractères imprimés. Application à un système de lecture pour aveugles.
Thèse de doctorat d'Université. 04.04.79
Paris VIII - ENST - E - 79 003.

5 - J. LOPEZ KRAHE, Z. EL SAYED, J. FLEURET
Theoretical Optimization of a character encoding filter (soumis à "Signal Processing").