Classification géométrique par polytopes de contraintes. Performances et intégration

Geometric Classification by Stress Polytopes. Performances and Integration.

par J. MITÉRAN, P. GORRIA, M. ROBERT*

Université de Bourgogne, Laboratoire GERE
12, rue de la Fonderie – F-71200 LE CREUSOT
** Université de Montpellier, Laboratoire LIRMM*
161, rue Ada – F-34392 Montpellier

Résumé

Nous présentons dans cet article un opérateur de classification rapide adapté au traitement d'images, ses performances en classification, ainsi que son intégration dans un circuit ASIC. Pour effectuer une segmentation ou un classement d'images en vue de la détection de défauts, il est souvent impossible de trouver un nombre réduit de paramètres caractéristiques pertinents qui permettent de discriminer les classes. Nous proposons une méthode de classification géométrique par apprentissage de polytopes de contraintes, qui autorise l'utilisation d'un grand nombre de paramètres et assure une vitesse de décision élevée. L'opérateur de décision associé à cette classification a été intégré sous forme de circuit précaractérisé dont la simplicité de mise en œuvre, la rapidité et la robustesse en classification sont des qualités qui lui permettent de rivaliser avec les opérateurs neuronaux.

Mots clés : Segmentation, Image, Temps réel, ASIC, Classification, Apprentissage, Polytope, Hyperrectangle, performances.

Abstract

We present in this paper a fast classification operator suitable for image processing, the performances of this operator as well as its implementation in the form of an ASIC. In image segmentation and classification in view of defect detection, it is often impossible to find a reduced set of pertinent characteristic parameters which allows to distinguish the classes. We propose herein a geometric classification method by stress polytop training which allows the use of a great number of parameters and ensures a high decision speed. The decision operator associated with the classification has been implemented in Standard Cell and Full Custom. Its ease of use, rapidity, and robustness in classification are the major qualities which enable it to compete with neural operators.

Key words : *Segmentation, Image, Real Time, ASIC, Classification, Training, Polytope, Hyperrectangle, performances.*

1. Introduction

La segmentation automatique d'images est réalisable en temps réel, grâce notamment à une application particulière de la classification par apprentissage, qui consiste à attribuer une étiquette ou classe à un objet défini par un ensemble de paramètres. De nombreux travaux ont été menés, pour réaliser des procédures de classification non paramétrique, utilisant au mieux les critères de proximité [9], [11], [1], [10]. Suivant le modèle de découpage de l'espace des attributs choisi, ces travaux conduisent à une réduction du nombre d'exemples d'apprentissage nécessaires, à une accélération du processus d'apprentissage, ou à l'optimisation des performances du classifieur. Un algorithme de regroupement sous forme d'hyperrectangles [9] a été proposé et amélioré en vue d'accélérer la phase d'apprentissage au détriment des performances.

Une approche connexionniste de la classification supervisée permet également de segmenter une image en temps réel, par classification des pixels qui la constituent. Le réseau de neurones est alors alimenté par des paramètres de texture [3], ou tout autre paramètre caractéristique calculé dans des fenêtres locales de petites dimensions. Les neurones de sortie indiquent la classe à attribuer au pixel traité [2], [16]. Malheureusement, outre le fait que la conception même de l'architecture neuronale peut être une opération complexe [15], la vitesse de convergence de l'algorithme d'apprentissage est fortement dépendante du choix des paramètres utilisés et de leurs distributions. Ainsi le temps d'apprentissage n'est pas connu a priori. De plus l'intégration d'un grand nombre

de neurones sous forme d'opérateur câblé est encore limitée par le fait que chaque neurone nécessite la réalisation de fonctions coûteuses en surface de silicium, ou encore un très grand nombre de composants dans le cas de [4] où deux FPGAs sont utilisés pour chaque unité cachée.

Le but recherché ici est d'obtenir un classifieur robuste en classification, c'est–à–dire peu influencé par la présence de paramètres peu discriminants, tout en étant intégrable sous forme de composant unique capable de classer les pixels à la cadence vidéo.

Nous présentons pour cela une méthode de classification géométrique par hyperrectangles de contraintes (polytopes particuliers, dont les faces sont orthogonales entre elles et parallèles aux axes de l'espace des paramètres) et l'opérateur décisionnel correspondant. Ces travaux ont conduit à l'intégration de l'opérateur décisionnel sous forme de circuit précaractérisé (ASIC) assurant une fréquence de décision élevée (10 Mhz). Ce classifieur peut donc être utilisé pour des traitements de bas niveau de segmentation en temps réel, où la durée de décision ne doit pas excéder 125 ns par pixel.

Les algorithmes de classification et de décision sont décrits au paragraphe 2. Au paragraphe 3 sont présentées les performances théoriques et pratiques de l'opérateur. L'architecture du circuit décisionnel est décrite au paragraphe 4, accompagnée d'un exemple d'application en traitement de bas niveau.

2. Description des polytopes de contraintes

Cette méthode géométrique permet de classer un élément grâce à la connaissance d'un certain nombre de paramètres ou attributs qui doivent être caractéristiques de la classe de l'élément considéré, et grâce à une phase d'apprentissage du système. Pendant cette phase, le système partitionne l'espace des attributs, de manière à le séparer en deux régions, chacune caractéristique d'une classe. La phase de décision consiste ensuite à mesurer les attributs d'un nouvel élément, et à vérifier à quelle région de l'espace, donc à quelle classe il appartient.

2.1. PHASE D'APPRENTISSAGE

2.1.1. *Saisie d'un échantillonnage; détermination des contraintes locales*

Cette étape de l'apprentissage consiste à collecter un ensemble $E_p = e_1, e_2, \ldots, e_p$ de p échantillons les plus représentatifs possibles des diverses classes en présence et d'associer à chaque échantillon une contrainte locale. Chaque échantillon e_i est constitué d'un vecteur attribut $(x_1, x_2, \ldots, x_N)$ dans un espace de dimension N et de sa classe $C(e_i)$ correspondante. En pratique, dans le cas de la segmentation d'images, le nombre d'échantillons disponibles est égal au nombre de pixels des images de référence (plusieurs milliers), et la dimension de l'espace des attributs peut être très élevée (5 à 20 paramètres).

À chaque échantillon est associée une région de l'espace, telle que tout point de cette région soit considéré comme étant de la classe de l'échantillon. Cette contrainte locale garantit le bon reclassement de l'échantillon.

Nous déterminons pour cela un hyperrectangle $H(e_i)$, qui constitue une contrainte de proximité envers tous les éléments de E_p qui ne sont pas de classe $C(e_i)$. Les frontières de cette contrainte locale sont obtenues en recherchant dans toutes les directions (à « gauche » et à « droite » du point, pour chaque paramètre, soit $2N$ directions), les plus proches voisins de classe différente de $C(e_i)$. Afin d'obtenir une contrainte définie par un hyperrectangle, la distance utilisée pour la recherche des plus proches voisins est la distance du « max » :

$$d_\infty(x, y) = max_{k=1,\ldots,N}|x_k - y_k| \qquad [5] \qquad (1)$$

En effet, la boule, c'est–à–dire l'ensemble des points équidistants d'un point central donné, est un hyperrectangle, si la distance choisie est celle du « max ».

Soit dp la distance de l'échantillon au plus proche voisin défini précédemment, la frontière est alors placée à une distance $df = dp \cdot R$ de l'échantillon. La figure 1 représente un échantillon e_i et l'hyperrectangle $H(e_i)$ associé pour un rapport $R = 0,4$.

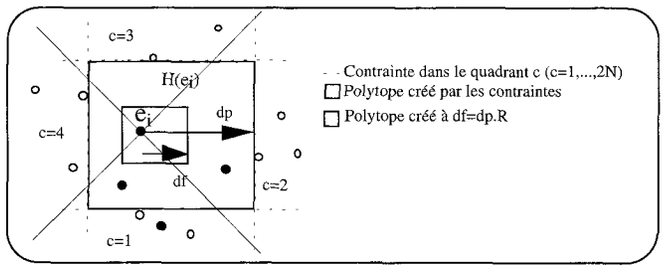


Figure 1. – Hyperrectangle $H(e_i)$ attaché à l'échantillon e_i.

Chaque hyperrectangle est ainsi défini par une borne inférieure et une borne supérieure pour chaque paramètre.

Le rapport R caractérise le degré de similitude que doit vérifier un point, pour être classé dans la classe du point qui engendre cette contrainte. La tolérance de positionnement sur un point sera d'autant plus faible que le rapport R sera petit. En effet, si R tend vers zéro, alors la contrainte locale est réduite au seul point qui l'engendre, par contre si R tend vers 1, la fluctuation est maximale, à l'exclusion des points de classes différentes. Un objet ne devant en aucun cas être classé dans deux classes distinctes à la fois, le rapport R doit rester inférieur ou égal à $0,5$.

Une représentation de l'espace des attributs pour un échantillonnage de vecteurs à deux paramètres, et pour un exemple à deux classes (les points de classe $C0$ sont en blanc et $C1$ en noir) est donnée figure 2. La figure 3 représente ces mêmes échantillons avec leurs contraintes locales associées.

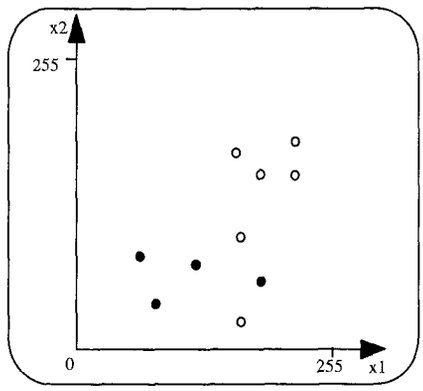


Figure 2. – Echantillonnage

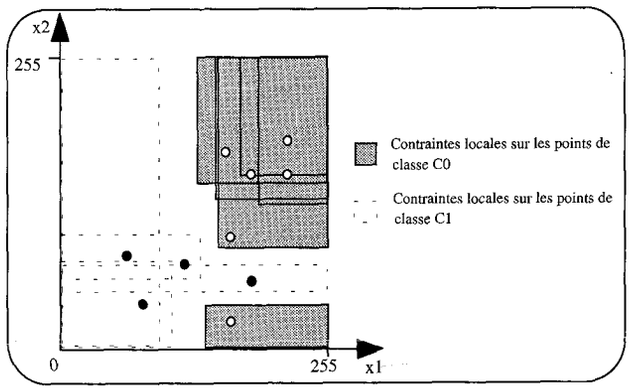


Figure 3. – Contraintes locales, R = 0,3.

L'algorithme d'obtention des contraintes locales est le suivant :

1. Choisir un rapport R ef faire 2.
2. Pour tous les échantillons e_i faire 3.
3. Pour les directions $d = 1, \ldots, 2N$ (donc pour tous les paramètres), faire 4 et 5.
4. Rechercher le plus proche voisin e_j (au sens de la distance du max) de e_i, de classe $C(e_j)$ différente de $C(e_i)$ dans la direction d.
5. Mémoriser la position de la frontière définie par $df = dp \cdot R$, où dp est la distance du « max » entre les points e_i et e_j.

La complexité de l'algorithme est alors en $O(p^2N)$ si p est le nombre d'éléments et N le nombre de paramètres. Cette complexité ne pose pas de problème dans la plupart des cas pratiques rencontrés de segmentation, où l'apprentissage n'est réalisé qu'une fois. Les temps obtenus sur un calculateur de type 486DX50 sont par exemple de l'ordre de 25 secondes pour $p = 1000$ et $N = 8$, ou encore 2 minutes pour $p = 1000$ et $N = 50$. Il faut ici encore une fois mettre l'accent sur le fait que si le temps de décision doit être le plus court possible pour réaliser la segmentation d'images en temps réel, le temps d'apprentissage peut être plus important, limité par le seul besoin de confort lors du développement d'une application.

2.1.2. *Fusion des contraintes locales*

Une fois obtenu l'ensemble des contraintes locales de chaque vecteur attribut, il est nécessaire de fusionner ces contraintes dans des hyperrectangles plus larges, en préservant la non–superposition de ces nouvelles contraintes avec celles des classes différentes [9].

Soit H_1 et H_2 deux hyperrectangles de dimension N définis par les intervalles $I_{ik} = [a_{ik}, b_{ik}]$, où i désigne l'indice de l'hyperrectangle H_i.

Si $\times$ désigne le produit cartésien entre deux ensembles I_{ik}, alors :

$$H_1 = I_{11} \times I_{12} \times \ldots \times I_{1N} \tag{2}$$

et

$$H_2 = I_{21} \times I_{22} \times \ldots \times I_{2N} \tag{3}$$

La fusion de H_1 et H_2 est l'hyperrectangle H_3 défini de la façon suivante :

$$H_3 = I_{31} \times I_{32} \times \ldots \times I_{3N} \tag{4}$$

avec

$$I_{3k} = [min(a_{1k}, a_{2k}), max(b_{1k}, b_{2k})] \tag{5}$$

H_3 est donc l'hyperrectangle le plus petit qui contient à la fois H_1 et H_2. La fusion s'opère en associant deux à deux les contraintes d'une même classe. Si la fusion conduit à un hyperrectangle plus large ne chevauchant pas les contraintes d'une classe différente, alors cette fusion remplacera ces deux contraintes. L'algorithme de fusion est donc le suivant :

1. Pour tous les couples (H_1, H_2) de contraintes locales d'une même classe C, faire 2.
2. Calculer l'hyperrectangle H_3 défini par les limites de H_1 et H_2.
3. Si l'intersection de H_3 avec l'ensemble des hyperrectangles de classe différente de C est vide alors faire 4, sinon faire 5.
4. Remplacer H_1 et H_2 par H_3 et reprendre en 1.
5. Choisir un autre couple (H_1, H_2) et reprendre en 2.

La complexité peut être évaluée dans le cas le plus coûteux en calcul, c'est–à–dire lorsque aucun des polytopes ne fusionne. Le nombre de tests maximum est alors proportionnel à ph^3, où h est le nombre d'hyperrectangles avant fusion.

Le taux de fusion (le nombre de contraintes avant fusion, divisé par le nombre de contraintes après fusion) sera d'autant plus grand que le rapport R est faible. Si, pour des problèmes d'intégration, on souhaite réduire le nombre d'hyperrectangles, il est préférable de choisir un rapport faible, ce qui minimisera le nombre d'hyperrectangles, au détriment de la tolérance sur les attributs.

Voici à titre d'exemple (figure 4), le résultat obtenu par cette méthode sur la distribution à deux classes en dimension deux, présentée au paragraphe précédent.

Seul ce nouvel ensemble de contraintes sera utilisé lors de la phase de décision. Il est constitué d'un nombre nh d'hyperrectangles (polytopes de contraintes).

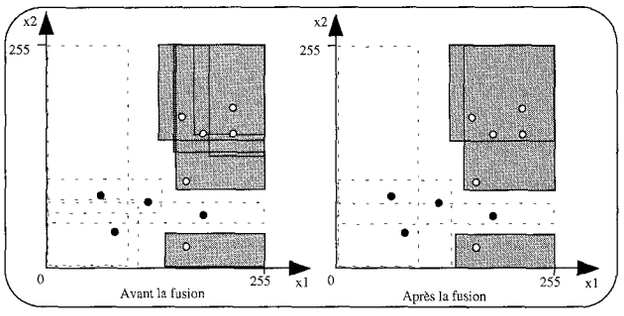


Figure 4. – Fusion des contraintes locales.

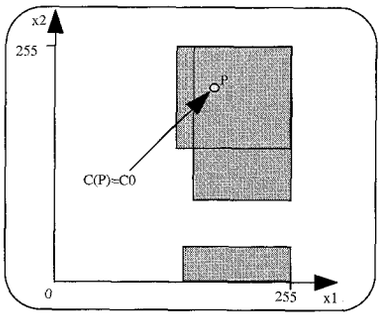


Figure 5. – Exemple de décision en dimension 2.

Il est clair que ce nombre nh peut dépendre de l'ordre des points d'apprentissage, puisque cet ordre intervient sur l'ordre de la fusion des hyperrectangles. Or il peut arriver que l'ordre de la fusion influe sur la possibilité de fusionner (par exemple si H_1 fusionne avec H_2, donnant H_3, qui ne peut fusionner avec H_4, alors que H_2 pouvait fusionner avec H_4). Toutefois, nous avons vérifié expérimentalement, en faisant varier cet ordre pour des distributions gaussiennes, que nh varie très peu (5% au maximum). Le gain possible sur nh est donc faible en regard du temps de calcul qui serait nécessaire si l'on voulait tester tous les ordres de fusion possibles.

2.2. PHASE DE DÉCISION

La phase de décision consiste à attribuer une classe à un vecteur attribut nouveau.

Le découpage de l'espace des attributs a été réalisé uniquement avec des hyperrectangles, qui définissent l'espace caractéristique d'une classe. L'orthogonalité des frontières autorise un traitement simultané des paramètres.

Soit P un nouvel échantillon à classer. Ce nouvel objet P sera classé dans la classe C_p, si et seulement si il appartient à au moins un hyperrectangle H_i de classe C_p (voir figure 5).

$$C(P) = C_p \Leftrightarrow \exists\, H_i / P \in H_i, i = 0, \ldots, nh. \qquad (6)$$

Or, l'appartenance de l'échantillon P à un hyperrectangle H_i est vérifiée si tous ses paramètres x_k vérifient leur appartenance aux intervalles I_{ik} correspondants aux projections de H_i sur chaque axe.

$$P \in H_i \Leftrightarrow \forall k, x_k \in I_{ik}, k = 1, \ldots, N. \qquad (7)$$

Enfin, l'appartenance d'un paramètre x_k à un intervalle I_{ik} donné est vérifiée aisément en contrôlant les deux conditions suivantes : $(x_k > a_{ik})$ et $(x_k < b_{ik})$.

Le contrôle d'appartenance d'un point P à une classe C_p se traduit donc par un ensemble de comparaisons à effectuer simultanément, sur chaque paramètre, pour tous les hyperrectangles de classe C_p.

La fonction logique résultante est :

$$C(P) = C_p \quad \Leftrightarrow \quad \sum_{i=1}^{i=nh} \prod_{k=1}^{k=N} ((x_k > a_{ik}) \cdot (x_k < b_{ik})) \qquad (8)$$

est vrai pour la classe C_p

Si ces opérations sont effectuées en parallèle, le temps de traitement de la phase décisionnelle est égal au temps nécessaire pour effectuer une simple comparaison.

2.3. CONCLUSION SUR LA RECHERCHE DU CLASSIFIEUR

Notre démarche nous a permis de bâtir un algorithme de partitionnement de l'espace des paramètres en hyperrectangles ou polytopes de contraintes, à partir d'algorithmes existants [9], [7] dont les performances en classification ont été étudiées et satisfont à nos besoins. Il est toutefois nécessaire de mieux caractériser les performances propres de notre algorithme, car des différences importantes existent entre les positions des frontières données par l'algorithme BEG de Ichino et celles obtenues par les polytopes de contraintes. Ces différences proviennent notamment de l'utilisation de la distance du max et du rapport R, qui nous permettent de placer les frontières de décision plus ou moins loin des points d'une classe, donc de donner plus ou moins de prépondérance à cette classe. Une autre différence provient également du temps de calcul : l'algorithme BEG est fondé sur une succession de fusions, ce qui oblige à parcourir un grand nombre de fois les échantillons d'apprentissage, alors que nous calculons directement les bornes des hyperrectangles, avant de faire une seule fusion.

Le choix des frontières orthogonales entre elles et parallèles aux axes de l'espace des paramètres permettra d'envisager sereinement un traitement parallèle, donc rapide et adapté à la segmentation d'images en temps réel.

Enfin, il est clair que l'algorithme que nous avons décrit n'est pas unique : l'important est seulement d'arriver à la structure en hyperrectangles, de manière à pouvoir exécuter la décision par le même opérateur.

3. Performances du classifieur

L'erreur dépend d'un certain nombre de facteurs (nombre de points d'apprentissage, corrélation entre les paramètres, nombre et pertinence de ces paramètres...), qu'il convient de faire varier afin de caractériser cette dépendance, pour ensuite utiliser le classifieur de manière optimale, c'est–à–dire dans les conditions où l'erreur est minimum.

L'étude des performances a été menée de façon théorique (grâce à l'établissement de la probabilité d'erreur théorique) et pratique (par la mesure de l'erreur de classement dans des cas réels).

Nous avons considéré un exemple à deux classes, et nous nous sommes placés dans le cas $R = 0,5$. Les résultats obtenus nous ont permis d'évaluer l'influence du nombre de points d'apprentissage, ainsi que celle du choix des paramètres, sur les performances de l'opérateur de classification.

3.1. PROBABILITÉ D'ERREUR

3.1.1. Établissement de la probabilité théorique d'erreur

Étant donné une distribution à deux classes $C0$ et $C1$ dans un espace E, discret et borné (pratiquement, les valeurs utilisées sont entières et comprises entre 0 et 255) à N paramètres, nous cherchons à obtenir la probabilité théorique Pe de mauvais classement d'une mesure lors de la phase de décision.

La probabilité d'erreur Pe est celle de classer un point de type $C0$ dans la classe de type $C1$, ou de classer un point de type $C1$ dans la classe de type $C0$.

Si $n0$ et $n1$ sont les nombres d'échantillons a priori des classes $C0$ et $C1$, ceci se traduit par :

$$Pe = Pm0\frac{n0}{n1+n0} + Pm1\frac{n1}{n1+n0} \qquad (9)$$

Où $Pm0$ est la probabilité de mal classer un élément de classe $C0$, $Pm1$ la probabilité de mal classer un élément de classe $C1$. Il nous faut donc déterminer ces deux probabilités, sachant que $Pm0 = 1 - Pb0$ et $Pm1 = 1 - Pb1$, où $Pb0$ (respectivement $Pb1$) est la probabilité de bien classer un élément de classe $C0$ (respectivement $C1$).

Pour déterminer $Pb0$, nous allons considérer un point A de coordonnées $(a_1, \ldots, a_N)$ dans l'espace E, puis les conditions pour que ce point A de classe $C0$ soit bien classé. Ce point A étant quelconque, la probabilité de bien le classer sera la somme de tous les cas indépendants représentés par tous les A possibles.

Nous pouvons donc exprimer $Pb0$ comme étant la somme de tous les cas indépendants suivants : un échantillon A est bien classé dans $C0$ si, avant sa prise en compte, les hyperrectangles de classe $C0$ ont été construits de manière à ce que ce point soit dans au moins un de ces hyperrectangles, c'est–à–dire si son plus proche voisin au sens de la distance du max est un élément de classe $C0$.

$$Pb0 = \sum_E P0(A) \cdot Pv0 \qquad (10)$$

Où $P0(A)$ représente la probabilité pour qu'un élément de classe $C0$ soit au point A et $Pv0$ la probabilité pour que le plus proche voisin au sens de la distance maximum de l'élément de classe $C0$ arrivant en A soit un élément de classe $C0$, ce qui correspond à un bon classement lors de la décision.

$Pv0$ s'exprime de la manière suivante :

$$Pv0 = \sum_{e=1}^{emax} Pp0(H_{e-1}) \cdot Pp1(H_{e-1}) \cdot Pe0(H_e - H_{e-1}) \qquad (11)$$

Où He est un hypercube centré au point A, et dont tous les côtés sont égaux à $2e$, avec e entier.
En effet, pour que le plus proche voisin d'un élément A de classe $C0$ soit de la classe $C0$, il faut que trois conditions soient vérifiées simultanément :

- aucun élément de classe $C0$ ne doit exister à l'intérieur de l'hypercube d'éloignement $e-1$. Cette condition est exprimée par le premier terme : $Pp0(H_{e-1})$.
- aucun élément de classe $C1$ ne doit exister à l'intérieur de l'hypercube d'éloignement $e-1$: c'est le deuxième terme : $Pp1(H_{e-1})$.
- un élément de type $C0$ doit exister dans la couronne comprise entre les hypercubes d'éloignement $e-1$ et e par rapport à A : c'est le dernier terme : $Pe0(H_e - H_{e-1})$.

La probabilité d'erreur totale est la somme de ces événements indépendants, pour e variant de 1 à $emax$, $emax$ étant atteint lorsque l'hypercube He est borné par la taille maximum de l'espace.

Pour un espace à deux paramètres, nous avons représenté ces conditions sur la figure 6.

Pour respecter l'algorithme de fabrication des hyperrectangles, nous avons donné la priorité aux éléments de classe $C0$, ce qui

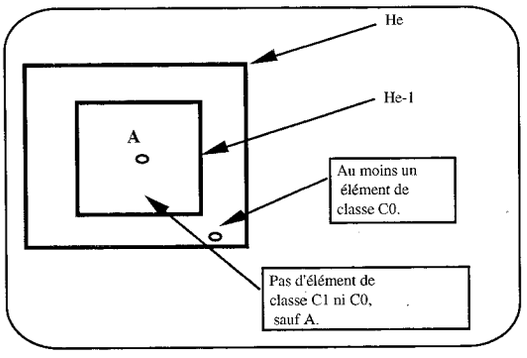


Figure 6. – Conditions pour PvO.

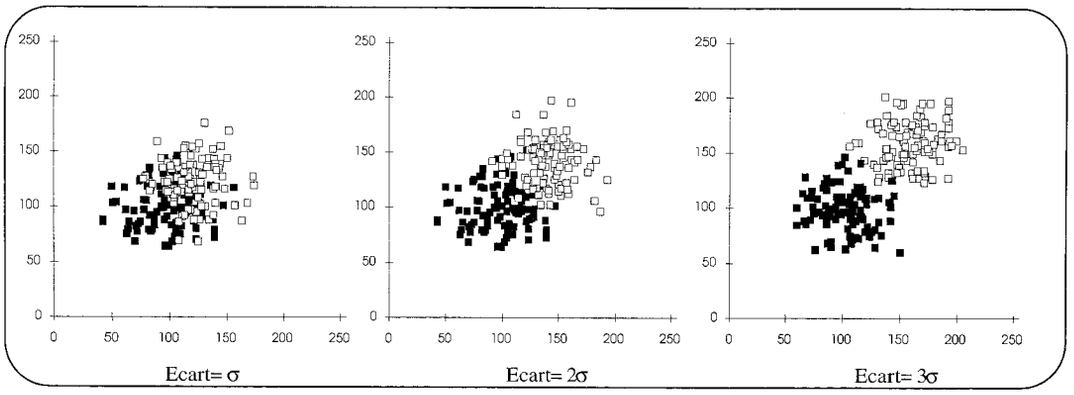


Figure 7. – Exemples de distributions gaussiennes.

revient à dire que l'on accepte la présence d'un élément de classe $C1$ dans la couronne.

$Pv0$ est donc la somme sur tous les cas possibles (tous les éloignements e) de la configuration précédente.

De plus, si nous désignons par $S0_{e-1}$ (resp. $S1_{e-1}$), la probabilité d'avoir un élément de classe $C0$ (resp. $C1$) dans l'hypercube H_{e-1}, dans le cas $n0 = 1$ (resp. $n1 = 1$)., nous avons :

$$Pp0(H_{e-1}) = (1 - S0_{e-1})^{n0} \tag{12}$$

$$Pp1(H_{e-1}) = (1 - S1_{e-1})^{n1} \tag{13}$$

Avec :

$$S0_e = \sum_{k=1}^{k=N} \left(\sum_{i=ak-e}^{i=ak+e} P0(i) \right) \tag{14}$$

$$S1_e = \sum_{k=1}^{k=N} \left(\sum_{i=ak-e}^{i=ak+e} P1(i) \right) \tag{15}$$

La probabilité $Pe0(H_e - H_{e-1})$ est une probabilité conditionnelle, puisque nous savons qu'il n'y a pas d'élément de classe $C1$ ni de classe $C0$ dans $He - 1$, avant de déterminer $Pe0(H_e - H_{e-1})$.

Soit :

$$Pe0(H_e - H_{e-1}) = 1 - \left(1 - \frac{S0_e - S0_{e-1}}{1 - S0_{e-1}} \right)^{n0} \tag{16}$$

Finalement :

$$Pv0 = \sum_{e=1}^{emax} (1 - S0_{e-1})^{n0} \cdot (1 - S1_{e-1})^{n1} \left[1 - \left(1 - \frac{S0_e - S0_{e-1}}{1 - S0_{e-1}} \right)^{n0} \right] \tag{17}$$

De plus, $Pm1 = 1 - Pb1$, où $Pb1$ est la probabilité de bien classer un élément de classe $C1$.

$$Pm1 = 1 - Pb1 = 1 - \left(\sum_E P1(A) \cdot (1 - Pv0) \right) \tag{18}$$

$$Pm1 = \sum_E P1(A) \cdot Pv0 \tag{19}$$

$P1(A)$ représentant la probabilité pour qu'un élément de classe $C1$ soit au point A.

Finalement :

$$Pe = \frac{n0}{n1+n0} \left[1 - \sum_E P0(A) \cdot Pv0 \right] + \frac{n1}{n1+n0} \sum_E P1(A) \cdot Pv0 \tag{20}$$

Il est à noter que cette formulation de la probabilité d'erreur est applicable quelle que soit la distribution des éléments de la classe $C0$ et de classe $C1$, et quel que soit le nombre de paramètres N.

3.1.2. Étude pratique

L'étude a été menée de façon pratique, c'est–à–dire sur des tirages aléatoires réels. Nous avons choisi des distributions gaussiennes pour chaque classe, en suivant une démarche analogue à celle menée par [11] pour caractériser les performances du classifieur « Riffle ».

Les points de mesure, provenant de moyennes faites entre plusieurs réalisations, ont été reportés sur la figure 9. Ils ont été obtenus à partir de distributions telles que celles reportées figure 7, c'est–à–dire où les valeurs moyennes des distributions de chaque classe sont telles que (voir la figure 8) :

$$Ecart = m_1 - m_0 = n \cdot \sigma \tag{21}$$

où m_i est la valeur moyenne de la distribution gaussienne i, sur chaque axe.

σ est l'écart–type de la distribution (nous l'avons choisi égal à 20 pour toutes les distributions).

Pour déterminer l'erreur, nous avons, pour chaque mesure, créé deux ensembles distincts d'échantillons. L'un est utilisé en apprentissage, et l'autre, contenant un grand nombre de points, est utilisé pour la vérification. Les deux ensembles sont construits avec les mêmes caractéristiques de distribution.

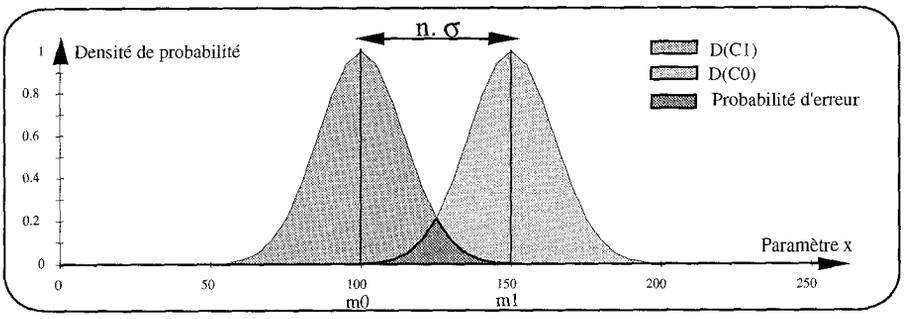


Figure 8. – Distributions gaussiennes à deux classes, N = 1.

3.1.3. Résultats numériques et comparaison à l'optimum

Pour une configuration donnée, du type de la figure 8, l'erreur optimum de classification ne peut être inférieure à l'aire de l'intersection entre les deux gaussiennes [5].

Pour pouvoir comparer les performances de notre méthode à l'optimum de classification, nous avons choisi des distributions gaussiennes pour chaque classe, dans les trois cas : théorique (Pe), pratique (tirages aléatoires) et optimum.

Dans le cas théorique, la probabilité d'erreur Pe est obtenue à partir de (20), avec $n0 = n1 = 50$, et en utilisant pour $P0(A)$ et $P1(A)$ des formes de distribution gaussiennes. Pour $N = 1$, ces distributions s'expriment par :

$$P0(x) = \frac{1}{\sigma \cdot \sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-m_0}{\sigma})^2} \qquad (22)$$

et

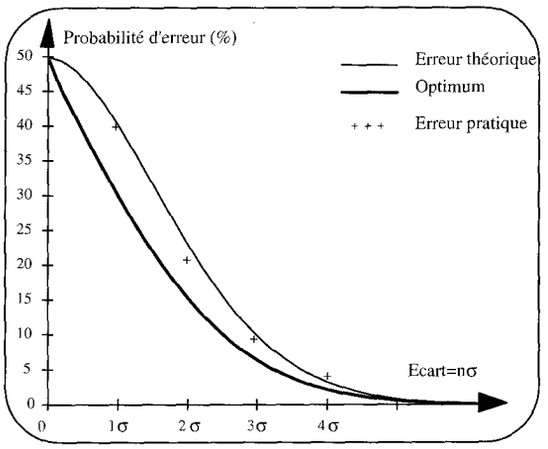
$$P1(x) = \frac{1}{\sigma \cdot \sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-m_1}{\sigma})^2} \qquad (23)$$

Nous avons alors fait varier les différents paramètres caractéristiques des distributions (écart–type σ, moyenne m). Nous avons obtenu des résultats numériques, reportés figure 9.

Nous pouvons d'abord remarquer la très bonne adéquation entre les résultats numériques provenant de la probabilité théorique d'erreur et ceux provenant des tirages aléatoires réels.

D'autre part, à partir de $n = 3$, l'erreur produite par le classifieur se rapproche notablement de l'optimum. L'erreur est alors inférieure à 10%, ce qui permet d'utiliser le classifieur dans de bonnes conditions.

Nous avons également reporté dans le tableau précédent les valeurs numériques des performances données par la méthode des K plus proches voisins ($Kppv$) avec $K = 1$ et $K = 7$. Cette méthode consiste à classer un élément dans la classe déterminée comme majoritaire parmi les K plus proches voisins de l'élément à classer [5]. La distance utilisée ici est la distance euclidienne. On peut remarquer que les méthodes sont comparables en termes de performances, alors que la méthode des $Kppv$ n'est pas une méthode compilée (au sens où elle ne réduit pas la quantité d'informations contenue dans l'apprentissage) et ne peut être intégrée dans un seul composant dès lors que la dimension du problème devient importante.



	Optimum (%)	Théorique (%)	Pratique (%)	1 ppv (%)	7 ppv (%)
$n = 0$	50	50	50	50	50
$n = 1$	30	40	40	40	34
$n = 2$	15	23	21	18	16
$n = 3$	6	10	9	9	7
$n = 4$	2	3	4	2,5	2,2

Figure 9. – Erreurs théoriques, pratiques et optimum.

3.2. INFLUENCE DU NOMBRE D'ÉLÉMENTS D'APPRENTISSAGE

Pour étudier l'influence du nombre de points p d'apprentissage sur la probabilité d'erreur, nous avons déterminé cette probabilité pour plusieurs valeurs de p ainsi que pour plusieurs écarts entre $m0$ et $m1$ (figures 10 et 11, pour lesquelles $p = n0 = n1$) :

Nous pouvons tirer plusieurs enseignements de ces résultats : contrairement au cas traité par [13] qui pourrait nous faire penser

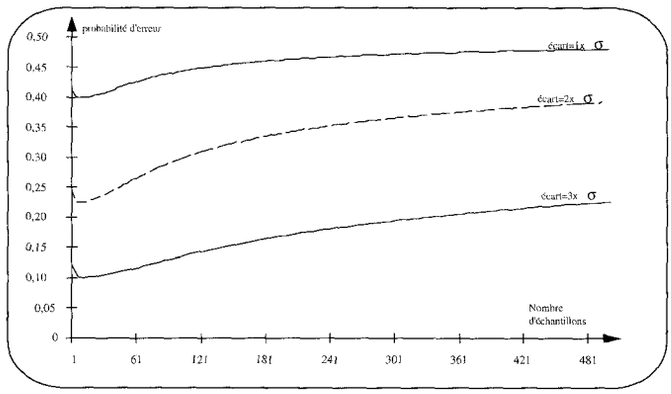


Figure 10. – Influence de l'échantillon initial, N = 1.

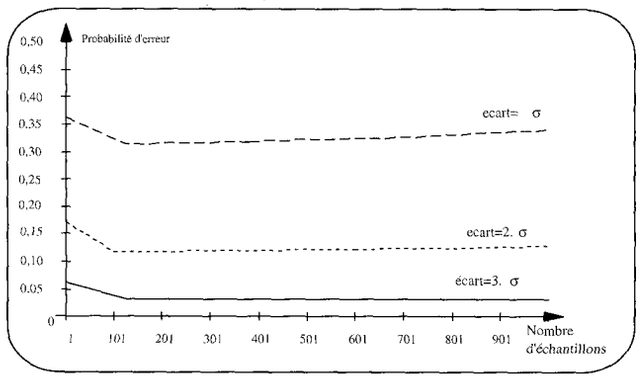


Figure 11. – Influence de l'échantillon initial, N = 2.

qu'il est préférable de prendre un ensemble d'apprentissage le plus vaste possible, on peut s'apercevoir qu'il existe un optimum au–delà duquel le pourcentage d'erreur augmente, et ceci quel que soit l'écart entre $m0$ et $m1$, et quelle que soit la dimension N.

Deux phénomènes expliquent la présence de cet optimum, ainsi que sa moindre importance lorsque la dimension augmente. Tout d'abord, et essentiellement pour N petit $(1, 2)$, la saturation de l'espace discret intervient rapidement et fait tendre l'erreur vers 50% pour un nombre élevé de points d'apprentissage : tout point peut alors être déclaré de classe $C0$ ou $C1$ (points ambigus). Intervient alors un second phénomène : le choix de la priorité pour l'une des deux classes, ce qui signifie qu'un point ambigu est considéré comme étant de classe $C0$. Un trop grand nombre de points d'apprentissage fait que seuls les points de classe $C0$ sont pris en compte par le système. Nous avons représenté un exemple pour $N = 1$, sur les figures 12a et 12b.

Nous avons représenté sur la figure 12a, la répartition des points de chaque classe dans l'espace discret des paramètres. Certains points sont redondants (plusieurs points de même classe au même endroit) : l'algorithme ne prend pas en compte cette redondance, et se comporte comme s'il n'y avait qu'un point, à un endroit donné de l'espace. La priorité donnée à la classe $C0$ fait que le système considère les points ambigus comme de classe $C0$. Les

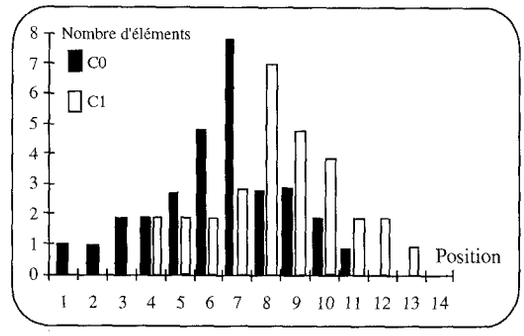


Figure 12a. – Echantillon réel.

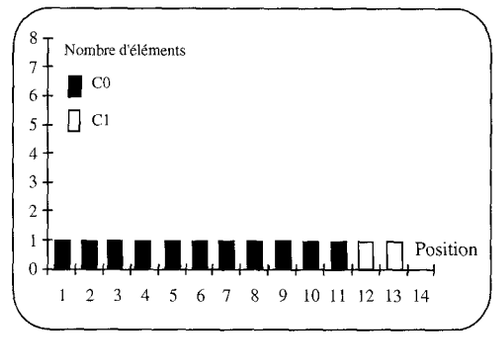


Figure 12b. – Points traités par le système.

points traités par le système sont ceux représentés figure 12b. Il est clair que si l'on avait encore augmenté le nombre d'éléments d'apprentissage, l'espace aurait été finalement uniquement rempli de points de classe $C0$.

Il est clair que lorsque N augmente, l'importance de ce phénomène de saturation est moindre (il faudrait énormément d'échantillons pour saturer l'espace à 4 paramètres, chacun pouvant varier de 0 à 255). Par contre, dans le cas de deux gaussiennes peu séparées (paramètres peu discriminants), le phénomène de priorité interviendra toujours de façon plus importante.

Toutes les mesures précédentes ont été réalisées avec $n0 = n1$. Or il est fréquent que dans la pratique les nombres d'échantillons disponibles soient très différents selon les classes. Nous avons donc également caractérisé le classifieur en fonction du rapport $r = \frac{n1}{n0}$ (voir figure 13).

Nous avons représenté l'erreur commise sur les points de classe $C0$ en fonction de celle commise sur les points de classe $C1$, pour différentes valeurs de r (les distributions utilisées sont gaussiennes, avec un écart de 2σ, et $N = 2$). Cette représentation met en évidence le fait que le rapport r donnant le minimum d'erreur pour les deux classes simultanément est $r = 1$. Dans la pratique, l'utilisateur devra choisir de privilégier le mauvais classement de l'une ou l'autre des classes, si r est différent de 1 (par exemple privilégier la classe des points des contours par rapport à ceux du fond de l'image dans l'exemple donné à la fin de cet article).

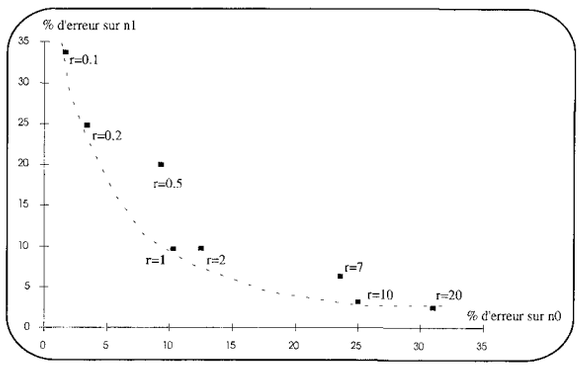


Figure 13. – Erreur de classification en fonction de r.

3.3. INFLUENCE DU CHOIX DES PARAMÈTRES

3.3.1. Influence du nombre de paramètres

Dans la pratique, la difficulté de trouver des paramètres très discriminants oblige le praticien à choisir plusieurs paramètres moyennement discriminants. Les résultats donnés par notre opérateur lorsque l'on ajoute progressivement des paramètres également discriminants (distributions gaussiennes de mêmes caractéristiques) sont reportés figure 14.

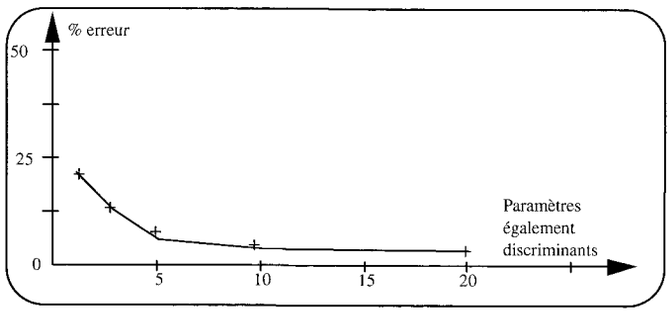


Figure 14. – Ajout de paramètres également discriminants.

Il est clair que l'augmentation du nombre de paramètres également discriminants améliore les performances du système, avec bien sûr un comportement asymptotique : dans cette configuration, une dizaine de paramètres conviendrait parfaitement, et en choisir plus alourdirait inutilement le système.

3.3.2. Robustesse du classifieur

Dans le but d'améliorer les performances de son classifieur, le praticien peut avoir recours à la sélection automatique de paramètres parmi un échantillon de départ, sans toujours connaître leur pertinence. Nous avons montré que le classifieur résistait à l'ajout involontaire de paramètres non pertinents, s'il existe au moins un paramètre discriminant (voir figure 15).

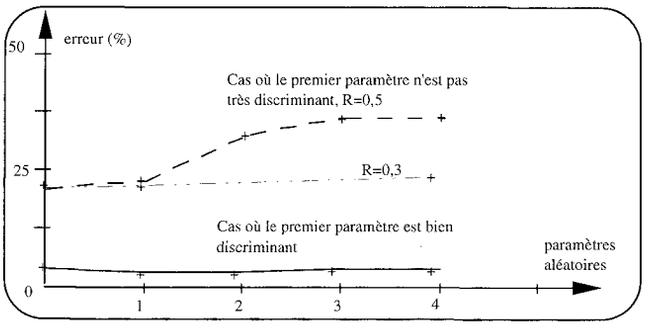


Figure 15. – Ajout de paramètres aléatoires.

Au cas où le premier paramètre n'est pas suffisamment discriminant, on constate une dégradation des performances. Ceci est dû à la présence de paramètres aléatoires, qui empêchent la fusion des hyperrectangles, donc la bonne généralisation du classifieur. Le système apprend de plus en plus « par cœur » un échantillonnage qui devient de plus en plus aléatoire! Nous avons vérifié ce phénomène en diminuant le rapport R, donc en augmentant la possibilité de fusion. Les performances ne varient alors plus en fonction du nombre de paramètres aléatoires, car le système peut généraliser sans être influencé par ces paramètres.

Notre méthode se distingue ici de certaines méthodes de classification comme celle des agrégats, qui dans le cas d'ajout de paramètres aléatoires, ne donnent pas de bons résultats [6].

3.3.3. Influence de la corrélation entre paramètres

L'influence de la corrélation entre paramètres a également été étudiée, en soumettant au classifieur deux paramètres et en faisant progressivement varier le cœfficient de corrélation qui les lie. Les performances obtenues varient de la manière suivante : l'erreur de classement augmente en même temps que le cœfficient de corrélation, mais reste toujours en deçà de l'erreur obtenue avec un seul des deux paramètres (voir figure 16).

Ceci nous permet de conclure que même si le choix de paramètres décorrélés est le meilleur au point de vue des performances, il

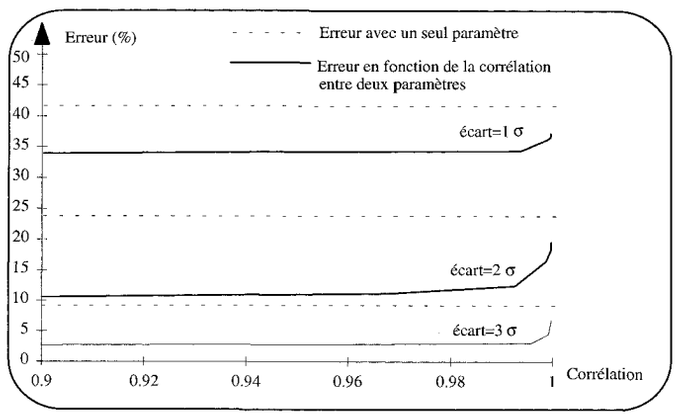


Figure 16. – Influence de la corrélation.

reste avantageux d'augmenter le nombre de paramètres, et les conclusions tirées de la figure 15 restent valables également.

Afin de caractériser encore mieux le classifieur à la fois en termes de performances et de complexité, nous avons étudié le nombre d'hyperrectangles créés en fonction du nombre d'échantillons d'apprentissage, pour deux écarts entre les distributions (toujours gaussiennes) et deux valeurs du rapport R. Les mesures ont été faites avec 2 ou 4 paramètres suivant les écarts entre les distributions (figure 17).

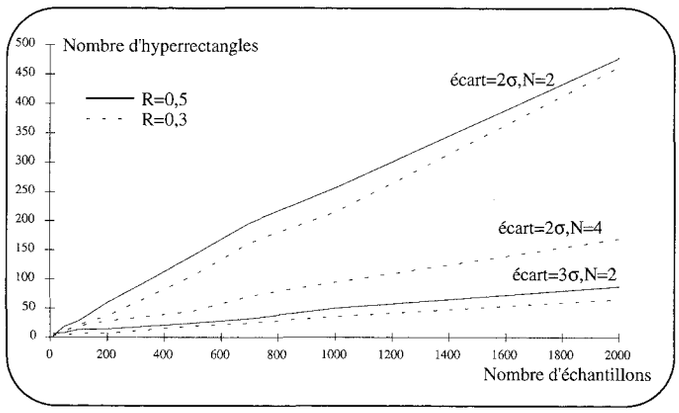


Figure 17. – Influence de l'échantillon sur le nombre d'hyperrectangles.

Nous pouvons constater une croissance quasi–linéaire du nombre d'hyperrectangles en fonction du nombre d'éléments d'apprentissage. Pour un nombre donné de points d'apprentissage, l'augmentation du nombre de paramètres ou de la pertinence de ces paramètres, diminue le nombre d'hyperrectangles, tout en améliorant les performances. Ces variations influent directement sur l'occupation des composants pour la réalisation de la phase de décision en temps réel. La comparaison des données pour $N = 2$ et $N = 4$ à écart constant nous montre qu'il est avantageux d'augmenter le nombre de paramètres discriminants (en gardant à l'esprit la limitation de 8 paramètres imposée par la réalisation temps réel), puisque les performances s'améliorent (voir figure 14) alors que l'occupation des composants (produit du nombre d'hyperrectangles par le nombre de paramètres) reste à peu près constante.

3.4. CONCLUSION

L'étude des performances du classifieur a donc été menée de manière théorique, ce qui a permis d'évaluer l'opérateur pour les grands nombres de points d'apprentissage, et surtout nous a donné une probabilité d'erreur dont la formule est indépendante de la distribution, indépendance qui permet de s'adapter très rapidement aux besoins.

L'étude pratique, même si elle est figée au cas de distributions gaussiennes, nous a donné des résultats pour un grand nombre de paramètres ($N > 10$), ce que nous ne pouvions obtenir avec la formule théorique, qui donne des temps de calcul très importants.

Au point de vue quantitatif, les performances obtenues sont tout à fait satisfaisantes dès que l'on est en présence de paramètres suffisamment discriminants : c'est le problème inhérent à toute classification. Toutefois, la robustesse et la résistance à la corrélation entre paramètres, ainsi que la simplicité de l'opérateur de décision permettent à cette méthode d'être employée dans la pratique.

L'aspect apprentissage–décision et la capacité à généraliser l'apprentissage grâce à la phase de fusion des hyperrectangles, font que la méthode s'approche du concept des réseaux de neurones.

4. Intégration et résultats

4.1. INTÉGRATION

Afin de réaliser une classification en temps réel, nous avons développé un circuit $VLSI$ qui effectue en parallèle les opérations décrites précédemment pour des vecteurs attributs de dimension huit.

La figure 18 illustre l'organisation physique du composant, implanté sous forme de 256 cellules élémentaires. La figure 19 présente la structure d'une de ces cellules, composée d'un élément x_t du vecteur attribut, d'une borne inférieure a_{ik}, d'une borne supérieure b_{ik} et d'un ensemble logique de comparaison.

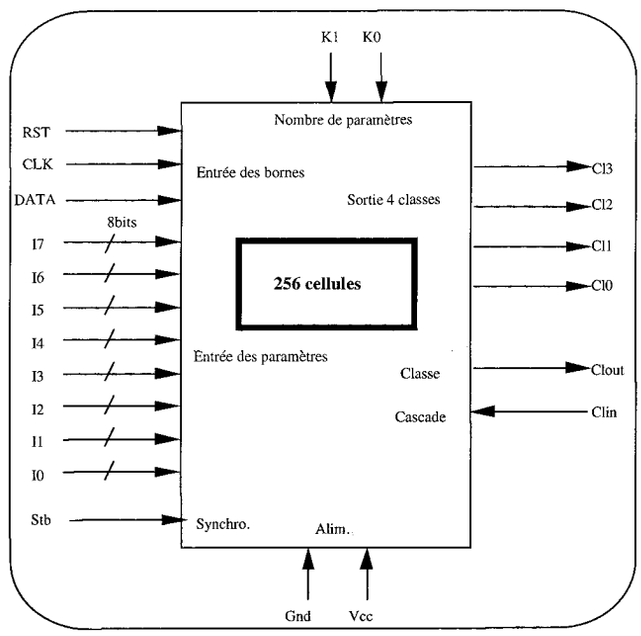


Figure 18. – Organisation physique.

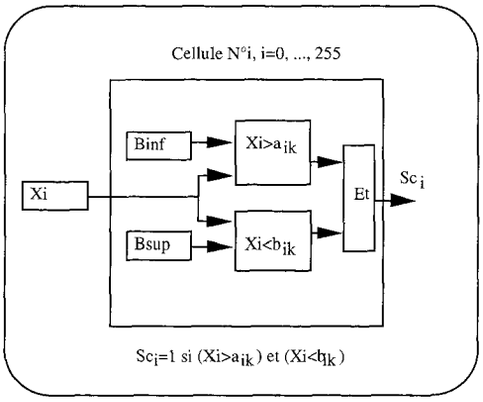


Figure 19. – Cellule élémentaire.

Les bornes a_{ik} et b_{ik} sont chargées en série à la mise sous tension du circuit. Les signaux $K1$ et $K0$ permettent de configurer l'ASIC pour 2, 4 ou 8 paramètres.

Chacune des cellules permet de vérifier la condition d'appartenance de l'attribut x_k à un intervalle $I_k = [a_k, b_k]$ donné. Si cette condition est vérifiée, la sortie Sc_i prend la valeur logique 1, sinon elle prend la valeur logique 0.

Le circuit a été prévu pour être utilisé soit avec deux classes (appartenance ou non à un polytope), soit en multi–classe, avec 4 classes. Dans ce dernier cas, les polytopes doivent couvrir tout l'espace des paramètres. Tout point appartient alors à au moins un polytope. C'est la sortie Cl_i (avec $i = 0, \ldots, 3$), valant 1, qui donne la classe du point.

Un décodeur de sortie 256 vers 4 indique en sortie le rang de l'ensemble de cellules qui prend la valeur logique 1, ce qui caractérise la classe du vecteur attribut. Ce décodeur est constitué de portes « ET/OU », configurables suivant la valeur de $K1$ et $K0$.

En effet, si nous prenons l'exemple à deux paramètres : pour déclarer qu'un point appartient à un polytope donné, défini par les cellules i et $i+1$, il faut que les sorties Sc_i et Sc_{i+1} soient en même temps égales à 1. Il faut donc un ET entre les deux sorties pour exprimer l'appartenance à ce rectangle (c'est le rôle de la première couche de ET située dans le bloc de cellules). Enfin, pour exprimer le fait qu'un point peut appartenir à plusieurs rectangles, il suffit de faire des OU entre les 128 sorties obtenues. Ceci est réalisé grâce aux trois couches de portes ET/OU du décodeur de sortie, configurées en OU. Le nombre maximum de polytopes est alors de 128 si l'on a deux classes (32 pour 4 classes).

Pour quatre paramètres, un hyperrectangle est défini par les cellules i, $i+1$, $i+2$, $i+3$. Il faut donc réaliser un ET entre les quatre sorties correspondantes pour déterminer l'appartenance d'un point à ce polytope. Ce ET est effectué par d'abord deux ET à deux entrées de la couche du bloc de cellules, puis par un ET à deux entrées provenant de la première couche du décodeur de sortie (figure 20). Cette couche de portes ET/OU doit donc alors être configurée en ET, alors que les deux couches suivantes doivent l'être en OU. Le nombre maximum de polytopes est alors de 64.

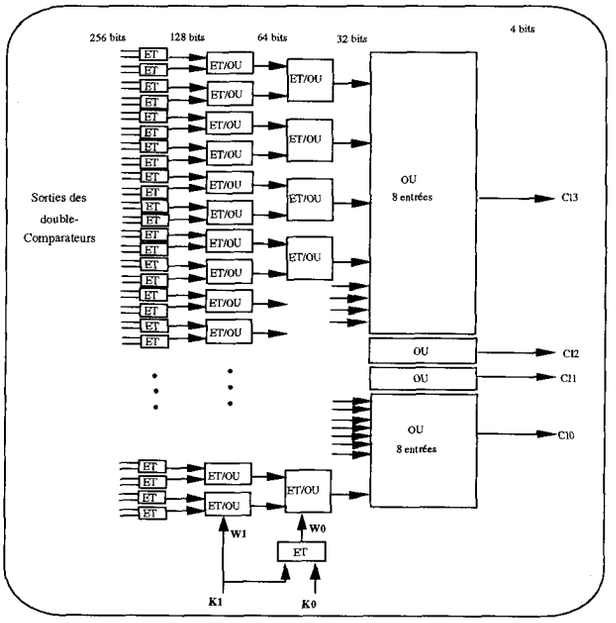


Figure 20. – Décodeur de sortie.

Pour huit paramètres, il faut regrouper les cellules par deux, puis encore par deux, par des ET, avant d'effectuer les OU. Seule la dernière couche sera donc configurée en OU, alors que les autres seront en ET. Le nombre maximum de polytopes est alors de 32.

Ces contraintes expliquent l'architecture que nous avons proposée pour le décodeur de sortie.

Le circuit effectue l'ensemble des comparaisons $(x_k > a_{ik})$ et $(x_k < b_{ik})$ en moins de $125ns$ et fournit en sortie la classe du vecteur attribut présenté en entrée. Si l'on souhaite augmenter le nombre de paramètres, le nombre de classes, ou le nombre de polytopes, il est possible d'augmenter le nombre de cellules en cascadant indéfiniment les composants.

Ce circuit a été développé dans l'environnement Cadence Solo 2130 en technologie $1.2\mu m$, et a conduit à une intégration des 512 comparateurs dans un composant de moins de $40mm^2$. Pour mener à bien l'intégration d'un si grand nombre de comparateurs sur un même composant, deux approches ont été nécessaires [14], [12]. La première approche, dont le dévelopement est le plus rapide et le plus aisé est dite « Standard Cell » et utilise des composants provenant d'une bibliothèque. Cette méthode permet l'intégration de seulement 64 doubles comparateurs dans un seul composant, c'est pourquoi la deuxième approche dite « Full Custom » a été menée. L'intégration transistor par transistor est effectivement la plus performante, bien que beaucoup plus complexe à mettre en œuvre. C'est cette dernière méthode qui permet l'intégration de 256 doubles comparateurs (soit par exemple 64 polytopes à 4 paramètres) dans un seul composant. L'optimisation a été réalisée au niveau de la cellule élémentaire, et surtout au niveau du routage entre ces cellules (20 mm^2 en Full Custom contre 100 mm^2 en Standard Cell).

4.2. RÉSULTATS

4.2.1. Segmentation de textures

L'image de la figure 21a représente deux objets texturés, posés sur un fond de texture légèrement différente. À titre indicatif, nous avons montré l'histogramme de l'image choisie (figure 21c). Cet histogramme est unimodal : il n'est donc pas possible d'effectuer directement à partir de celui–ci une segmentation correcte, séparant les objets du fond de l'image.

Grâce à un logiciel de dessin, nous avons segmenté cette image, en créant une image composée de trois zones (voir figure 21d). La première zone, en blanc, représente les points de classe 0 : ce sont les points des objets. La deuxième zone, laissée en niveau de gris, est constituée des points de classe 1 : c'est le fond de l'image. La troisième zone, en noir, n'est pas prise en compte car ces points sont ambigus et pourraient introduire des erreurs lors de l'apprentissage. Le logiciel d'échantillonnage prend alors en compte un certain nombre de points de l'image de référence 1 (nombre déterminé par l'utilisateur), calcule pour chacun les paramètres spécifiés par l'opérateur, et les range suivant leur classe (donnée par l'image composée des trois zones) dans un fichier de mesures.

Le fichier est fourni à l'algorithme d'apprentissage, qui élabore les polytopes de contrainte. Nous avons alors vérifié le bon classement des points de l'image d'apprentissage, sur la figure 21b. Nous pouvons ensuite mesurer les mêmes paramètres que précédemment sur chaque point d'une nouvelle image (Figure 21e), et effectuer la décision correspondante en temps réel. Ceci donne l'image 21f.

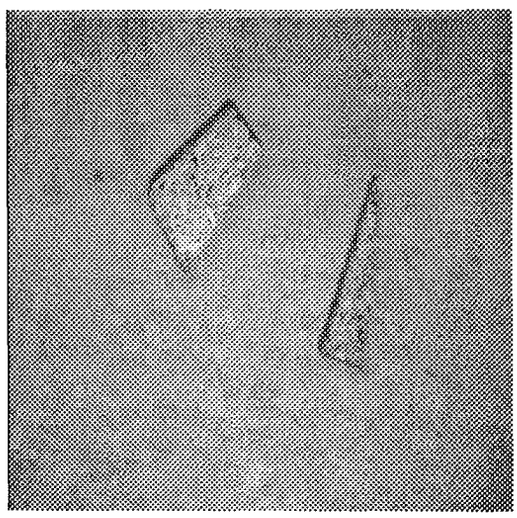


Figure 21a. – Image d'apprentissage 1-1.

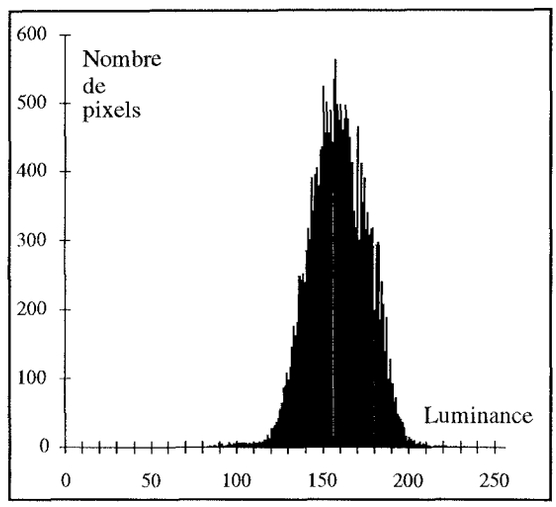


Figure 21c. – Histogramme de l'image 1-1.

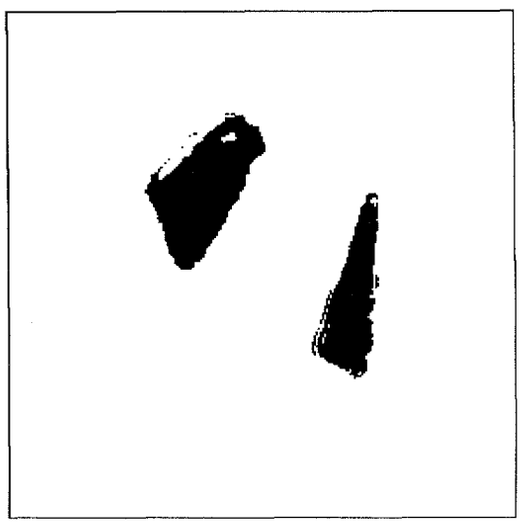


Figure 21b. – Image 1-1 segmentée par classification.

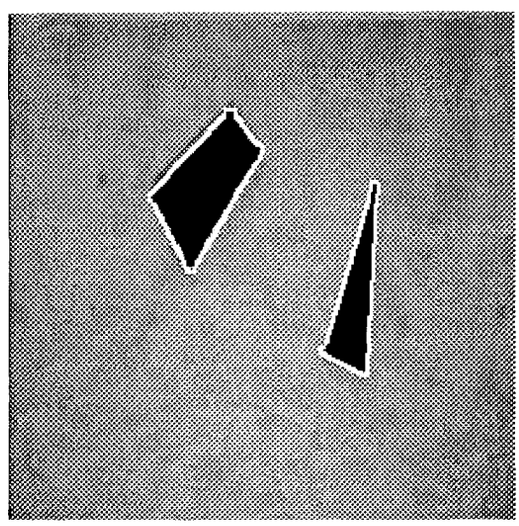


Figure 21d. – Image 1-1 segmentée "à la main".

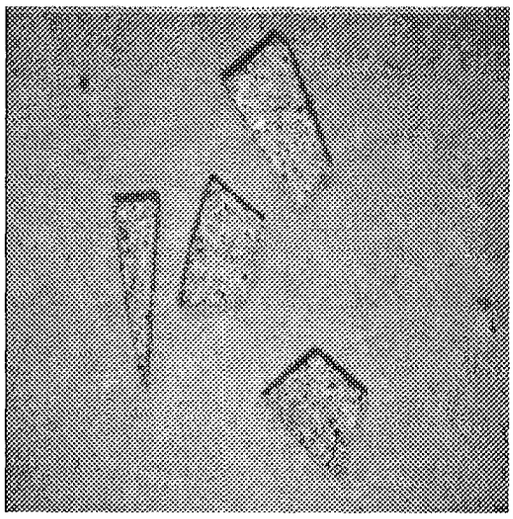


Figure 21e. – Image d'exemple 1-2.

Figure 21f. – Image 1-2 segmentée par classification.

Les paramètres utilisés dans cet exemple sont la luminance moyenne et le gradient moyen dans des fenêtres de différentes tailles autour du point considéré. Il s'agit ici du gradient de Roberts, que nous avons utilisé car les images de cet exemple présentent un rapport signal sur bruit élevé (le diagramme de Pratt nous montre que dans ce cas, le gradient de Roberts est suffisamment performant), et parce que nous disposons d'un ASIC permettant son calcul en temps réel.

Pour cet exemple, les 8 paramètres utilisés sont :

- paramètre $0 = \text{gradmoy}(p, 6)$;
- paramètre $1 = \text{moy}(p, 4)$;
- paramètre $2 = \text{moy}(p, 8)$;
- paramètre $3 = \text{moy}(p, 12)$;
- paramètre $4 = \text{moy}(p, 18)$;
- paramètre $5 = \text{gradmoy}(p, 12)$;
- paramètre $6 = \text{moycour}(p, 20)$;
- paramètre $7 = \text{moycour}(p, 10)$;

Gradmoy(p, i) représente le gradient moyen dans une fenêtre carré, centrée autour de p, de côté $2 \cdot i$.

Moy(p, i) représente la luminance moyenne dans le même type de fenêtre autour de p.

Moycour(p, i), représente la moyenne des valeurs trouvées dans la couronne d'un pixel de large, centrée en p, de côté intérieur $2 \cdot i$.

Il n'est pas possible de représenter l'espace des paramètres dans son intégralité, mais nous pouvons donner ces deux projections, représentant les deux premiers paramètres d'une part, et les paramètres 4 et 7 d'autre part, sur la figure 22. Les carrés noirs représentent les points du fond de l'image, et les blancs représentent ceux des objets.

Le faible nombre final de polytopes par rapport au nombre de points d'apprentissage tend à montrer que le choix des paramètres est bon. Toutefois une étude menée grâce à la sélection automatique de paramètres (alors qu'ils avaient été choisis empiriquement jusque là), montre que 4 paramètres suffisent pour obtenir une image aussi bonne en décision, avec seulement une vingtaine de polytopes, ce qui est réalisable en temps réel, puisque l'exemple demande alors 80 doubles comparateurs.

4.2.2. *Détection de contours*

La détection de contours dans une image n'est autre qu'une forme particulière de segmentation dans laquelle on cherche à mettre en valeur les contours des objets présents dans la scène étudiée, c'est-à-dire les zones où le gradient de luminance est important [8] (transitions rapides entre les niveaux de gris de faibles valeurs et ceux de fortes valeurs).

La figure 23a représente l'image d'apprentissage, et 23b la même image segmentée par un algorithme classique de détection de contour (gradient de Roberts). Ces deux images ont été fournies au système d'apprentissage, qui a élaboré les contraintes selon l'algorithme décrit précédemment. Nous avons ensuite appliqué la phase de décision sur l'image 23c, ce qui nous a donné l'image segmentée 23d.

Les paramètres utilisés ici sont simplement les valeurs de luminance des quatre points $p(l, c), p(l+1, c), p(l+1, c+1), p(l, c+1)$, où $p(l, c)$ est le point courant en ligne l et colonne c. L'allure de l'espace des paramètres, pour les paramètres 0 et 1, est représentée figure 24. Nous avons obtenu une détection de contour dont les performances sont tout à fait semblables à l'algorithme de départ, et réalisable en temps réel. Nous avons vérifié le classement des points de l'image du bureau à partir de l'image segmentée par l'algorithme de détection de contours. Les erreurs de classements sont de 2,6% pour les points de contours et 4,5% pour les points du fond de l'image.

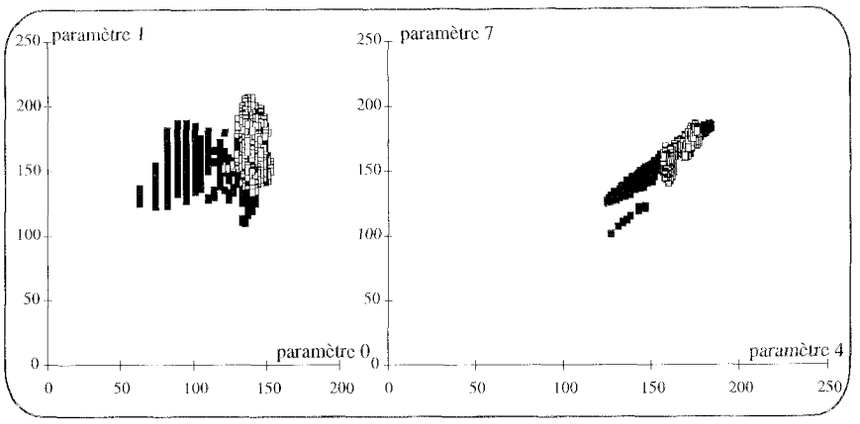


Figure 22. – Espace des paramètres de l'exemple 1.

Figure 23a. – Image d'apprentissage 2.

Figure 23b. – Image d'apprentissage 2 segmentée.

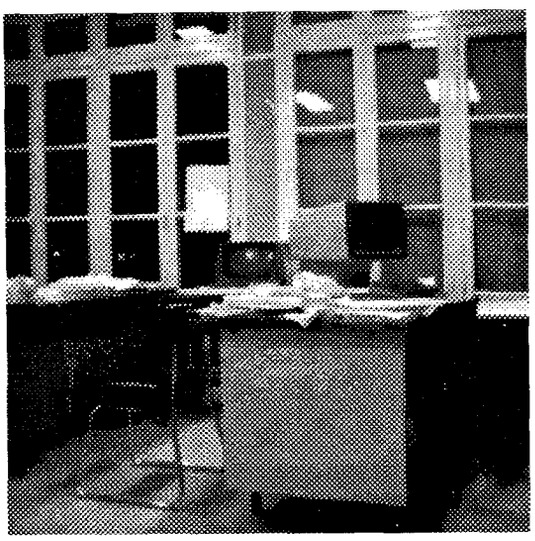


Figure 23c. – Image d'exemple 2.

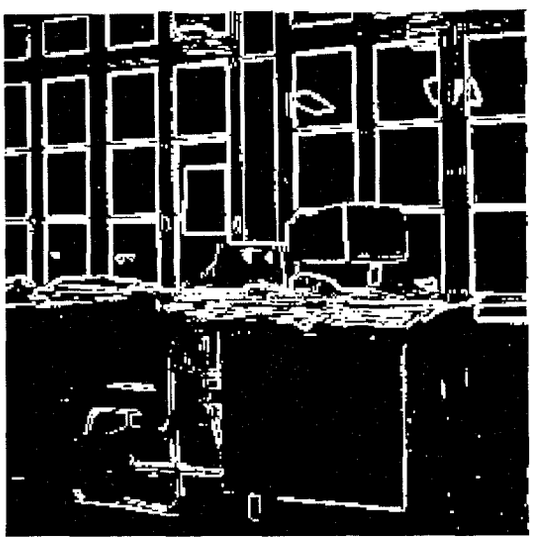


Figure 23d. – Image d'apprentissage 2 segmentée.

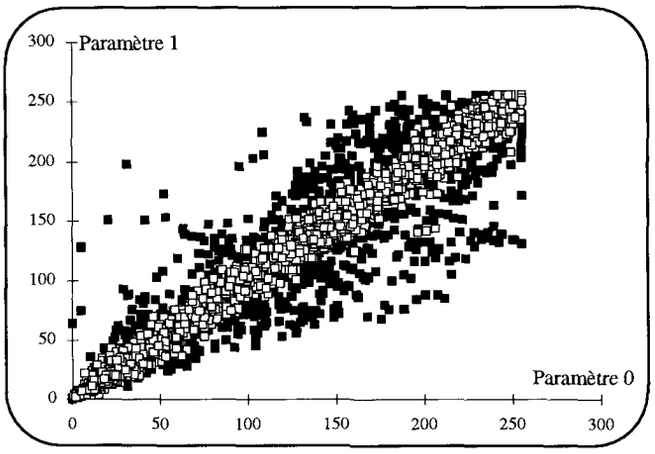


Figure 24. – Espace des paramètres pour la détection de contour.

Étant donné la dissemblance entre l'image d'apprentissage et celle utilisée pour la décision, ces performances sont tout à fait remarquables et prouvent une fois de plus le bien–fondé de cette méthode. Il est à noter qu'une recherche plus approfondie concernant la détection de contour par apprentissage est en cours au Laboratoire GERE et fait l'objet d'un travail de thèse. La recherche est basée sur une analyse multirésolution (transformée en ondelettes) qui alimente le classifieur en paramètres calculés grâce à des images de contours obtenues par des filtres comme celui de Canny–Deriche.

5. Conclusion

La méthode de classification géométrique que nous avons mise au point et caractérisée permet le traitement d'images en temps réel et notamment la détection de défauts, des applications en télésurveillance étant également à l'étude. Les performances ont été étudiées de manière théorique et pratique, ainsi que l'influence du nombre d'éléments d'apprentissage et du choix des paramètres. Malgré les concessions faites pour obtenir le temps réel, en découpant l'espace avec des hyperrectangles, les performances sont proches de l'optimum. L'opérateur de décision associé à cette classification a été intégré sous forme de circuit précaractérisé dont la simplicité de mise en œuvre, la rapidité et la robustesse en classification sont des qualités qui lui permettent de rivaliser avec les opérateurs neuronaux.

BIBLIOGRAPHIE

[1] F. BARLACH, « Payphone coin validation using pattern recognition », *Pattern Recognition*, vol. 23, N° 3,4, 1990, pp. 379-384.

[2] W.E. BLANZ, S.L. GISH, « A real–time segmentation system using a connectionist classifier architecture », *International journal of pattern recognition and artificial intelligence*, vol. 5, N° 4, 1991, pp. 603-617.

[3] G. BUREL, J.Y. CATROS, H. HENOCQ, « Caractérisation et classification de textures sur images naturelles », *Traitement du signal*, vol. 9, N°1, 1992, pp. 33-43.

[4] C.E. COX, W.E. BLANZ, « GANGLION — A real Field–programmable gate array implementation of a connexionist classifier », *IEEE journal of solid state circuits*, vol. 27, N° 3, Mars 1992, pp. 288-299.

[5] B. DUBUISSON, « Diagnostic et reconnaissance des formes », *Traité des nouvelles technologies*, Hermès, Paris, 1990, chapitre 3.

[6] R.O. DUDA, P.E. HART, « Pattern classification and scene analysis », New–York, Wiley, 1973, pp. 230-243.

[7] Y. GOUR, « Contribution à la reconnaissance automatique des images. Classification des défauts par polytopes de contrainte ». Thèse, Université de Bourgogne, 1991.

[8] R. HORAUD, O. MONGA, « Vision par ordinateur. Outils fondamentaux ». *Traité des nouvelles technologies*, Hermès, Paris, 1993, pp 23-40.

[9] M. ICHINO, « A nonparametric multiclass pattern classifier », *IEEE trans. on systems, man, and cybernetics*, vol. 9, N° 6, Juin 1979, pp. 345-352.

[10] J.M. JOLION, P. MEER, S. BATAOUCHE, « Robust clustering with applications in computer vision », *IEEE trans. on pattern analysis and machine intelligence*, vol. 13, N° 8, Août 1991.

[11] G. MATTEWS, J. HEARNE, « Clustering without a metric », *IEEE trans. on pattern analysis and machine intelligence*, vol. 13, N° 2, Février 1991, pp. 175-184.

[12] J. MITÉRAN, P. GORRIA, M. ROBERT, « Performances et intégration d'un opérateur de classification géométrique par apprentissage. Applications en traitement d'images », Thèse, Université de Bourgogne, 1994.

[13] S.J. RAUDYS, A.K. JAIN, « Small sample size effects in statistical pattern recognition : recommendations for practitioners », *IEEE trans.on pattern analysis and machine intelligence*, vol. 13, N° 3, Mars 1991, pp. 252-264.

[14] M. ROBERT, M. PAINDAVOINE, P. GORRIA, « Architectures for integration of real time image processing systems. », *IEEE workshop on VLSI Signal Processing*, Napa California, Octobre 1992, pp. 267-276.

[15] J. SAGAUT, A. ACCES, « La conception d'une architecture neuronale », *Veille Technologique*, N° 6, Mars-Avril 1994, pp. 27-30.

[16] A. TABATABAI, T.P. TROUDET, « A neural net based architecture for the segmentation of mixed gray level and binary pictures », *IEEE trans. on circuits and system*, vol. 38, N° 1, Janvier 1991, pp. 66-77.

Manuscrit reçu le 16 novembre 1993.

LES AUTEURS

Johel MITÉRAN

Johel Mitéran a reçu une Maîtrise de Physique et Applications en 1990 et un Diplôme d'Études Approfondies (DEA de Physique) en 1991 à l'Université de Bourgogne. Il poursuit actuellement ses études au Laboratoire GERE de l'IUT du Creusot rattaché à l'Université de Bourgogne, afin d'obtenir un Doctorat d'Université. Il s'intéresse au traitement d'images en temps réel, traitement du signal, reconnaissance des formes, et travaille actuellement à la mise au point et la caractérisation d'un classifieur applicable à la télésurveillance.

Patrick GORRIA

Patrick Gorria est Ingénieur de l'École Nationale Supérieure des Arts et Métiers (1980) et Docteur d'Université (1984). Actuellement Professeur à l'IUT du Creusot, il a contribué au démarrage d'une activité de recherche en traitement d'images au laboratoire GERE de l'IUT du Creusot. Membre du GDR 134, il dirige des travaux de recherche portant sur la détection de défauts par vision artificielle et l'intégration d'opérateurs de classification en temps réel.

Michel ROBERT

Michel Robert a reçu un Doctorat en Ingénierie Électronique de l'Université de Montpellier II en 1987. De 1980 à 1984, il est employé par Texas Instrument dans la division semiconducteurs, comme ingénieur de test et développement spécialisé dans le programme standard cell. Il rejoint la Laboratoire d'Automatique et de Microélectronique de Montpellier, en 1984. Professeur en microélectronique de l'Université de Montpellier II, ses recherches actuelles incluent les différents aspects du développement des VLSI, comme les méthodologies de conception des ASIC, le routage automatique, l'optimisation des cellules, principalement en technologie CMOS.