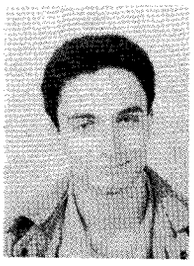
Quantification vectorielle d'images par le réseau neuronal de Kohonen

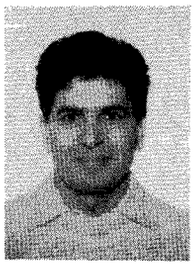
Vector quantization of images using Kohonen neural network



Éric LE BAIL

ENST, 46, rue Barrault, 75634 PARIS.

Éric Le Bail, ancien élève de l'École Nationale Supérieure des Télécommunications (promotion 1989), travaille actuellement à l'université d'Ottawa, au Centre de Recherche en Communications Médicales du département de génie électrique. Ses centres d'intérêt sont liés aux réseaux neuronaux et au traitement des images. Ses recherches en cours, dans le cadre du projet IRIS (Integrated Radiographic Information System), portent sur les problèmes de traitement d'images dans les stations de travail pour médecins.



Amar MITICHE

INRS-Télécommunications, 3, place du Commerce, VERDUN, QUÉBEC H3E 1H6, CANADA.

Amar Mitiche a obtenu une licence ès-sciences en mathématiques pures à l'Université d'Alger et une thèse de doctorat (Ph.D.) en informatique à l'Université du Texas à Austin. Son travail de thèse à l'Université du Texas à Austin concernait l'analyse et la discrimination des textures. Après un séjour au Laboratory for Image and Signal Analysis de l'université du Texas à Austin où il s'est intéressé à divers problèmes de vision par ordinateur, il est professeur à l'Institut National de La Recherche Scientifique (INRS-Télécommunications) à Montréal, Québec, Canada, depuis 1985. Il s'intéresse actuellement, surtout du point de vue méthodologique, au calcul et à l'interprétation du mouvement, la reconnaissance des formes, l'intégration de sources d'information multiples, et aux réseaux neuronaux.

RÉSUMÉ

Cet article étudie les possibilités offertes par l'utilisation du réseau neuronal de Kohonen pour la quantification vectorielle d'images. Quelques résultats théoriques sur la convergence du processus d'apprentissage du réseau précéderont les essais réalisés sur des images. On comparera d'abord, à l'aide d'une image test, les performances obtenues pour des dictionnaires de taille et de dimension différentes. On poursuivra alors les essais avec les meilleures combinaisons, en utilisant finalement plusieurs images pour élaborer les dictionnaires. Cette étude se particularise par l'emploi de cinq réseaux en parallèle, ainsi que par la variation de plusieurs paramètres, longueur de la séquence d'entraînement, nombre de classes de vecteurs, dimension des vecteurs, et nombre de vecteurs utilisés pour le codage.

MOTS CLÉS

Traitement d'images, vision par ordinateur, quantification vectorielle d'images, réseau de Kohonen, blocs, taux de compression, erreur quadratique moyenne, rapport signal à bruit, dictionnaires.

SUMMARY

In this article, the possible use of Kohonen's neural network to vector quantize images is investigated. Some theoretical results on convergence of the training process are first given. Then, results obtained for various codebook sizes and input dimensions are compared. Tests are then performed with the best parameter values, using several images to design codebooks. This approach is based on the concurrent use of five networks where the effect of various relevant parameters is studied, such as number of classes of vectors, vectors dimension, and number of vectors used for coding.

KEY WORDS

Image processing, computer vision, image vector quantization, Kohonen network, blocs, compression, mean squared error, signal to noise ratio, codebooks.

1. Introduction

Depuis plusieurs années, de nombreuses techniques de quantification vectorielle se sont développées, tant en traitement de la parole qu'en traitement d'images, deux domaines où les problèmes de réduction du débit de bits transmis sont particulièrement importants. En traitement d'images, les applications sont nombreuses et en rapide développement : images satellite, vidéo-conférence, transmission de fac-similés de documents imprimés, images de télévision. La compression permet en outre de diminuer considérablement les coûts de stockage, un facteur essentiel en particulier pour l'archivage d'images médicales ou d'empreintes digitales. On trouvera dans [1] une présentation des principales méthodes de quantification vectorielle appliquée aux images. Le principal problème rencontré est le coût important de stockage des dictionnaires (codebook), c'est-à-dire des vecteurs utilisés pour le codage. De plus, la plupart des techniques classiques ne sont pas optimales, car elles utilisent des données corrélées entre elles.

A l'origine, l'algorithme mis au point par Kohonen permet de former, de façon autonome, des cartographies de propriétés données (self-organizing feature maps). Ce type de cartographies est typiquement réalisé par le cerveau, à partir d'entrées sensorielles tactiles, visuelles ou acoustiques. En général, le système nerveux, au cours des premières étapes de son développement, auto-organise ces cartographies qui ne sont donc pas pré-spécifiées génétiquement. Une fois réalisées, elles conservent, suivant les cas, plus ou moins de plasticité. Le réseau de Kohonen n'est pas destiné à modéliser fidèlement les processus à l'œuvre dans le cerveau, mais tente d'en reproduire les traits essentiels, tout en restant raisonnablement simple du point de vue des calculs. D'autres auteurs ont proposé des solutions différentes [2 à 9], mais le réseau de Kohonen est le plus apte à être utilisé comme quantificateur vectoriel. En effet, il réalise la projection d'un espace d'entrées dans un ensemble fini de vecteurs de sortie, suivant une procédure proche de celle mise en œuvre dans l'algorithme de K-means, une méthode qui conduit à un quantificateur optimal localement. De plus, dans les méthodes classiques, la constitution du dictionnaire initial est un problème très délicat, alors que, d'après [10] et [11], les valeurs de convergence des vecteurs du réseau de Kohonen ne dépendent pas des valeurs initiales.

Par rapport aux travaux déjà effectués sur l'utilisation du réseau de Kohonen comme quantificateur vectoriel [12], l'approche décrite ici se propose d'indiquer le bénéfice que l'on peut attendre d'une telle utilisation, en fonction de plusieurs paramètres définis ci-dessous. Dans quelques cas, on essaiera de pousser les calculs afin d'avoir une idée des performances maximales du système. En aucun cas, l'étude présente n'est-elle exhaustive, elle vise plutôt à donner une bonne idée des différentes possibilités offertes.

Le prochain chapitre introduira la notion de quantificateur vectoriel. Ensuite, nous présenterons le réseau de Kohonen, avec quelques données théoriques, avant d'exposer les différents résultats expérimentaux.

2. La quantification vectorielle

2.1. Définitions et propriétés

On peut définir un quantificateur vectoriel q par une application d'un ensemble E dans un ensemble $F \subset E$:

$$q: \begin{array}{c} E \to F \subset E \\ \mathbf{x}=(x_0, \ldots, x_{k-1}) \to q(\mathbf{x}) \\ =\hat{\mathbf{x}} \in \hat{A}=\{\mathbf{y}_i, i=1, \ldots, N\}, \\ \mathbf{y}_i \in F \end{array}$$

E est partitionné par $S=\{S_i, i=1, \ldots, N\}$, avec $S_i=\{\mathbf{x}/q(\mathbf{x})=\mathbf{y}_i\}$.

La distortion entre $\mathbf{x}$ et $\hat{\mathbf{x}}$ est donnée par la grandeur positive ou nulle $d(\mathbf{x}, \hat{\mathbf{x}})$. $\hat{A}$ est alors un dictionnaire de taille N et de dimension k.

Dans la quantification scalaire, chaque coordonnée x_i de $\mathbf{x}$ serait associée à un quantificateur $q_i: x_i \to y_i$. Si l'on considère le débit d'informations, on a $B=\log_2 N$ bits/vecteur, et R, le nombre de bits par dimension, est égal à B/k bits/dimension. On voit donc que la quantification vectorielle permet d'obtenir $R < 1$, ce qui est impossible en quantification scalaire.

Le quantificateur optimal est celui qui minimise la distortion pour une séquence aléatoire de vecteurs $\mathbf{X}=(X_0, \ldots, X_{k-1})$, où X est de probabilité $p(\mathbf{X})$.

Si on définit alors la performance du quantificateur q par $D(q)=E(d(\mathbf{X}, q(\mathbf{X}))$, q^* est optimal si, pour tout quantificateur ayant le même nombre de vecteurs de reproduction que q^*, $D(q^*) \leqq D(q)$.

La donnée de $\hat{A}$ et de S déterminant entièrement q, on peut écrire $D(q)=D(\hat{A}, S)$.

Expression de $D(\hat{A}, S)$;

$$D(\hat{A}, S)=E(d(\mathbf{X}, q(\mathbf{X})) = \sum_{i=1}^{N} \Pr(\mathbf{x} \in S_i) \int d(\mathbf{x}, \mathbf{y}_i/\mathbf{x} \in S_i)\, p(\mathbf{x}/\mathbf{x} \in S_i)\, d\mathbf{x}.$$

Deux propriétés importantes des quantificateurs vectoriels sont à l'origine de la plupart des algorithmes utilisés dans les réalisations classiques :

1. Si $\hat{A}$ est donné, alors la meilleure partition possible de l'espace d'entrée est $P(\hat{A})$, obtenue en faisant correspondre chaque X au vecteur $\mathbf{y}_i$ de $\hat{A}$ minimisant $d(\mathbf{x}, \mathbf{y}_i)$. C'est la règle dite « du plus proche voisin ».

2. Si S est donnée, alors supposons que pour tout S de probabilité non nulle dans un espace k-euclidien,

$$\exists\, \hat{\mathbf{x}}(S)/E(d(\mathbf{X}, \hat{\mathbf{x}}(S))/\mathbf{X} \in S) = \min_{\mathbf{u}} E(d(\mathbf{X}, \mathbf{u})/\mathbf{X} \in S)$$

$\hat{\mathbf{x}}(S)$ correspond donc à un barycentre généralisé de S. Dans ces conditions, le meilleur alphabet de reproduction est $\hat{\mathbf{x}}(S)=\{\hat{\mathbf{x}}(S_i), i=1, \ldots, N\}$.

2.2. Deux algorithmes : K-means et Isodata

Outre leur similitude avec les équations régissant l'évolution du réseau de Kohonen, ces deux algo-

rithmes présentent l'intérêt d'être à la base de nombreuses méthodes classiques de quantification vectorielle. Partant de M vecteurs d'entraînement $\mathbf{x}(n)$, $1 \leqq n \leqq M$, le problème est de diviser cet ensemble en K groupes S_i tels que les deux conditions d'optimalité soient satisfaites. K-mean est un algorithme conduisant à un optimum local satisfaisant aux deux conditions :

1. choisir K vecteurs $\mathbf{y}_i(0)$, initialiser m, l'indice des itérations, à 0;
2. classification : à l'itération m, ranger les vecteurs $\{\mathbf{x}(n), n=1, \ldots, M\}$ dans les $S_i(m)$ en appliquant la règle du plus proche voisin à partir des vecteurs $\mathbf{y}_j(m)$;
3. mise à jour des vecteurs codes : $\mathbf{y}_j(m+1) =$ barycentre $(S_i(m))$, $1 \leqq i \leqq K$;
4. test de fin sur le gain en distortion par rapport à l'itération précédente. Si le résultat est négatif, aller en 2.

On voit que dans K-means, on a besoin de toute la séquence d'entraînement à chaque itération, et on n'a pas de quantificateur tant que la procédure n'est pas terminée. C'est donc plutôt une méthode de groupement que de quantification. En quantification, on cherche à fixer les groupes pour les utiliser sur des données autres que celles ayant servi à l'entraînement, alors qu'ici, les données doivent faire partie de la séquence d'entraînement, et les groupes changent à chaque itération. Une méthode similaire, qui produit à chaque étape un QV, est l'algorithme de Linde, Buzo et Gray (LBG) [13]. Avec cet algorithme, une partie seulement de la séquence d'entrée est utilisée pour construire le quantificateur, et, à condition que la séquence de vecteurs produits par la source soit stationnaire et ergodique, la réalisation reste pratiquement optimale pour n'importe quel vecteur produit ultérieurement. Quant à la méthode Isodata, elle ne diffère de K-means qu'en un seul point : à chaque étape, un seul vecteur est incorporé, au lieu d'utiliser toute la séquence d'entraînement.

3. Le réseau de Kohonen

3.1. Fonction et organisation

Le réseau de Kohonen réalise la liaison entre un ensemble de I entrées et un ensemble de J sorties, par l'intermédiaire de $I \times J$ coefficients ou poids (*fig.* 3.1). Une propriété importante du réseau est que les poids évoluent de façon à conserver entre les vecteurs de sortie les relations topologiques présentes entre les vecteurs d'entrée. On dit que le réseau s'auto-organise (self-organizing).

L'espace d'entrée, E, est de dimension I, et on en tire des vecteurs $\mathbf{x} = (x_1, \ldots, x_I)^T$. Chaque entrée x_i est reliée à J nœuds de sortie, ou neurones, par J coefficients μ_{ij}. Chaque sortie j peut donc être considérée comme portant un vecteur image $\boldsymbol{\mu}_j = (\mu_{1j}, \ldots, \mu_{Ij})^T$. Les J sorties sont disposées de façon ordonnée, de manière à définir un voisinage physique ou topologique de chaque nœud. En général, on les dispose en ligne ou en grille.

L'entraînement consiste à présenter successivement au réseau des vecteurs d'entrée $\mathbf{x}$. A chaque fois, c'est-à-dire à chaque itération, l'opération de base consiste à déterminer le nœud j^* portant le vecteur de poids $\boldsymbol{\mu}_{j^*}$ le plus proche de $\mathbf{x}$, puis à le mettre à jour, lui ainsi que ses voisins sur la grille. On peut, soit définir un voisinage décroissant dans le temps, soit mettre à jour tous les poids de tous les nœuds, à chaque itération, par une quantité fonction décroissante de la distance au nœud j^*. Dans l'algorithme ci-dessous, c'est la seconde solution qui a été retenue.

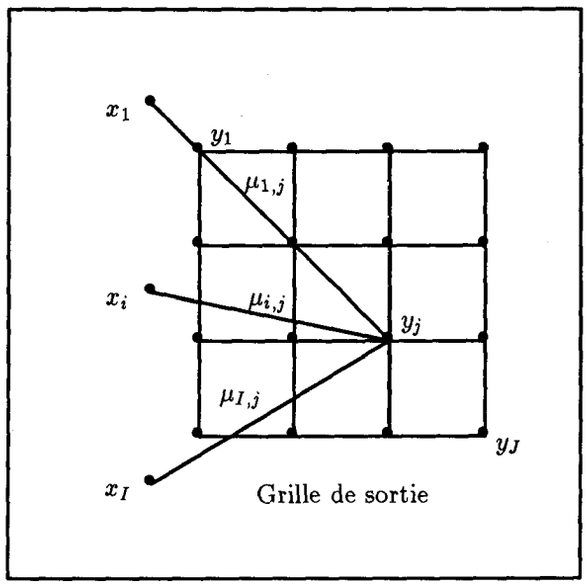


Fig. 3.1. — Réseau de Kohonen.

3.2. Algorithme d'évolution du système

L'ensemble des opérations peut se décomposer en quatre temps :

1. Initialisation des poids μ_{ij}, $i \in [1, I]$, $j \in [1, J]$, à de petites valeurs aléatoires.
2. Prendre une nouvelle entrée $\mathbf{x}^n$, et calculer sa distance à chacun des vecteurs $\boldsymbol{\mu}_j^n$, $j \in [1, J]$.
3. Sélectionner le nœud j^* le plus proche de l'entrée.
4. Mettre à jour les poids suivant la formule

$$\mu_{ij}^{n+1} = \mu_{ij}^n + \varepsilon_n h_n^{j, j^*} (x_i^n - \mu_{ij}^n)$$

$$i \in [1, I], \qquad j \in [1, J]$$

$$\varepsilon_n = \varepsilon_i \left(\frac{\varepsilon_f}{\varepsilon_i} \right)^{n/n_{\max}}$$

$$h_n^{j, j^*} = \exp - \frac{\| \mathbf{j} - \mathbf{j}^* \|^2}{2 \sigma_n^2}$$

$$\sigma_n = \sigma_i \left(\frac{\sigma_f}{\sigma_i} \right)^{n/n_{\max}}.$$

Dans cette expression, μ_{ij}^n représente le poids entre l'entrée i et la sortie j, à l'itération n. Le nœud j^* est celui qui, au temps n, est le plus proche de l'entrée $\mathbf{x}^n = (x_1^n, \ldots, x_I^n)^T$, au sens de la distance euclidienne. j^* vérifie donc

$$\| \boldsymbol{\mu}_{j^*}^n - \mathbf{x}^n \| = \min_{j \in [1, J]} \| \boldsymbol{\mu}_j^n - \mathbf{x}^n \|.$$

La fonction $h_n^{j,j*}$ définit l'étendue de l'influence du point j^* sur les points j, et décroît avec la distance entre les positions **j** et **j*** de ces nœuds dans la grille de sortie. Cette dernière est organisée en carré de dimensions $\sqrt{J} \times \sqrt{J}$. Quant à ε_n, il définit l'amplitude des changements apportés aux poids à chaque itération.

Les paramètres ε_i, ε_f, σ_i et σ_f doivent être choisis de façon à ce que le système converge, ce qui comprend en fait deux phases :

1. Ordonnancement des poids $\boldsymbol{\mu}_j$.
2. Convergence des vecteurs poids vers leurs valeurs finales.

Il est possible de trouver des valeurs pour les paramètres permettant de remplir la deuxième condition sans que la première ne soit vérifiée. Comme il n'y a pas de méthode pour trouver des valeurs quasi optimales, on les a déterminées à la suite de plusieurs essais sur des entrées de lois de probabilité simples. Le fait empirique que le comportement du réseau de Kohonen ne soit pas perturbé par la nature de l'entrée justifie l'utilisation de ces paramètres quand on remplace ces entrées statistiquement simples par des images.

Note : on pourrait présenter le réseau de manière plus générale, comme une relation entre un espace d'entrées et un réseau d'unités ou neurones. Dans ce cas, l'espace de sortie peut être de dimension différente de l'espace d'entrée, et le choix du vecteur « le plus proche » devient celui du vecteur présentant la réponse maximale à l'entrée.

On voit que l'algorithme de Kohonen est une méthode de groupement plutôt qu'un quantificateur; toutefois il est facile d'obtenir un quantificateur en utilisant le dictionnaire, obtenu après entraînement sur une séquence d'images, pour coder d'autres images.

3.3. Quelques données théoriques

Très peu de résultats théoriques ont été démontrés sur le réseau de Kohonen. On peut citer l'ouvrage de référence de Kohonen [14], mais les résultats auxquels il aboutit restent très intuitifs, et sont déduits à partir de situations particulièrement simples (entrée scalaire de distribution uniforme) et de données expérimentales.

D'autres résultats ont été établis par H. Ritter et K. Schulten [15], dans le cas d'entrées de loi de probabilité discrète. Considérant l'algorithme comme un processus de Markov, ils montrent que l'on peut lui associer un potentiel qui, en moyenne, décroît. Ils insistent aussi sur le fait que ce potentiel présente de nombreux minimums locaux, ce qui indique que les différents paramètres du système ne peuvent être choisis au hasard si l'on veut atteindre un état satisfaisant de convergence.

Ils ont également étudié [16] le comportement du système en se plaçant dans un état stationnaire, en tirant des conclusions dans quelques cas simples. Ils aboutissent à une équation analytique décrivant le système à l'équilibre, mais trop complexe pour pouvoir être résolue dans le cas général. Dans le cas d'une entrée scalaire et de nœuds disposés en ligne, ils montrent que la répartition des poids n'est pas proportionnelle à la probabilité de l'entrée, ainsi que supposé par Kohonen, mais proportionnelle à celle-ci à la puissance 2/3.

On se place dans le cas où l'entrée x est scalaire, et possède une densité de probabilité $p(x)$ sur un support $[a, b]$. On suppose que les μ_j, $j \in [1, N]$, sont déjà ordonnés, ce qui n'enlève rien à la généralité des résultats évoqués ci-dessous comme on le verra un peu plus loin. On dispose les N nœuds en ligne, et on utilise l'algorithme tel que présenté par Kohonen [14], où, au lieu de changer tous les poids avec des amplitudes différentes, on ne modifie que ceux appartenant à un voisinage donné du point le plus proche de l'entrée.

A l'itération n, on a $\mu_i^{n+1} = \mu_i^n + \alpha_n (x^n - \mu_i^n)$, avec $0 < \alpha_n < 1$, $\alpha_n \searrow 0$, et $i \in N_{i*} = \{\max(1, i^* - 1), i^*, \min(N, i^* + 1)\}$, i^* repérant le nœud le plus proche de x^n.

On peut montrer que si i = i*, alors

(i) $d(\mu_i^{n+1}, \mu_{i-1}^{n+1}) \leqq d(\mu_i^n, \mu_{i-1}^n)$,

(ii) $d(\mu_i^{n+1}, \mu_{i+1}^{n+1}) \leqq d(\mu_i^n, \mu_{i+1}^n)$.

Il suffit de montrer (i), le problème (ii) étant similaire. On suppose l'ordre croissant, d'où

$$d(\mu_i^n, \mu_{i-1}^n) = \mu_i^n - \mu_{i-1}^n.$$

Donc

$$\mu_i^{n+1} - \mu_{i-1}^{n+1} = \mu_i^n - \mu_{i-1}^n + \alpha_n (\mu_{i-1}^n - \mu_i^n).$$

Le premier membre est positif ou nul puisque $0 < \alpha_n < 1$. Cette relation montre :

1. que l'ordre ne peut être perturbé, une fois établi, quand $n \to \infty$;
2. que $\mu_i^{n+1} - \mu_{i-1}^{n+1} = (1 - \alpha_n)(\mu_i^n - \mu_{i-1}^n)$. Comme $\alpha_n \to 0$, on atteint un état stable quand $n \to \infty$.

Autre résultat : les μ_j^n sont bornés.

On part de $\mu_1^n \leqq \ldots \leqq \mu_N^n$. Il suffit de montrer que μ_N^n est borné. Une condition suffisante est de prendre $\mu_N^0 \in [a, b]$ et $\mu_1^0 \in [a, b]$. μ_N^n ne peut augmenter que si $x_n \in [\mu_N^n, b]$. Mais μ_N^{n+1} se trouve entre μ_N^n et x_n, donc reste plus petit que b. De la même façon on montrerait que μ_1^n est toujours supérieur ou égal à a.

Les μ_j^n convergent donc à l'intérieur de $[a, b]$.

Car pour $i \in V_{i*}^n$,

$$|\mu_i^{n+1} - \mu_i^n| = \alpha_n |x_n - \mu_i^n| \underset{n \to \infty}{\to} 0,$$

et comme les valeurs sont bornées, avec les hypothèses faites plus haut tous les μ_i^n convergent dans l'intervalle $[a, b]$.

Une façon d'interpréter l'équation d'évolution du réseau dans sa forme générale présentée plus haut est de remarquer que c'est une équation différentielle du premier ordre non linéaire avec des coefficients aléatoires. La différence $(x_i^n - \mu_{ij}^n)$ correspond à une estimation de la variable aléatoire x_i^n par le terme μ_{ij}^n, et on ajoute au vecteur $\boldsymbol{\mu}_j^n$ un vecteur estimé, multiplié par une séquence α_n dont le rôle est de rendre les variations progressives. De ce point de vue, on s'attend donc à ce que chaque vecteur $\boldsymbol{\mu}_j^n$ tende vers

l'espérance mathématique de $\mathbf{x}^n$ prise sur l'intervalle S_j^n, défini comme l'ensemble des vecteurs $\mathbf{x}^n$ susceptibles de modifier $\boldsymbol{\mu}_j^n$. Si on note $E_j^n = E(\mathbf{x}/\mathbf{x} \in S_j^n)$, alors on aimerait montrer que

$$\lim_{n \to \infty} \boldsymbol{\mu}_j^n = \lim_{n \to \infty} E_j^n.$$

La principale difficulté, quand on essaie de trouver la valeur limite des poids, est que les théorèmes classiques de régression non linéaire ne semblent pas applicables [17]. En effet, prenons comme modèle de l'algorithme celui utilisé par H. Ritter et K. Schulten [15], c'est-à-dire celui que nous avons choisi pour les expériences. Ce qu'il faut noter, c'est qu'à chaque itération, le poids gagnant est choisi aléatoirement, et que, selon sa valeur, les modifications apportées à chacun des poids seront plus ou moins importantes. On a donc, pour chaque poids, un ensemble de N fonctions distinctes (N désignant le nombre de poids), chaque fonction correspondant à une valeur donnée du poids gagnant. Et, à chaque itération, l'une de ces fonctions s'applique aléatoirement au poids. Les théorèmes nous permettraient de conclure si la façon d'appliquer ces fonctions était déterministe, ce qui n'est pas le cas.

3.4. Propriétés importantes du réseau de Kohonen

1. La fonction de densité de points des vecteurs $\boldsymbol{\mu}_j$ va tendre vers la densité de probabilité $p(\mathbf{x})$ de l'entrée, et les $\boldsymbol{\mu}_j$ vont s'ordonner en fonction de leur similarité mutuelle.

2. Dans le cas où on se restreint à x scalaire, avec N sorties et un voisinage

$$V_{i*} = \{\max(1, i^* - 1), i^*, \min(N, i^* + 1)\},$$

alors, en partant de $\mu_j(0)$ aléatoires, les $\mu_j(n)$ vont s'ordonner de façon croissante ou décroissante. Une fois que l'ordre est établi, il est maintenu quel que soit n.

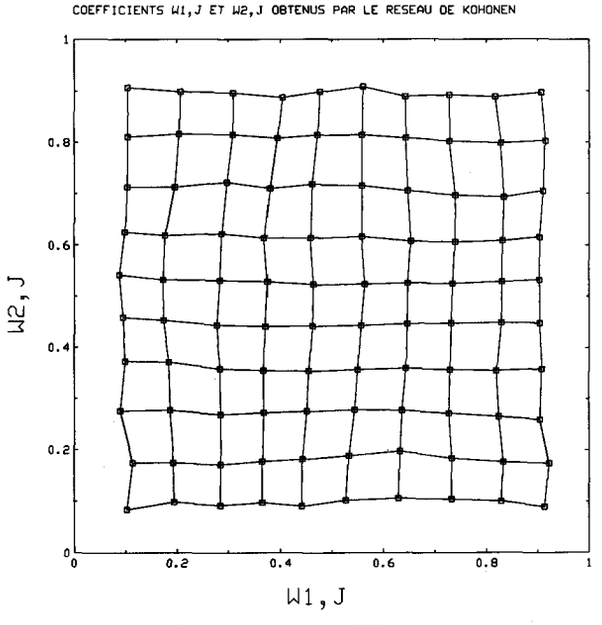


Fig. 3.2. — Réponse du réseau de Kohonen à une entrée de loi de probabilité uniforme. 10^4 itérations; 100 points.

Bien sûr, dans le cas 2, 1 est toujours vérifié, ce qui signifie que si l'on considère les $\mu_j(n)$ sur l'axe des réels, leur distribution va s'établir en suivant la loi de probabilité $p(x)$ de x. Comme illustration de ces propriétés, on se reportera à la figure 3.2, obtenue avec une entrée bidimensionnelle de loi de probabilité uniforme sur $[0, 1] \times [0, 1]$. Sur cette figure, on a relié entre eux les points voisins dans la grille de sortie.

Note : la propriété 1 a été énoncée par Kohonen, mais elle est surtout le résultat d'expériences répétées que de démonstrations formelles [14], et ne semble vraie que dans certains cas précis [16].

4. Résultats expérimentaux

4.1. Démarche adoptée

Les images utilisées sont du type de celles de vidéoconférence, en noir et blanc, de format 512×424 pixels, chaque pixel étant codé sur 8 bits, entre 0 et 255. On vérifie dans un premier temps que les valeurs de la luminance s'étendent effectivement de 0 à 255. Si cela n'avait pas été le cas, on aurait pu, pour un codage optimal, commencer par réduire le nombre de bits nécessaires pour chaque pixel. Lors de l'entraînement, chaque image est découpée en blocs carrés, de 4 à 49 pixels, qui constitueront les vecteurs d'entrée. Chaque bloc est ensuite dirigé vers l'un des cinq réseaux qui correspondent à cinq types de blocs :

- type 1 : frontières horizontales;
- type 2 : frontières verticales;
- type 3 : frontières diagonales selon la première bissectrice d'un repère classique $(0, x, y)$;
- type 4 : frontières diagonales selon la seconde bissectrice;
- type 5 : blocs uniformes.

Pour déterminer le type d'un bloc, on calcule quatre gradients associés aux quatre directions horizontale, verticale et diagonales. Si le gradient le plus élevé est inférieur à un seuil, égal à 5, le bloc est dit uniforme. Sinon, il est du type correspondant au plus fort gradient.

L'intérêt d'employer plusieurs réseaux est d'augmenter la qualité de la restitution en regroupant dans une même classe tous les vecteurs qui sont du même type, c'est-à-dire qui conduiront à des vecteurs poids semblables.

Plusieurs paramètres sont modifiables :

- les paramètres du réseau, fixés au cours des tests aux valeurs suivantes : $\varepsilon_i = 1$, $\varepsilon_f = 5.10^{-4}$, $\sigma_i = 1$, $\sigma_f = 10^{-2}$. Ce sont les valeurs qui ont été trouvées optimales au cours des essais effectués avec des entrées limitées à des variables aléatoires de densités connues, et on les utilise ici en arguant du fait que ces valeurs ne dépendent que peu du type d'entrée;
- la taille des blocs, entre 2×2 et 7×7;
- le nombre initial d'itérations désiré;
- le nombre initial de vecteurs pour chaque réseau.

Remarques

Premièrement, en choisissant un nombre d'itérations fini, on fait une approximation en supposant que la

convergence est atteinte, approximation négligeable si le nombre d'itérations est convenablement fixé.

Deuxièmement, on fait l'hypothèse qu'en changeant le nombre de vecteurs d'un des réseaux de Kohonen, les paramètres ε et σ sont encore valables. Elle est fondée sur la constatation que ces paramètres ont conduit à de très bons résultats dans le cas d'entrées de densités de probabilité simples, pour des grilles de tailles comprises entre 9 et 225 points.

Troisième remarque, sur le choix des divers types de blocs : si l'on voulait faire une classification complète comme dans des procédés classiques de quantification, on serait obligé de définir des types en nombre croissant avec la taille des blocs, conduisant rapidement à une grande complexité d'implémentation peu compatible avec le but initial, voir si l'utilisation de réseaux de Kohonen permet un gain par rapport aux méthodes classiques, gain pouvant être du domaine de la facilité de mise en œuvre. Ainsi, on renonce à des méthodes de classification trop lourdes, et on choisit cinq types de blocs qui ont l'avantage d'être compatibles avec une taille variable de blocs.

L'essentiel des tests a été mené en utilisant cinq classes de blocs de 2×2. Avec plus de classes et des blocs plus grands, les résultats seraient meilleurs, particulièrement en ce qui concerne le nombre de bits par pixel. Cependant, le but n'est pas ici de réaliser un seul dictionnaire d'excellente qualité, mais de comparer l'influence de divers paramètres à l'aide de plusieurs dictionnaires relativement petits.

On trouvera plus de résultats numériques dans [18], ceux reproduits plus bas étant néanmoins les plus représentatifs.

Les critères de qualité pour les images codées sont de trois ordres :

1. Pour chaque image reconstruite, on calcule, pour chaque type de blocs, une erreur quadratique moyenne (EQM) par pixel selon la formule suivante :

$$\mathrm{EQM}_i = \frac{1}{\mathrm{TN}_i} \sum_{k=1}^{\mathrm{N}_i} (\| \mathbf{x}_k - q_i(\mathbf{x}_k) \|)^2,$$

où $\mathbf{x}$ représente un bloc, T le nombre de pixels par bloc, $1 \leqq i \leqq 5$, i désigne le type du bloc, q_i le réseau de Kohonen associé, et N_i le nombre de blocs de type i. On aura l'erreur quadratique moyenne totale EQM en calculant

$$\mathrm{EQM} = \frac{1}{\mathrm{N}} \sum_{i=1}^{5} \mathrm{N}_i \, \mathrm{EQM}_i.$$

2. Parallèlement, on calcule un rapport signal sur bruit de la façon suivante :

$$\mathrm{MSNR} = 10 \log \frac{m^2}{\mathrm{EQM}}.$$

Le MSNR (Mean Signal To Noise Ratio) est calculé par rapport à la valeur moyenne de la luminance, m, ici égale à 130. Il est donc moins élevé que le PSNR (Peak SNR), d'environ 6 dB pour les images utilisées, le PSNR étant calculé par rapport à la valeur maximale de la luminance.

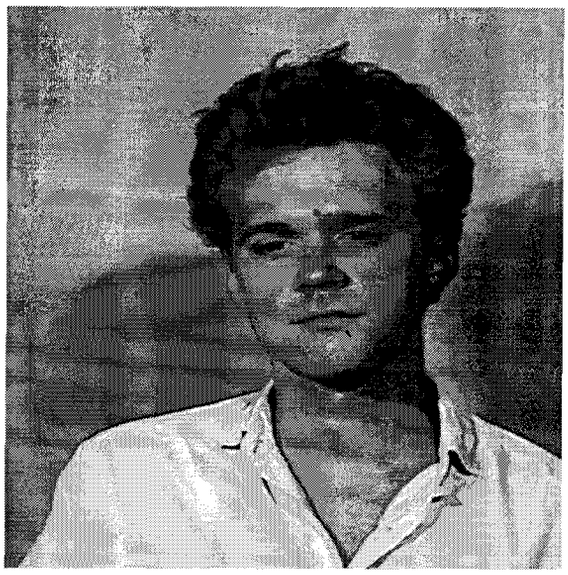


Fig. 4.1. — JPC; image originale.

3. Enfin, on a tenu compte de la qualité visuelle des images. Quant au nombre de bits par pixel, il est calculé suivant la formule

$$\frac{\log_2 \text{nombre de vecteurs}}{\text{nombre de pixels par bloc}}.$$

En divisant par 8, on obtient le taux de compression.

4.2. Première série de tests

4.2.1. *Variation de la taille des blocs*

Dans cette section, on utilise une seule image, JPC, pour évaluer l'influence du nombre de vecteurs par classe et celle de la taille des blocs sur la qualité de restitution. Procédant ainsi, on est conscient du fait que les performances seront meilleures à celles qui auraient été obtenues en utilisant plusieurs images et/ou en codant une image ne faisant pas partie de la séquence d'entraînement. Mais le but est seulement de comparer les résultats en faisant varier les paramètres cités ci-dessus. Dans les sections suivantes, on fera des tests plus poussés pour la ou les combinaisons taille de blocs/nombre de vecteurs par classe les plus intéressantes.

On note J le nombre de vecteurs pour chacune des cinq classes, et P, qui est un multiple du nombre de blocs par image, indique le nombre d'itérations. Pour que les comparaisons entre plusieurs tailles de blocs soient plus licites, on prendra P plus élevé pour des grands blocs. En effet, pour un même P, les petits blocs seront beaucoup plus proches de la convergence, à cause du nombre plus élevé d'itérations. On fait varier J entre 4 et 100, ce qui correspond à des dictionnaires de taille 20 à 500. La taille des blocs est prise égale à 2×2, 3×3, 5×5 et 7×7.

Les résultats permettent de mettre en évidence plusieurs phénomènes :

1. Dans le cas des blocs de 2×2, on note le fait suivant, moins évident dans les autres cas : pour un nombre d'itérations donné, les performances n'évo-

TABLEAU 4.I

J	Itérations	Blocs EQM	EQM_i	Bits/pel MSNR	J	Itérations	Blocs EQM	EQM_i	Bits/pel MSNR
4	54272 (P=1)	2×2 234,2	465,2 936,4 165,9 199,0 258,0	1,08 18,4 dB	4	108544 (P=2) *Fig.* 4.5	2×2 104,6	103,6 145,9 156,4 141,8 65,3	1,08 21,9 dB
100	108544	2×2 12,9	20,7 65,1 21,9 20,4 4,5	2,24 31,0 dB	100	217088 (P=4) *Fig.* 4.2	2×2 8,0	10,2 26,6 15,2 14,1 1,9	2,24 33,1 dB
4	48242 (P=2)	3×3 352,6	476,1 435,9 460,5 407,7 142,2	0,48 16,6 dB	4	96484 (P=4)	3×3 131,2	155,6 142,7 151,1 268,5 45,0	0,48 21,0 dB
100	48242	3×3 26,0	32,1 28,8 43,6 37,4 8,6	0,99 28,0 dB	100	96484 *Fig* 4.3	3×3 14,1	18,7 17,9 23,0 19,7 2,7	0,99 30,6 dB
16	34736 (P=4)	5×5 55,3	61,3 51,5 61,0 57,3 12,8	0,25 24,7 dB	16	69472 (P=8)	5×5 51,3	57,6 47,7 58,5 54,9 6,9	0,25 25,0 dB
100	34736	5×5 32,1	31,6 30,1 39,3 35,8 5,5	0,36 27,1 dB	16	26586 (P=6)	7×7 71,8	75,7 69,0 87,9 56,5 54,8	0,13 23,6 dB
16	44310 (P=10)	7×7 71,6	77,4 70,2 76,6 62,6 30,5	0,13 23,6 dB	100	44310 *Fig.* 4.4	7×7 32,1	33,4 30,9 36,5 28,3 30,4	0,18 27,1 dB

luent plus à partir d'une certaine valeur de J. Donc, si l'on veut une excellente performance, il ne suffit pas d'augmenter J, il faut augmenter le nombre d'itérations. Si on se fixe une valeur seuil pour la qualité, on peut ainsi choisir le J minimum correspondant, en sachant que choisir des valeurs plus élevées pour J ne conduirait qu'à diminuer le taux de compression et à augmenter le coût de stockage des dictionnaires. On pourrait ainsi établir des tables de correspondance, pour chaque taille de blocs, entre le nombre de passages P et la valeur de J au-delà de laquelle on n'observe plus aucune amélioration.

2. Aussi bien pour le 5×5 que pour le 7×7, les performances, bien que la convergence semble atteinte, ne sont pas très bonnes. Ceci peut signifier au moins deux choses, soit qu'on ne puisse obtenir

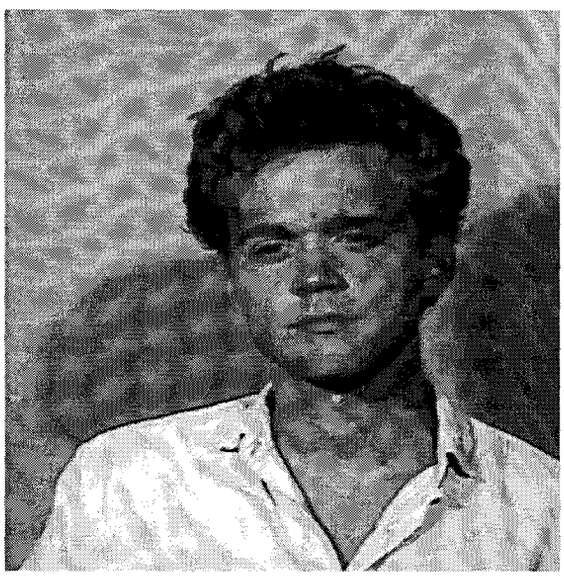


Fig. 4.2. — Blocs 2 × 2; J = 100; 5 classes; P = 4; JPC.

Fig. 4.3. — Blocs 3 × 3; J = 100; 5 classes; P = 4; JPC.

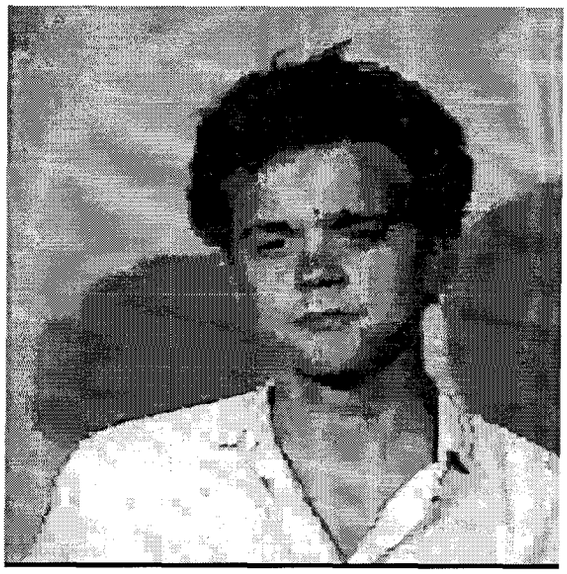


Fig. 4.4. — Blocs 7 × 7; J = 100; 5 classes; P = 10.

Fig. 4.5. — Blocs 2 × 2; J = 4; 5 classes; P = 2.

de bons résultats avec de gros blocs, soit, ce qui est plus probable, que le fait de n'avoir que cinq types de blocs se fait d'autant plus sentir que les blocs sont grands. Avec de grands blocs, la classification en cinq classes devient tout à fait insuffisante pour rendre compte de la diversité des blocs possibles. Dans le tableau 4.1, où ne sont reportés que quelques essais significatifs, les valeurs de l'EQM sont données pour chaque type de blocs, de 1 à 5.

3. Les performances diminuent quand la taille du bloc augmente, l'effet de bloc devenant en outre de plus en plus sensible. Encore assez peu perceptible pour le 3 × 3 (*fig.* 4.3), il nuit à la qualité pour le 5 × 5 et le 7 × 7 (*fig.* 4.4).

4. Les taux de compression sont très satisfaisants, même pour le 2 × 2 où on reste en deçà de 2,3 bits/pel.

5. Dans le tableau 4.1, on remarque que l'EQM des blocs du type 5 (uniformes) est bien inférieure à celle des autres. Cela provient du fait que les images choisies présentent simultanément des contrastes importants et de grandes zones uniformes, où les risques d'erreurs de classification sont très faibles.

Il n'est pas possible d'étudier chaque cas en détail, c'est pourquoi on se limite dans la suite à une seule taille de blocs, le 2 × 2, qui a fourni les meilleurs résultats (*fig.* 4.2, à comparer à l'original, *fig.* 4.1). Mais, au vu des expériences précédentes, il semble que des blocs plus gros soient aussi intéressants, surtout si l'on applique un algorithme de lissage pour atténuer la gêne causée par l'effet de bloc.

Noter enfin que l'image obtenue pour J = 4, des blocs 2 × 2, cinq classes et P = 2 est, malgré un dictionnaire de taille 20 seulement, tout à fait reconnaissable (*fig.* 4.5).

4.2.2. *Variation du nombre de classes*

On se limite désormais à des blocs de 2 × 2, puisqu'ils ont fourni les meilleurs résultats. On recommence les tests avec trois types au lieu de cinq, les types diagonaux étant supprimés.

TABLEAU 4.2

Trois types de blocs de 2 × 2. *P* = 2

J	EQM_i	Bits/pel MSNR	EQM
4	161,6 250,7 67,2	0,90 21,6 dB	113,6
16	50,4 56,4 10,2	1,39 28,1 dB	25,3
100	24,2 30,5 4,7	2,06 31,2 dB	12,5

On peut faire plusieurs remarques sur les résultats (tableau 4.2). Tout d'abord, la convergence est plus rapide avec trois classes qu'avec cinq. Si l'on est limité en temps de calcul, ce peut être un facteur déterminant. En outre, pour P = 4 et cinq classes, les temps de calcul deviennent très importants si l'on veut de très bonnes performances. On ne fera donc pas de tests avec plus de cinq classes.

Les résultats obtenus montrent que l'utilisation de trois classes conduit à de meilleurs résultats, pour P inférieur ou égal à 4. Cela est dû au fait que la convergence est atteinte plus rapidement pour trois classes. Par conséquent, plus le nombre d'itérations augmente, plus l'utilisation de cinq classes au lieu de trois est intéressante, la différence dans le nombre de vecteurs expliquant le gain en qualité. Remarquons également qu'à paramètres égaux, la qualité visuelle des images codées avec cinq classes est supérieure.

4.2.3. *Essais avec plusieurs images*

Dans cette section, on entraîne le réseau avec quatre images, chacune étant lue une fois, ce qui correspond à 217 088 itérations. J est pris égal à 100, et, pour les raisons données dans la section précédente, on choisit d'utiliser cinq classes de blocs. Un avantage supplémentaire est une meilleure restitution pour des images ne faisant pas partie de la séquence d'entraînement

TABLEAU 4.3

Entraînement avec 4 images.
Codage d'une image extérieure. P=4.

J	EQM_i	Bits/pel MSNR	EQM
64	17,3	2,08	24,3
	54,7		
(5 classes)	32,2	28,3 dB	
	32,0		
	17,1		
100	17,3	2,24	21,1
	54,7		
(5 classes)	24,4	28,9 dB	
	26,1		
	17,0		
64	33,7	1,90	21,4
	32,2		
(3 classes)	14,9	28,8 dB	
100	29,4	2,06	21,9
	36,7		
(3 classes)	16,2	28,7 dB	

(tableau 4.3). Cette propriété provient sans doute de la plus grande taille du dictionnaire.

Les images qui n'appartiennent pas à la séquence d'entraînement (JPC, DTT et LLL) donnent des résultats aussi bons que les autres (DED, HLN, MRD et RDL). Les performances inférieures obtenues avec JPC sont sans doute dues au fait que cette image présente des nuances très fines de blanc, absentes de celles de la séquence. Dans le cas de DED (*fig.* 4.6), il est pratiquement impossible, à l'œil nu, de distinguer l'image codée de l'originale.

Fig. 4.6. — Blocs 2 × 2; J = 100; 5 classes; P = 4; DED.

Dans le tableau 4.4, les résultats sont donnés pour P = 8. Par rapport à P = 4, les gains varient de 0,3 à 1,1 dB, les plus élevés correspondant aux images de moindre MSNR.

4.2.4. *Influence du dictionnaire initial*

Il semble qu'elle ne soit déterminante en ce qui concerne les performances atteintes. En effet, si, réinitialisant les valeurs du système, on prend comme dictionnaire initial l'un de ceux établis à partir de quatre images, avec P = 4, et qu'on entraîne le système avec trois images, différentes des quatre premières, on obtient des résultats similaires à ceux obtenus en partant d'un dictionnaire aléatoire (tableau 4.5). Prendre un dictionnaire initial aléatoire ne fait que ralentir la convergence, mais n'empêche pas cette dernière. Cependant, il est probable que cette indépendance apparente ne soit vraie qu'à l'intérieur d'un domaine de convergence à définir.

TABLEAU 4.4

Image	EQM_i	Bits/pel	MSNR	EQM
DED	6,4	2,24	34,3 dB	6,1
	9,3	*Fig.* 4.6		
	10,0			
	10,4			
	1,3			
HLN	10,5	2,24	32,4 dB	9,3
	18,9			
	18,3			
	18,4			
	1,2			
MRD	9,1	2,24	32,9 dB	8,4
	16,3			
	13,9			
	14,2			
	1,5			
RDL	6,0	2,24	33,5 dB	7,3
	10,9			
	13,1			
	12,9			
	1,3			
JPC	26,3	2,24	30,0 dB	16,4
	31,4			
	21,2			
	23,4			
	10,8			
DTT	9,3	2,24	32,8 dB	8,6
	27,8			
	15,6			
	15,4			
	1,9			
LLL	10,2	2,24	33,5 dB	7,3
	15,9			
	13,4			
	12,2			
	1,9			

TABLEAU 4.5

P = 3. Blocs 2 × 2. 5 classes.

J	EQM	EQM_i	Bits/pel MSNR	Dictionnaire initial
64	13,5	16,2	2,08	Non aléatoire
		26,3		
		23,0	30,8 dB	
		19,9		
		6,3		
64	15,6	23,6	2,08	Aléatoire
		47,6		
		23,1	30,2 dB	
		21,0		
		9,0		
100	11,0	15,3	2,24	Non aléatoire
		26,3		
		18,6	31,7 dB	
		16,8		
		4,8		
100	13,7	23,6	2,24	Aléatoire
		47,6		
		20,0	30,8 dB	
		17,8		
		8,2		

4.2.5. *Comparaison avec la méthode Isodata*

Pour comparer avec Isodata, on élabore un dictionnaire initial en utilisant la méthode dite par « splitting » présentée par Linde, Buzo et Gray [13]. On commence donc par déterminer, pour chacune des cinq classes, le barycentre des vecteurs constituant la séquence d'entrée. On passe ensuite à un dictionnaire de dimension 2 en ajoutant ou retranchant un même vecteur de perturbation à chaque barycentre. On recalcule les barycentres des vecteurs associés à chacun des deux nouveaux vecteurs en utilisant la règle du plus proche voisin, et on peut de nouveau diviser chaque barycentre en deux nouveaux vecteurs. D'autres essais, effectués en constituant le dictionnaire initial à partir de vecteurs pris dans la séquence d'entraînement, donnent des résultats tout à fait mauvais, avec un MSNR de l'ordre de 11 dB. Par contre, en utilisant le « splitting », avec 64 vecteurs par classe, 5 classes et des blocs de 2×2, on obtient de bien meilleurs résultats (tableau 4.6). Plusieurs remarques s'imposent : tout d'abord, pour le codage d'une image appartenant à la séquence d'entraînement, les performances entre les deux méthodes, Kohonen et Isodata, sont en faveur de la première, qui fournit des images plus nettes, aux contours plus précis (comparer les figures 4.7 et 4.8). De plus, le codage d'une image extérieure à la séquence semble être meilleur en utilisant Kohonen (tableaux 4.6 et 4.4). Les images servant à l'entraînement étant les mêmes que dans la section 4.2.3, le codage de JPC, bien que cette fois la convergence soit complète (résultats identiques pour $P = 16$ et $P = 32$), ne donne plus que 28,2 dB, au lieu de 30. Ajoutant à ce qui précède que l'obtention du dictionnaire initial est immédiate si l'on choisit d'utiliser Kohonen, cette dernière technique est préférable à Isodata sur les tests effectués.

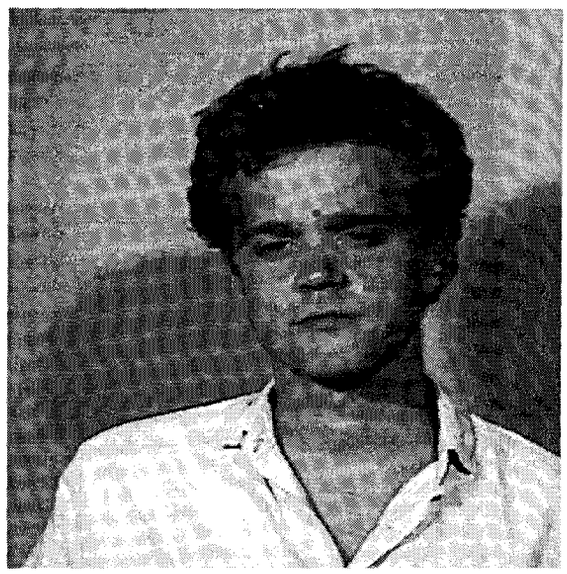


Fig. 4.7. — Isodata. Blocs 2×2. $J = 64$. 5 classes. $P = 10$. Dictionnaire initial par splitting. MSNR = 30,8 dB.

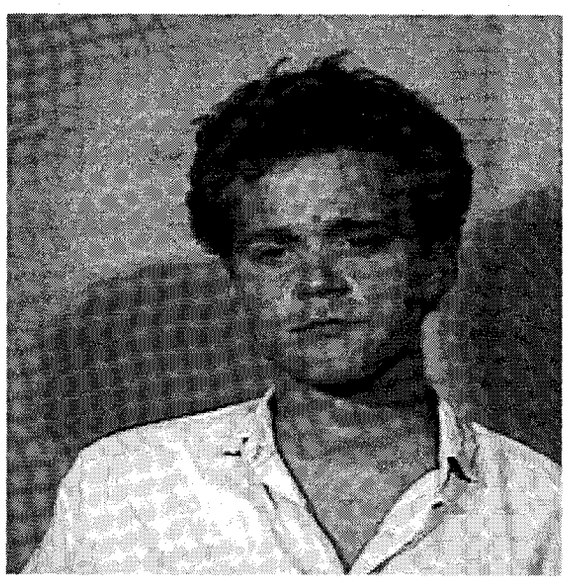


Fig. 4.8. — Kohonen. Blocs 2×2. $J = 64$. 5 classes. $P = 4$. MSNR = 32,4 dB.

TABLEAU 4.6

Algorithme d'Isodata. Image codée : JPC.

EQM_i	Bits/pel P	MSNR	EQM	Images servant à élaborer le dictionnaire
6,9 8,4 30,0 25,4 1,4	2,08 10	30,8 dB	13,4	JPC *Fig.* 4.9
17,8 23,9 47,7 46,8 95,0	2,08 8	23,6 dB	71,5	DED HLN MRD RDL
10,9 12,0 36,8 31,0 17,7	2,08 16	28,2 dB	24,9	DED HLN MRD RDL

Conclusion

Le système étudié pour quantifier vectoriellement des images à l'aide de plusieurs réseaux de Kohonen en parallèle permet de tirer plusieurs conclusions.

En premier lieu, les performances se sont révélées insuffisantes pour des blocs de taille supérieure à 3×3 pixels, les tests effectués pour différentes valeurs de P montrant que l'on était arrivé près des valeurs limites. Cela est probablement dû au fait que l'on s'est restreint à cinq types distincts de blocs, un nombre sans doute insuffisant pour tenir compte de la diversité des types possibles pour des blocs comprenant beaucoup de pixels. Le taux de compression augmentant simultanément à la taille du bloc, il s'avérerait sans aucun doute intéressant de tenter des classifications plus fines. En augmentant la taille du bloc, on peut de plus se permettre d'utiliser un nombre élevé de vecteurs tout en restant en deçà de 1 bit/pel. Avec des blocs de 16 pixels et plus de classes, 20 ou 30 par exemple, on devrait atteindre des résultats

extrêmement satisfaisants, tout en conservant moins de 1 bit/pel aussi longtemps que le nombre de vecteurs est inférieur à 2^{16}.

En second lieu, pour les blocs ayant donné les meilleurs résultats, ceux de 4 pixels, on a vu qu'à un nombre d'itérations donné correspond une performance maximale, atteinte pour une certaine valeur de J, et qu'aucune amélioration n'est apportée en prenant un J supérieur.

Ensuite, toujours pour des blocs de 2×2 pixels, et dans le cas où on code une image ayant servi à l'entraînement, les données numériques donnent l'avantage aux images codées en utilisant trois classes seulement au lieu de cinq, mais, visuellement, celles codées avec cinq classes sont plus fidèles à l'original.

Le choix du dictionnaire initial ne semble pas influer sur les valeurs finales, du moins à l'intérieur d'une zone de convergence qui resterait à définir.

Comparé à la méthode Isodata, le réseau de Kohonen a produit des images plus nettes, et s'est montré meilleur dans le cas de la restitution d'images extérieures à la séquence d'entraînement.

L'avantage de ce système est sa simplicité de mise en œuvre, la faible taille du dictionnaire assurant un faible coût de stockage, ainsi que des taux de compression intéressants, correspondant toujours à moins de 2,3 bits/pixel dans les tests effectués. De plus, le dictionnaire initial peut être choisi aléatoirement, à la différence d'autres méthodes de quantification vectorielle, où ce choix met en œuvre de nombreux calculs, et constitue un facteur déterminant pour la qualité du quantificateur final.

Quant aux performances, elles sont tout à fait satisfaisantes, aussi bien pour des images ayant servi à l'entraînement que pour d'autres, différentes, ce qui permet d'envisager d'utiliser le réseau de Kohonen pour quantifier des images. Les dictionnaires, obtenus à partir de quatre images, permettent de coder correctement n'importe quelle image du même type. Il serait intéressant de tester le système sur des images quelconques. Au besoin, on peut facilement introduire une nouvelle image dans le dictionnaire, sans pour autant être obligé de recommencer tout l'entraînement, puisque l'algorithme de Kohonen n'a pas besoin de toute la séquence à chaque itération, une caractéristique qui confère au système beaucoup d'adaptabilité.

Remerciements

Cette recherche a été rendue possible par les subventions A6022 et A4234 du CRSNG (Conseil de Recherches en Sciences Naturelles et en Génie du Canada). Nous remercions vivement Dr Biren Prasada pour ses commentaires, ses encouragements, et l'utilisation de fonds provenant de la subvention A6022 dont il est le bénéficiaire. Nous remercions également Gregory Onyszchuk pour les discussions que nous avons eues avec lui tout au long de ce travail.

Manuscrit reçu le 17 *avril* 1989.

BIBLIOGRAPHIE

[1] N. M. Nasrabadi et R. A. King, Image Coding Using Vector Quantization: A Review, *IEEE Transactions on Communications*, 36, n° 8, August 1988.

[2] G. M. Edelman et L. H. Finkel, Neuronal group selection in the cerebral cortex, *1st Symposium of the Neurosciences Institute*, La Lolla, California, October 3-8, 1982.

[3] A. Takeuchi et S. Amari, Formation of topographic maps and columnar microstructures, *Biol. Cybern.*, 35, 1979, p. 63-72.

[4] D. J. Willshaw et C. von der Malsburg, How patterned neural connections can set up by self-organization, *Proc. R. Soc.*, London Ser. B, 194, 1976, p. 431-445.

[5] D. J. Willshaw et C. von der Malsburg, A marker induction mechanism for the establishment of ordered neural mappings: its application to the retinotectal problem, *Proc. R. Soc.*, London Ser. B, 287, 1979, p. 203-243.

[6] S. Grossberg, On the development of feature detectors in the visual cortex with applications to learning and reaction-diffusion systems, *Biol. Cybern.*, 21, 1976, p. 145-159.

[7] S. Grossberg, Adaptative pattern classification and coding of neural feature detectors, *Biol. Cybern.*, 23, 1976, p. 121-134.

[8] R. Linsker, Self-Organization in a Perceptual Network, *Computer*, March 1988.

[9] S. P. Luttrell, Self-Organizing Multilayer Topographic Mappings, *IEEE International Conference on Neural Networks*, San Diego, California, July 24-27, 1988, p. 1-93.

[10] T. Kohonen, Self-organized Formation of Topologically Correct Feature Maps, *Biol. Cybern.*, 43, 1982, p. 59-69.

[11] T. Kohonen, Analysis of a Simple Self-organizing Process, *Biol. Cybern.*, 44, 1982, p. 135-140.

[12] N. M. Nasrabadi et Y. Feng, Vector Quantization of Images Based Upon the Kohonen Self-Organizing Feature Maps, *IEEE International Conference on Neural Networks*, San Diego California, July 24-27, 1988, p. I-101.

[13] Y. Linde, A. Buzo et R. M. Gray, An Algorithm for Vector Quantizer Design, *IEEE transactions on Communications*, Jan. 1980.

[14] T. Kohonen, *Self-organization and Associative Memory*, Springer-Verlag, Berlin, Heidelberg, 1984.

[15] H. Ritter et K. Schulten, Kohonen's Self-Organizing Maps: Exploring their Computational Capabilities, *IEEE International Conference on Neural Networks*, San Diego California, July 24-27, 1988, p. I-109.

[16] H. Ritter et K. Schulten, On the Stationary State of Kohonen's Self-Organizing Sensory Mapping, *Biol. Cybern.*, 54, 1986, p. 99-106.

[17] A. E. Albert et L. A. Gardner, *Stochastic Approximation and Nonlinear Regression*, MIT Press, 1967.

[18] E. Le Bail et A. Mitiche, *Quantification vectorielle d'images : utilisation du réseau neuronal de Kohonen*, INRS, rapport 89-06, Montréal, 1989.