Localisation et reconnaissance de visages en temps réels avec un réseau de neurones RBF : algorithme et architecture

Real Time Faces Localisation and Recognition with a RBF neural network : Algorithm and Architecture

par F. YANG, M. PAINDAVOINE, N. MALASNÉ

Laboratoire Le2i, Aile de l'Ingénieur – Mirande, Université de Bourgogne, BP400 – 21011 Dijon cedex. Tél : 03 80 39 68 48 Fax : 03 80 39 59 10
fanyang@u-bourgogne.fr
paindav@u-bourgogne.fr

résumé et mots clés

Cet article décrit un système de vision temps réel permettant de localiser des visages dans des séquences vidéo ainsi que de reconnaître leur identité. Ces processus sont effectués en combinant des techniques de traitements d'images et des méthodes de réseaux de neurones. La robustesse du système a été évaluée quantitativement sur un corpus de 8 séquences vidéo. Dans le but de comparer les performances avec les autres méthodes existantes, nous avons également testé notre modèle en utilisant la banque de visages standard ORL. Le système a aussi été implanté sur deux architectures électroniques à base de composants spécialisés ZISC et de FPGA. Nous analysons la complexité de l'algorithme et nous présentons les résultats des implantations architecturales en termes de ressources matérielles et de vitesse de traitement.

Localisation et reconnaissance de visages, réseaux de neurones RBF, implantation en temps réel, FPGA, processeur ZISC.

abstract and key words

This paper describes a real time vision system, allowing to localize faces in video sequences and to recognize their identity. These processes are based on combining techniques of image processing and neural network approach. The robustness of this system has been evaluated quantitatively on 8 video sequences. We have tested our model using the ORL database in order to compare performances with other systems. The system has also been implanted on electronic architectures based on dedicated chip ZISC and FPGA. We analyse the algorithm complexity and we present results of hardware implementations in terms of used resources and processing speed.

Faces localization and recognition, RBF neural networks, real-time implementation, FPGA device, ZISC chip.

1. introduction

La localisation et la reconnaissance de visages dans des séquences d'images rendent possibles de nombreuses applications, telles que l'interaction homme-machine, la surveillance, la vidéo-conférence et le développement d'outils perceptuels. Résoudre ces problèmes difficiles grâce à la vision par ordinateur apporte une souplesse inégalée à l'utilisateur.

Le traitement automatique d'images de visages a été abordé de multiples manières dans la littérature, par exemple à l'aide de réseaux de neurones artificiels [1] [2] [3] [4], de chaînes de Markov [5], de graphes déformables [6] ou d'« eigenfaces » [7] [8]. Certains chercheurs exploitent le facteur de couleurs ou de convexité. M. Rosenblum *et al.* [9] ont développé un système de reconnaissance d'expressions humaines basé sur l'analyse du mouvement en utilisant un réseau de neurones de type RBF (Radial Basis Function). A.J. O'Toole *et al.* [10] ont montré que les classifieurs de type RBF possèdent une grande capacité de généralisation pour reconnaître ou identifier des visages de 3/4 profil ou de profil. A.J. Howell et H. Buxton [11] ont comparé les performances de reconnaissance de visages d'un réseau RBF avec les autres systèmes en utilisant des images statiques et des séquences d'images. Le réseau RBF a toujours donné des résultats très satisfaisants dans les applications de reconnaissance de visages.

Notre but final est d'analyser en temps réel des séquences vidéo pour répondre aux questions suivantes : les personnes intéressantes se trouvent elles dans la scène ? Si la réponse est Oui, quelle est alors leur position à chaque instant ? Nous visons à réaliser un système de localisation et de reconnaissance de visages pour des applications grand public ou industrielles. Ce système doit être exploité dans des conditions générales : l'environnement (le fond, le décor, l'éclairage ...) est quelconque, les visages sont de profil ou de face. L'algorithme doit être suffisamment simple pour permettre le traitement en temps réel sur un système électronique. Les classifieurs de type RBF ont l'avantage de posséder une bonne capacité de généralisation et nécessitent un volume de calculs modéré. De tels calculs peuvent être implantés sur des architectures électroniques, massivement parallèles ce qui permet d'accélérer la vitesse de traitement. C'est pour ces raisons que nous avons décidé d'utiliser un réseau de neurones RBF.

Dans les sections 2 et 3, nous décrivons d'abord le principe des réseaux de type RBF et la structure du modèle utilisé. Ensuite, le système de localisation et de reconnaissance de visages est évalué quantitativement sur un corpus de 8 séquences vidéo avec de multiples configurations (réduction de la taille des données, utilisation de différentes mesures de distance et de différentes fonctions de réponse). La robustesse de la méthode est aussi testée avec le changement d'échelle dans la section 4. Nous avons appliqué notre modèle sur la banque de visages standard ORL afin de comparer les performances avec les autres systèmes. Les résultats sont présentés dans la section 5.

Nous abordons les réalisations matérielles du modèle en temps réel dans les sections 6, 7, 8. Ces implantations ont été effectuées sur deux systèmes électroniques du commerce à base de FPGA : la carte $Neurosight$ et la carte $MEMEC$. La première comporte des composants spécialisés ZISC, alors que la seconde est plus généraliste.

Nous analysons la complexité de l'algorithme et nous décrivons chaque implantation matérielle : l'architecture de la carte utilisée, les entrées-sorties, les mémoires et les opérations effectuées. Nous présentons les résultats en termes de ressources matérielles utilisées et de vitesse de traitement. En perspectives, nous proposons des améliorations éventuelles pour chaque architecture.

2. principe du réseau RBF

2.1. méthode, architecture et fonctionnement du réseau de type RBF

Les réseaux de type RBF [12] sont des réseaux à trois couches qui ont comme origine une technique d'interpolation. Cette technique s'avère être à la fois rapide et efficace, en particulier pour les applications de classification. Son principe est d'approximer un comportement désiré par une collection de fonctions (appelées fonctions-noyau). Les fonctions-noyau de la méthode RBF sont locales, c'est-à-dire qu'elles ne donnent des réponses utiles que dans un domaine d'influence délimité par un *seuil de distance*. Ce seuil est défini autour d'un point, le *noyau* (ou *centre*). La distance Euclidienne est généralement utilisée pour la mesure de distance :

$$d(\boldsymbol{x}) = ||\boldsymbol{x} - \boldsymbol{c}_i|| \tag{1}$$

Où $d(\boldsymbol{x})$ mesure la distance entre le vecteur $\boldsymbol{x}$ et le centre $\boldsymbol{c}_i$ de la fonction f_i. La fonction de la réponse la plus utilisée dans ce cadre est la Gaussienne :

$$f_i(\boldsymbol{x}) = \exp^{\frac{-d(\boldsymbol{x})^2}{\sigma_i^2}} \tag{2}$$

Ainsi, chaque fonction-noyau est décrite par deux paramètres : la position de son centre $\boldsymbol{c}_i$ et le seuil de distance σ_i.

Pour approximer un comportement donné, les fonctions-noyaux associées à une catégorie sont assemblées de façon à couvrir avec leurs domaines d'influence l'ensemble des vecteurs de cette catégorie. Ces fonctions sont ensuite pondérées puis sommées

pour produire une valeur de sortie. La figure 1 présente la méthode RBF sous la forme d'un réseau à trois couches : une couche d'entrée, une couche cachée composée des fonctions-noyaux, et une couche de sortie dont les neurones sont généralement animés par une fonction d'activation linéaire.

2.2. apprentissage du réseau RBF

L'apprentissage des réseaux RBF commence par la paramétrisation des fonctions-noyaux. Nous devons d'abord acquérir une collection d'exemples pour chaque catégorie. Certains exemples d'une catégorie peuvent alors être assez ressemblants. Dans ce cas, nous cherchons à les fusionner en déterminant un unique vecteur d'apprentissage les représentant. A chaque vecteur d'apprentissage $\boldsymbol{c}_i$ est associé un seuil de distance σ_i délimitant le domaine d'influence. Pour calculer les noyaux $(\boldsymbol{c}_i, \sigma_i)$, nous avons adapté la méthode de Musawi *et al.* [13]. Cette technique, appelée « convergent K-means clustering » nous permet d'obtenir une bonne capacité de généralisation et un temps d'apprentissage raisonnable. Nous décrivons ci-après les différentes étapes de cette technique :

Etape 0 Chaque catégorie est représentée par un certain nombre de vecteurs d'exemples. A chacun d'entre eux, on associe un seuil de distance nul.

Etape 1 Choisir aléatoirement un vecteur d'exemple $\boldsymbol{c}_k$ (initialement $\sigma_k = 0$).

Etape 2 Trouver, dans la même catégorie de ce vecteur choisi, le vecteur d'exemple $\boldsymbol{c}_l$ associé à son seuil de distance σ_l (initialement $\sigma_l = 0$) le plus proche de $\boldsymbol{c}_k$:

2.1 Calculer le barycentre $\boldsymbol{c}_i = \dfrac{\boldsymbol{c}_k + \boldsymbol{c}_l}{2}$.

2.2 A ce nouveau vecteur, nous associons un seuil de distance $\sigma_i = \dfrac{||\boldsymbol{c}_k + \boldsymbol{c}_l||}{2} + \sigma_l$.

2.3 calculer la distance D entre $\boldsymbol{c}_i$ et le vecteur le plus proche parmi les autres catégories.

2.4 si $D > \mu\sigma_i$, alors la fusion est acceptée, et le nouveau vecteur $\boldsymbol{c}_i$ remplace les vecteurs $\boldsymbol{c}_k$ et $\boldsymbol{c}_l$. Sinon, la fusion est rejetée. Nous gardons le vecteur $\boldsymbol{c}_k$ et nous lui associons le seuil de distance $\sigma_k = \dfrac{D'}{2}$. D' est la distance entre $\boldsymbol{c}_k$ et le vecteur le plus proche parmi les autres catégories.

2.5 Répéter la séquence d'opérations à partir de l'étape 2 en considérant $\boldsymbol{c}_i$ comme le nouveau $\boldsymbol{c}_k$ et ceci jusqu'à ce que tous les vecteurs de la même catégorie aient été examinés.

Etape 3 Répéter l'opération à partir de l'étape 1 pour les autres catégories.

L'intérêt d'un tel apprentissage est le contrôle explicite de la tendance du réseau à généraliser ses connaissances grâce au paramètre μ. La valeur de μ est réglée entre 1 et 3 comme cela a été montré dans [13]. Si $1 \leqslant \mu \leqslant 2$, alors on autorise un éventuel recouvrement entre les domaines d'influences. De la même façon, si $2 \leqslant \mu \leqslant 3$, alors les domaines d'influences seront tous séparés. Enfin, si $\mu = 2$, alors on permet aux domaines d'influences d'être contigus. Notre choix porte donc sur $\mu = 2$ qui permet une bonne capacité de généralisation et en même temps, évite toutes les confusions possibles entre les catégories.

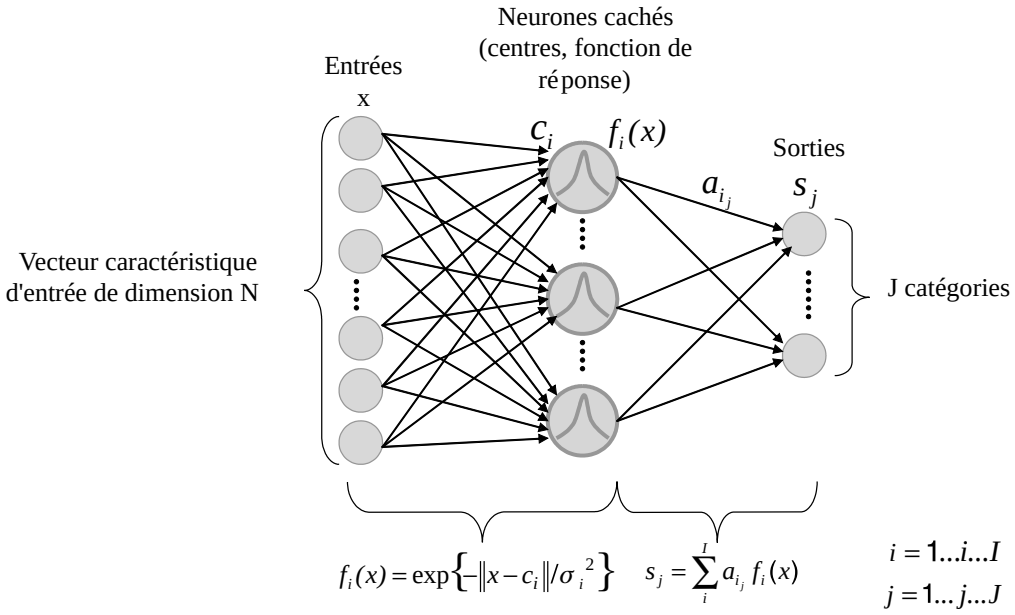

Figure 1. – Architecture générale d'un réseau RBF.

La figure 2 donne un aperçu de la technique « convergent K-means clustering ».

L'apprentissage des poids de la couche de sortie peut être réalisé par des techniques d'apprentissage simples. Dans notre cas, nous utilisons une méthode de matrice pseudo-inverse.

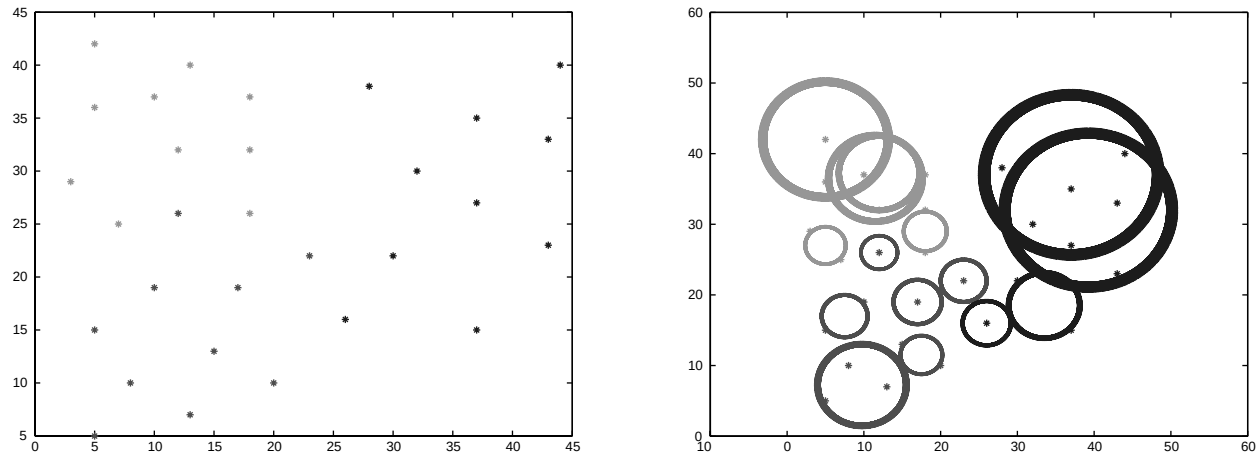

Figure 2. – Application de l'algorithme d'apprentissage avec $\mu = 2$ sur des vecteurs exemples de trois catégories : (a) Répartition des vecteurs exemples dans l'espace 2D. (b) Visualisation des nouveaux vecteurs associé à leur rayon respectif suite à la fusion des vecteurs exemples.

3. localisation et reconnaissance de visages – description du système

La figure 3 expose l'architecture de notre système de localisation et de reconnaissance de visages à 4 échelles. E. Viennet [14] a montré dans sa thèse qu'un rapport de 1.5 entre deux échelles voisines est suffisant pour détecter en continu un visage dont la taille varierait progressivement de l'échelle 1 à cette échelle 1.5. Nous utilisons donc ce rapport dans notre système. Les images à différentes échelles sont obtenues par sous échantillonnage de l'image originale.

Quelle que soit l'échelle de l'image sous échantillonnée utilisée, nous scrutons toujours dans celle-ci des imagettes de taille 40×32. Ainsi, la taille des visages dans les séquences originales peut varier de 40×32 pixels à 135×108 pixels. La base d'apprentissage étant alors seulement composée à partir des vignettes de visages de taille 40×32 pixels, l'apprentissage du

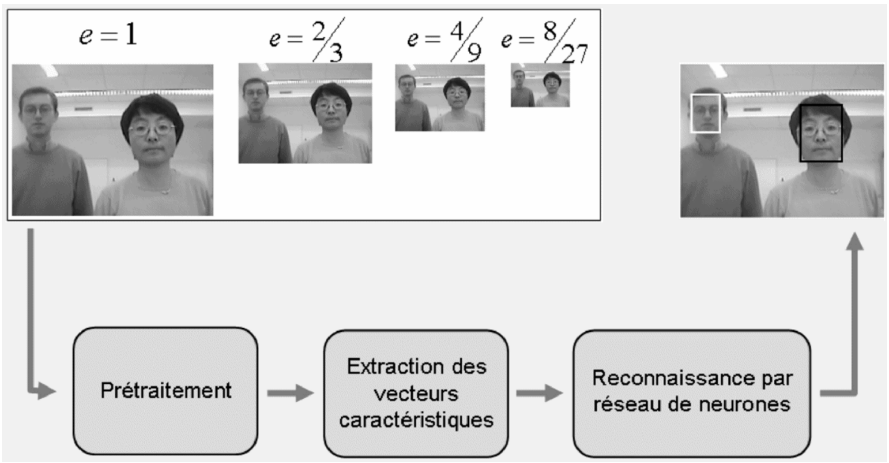

Figure 3. – Schéma global de localisation et de reconnaissance de visages.

réseau RBF est simplifié. Il est à noter que cette structure peut facilement être adaptée à un nombre d'échelles quelconque.

Nous réalisons des détections globales dans une image complète : nous découpons l'image en imagettes successives de taille 40×32 pixels (balayage de l'image avec un masque). Chaque imagette subit d'abord un pré-traitement de filtrage passe bas suivant les lignes. Ensuite, un extracteur forme un vecteur caractéristique d'entrée du réseau RBF qui détermine la présence, la position, l'identité et l'échelle des visages en cherchant les points maxima parmi les réponses du réseau. Le fonctionnement des modules de pré-traitement et d'extraction est présenté dans la section 4.2.

L'architecture du réseau RBF est prédéfinie par le nombre de personnes à reconnaître, avec une structure modulable : la connexion entre la couche cachée et la couche de sortie est locale, c'est-à-dire que les centres associés à chaque catégorie ne sont connectés qu'à un neurone de sortie qui indique si le stimulus appartient à cette catégorie. Cette structure du réseau fournit les avantages de réduire la complexité de l'algorithme et l'espace mémoire nécessaire pour stocker la matrice de poids de la couche de sortie. Elle facilite aussi l'exploitation du parallélisme intrinsèque de l'algorithme en éliminant une grande partie de l'interdépendance des données. Ce sont des points cruciaux pour la réalisation d'une implantation matérielle en temps réel.

4. résultats expérimentaux – évaluation du système

Nous présentons dans cette section les résultats obtenus par le système de localisation et de reconnaissance de visages. Une évaluation détaillée des performances de chaque configuration du système nous a permis de quantifier le comportement de celui-ci. Des tests ont aussi été effectués pour détecter et localiser les visages de différentes tailles. On estime l'efficacité à partir de fausses alarmes et de non-détections. L'évaluation est basée sur un corpus de 8 séquences de 256 images, soit un total de 2048 scènes.

4.1. présentation des séquences vidéo

Les 4 premières séquences sont filmées dans une salle sans aucun éclairage spécifique. Ces images de 240×320 pixels contiennent zéro, une, deux ou 3 personnes libres de leur mouvement. Il y a peu de changement d'éclairage et d'échelle de visages. La figure 4 affiche quelques images extraites de différentes séquences du corpus. Nous pouvons observer certains visages de profil ou inclinés. Nous avons choisi *Seb* et *Soph* comme personnes à localiser et à identifier dans ces séquences (voir Tableau 1). Nous avons utilisé la même base d'apprentissage qui contient 12 imagettes de visages pour chaque personne afin de comparer les différentes configurations du système (voir Figure 5).

Tableau 1. – Composition des 4 premières séquences du corpus : Chaque séquence est composée de 256 images. Nb. Soph, Nb. Seb et Nb. Domi indiquent respectivement le nombre de présences de chaque personne dans une séquence. Fond correspond au nombre d'images pour lesquelles aucun visage n'est présent.

	Nb. Soph	Nb. Seb	Fond	Nb. Domi
Séquence 1	256	192	0	0
Séquence 2	256	188	0	0
Séquence 3	228	204	20	60
Séquence 4	256	216	0	256

4.2. variation de la taille de données

Nous avons d'abord testé le système en introduisant une imagette brute de la taille 40×32 au réseau RBF. Le module de prétraitement effectue un simple lissage suivant les lignes pour atténuer le bruit dans l'image lié au capteur. Ensuite le module d'extraction convertit chaque imagette en un vecteur : chaque pixel de l'imagette correspond à une composante du vecteur. Le vecteur d'entrée du réseau RBF possède $40 \times 32 = 1280$ composantes.

Dans l'idée d'accélérer le calcul, nous avons ensuite essayé de réduire le nombre de composantes à traiter. Le module d'extraction réalise alors un sous-échantillonnage sur chaque ligne avec un pas de 4, 8 et 16 pixels pour former les vecteurs caractéristiques d'entrée du réseau.

Cette méthode de réduction de la taille des vecteurs est basée sur la propriété de symétrie par rapport à un axe vertical de la structure d'un visage (sous-échantillonnage suivant les lignes). Le choix du pas de sous-échantillonnage (4, 8, ou 16) qui est une puissance de 2 provient de notre souhait de simplifier l'implantation architecturale (voir la section 6). Ces différents pas de sous-échantillonnage (4, 8, ou 16) ont été étudiés sur nos séquences d'images présentées dans le tableau 1.

Le tableau 2 donne les résultats de ces tests, puis quelques images traitées sont affichées (voir Figure 6). L'efficacité du système est définie ci-dessous :

Résultat correct : les visages à identifier ont été correctement localisés et reconnus,

Non-détection : un visage à identifier n'a pas été localisé,

Fausse alarme : un visage a été localisé là où il n'y en a pas.

Figure 4. – Quelques images extraites des 4 premières séquences du corpus.

Figure 5. – Base d'apprentissage : ces exemples d'apprentissages sont choisis pour couvrir le plus possible les variations des positions des visages : profil, inclinaison, ...

Figure 6. – Images résultats de localisation et de reconnaissance de visages.

Tableau 2. – Résultats de localisation et de reconnaissance de visages avec variation de la taille de données : la longueur du vecteur d'entrée du réseau RBF est respectivement réduite à 320, 160 et 80. La colonne *Soph et Seb* correspond au nombre total de visages à reconnaître pour les séquences 1,2,3 et 4.

	Nb.composantes	Soph et Seb	N-Détec	F-Alarme	%Res-Correct
Imagette brute	1280	1796	140	0	92.2
Sous-échantillonnage 1 pixel/4 par ligne	320	1796	76	0	95.8
Sous-échantillonnage 1 pixel/8 par ligne	160	1796	120	0	93.3
Sous-échantillonnage 1 pixel/16 par ligne	80	1796	170	60	87.1
Sous-échantillonnage 1 pixel/8 par ligne 1 pixel/2 par colonne	80	1796	170	55	87.5

Les performances chutent rapidement quand le nombre de composantes passe de 160 à 80. Les fausses alarmes apparaissent à partir d'un sous-échantillonnage de 1 pixel/16. On remarque que le résultat obtenu avec un sous-échantillonnage de 1 pixel/16 par ligne est presque identique à celui d'un sous-échantillonnage de 1 pixel/8 par ligne et de 1 pixel/2 par colonne. Avec un sous-échantillonnage de 1 pixel/4 par ligne, nous avons obtenu des performances supérieures par rapport à une imagette brute. En effet, le fait de prendre seulement un pixel sur 4 rend le système moins sensible aux détails des visages, ce qui permet une meilleure généralisation du réseau RBF.

4.3. mesure de distance

Les résultats précédents ont été obtenus en appliquant la distance Euclidienne pour mesurer la différence entre le vecteur d'entrée $\boldsymbol{x}$ et un centre (noyau) $\boldsymbol{c}$ du réseau RBF :

$$d_2(\boldsymbol{x}) = \sqrt{\sum_{1 \leqslant n \leqslant N} (x_n - c_n)^2} \qquad (3)$$

avec N : le nombre de composantes du vecteur d'entrée.

Certaines publications [15] [16] [17] montrent que la distance $d_1(\boldsymbol{x})$ s'adapte mieux à un environnement bruité :

$$d_1(\boldsymbol{x}) = \sum_{1 \leqslant n \leqslant N} |x_n - c_n| \qquad (4)$$

Une autre mesure de distance ne compte que le nombre de composantes pour lesquelles la différence entre x_n et c_n est supérieure à un seuil δ :

$$d_0(\boldsymbol{x}) = \sum_{1 \leqslant n \leqslant N} 1 \quad \forall\, |x_n - c_n| > \delta \qquad (5)$$

Nous avons testé le comportement du système en utilisant ces 3 mesures de distance. Le seuil δ a été réglé expérimentalement à 10. Notre expérience indique que les meilleures performances pour cette application de localisation et de reconnaissance de visages sont obtenues avec $d_0(\boldsymbol{x})$ (voir Tableau 3). En effet, cette mesure de distance ne prend en compte que les composantes du vecteur à tester suffisamment différentes de celles du vecteur d'apprentissage.

Tableau 3. – Résultats de localisation et de reconnaissance de visages avec différentes mesures de distance : les performances sont obtenues avec le sous-échantillonnage de 1 pixel/4 par ligne.

	Nb.composantes	Soph et Seb	N-Détec	F-Alarme	%Res-Correct
Distance $d_2(\boldsymbol{x})$	320	1796	76	0	95.8
Distance $d_1(\boldsymbol{x})$	320	1796	64	0	96.4
Distance $d_0(\boldsymbol{x})$	320	1796	32	0	98.2

4.4. fonction de la réponse RBF

Dans le cadre du réseau RBF, la fonction de la réponse la plus utilisée est la fonction Gaussienne :

$$f_i(\boldsymbol{x}) = \exp^{\frac{-d(\boldsymbol{x})^2}{\sigma_i^2}} \quad (6)$$

avec $d(\boldsymbol{x})$: la mesure de distance entre le vecteur $\boldsymbol{x}$ et le centre $\boldsymbol{c}_i$. Nous avons exploité la possibilité de remplacer la fonction Gaussienne par une simple fonction Porte afin de faciliter l'implantation sur une architecture électronique. La largeur de la fonction Porte correspond au seuil de distance σ_i :

$$f_i(\boldsymbol{x}) = \begin{cases} 1 & d(\boldsymbol{x}) \leqslant \sigma_i \\ 0 & d(\boldsymbol{x}) > \sigma_i \end{cases} \quad (7)$$

Si au moins une des fonctions-noyau (centre) de la même catégorie produit une réponse active, alors le neurone de sortie lié à cette catégorie donne une réponse positive. Ceci se matérialise par la fonction *OU logique* entre les sorties des neurones de la couche cachée.

Les résultats comparatifs sont présentés dans le tableau 4. Par rapport à la fonction Gaussienne, le nombre de non-détections a beaucoup augmenté avec la fonction Porte. Le taux de réussite diminue de 98.2 % à 93.4 %. Ceci peut s'expliquer par le fait que le réseau RBF utilisant la fonction Porte restreint la capacité de généralisation par manque d'interactions entre les centres d'une même catégorie : le système ne détecte que des visages assez proches des exemples d'apprentissage.

Parmi toutes les configurations du système, la meilleure performance a été obtenue avec le sous-échantillonnage de 1 pixel sur 4 (le nombre de composantes du vecteur d'entrée = 320), la mesure de distance $d_0(\boldsymbol{x})$ et la fonction de réponse Gaussienne : le taux de réussite est de 98.2%. Presque tous les visages sont bien détectés, localisés et identifiés dans ces 4 séquences d'image, soit un total de 1024 scènes. Il est à noter que nous avons réussi à regrouper 12 vecteurs d'exemples d'apprentissage par personne en 6 ou 8 neurones cachés par personne pour chaque configuration en utilisant l'algorithme de « clustering ».

4.5. variation de l'échelle de visages

Nous avons aussi appliqué notre système de localisation et de reconnaissance de visages sur 4 nouvelles séquences d'images dans lesquelles on observe une grande variation de la taille des visages et des variations d'éclairage. Nous avons ajouté dans le module « Extraction des vecteurs caractéristiques » une étape de traitement qui consiste à centrer et à normaliser les vecteurs d'entrée du réseau RBF afin de rendre le système moins sensible aux variations de luminance.

Les images originales sont d'abord sous-échantillonnées en 4 échelles avec un rapport d'échelle égal à 1.5. Ensuite, ces images sous-échantillonnées sont présentées au système de localisation et de reconnaissance de visages. Quelque soit l'échelle de l'image sous-échantillonnée utilisée, nous scrutons toujours dans celle-ci des visages de taille 40×32. Ainsi, la taille des visages dans les séquences originales peut varier de 40×32 pixels à 135×108 pixels. Par exemple, si l'on détecte un visage avec la deuxième échelle, la taille réelle de ce visage dans l'image originale est de 60×48 pixels (voir Figure 3).

Le tableau 5 donne les résultats comparatifs de localisation et de reconnaissance de visages avec nos 4 séquences d'images. Ces résultats ont été obtenus avec ou sans l'étape de centrage et de normalisation des vecteurs d'entrée. Cette étape supplémentaire nous permet d'augmenter de 4.7 % le taux de réussite pour ces séquences où l'on observe des variations d'éclairage. La figure 7 montre quelques images résultats de localisation et de reconnaissance de visages.

Le temps d'exécution de notre algorithme de localisation et de reconnaissance de visage, implanté sous Matlab sur un PC PIII-

Tableau 4. – Résultats de localisation et de reconnaissance de visages avec différentes fonctions de réponse : les performances sont obtenues avec un sous-échantillonnage de 1 pixel/4 et la mesure de distance $d_0(\boldsymbol{x})$.

Fonction de réponse	Nb.composantes	Soph et Seb	N-Détec	F-Alarme	%Res-Correct
Gaussienne	320	1796	32	0	98.2
Porte	320	1796	118	0	93.4

Tableau 5. – Résultats obtenus pour la reconnaissance, en utilisant la configuration suivante : sous-échantillonnage de 1 pixel/4, distance $d_1(\boldsymbol{x})$ et fonction de réponse Porte. C. et N. **représentent l'étape de centrage et de normalisation des vecteurs d'entrée du réseau RBF.**

	Nb.images	Nb.visages à reconnaître	N-Détec	F-Alarme	%Résultat Correct
Sans C. et N.	256×4	1492	152	24	88.2
Avec C. et N.	256×4	1492	105	0	92.9

Figure 7. – Images résultats de localisation et de reconnaissance de visages.

700Mhz, est de 64,8 secondes pour le traitement d'une image avec l'échelle 1 et de 130,5 secondes pour le traitement complet avec 4 échelles. Rappelons que ces images sont de taille 240×320 pixels.

5. reconnaissance de visages de la base de données ORL

Dans l'idée de vérifier et de comparer la robustesse de notre système par rapport aux méthodes existantes, nous avons adapté notre algorithme de localisation et de reconnaissance de visages pour une application de simple reconnaissance de visages que nous testons sur la base de données standard ORL (Olivetti Research Laboratory) [18]. L'ensemble de la base de données contient 400 images (40 personnes $\times$ 10 images). Ces images sont de taille 112×92 pixels et sont codées sur 8 bits. La figure 8 présente quelques extraits de la base ORL.

Après sous-échantillonnage, nous disposons de 400 visages au format 16×16 [15] et la classification a été directement appliquée sur ces visages statiques avec le réseau RBF, le nombre de personnes à reconnaître étant fixé à 40. Nous utilisons $p = 1,3,5$ images aléatoirement choisies de chaque individu pour constituer la base d'apprentissage, et le reste de visages pour tester le système. Plusieurs exécutions ont été réalisées avec chaque valeur de p pour répartir aléatoirement la base d'apprentissage et celle de test. Ensuite, leurs résultats sont moyennés et présentés dans le tableau 6. Les performances sont définies ci-dessous :

Résultat correct : reconnaissance correcte d'un visage,

Non-reconnaissance : un visage n'a pas été reconnu,

Confusion : un visage est confondu avec un intrus.

Figure 8. – Quelques extraits de la base ORL.

Tableau 6. – Résultats de reconnaissance de visages avec la base ORL : ces résultats ont été obtenus avec la mesure de distance $d_0(x)$ et avec la fonction de réponse Gaussienne. Ici, Nb. Reconnu représente le nombre de visages devant être reconnus et Nb. Intrus indique le nombre de visages devant être rejetés.

Nb. Exemples/individu	$p = 1$	$p = 3$	$p = 5$
Nb. Exemple	40	120	200
Nb. Test	14400	11200	8000
Nb. Reconnu	360	280	200
Nb. Intrus	14040	10920	7800
N-Reconnaissance	68	47	21
Confusion	1026	682	226
% Résultat correct	92.4	93.5	96.9

Tableau 7. – Comparaison des taux de reconnaissance de différentes méthodes utilisant la base ORL.

Méthodes	$p = 1$	$p = 3$	$p = 5$
Sim & *al.*	75.1 %	92.2 %	97.1 %
Howell & Buxton	84 %	91 %	95 %
Slimane & *al.*	80 %	89 %	96 %

Les performances du tableau 6 sont obtenues à partir des courbes *Nombre-Non-Reconnu* en fonction du *Nombre-Confusion* (voir Figure 9). Par exemple, si nous prenons $p = 1$, alors chaque neurone de sortie (associé à une catégorie) devrait reconnaître 9 visages de sa catégorie et rejeter les $9 \times 39 = 351$ visages correspondant aux autres catégories (intrus). Avec $p = 5$, nous avons un taux de réussite de 97 %, qui est très proche ou supérieur aux performances annoncées dans la littérature [15] [5] [11] (voir Tableau 7) . Notre système présente des avantages de volume de calcul modéré et d'espace de mémoire nécessaire limité.

Par exemple, quand le nombre de vecteurs d'apprentissage $p = 5$ par individu, notre algorithme de « clustering » nous permet de fusionner ces 5 vecteurs en seulement 2 ou 3 centres par individu.

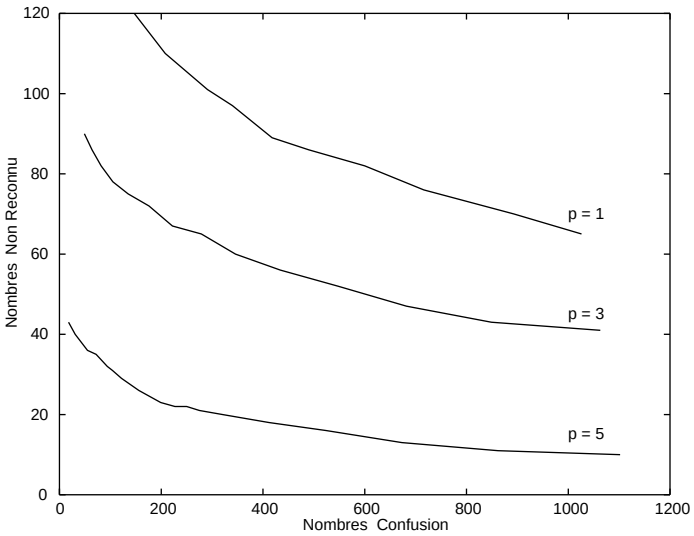

Figure 9. – Performances de reconnaissance de visages avec la base ORL.

6. adéquation algorithme architecture – Analyse de complexité

Notre but final est de réaliser un système embarqué permettant la localisation et la reconnaissance de visages en temps réel. C'est pourquoi nous considérons l'aspect « Adéquation Algorithme Architecture » tout au long du développement du système. Par exemple, le fait que le pré-traitement de filtrage passe bas (voir la section 4.2) ne soit effectué que sur les lignes de chaque imagette nous permet d'éviter l'utilisation d'une mémoire FIFO de grande taille car les valeurs des pixels sortent de la caméra CCD (ou CMOS) une ligne après l'autre. De même, nous cherchons à réduire le volume de calcul. Avec l'algorithme de « clustering », nous avons réussi à réduire le nombre de neurones de la couche cachée par la fusion des exemples d'apprentissage. Dans cette section, nous allons d'abord définir les paramètres et la configuration du système à implanter sur les cartes électroniques. Ensuite, nous effectuons une analyse de complexité afin de connaître le nombre d'opérations à réaliser.

6.1. choix des paramètres et de la configuration du système à implanter

Pour localiser et reconnaître des visages dans des séquences vidéo, nous utilisons des exemples d'apprentissages pour entraîner le réseau RBF. Cette phase d'apprentissage est décomposée en deux étapes : configuration de la couche cachée (nombre de neurones cachés, leur position de centre et leur seuil) et définition des poids de connexion entre la couche cachée et la couche de sortie. La première étape est réalisée avec l'algorithme « clustering » (voir la section 2.2) et la seconde utilise la méthode de matrice pseudo-inverse. Une fois l'apprentissage du réseau accompli, le système analyse, en phase de test, les imagettes en donnant les réponses sur la présence, la position, l'identité et l'échelle des visages des personnes apprises. Dans les applications visées par cette étude, le traitement en temps réel est surtout important pour cette phase de test. Dans les présentations ci-dessous, on considère que l'apprentissage du réseau est d'abord réalisé sur un PC et ses résultats sont ensuite téléchargés sur des cartes électroniques. Les implantations matérielles sont ainsi effectuées uniquement pour la phase de test.

Afin d'accélérer les calculs pendant la phase de test, nous allons réduire la complexité de celle-ci. Dans la méthode présentée en section 4.2, les vecteurs caractéristiques d'entrée du réseau RBF sont extraits en filtrant les imagettes puis en les sous-échantillonnant suivant les lignes. En prenant un pixel sur 4, nous obtenons dans ce cas 320 composantes par vecteur. Pour limiter les calculs, nous approximons ces traitements en calculant sur chaque ligne d'une imagette 8 valeurs moyennes de 4 pixels consécutifs (soient $8 \times 40 = 320$ composantes par imagette).

Dans la section 4.3, nous avons testé trois calculs de distances pour mesurer la similarité d'un vecteur en entrée par rapport à un neurone caché du réseau (voir Tableau 3). La distance $d_2(\boldsymbol{x})$ est la plus compliquée à réaliser et donne des performances moins bonnes. Les équations (4) et (5) montrent que la complexité de la distance $d_1(\boldsymbol{x})$ est presque identique à la distance $d_0(\boldsymbol{x})$. En utilisant la distance $d_0(\boldsymbol{x})$, nous obtenons des performances légèrement supérieures par rapport à la distance $d_1(\boldsymbol{x})$, cependant le seuil δ de cette distance est difficile à régler. Notre choix s'est donc porté sur la distance $d_1(\boldsymbol{x})$.

De même, dans la section 4.4, nous avons évalué deux fonctions de réponses : Gaussienne et Porte. La fonction de réponse Porte est beaucoup plus facile à implanter par rapport à la fonction Gaussienne et donne des performances proches de 90 % de taux de réussite comme présentées dans les sections 4.4 et 4.5. Nous avons décidé d'implanter cette fonction de réponse dans un premier temps. En conséquence l'analyse de complexité présentée ci-dessous, est effectuée avec la distance $d_1(\boldsymbol{x})$, la fonction de réponse Porte et l'extraction des vecteurs d'entrée par la méthode du moyennage sur 4 pixels consécutifs.

6.2. analyse de complexité de la phase de test

L'objectif de cette section est de calculer le nombre exact d'opérations à effectuer pour réaliser la localisation et la reconnaissance de visages. Pour ce faire, nous décomposons l'analyse de complexité en plusieurs étapes en commençant par définir la taille des images à tester.

6.2.1. taille des images à tester

Supposons que chaque image originale d'une séquence a la dimension suivante : L_1 lignes et C_1 colonnes soit $L_1 \times C_1$ pixels par image. L'image est d'abord sous échantillonnée afin de travailler avec E échelles. Nous considérons que le rapport r entre les échelles est constant (l'image à l'échelle e est r fois plus petite que l'image à l'échelle $e-1$). L'image originale correspond à l'échelle 1. La dimension des images aux différentes échelles s'exprime alors ainsi :

$$D_e = L_e \times C_e = \lfloor \frac{L_1}{r^{e-1}} \rfloor \times \lfloor \frac{C_1}{r^{e-1}} \rfloor \begin{cases} e = 1,2....,E \\ r = cste. \end{cases} \quad (8)$$

En relation avec la méthode décrite en section 3, nous utilisons 4 échelles avec un rapport $r = 1.5$ entre 2 échelles consécutives, d'où les dimensions des images à analyser :

$e = 1 : D_1 = L_1 \times C_1$

$e = 3 : D_2 = \lfloor \frac{L_1}{r} \rfloor \times \lfloor \frac{C_1}{r} \rfloor = \lfloor \frac{L_1}{1.5} \rfloor \times \lfloor \frac{C_1}{1.5} \rfloor = L_2 \times C_2$

$e = 3 : D_3 = \lfloor \frac{L_1}{r^2} \rfloor \times \lfloor \frac{C_1}{r^2} \rfloor = \lfloor \frac{L_1}{2.25} \rfloor \times \lfloor \frac{C_1}{2.25} \rfloor = L_3 \times C_3$

$e = 4 : D_4 = \lfloor \frac{L_1}{r^3} \rfloor \times \lfloor \frac{C_1}{r^3} \rfloor = \lfloor \frac{L_1}{3.375} \rfloor \times \lfloor \frac{C_1}{3.375} \rfloor = L_4 \times C_4$

Des travaux sur le traitement automatique de visages à multi-échelles [3] [14] montrent que le rapport $r = 1.5$ entre deux échelles voisines permet de détecter en continu un visage dont la taille peut varier progressivement de l'échelle 1 à l'échelle 1.5. En pratique, ce rapport fractionnaire nous amène à sous échantillonner nos images originales en conservant 2 pixels sur 3.

6.2.2. nombre d'imagettes à tester

Le traitement d'une image nécessite l'utilisation d'une fenêtre glissante de dimension $L_f \times C_f$ (voir Figure 10). La taille de cette fenêtre ainsi que les pas de déplacements de celle-ci suivant les lignes et les colonnes déterminent le nombre d'imagettes à tester V_f que nous calculons par l'équation :

$$V_f = \lfloor \frac{L - L_f}{p_c} + 1 \rfloor \lfloor \frac{C - C_f}{p_l} + 1 \rfloor \quad (9)$$

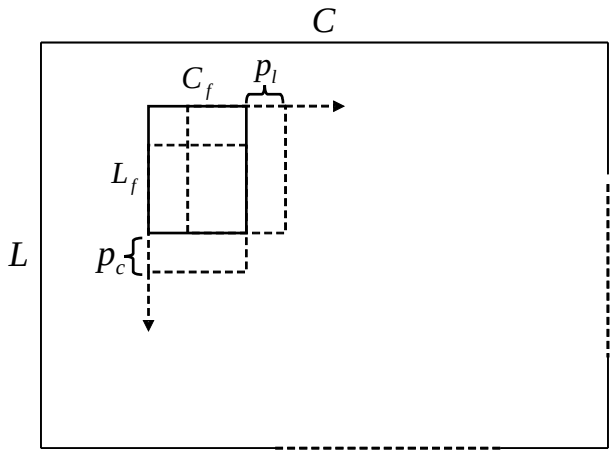

Figure 10. – Balayage de l'image avec une fenêtre glissante.

soit pour E échelles :

$$V_{f_E} = \sum_{e=1}^{E} (\lfloor \frac{L_e - L_{f_e}}{p_{c_e}} + 1 \rfloor \lfloor \frac{C_e - C_{f_e}}{p_{l_e}} + 1 \rfloor) \qquad (10)$$

avec les paramètres suivants :
L : nombre de lignes de l'image,
C : nombre de colonnes de l'image,
L_f : nombre de lignes de la fenêtre glissante,
C_f : nombre de colonnes de la fenêtre glissante,
p_l : pas de balayage suivant les lignes,
p_c : pas de balayage suivant les colonnes.

Nous utilisons la même taille de fenêtre de balayage pour chaque échelle ainsi que les mêmes pas de déplacement. Nous prenons donc L_f, C_f, p_l et p_c constants à chaque échelle. L'équation (10) devient alors :

$$V_{f_E} = \sum_{e=1}^{E} (\lfloor \frac{L_e - L_f}{p_c} + 1 \rfloor \lfloor \frac{C_e - C_f}{p_l} + 1 \rfloor) \qquad (11)$$

6.2.3. extraction des vecteurs d'entrée

L'extraction des composantes d'un vecteur d'entrée lié à une imagette $L_f \times C_f$ s'effectue en calculant les B valeurs moyennes de 4 pixels consécutifs sur chaque ligne (voir Figure 11). Nous effectuons le balayage vertical de l'image à analyser afin d'utiliser la redondance des composantes d'un vecteur avec son suivant. Comme cela est précisé dans les sections 7 et 8, la redondance est matérialisée par l'emploi d'une mémoire FIFO. Nous réduisons ainsi le volume de calculs liés à l'extraction. Les nombres d'additions et de divisions nécessaires au calcul du vecteur lié à la première imagette de chaque colonne sont donc :

$$A'_1 = (\frac{C_f}{B} - 1) \times B \times L_f \qquad (12)$$

B représente le nombre de composantes par ligne (valeurs moyennes). Par exemple, si la fenêtre glissante a une largeur de $C_f = 32$ pixels, alors, la valeur de B correspond à 8.

$$D'_1 = B \times L_f \qquad (13)$$

Pour former chaque nouveau vecteur à partir des imagettes suivantes de chaque colonne, il suffit d'ajouter au précédent $B \times p_c$ nouvelles composantes. En analysant l'image à une seule échelle, le nombre d'additions et de divisions nécessaires pour l'extraction de tous les vecteurs sont donc :

$$A_1 = (\frac{C_f}{B} - 1) \times B \times L \times \lfloor \frac{C - C_f}{p_l} + 1 \rfloor \qquad (14)$$

$$D_1 = B \times L \times \lfloor \frac{C - C_f}{p_l} + 1 \rfloor \qquad (15)$$

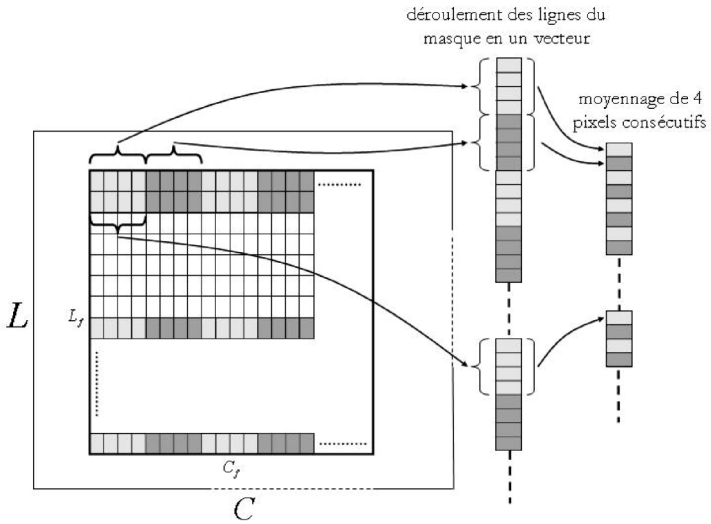

Figure 11. – Extraction des composantes d'un vecteur à tester.

6.2.4. calcul de distance et prise de décision

La mesure de distance à implanter est la distance $d_1(\boldsymbol{x})$ calculée entre les N composantes du vecteur d'entrée et les N composantes d'un neurone caché (voir équation (4)). Le nombre, la position de centre et le seuil de neurones cachés pour chaque personne apprise ont été définis pendant la phase d'apprentissage. Les résultats expérimentaux présentés dans la section 4 montrent que le nombre de 6 à 8 neurones cachés par personne suf-

fit pour reconnaître les différentes vues de cette personne (voir la section 4.4). Dans notre application, le nombre de personnes à reconnaître est fixé à 2. Le réseau RBF possède donc $I = 15$ neurones cachés après la phase d'apprentissage.

Le calcul de distance demande N soustractions et $N - 1$ additions par neurone caché. Nous avons alors :

$$A_2 = (N - 1) \times I \times V_f \tag{16}$$

$$S = N \times I \times V_f \tag{17}$$

avec :

A_2 : nombre d'additions nécessaires au calcul de la distance $d_1(\boldsymbol{x})$ pour une image.

S : nombre de soustractions nécessaires au calcul de la distance $d_1(\boldsymbol{x})$ pour une image.

N : nombre de composantes du vecteur d'entrée.

I : nombre total de neurones cachés.

Chaque neurone caché donne un résultat binaire avec la fonction de réponse Porte (voir équation (7)). La détermination de la réponse binaire d'un neurone caché nécessite une seule comparaison de la distance avec son seuil. Le nombre de comparaisons réalisées par les I neurones de la couche cachée pour traiter une image est donc :

$$C = I \times V_f \tag{18}$$

La prise de décision à la couche de sortie n'utilise que des fonctions *OU binaire*, le nombre de comparaisons est donné par l'équation suivante :

$$O = V_f \times \sum_{j=1}^{J} (I_j - 1) \tag{19}$$

O : nombre total de comparaisons *OU binaire*.

J : nombre de catégories apprises.

I_j : nombre de neurones cachés associés à la catégorie j.

Le nombre maximum d'opérations *OU binaire* est donné pour $J = 1$:

$$O = V_f \times (I - 1) \tag{20}$$

7. première implantation matérielle sur une architecture à base de processeurs ZISC

Nous présentons dans cette section la première implantation matérielle. Nous décrivons d'abord l'architecture de la carte utilisée. Ensuite, nous détaillons chaque étape de l'implantation et nous donnons aussi les résultats de cette réalisation en terme de ressources matérielles utilisées et de vitesse de traitement.

7.1. architecture du système électronique

Nous proposons une première implantation de notre algorithme de détection et de reconnaissance de visages sur la carte électronique commerciale *Neurosight* de la société *General Vision* [19]. La figure 12 présente la photo de cette carte. C'est un système embarqué capable d'apprendre et de reconnaître des objets dans des séquences vidéo. Elle est constituée d'un capteur d'images de type CMOS à sorties numériques, d'un FPGA permettant de programmer l'application désirée, d'un banc de mémoire utile à la mémorisation de l'image courante et enfin de processeurs spécifiques sur lesquels sont implantés des neurones de type RBF : ces composants sont seulement alimentés en données et réalisent toujours le même type de calcul sur celles-ci, d'où le terme « Zero Instruction Set Computer ». Cette carte est autonome du fait qu'elle est capable d'apprendre d'elle même des objets à reconnaître. Nous présentons plus en détail chacun des éléments de la carte.

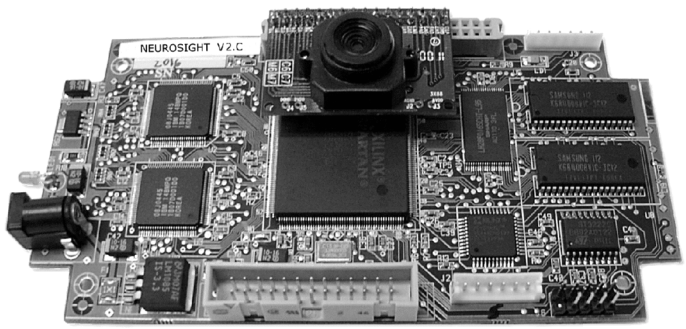

Figure 12. – Configuration de la carte Neurosight.

Processeur ZISC Le processeur ZISC réalise un réseau neuronal de type RBF. Ses performances le rendent capable d'effectuer en temps réel des applications de traitement du signal ou de traitement d'image. Nous pouvons citer par exemple la reconnaissance de caractères [20] ou le débruitage d'images [21].

La particularité du ZISC est qu'il intègre la phase d'apprentissage. Ceci le distingue de toutes les puces neuronales commerciales existantes. Ce composant ZISC ne nécessite aucune programmation, il suffit de lui présenter des vecteurs à apprendre ou à reconnaître. L'algorithme d'apprentissage intégré dans le ZISC est similaire à ce que nous avons présenté dans la section 2.2.

Dans le ZISC, chaque neurone caché stocke un vecteur appris (son centre), son seuil associé (définissant son domaine d'influence) ainsi que la catégorie à laquelle il appartient. Les vecteurs appris ou testés ont au maximum 64 composantes codées sur 8 bits. Le calcul de la distance entre un vecteur appris et un vecteur à tester utilise la norme $d_1(\boldsymbol{x})$ (voir équation (4)). La fonction de réponse de chaque neurone caché est la fonction Porte (voir équation (7)). Nous disposons de 2 composants ZISC sur la carte *Neurosight* et chaque ZISC possède 78 neurones cachés. Tous ces neurones cachés sont implantés en parallèle. La connectivité des processeurs entre eux est assurée par des liens directement câblés.

Imageur L'imageur est constitué d'un capteur d'image de taille 356×292 pixels maximum capable de fournir jusqu'à 60 trames par seconde en mode entrelacé ou 30 images par seconde en mode continu. La taille de l'image est modulable de 2×2 à 356×292 pixels. L'image est soit en couleur soit en noir et blanc (configurable). Il est à noter que le temps nécessaire à la sortie d'une image reste identique quelque soit les paramètres programmés. L'intérêt majeur d'un tel système imageur est de fournir directement une image numérique. Ce composant est complètement contrôlable via le FPGA.

FPGA Le FPGA est un SpartanII-50 de la société Xilinx [22]. Il constitue la partie de la carte directement programmable par l'utilisateur. Les ressources matérielles d'un FPGA sont principalement composées d'une matrice de blocs logiques interconnectés permettant de définir les fonctions souhaitées. Le composant utilisé contient 384 blocs logiques (CLBs) ainsi que 8 bancs mémoires de 4 Kbits chacun. Sur cette carte *Neurosight*, ce composant représente le coeur du système.

Mémoires SRAM Deux composants de mémoire SRAM sont disponibles sur cette carte. Ces bancs de mémoire sont communément utilisés pour stocker les données. Chaque banc a une configuration de 512 K $\times$ 8 bits, soit une capacité totale de 1 MOctets. L'accès à chacun des 2 bancs à partir du FPGA est indépendant. Par contre, le bus d'adresses est commun aux 2 bancs.

Mémoire Flash RAM Cette mémoire non volatile de 2 MOctets est connectée à la fois au FPGA et à un CPLD. Ce dernier composant a pour fonction de charger le fichier de configuration du FPGA contenu dans la mémoire Flash lorsque la carte est mise sous tension. Lors du développement d'une application, la configuration du FPGA peut aussi s'effectuer directement via le port JTAG de la carte.

Port série et port utilisateur Cette carte est dotée d'un port série classique et d'un port utilisateur. Dans notre application, ce port est configuré en port parallèle.

7.2. complexité du système

Avant de détailler l'implantation de la détection et de la reconnaissance de visages sur la carte *Neurosight*, nous précisons les paramètres du système réalisé. Tout d'abord, la taille d'images captées par l'imageur est configurée à 288×352 (format QCIF). Ensuite, nous choisissons dans un premier temps d'analyser des images seulement à l'échelle 1 qui représente le pire cas en terme de volume calculatoire. Le réseau de neurones RBF possède $I = 15$ neurones cachés (voir les sections 4.4 et 6.2.4). Enfin, les vecteurs appris ou à tester ont tous $N = 64$ composantes.

Pour s'adapter à la longueur maximale de 64 composantes imposée par le ZISC, nous effectuons d'abord un sous échantillonnage de l'image originale : nous gardons uniquement une ligne sur 4 d'où une image résultante de $L \times C = 72 \times 352$ pixels. Ensuite, une fenêtre glissante de la taille $L_f \times C_f = 8 \times 32$ est utilisée pour balayer cette nouvelle image avec les pas de déplacement $p_c = 1$ suivant les colonnes et $p_l = 2$ suivant les lignes. Les composantes d'un vecteur correspondent aux moyennes de 4 pixels successifs sur les lignes de l'imagette balayée, soit 8 composantes par ligne. Ce qui nous donne bien $8 \times 8 = 64$ composantes par vecteur. L'étude de complexité présentée dans la section 6 nous indique le nombre d'imagettes à traiter (voir équation (9)) :

$$V_f = \lfloor \frac{L - L_f}{p_c} + 1 \rfloor \lfloor \frac{C - C_f}{p_l} + 1 \rfloor$$
$$= \lfloor 72 - 8 + 1 \rfloor \lfloor \frac{352 - 32}{2} + 1 \rfloor = 10465 \qquad (21)$$

Le tableau 8 récapitule les résultats des applications numériques des équations (12) à (20) (voir la section 6), ce qui permet de définir le nombre d'opérations à effectuer par la carte *Neurosight* pour traiter une image. Ce nombre s'élève à un peu moins de $21 \cdot 10^6$ opérations par image.

7.3. implantation du système

Nous allons maintenant décrire l'implantation du système sur la carte *Neurosight*. La figure 13 présente l'organisation des tâches sur cette carte. Nous avons conçu les contrôleurs de chacun des composants ainsi que les liens de communications leur permettant de travailler ensemble. Cette logique de contrôle et

Tableau 8. – Tableau récapitulatif des opérations nécessaires au traitement d'une image sur la carte *neurosight*

	Additions	Soustraction	Divisions	Comparaisons	Nb.Total
Extractions des vecteurs	278208	0	92736	0	370944
Calculs de distance	$9{,}9 \cdot 10^6$	$10 \cdot 10^6$	0	0	$19{,}9 \cdot 10^6$
Réponse Neurones	0	0	0	303485	303485
Totaux	$10{,}17 \cdot 10^6$	$10 \cdot 10^6$	92736	303485	$20{,}57 \cdot 10^6$

l'extraction des vecteurs d'entrée sont implantées sur le FPGA. La programmation du FPGA est réalisée en langage VHDL. La figure 14 montre la hiérarchie du code VHDL écrit. Le contrôle du système s'effectue grâce à des machines d'états.

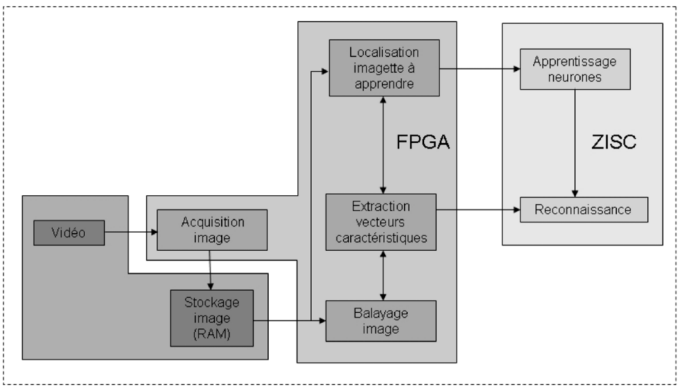

Figure 13. – Schéma de l'organisation des tâches sur la carte Neurosight.

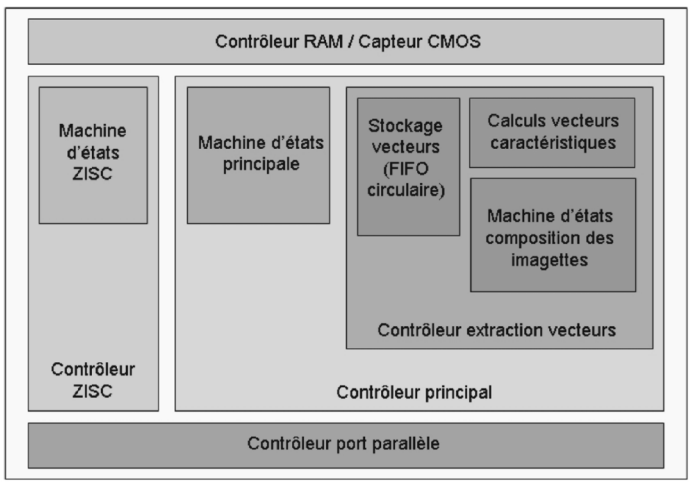

Figure 14. – Schéma de la hiérarchie du code VHDL : le contrôleur principal est au centre du système et fait le lien entre tous les éléments. Il contrôle la tâche d'extraction de vecteurs d'entrée, le réseau neuronal ZISC et les liens de communications externes.

Acquisition d'images L'utilisateur dispose d'une interface sur un PC hôte relié à la carte via le port parallèle classique. Il peut ainsi déclencher l'acquisition d'une image pour faire un apprentissage ou encore lancer le mode « reconnaissance » sur des séquences d'images. Lorsqu'une image est complètement chargée dans le banc mémoire de la carte (SRAM), l'extraction des vecteurs d'entrée successifs peut commencer avec la première imagette.

Extraction des vecteurs Pour extraire un vecteur à partir d'une imagette, le FPGA effectue d'abord des accès de la mémoire SRAM pour lire les valeurs de pixels correspondants, puis il calcule la moyenne de 4 pixels consécutifs pour former une composante. La valeur de cette composante est ensuite stockée dans un banc mémoire interne du FPGA que nous avons configuré comme une FIFO circulaire. Dès que la mémoire FIFO est remplie avec un vecteur de 64 composantes, le système calcule la position du premier pixel de l'imagette suivante puis attend à nouveau que la FIFO soit accessible.

Reconnaissance Lorsqu'un vecteur à tester est prêt dans la mémoire FIFO, il est chargé séquentiellement, composante par composante dans le processeur ZISC. Dès que cette étape du chargement est terminée, le processus de reconnaissance est activé. La machine d'états indique que la mémoire FIFO peut accueillir un nouveau vecteur. Le résultat de la reconnaissance est retourné depuis le ZISC vers le FPGA. Ce résultat et la position du vecteur courant sont stockés et rafraîchis. Quand l'analyse d'une image complète est accomplie, ces informations peuvent être éventuellement transmises au PC *via* le port parallèle pour l'affichage des résultats.

7.4. analyses matérielle et temporelle de l'implantation

Le tableau 9 indique les ressources matérielles disponibles ainsi que les ressources effectivement utilisées par le FPGA SparatanII-50 de la carte *Neurosight*. Nous pouvons remarquer que le système n'a besoin que de 40 % environ des ressources

fonctionnelles et de 12.5 % des ressources mémoires disponibles sur le FPGA pour cette application de détection et de reconnaissance de visage à l'échelle 1.

Tableau 9. – Ressources matérielles utilisées sur le FPGA Xilinx SpartanII-50 : un CLB correspond à 2 « slices » (« slice » = cellule logique de base du FPGA) et les pourcentages représentent les taux d'occupation des ressources.

	Slices	Blocs Mémoires RAM
SpartanII-50	768	8
Contrôleur principal	97	1
Contrôleur ZISC	52	0
Contrôleur RAM/CMOS	47	0
Contrôleur port parallèle	88	0
Logique contrôle global	8	0
Système complet	292 38%	1 12.5%

Nous allons maintenant étudier la vitesse de fonctionnement du système. Il faut pour cela définir les étapes séquentielles du processus de traitement. Les deux phases principales sont :

– Capture et mémorisation d'une image dans le banc mémoire,

– Traitement de toutes les imagettes contenues dans l'images les unes après les autres.

L'acquisition d'une image prend un temps incompressible quelle que soit la taille de l'image désirée. Le constructeur [23] donne un temps d'acquisition d'une image $T_{image} = 16{,}6$ ms. Le traitement des imagettes s'effectue selon le chronogramme présenté sur la figure 15. L'analyse temporelle détaillée [24] nous indique que le traitement d'une image complète de taille 288×352 nécessite $1{,}27 \cdot 10^6$ périodes d'horloges (T_{clk}). La fréquence de l'osillateur sur la carte est de 33 MHz ($T_{clk} = 30$ ns). Le traitement de toutes les imagettes est alors effectué en environ $T_{trait.} = 38$ ms. Le temps total nécessaire à une image est donc $T_{image} + T_{trait.} = 54{,}6$ ms. Le logiciel de développement nous indique que la fréquence maximale soutenue par le système est de 85MHz ($T_{clk} = 11{,}7$ ns, ce qui nous donne $T_{trait.} = 14{,}7$ ms). Dans ce cas, une image serait analysée en seulement 31,3ms.

Pour évaluer les performances de reconnaissance du système, nous avons effectué une simulation logicielle de cette première implantation. Nous avons utilisé les mêmes séquences d'images de test présentées dans la section 4.1. Nous respectons les paramètres suivants : vecteur d'entrée de 64 composantes, mesure de distance $d_1(\boldsymbol{x})$ et fonction Porte comme fonction de réponse. Le tableau 10 indique que le taux de reconnaissance correcte est de 85.3 %. Cette faible performance est dûe au fait que la longueur maximale des vecteurs d'entrée du ZISC est limitée à 64 composantes.

Tableau 10. – Résultats de la simulation logicielle de la première implantation

Nb. Visages à reconnaître	Nb. composantes	N-Détec	F-Alarme	%Res-Correct
1796	64	210	53	85.3

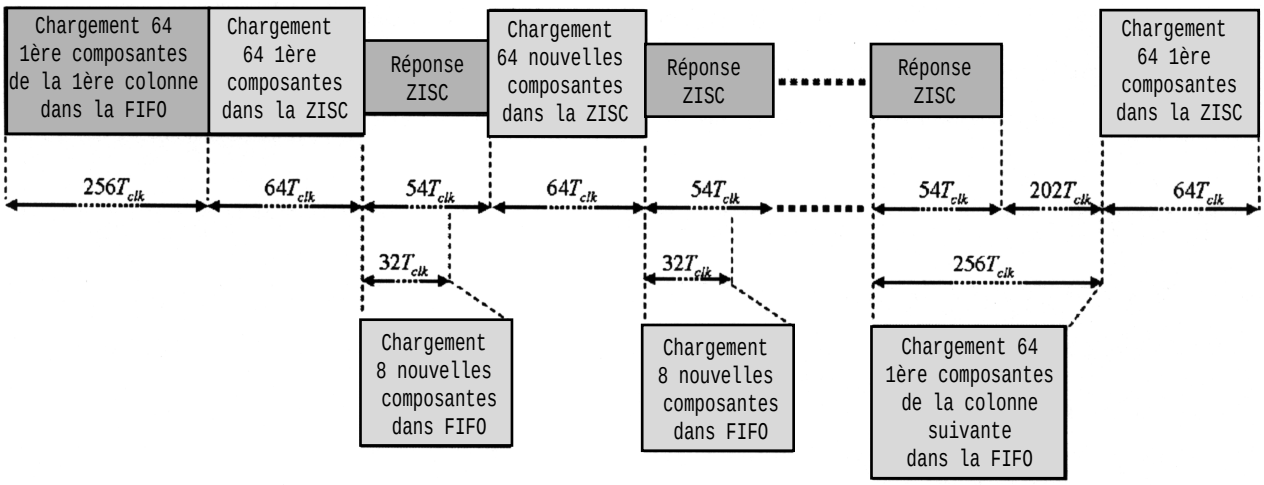

Figure 15. – Chronogramme du traitement par la carte Neurosight : le calcul d'une composante nécessite 4 périodes d'horloge, les vecteurs de 64 composantes sont chargés séquentiellement dans le ZISC. Le FPGA et le ZISC fonctionnent en mode « pipeline ».

8. seconde implantation matérielle sur une architecture à base d'un FPGA SpartanII-300

Nous proposons dans cette partie une seconde implantation matérielle. Tout d'abord, nous décrivons comme dans la section précédente l'architecture de la carte utilisée. Ensuite, nous détaillons chaque étape de l'implantation. Enfin, nous comparons les 2 implantations réalisées.

8.1. architecture de la carte et complexité du système

Nous utilisons pour cette seconde implantation une carte de développement de la société $MEMEC$ [25]. Cette plate-forme est basée sur un FPGA XILINX.SpartanII-300 contenant 3072 slices et 16 blocs mémoire internes de 4Kbits chacun. Un banc mémoire SRAM de 2 MOctets est implanté sur cette carte. Le module imageur peut directement être connecté sur la broche $Connecteur\ Capteur\ CMOS$. Le code VHDL écrit auparavant pour les interfaces (connecteur Capteur CMOS, port parallèle) reste le même. La différence fondamentale de cette carte $MEMEC$ avec la carte $Neurosight$ est qu'elle ne comporte aucun composant spécialisé intégrant des réseaux de neurones RBF. Nous avons implanté le réseau directement dans le FPGA. Contrairement à la réalisation précédente, nous avons maintenant la liberté d'implanter sur le FPGA des neurones cachés à notre convenance. Nous nous basons sur les résultats de simulations présentées dans la section 4. Nous considérons des imagettes à tester de taille $L_f \times C_f = 40 \times 32$ pixels. Les pas de balayage choisis sont $p_l = 2$ suivant les lignes et $p_c = 4$ suivant les colonnes. Les vecteurs d'entrée extraits des imagettes comportent 40 lignes $\times$ 8 composantes = 320 composantes. Les images à analyser ont toujours une taille de 288×352 pixels. Nous nous limitons là aussi à l'analyse de l'image avec son échelle d'origine (échelle = 1).

De la même façon que la section précédente, nous nous basons sur les équations établies dans la section 6 pour calculer la complexité de cette implantation. Dorénavant, le nombre d'imagettes à traiter est (voir équation (9)) :

$$\begin{aligned} V_f &= \lfloor \frac{L - L_f}{p_c} + 1 \rfloor \lfloor \frac{C - C_f}{p_l} + 1 \rfloor \\ &= \lfloor \frac{288 - 40}{4} + 1 \rfloor \lfloor \frac{352 - 32}{2} + 1 \rfloor = 10143 \end{aligned} \quad (22)$$

Le tableau 11 récapitule le nombre d'opérations à effectuer par la carte pour traiter une image originale. Ces résultats sont obtenus avec le nombre de neurones cachés $I = 15$ (voir section 6.2.4). Le nombre global de calculs s'élève à un peu moins de $99 \cdot 10^6$ opérations par image, ce qui représente environ 5 fois plus d'opérations que lors de l'implantation précédente.

8.2. implantation du système

Nous allons maintenant décrire l'implantation du système sur la carte $MEMEC$. La figure 16 présente l'organisation des tâches sur cette carte. Les contrôleurs des composants ainsi que leur lien de communications sont inchangés par rapport à la première implantation. L'ensemble du système est maintenant implanté sur le seul FPGA SpartanII-300 (à part la mémoire SRAM dédiée au stockage de l'image originale).

Les machines d'états contrôlant le système sont les mêmes que celles décrites à la section précédente (voir Figure 14) à la différence près que le ZISC est maintenant remplacé par un réseau neuronal intégré dans le FPGA.

Tableau 11. – Tableau récapitulatif des opérations nécessaires au traitement d'une image sur la carte $MEMEC$

	Additions	Soustraction	Divisions	Comparaisons	Nb.Total
Extractions des vecteurs	$1{,}11 \cdot 10^6$	0	370944	0	$1{,}48 \cdot 10^6$
Calculs de distance	$48{,}5 \cdot 10^6$	$48{,}7 \cdot 10^6$	0	0	$97{,}2 \cdot 10^6$
Réponses Neurones	0	0	0	294147	294147
Totaux	$49{,}6 \cdot 10^6$	$48{,}7 \cdot 10^6$	370944	294147	$98{,}97 \cdot 10^6$

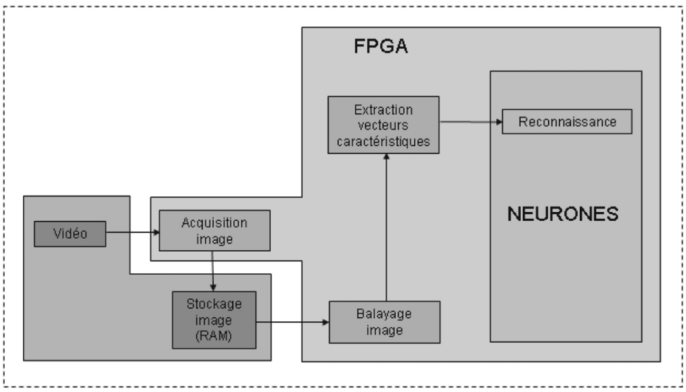

Figure 16. – Schéma de l'organisation des tâches sur la carte MEMEC.

Des implantations matérielles des réseaux de neurones RBF dans un FPGA ont été déjà réalisées pour d'autres applications [26] [27]. Il est important de préciser que le réseau implanté ici ne contient pas de logique d'apprentissage comme dans le ZISC. Cet apprentissage doit être réalisé par une machine hôte puis chargé dans le FPGA via le port parallèle ou la liaison JTAG (chargement lors de la configuration initiale du FPGA).

La figure 17 présente le schéma d'un neurone caché implanté sur le SpartanII-300. Les paramètres de chaque neurone caché (le vecteur appris, le seuil et la catégorie) sont stockés dans un banc de mémoire interne du FPGA. Les 320 composantes d'un vecteur à tester sont distribuées séquentiellement à chaque neurone caché via une FIFO. Chacun d'eux calcule les valeurs absolues des différences -composante par composante- entre le vecteur à tester et le vecteur appris. Un additionneur/accumulateur calcule en parallèle la somme de ces différences pour donner la mesure de distance $d_1(\boldsymbol{x})$. Enfin, cette distance est comparée au rayon afin de déterminer la réponse négative ou positive du neurone. Si le vecteur testé est déclaré comme reconnu, alors la catégorie à laquelle il appartient est indiquée en sortie du neurone. Dans le cas contraire, cette sortie est fixée à une valeur par défaut indiquant une non-reconnaissance.

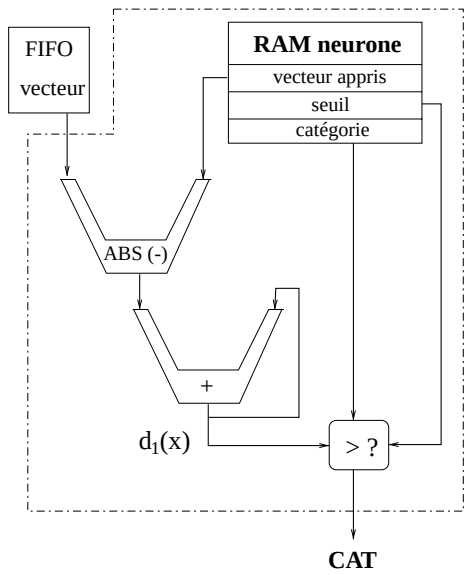

Figure 17. – Conception d'un neurone caché dans le FPGA.

8.3. analyses matérielle et temporelle de l'implantation

Comme pour la première implantation, nous décrivons la quantité de ressources utilisées en terme de *slices* et de blocs mémoires internes. Le tableau 12 présente les résultats de cette nouvelle implantation. Nous pouvons remarquer que le système utilise 27 % des ressources fonctionnelles, et 100 % des ressources mémoires disponibles sur le FPGA pour cette application de détection et de reconnaissance de visage à l'échelle 1.

Tableau 12. – Ressources matérielles utilisées sur le FPGA Xilinx SpartanII-300 : les pourcentages représentent les taux d'occupation des ressources.

	Slices	Blocs Mémoires RAM
SpartanII-300	3072	16
Contrôleur principal	97	1
Contrôleur neurones	52	0
15 neurones cachés	435	15
Contrôleur RAM/ Capteur CMOS	47	0
Contrôleur port parallèle	88	0
Logique contrôle global	108	0
Système complet	827 27 %	16 100 %

Nous allons maintenant étudier la vitesse de fonctionnement du système de cette seconde implantation. Il existe toujours les deux étapes séquentielles du processus que sont la capture d'une image dans le banc mémoire puis le traitement de celle-ci. L'acquisition d'une image se fait de la même façon que sur la carte *Neurosight*, c'est-à-dire $T_{image} = 16{,}6$ ms. Ensuite, le traitement des imagettes s'effectue selon le chronogramme présenté sur la figure 18. Le traitement d'une image complète nécessite $3{,}42 \cdot 10^6 T_{clk}$ [24]. La fréquence de l'oscillateur sur la carte est de 50 MHz ($T_{clk} = 20$ ns). Le traitement de toutes les imagettes est alors effectué en environ $T_{trait.} = 68{,}4$ ms.

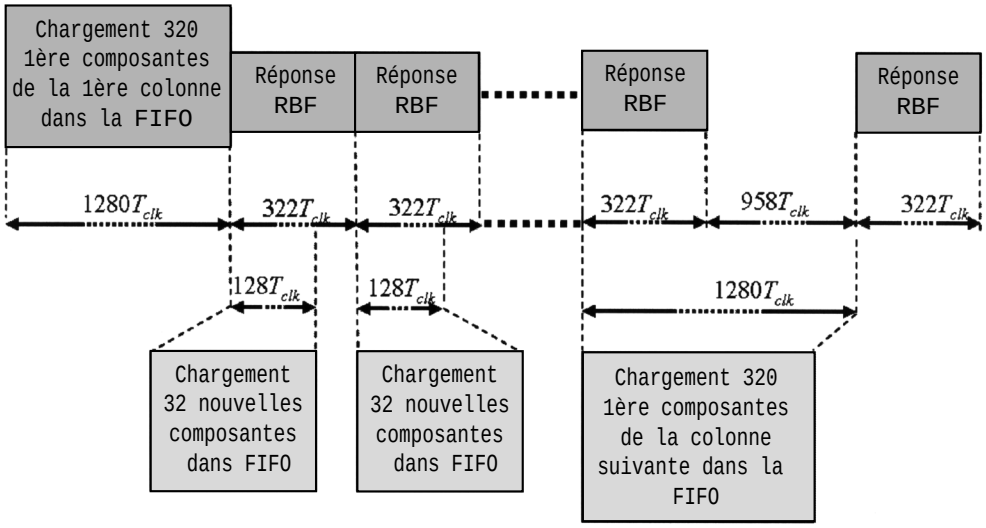

Figure 18. – Chronogramme du traitement par la carte *MEMEC*.

De même, nous avons effectué une simulation logicielle de cette seconde implantation. Le tableau 13 indique que le taux de reconnaissance correcte est de 92 %. Les vecteurs d'entrée du réseau RBF possèdent 320 composantes.

Tableau 13. – Résultats de la simulation logicielle de la seconde implantation.

Nb. Visages à reconnaître	Nb. composantes	N-Détec	F-Alarme	%Res-Correct
1796	320	123	20	92

8.4. comparaisons des deux implantations

Les ressources matérielles utilisées sur le FPGA de la carte *Neurosight* sont à peu près 3 fois plus faibles que celles utilisées sur le FPGA de la carte *MEMEC* en terme de slices (292 contre 827). D'autre part, seulement un bloc mémoire interne est utilisé sur la carte *Neurosight* alors que tous les blocs (16 blocs mémoire) le sont sur la carte *MEMEC*. Les taux d'occupation en terme de slices et de blocs mémoire des FPGA sont respectivement de 38 % et 12,5 % pour la carte *Neurosight* contre 27 % et 100 % pour la carte *MEMEC*. Par contre, le grand avantage de la carte *Neurosight* est de proposer 156 neurones totalement parallèles alors que nous disposons seulement de 15 neurones sur la carte *MEMEC*.

La vitesse de traitement est supérieure avec la carte *Neurosight* (25 images par seconde contre 14 images par seconde pour *MEMEC*) alors que la performance associée à cette carte est la plus faible (taux de réussite de 85 % contre 92 % pour la carte *MEMEC*). La différence de performance est essentiellement liée au fait que les vecteurs enregistrés sur le composant ZISC ont une longueur maximale de 64 composantes.

9. conclusions et perspectives

Nous avons établi un système permettant de détecter la présence de visages, de les suivre et de les reconnaître dans des séquences vidéo à l'aide d'un réseau de neurones de type RBF. La robustesse du système a été testée en utilisant 8 séquences vidéo et la banque de visages standard ORL. La meilleure performance a été obtenue avec la configuration suivante : un sous-échantillonnage d'un pixel/4 pour chaque ligne, la mesure de distance $d_0(\boldsymbol{x})$ et la fonction de réponse Gaussienne. En fait, la fonction de lissage comme pré-traitement suivi par un sous-échantillonnage des lignes et l'application de la distance $d_0(\boldsymbol{x})$ rendent le système moins sensible aux détails des visages et aux petites différences entre les exemples d'apprentissage et les imagettes à tester, d'où une meilleure généralisation.

Quant au problème d'éclairage, nous avons testé la méthode qui consiste à centrer et à normaliser les vecteurs d'entrée du réseau RBF afin de rendre le système moins sensible aux variations de luminance. Ceci augmente le taux de reconnaissance (environ 4,7%) dans des séquences vidéo où l'on observe une grande variation d'éclairage. Nous envisageons aussi une étape de pré-traitement par l'ajout de visages de synthèse à la base d'apprentissage, ceux-ci simulant les différentes conditions d'éclairage. Une autre solution basée sur la stratégie des réseaux de neurones évolutifs est de rafraîchir le système avec les derniers visages détectés et reconnus.

Nous avons démontré la faisabilité de détection et de reconnaissance de visages en temps réel en utilisant des cartes du commerce. L'application a été implantée d'abord sur la carte *Neurosight* à base du composant spécifique ZISC, ensuite sur la carte *MEMEC* à base de FPGA. Les vitesses de traitement sont respectivement de 25 images/seconde et de 14 images/secondes. Le nombre d'opérations réalisées par seconde sont de 0.53 Gops sur la carte *Neurosight* et de 1.39 Gops sur la carte *MEMEC*.

L'utilisation de la carte *Neurosight* nous évite la phase de conception du réseau mais le développement du contrôle des composants ainsi que des interfaces de communication reste à la charge de l'utilisateur. Le ZISC a pour avantage d'offrir une grande quantité de neurones cachés. De plus, son processus d'apprentissage intégré le distingue parmi toutes les autres puces neuronales. D'un autre côté, ses neurones cachés ont le défaut majeur de stocker des vecteurs de taille assez faible (64 composantes au maximum), ce qui n'est pas toujours adapté à la réalisation désirée. Une solution à ce problème est de proposer une nouvelle architecture du processeur ZISC avec une longueur accrue des vecteurs allant par exemple pour cette application jusqu'à 320 composantes.

En ce qui concerne la seconde implantation, la réalisation du réseau RBF sur le FPGA nous donne une grande souplesse. Nous pouvons choisir la mesure de distance à utiliser, la fonction de réponse ainsi que la longueur des vecteurs d'entrée. De plus, le fait de centraliser l'ensemble du système sur le même composant FPGA évite des processus de communications. Par contre, le nombre de neurones cachés est limité par les ressources matérielles disponibles sur le FPGA utilisé. A ce niveau nous pouvons améliorer cette approche en bénéficiant des performances des FPGA actuels en termes de nombre de portes logiques ($> 10^6$) et de taille mémoire (> 256 KOctets). Ceci nous permettrait d'intégrer un plus grand nombre de neurones dépassant ainsi la capacité du processeur ZISC.

Notre système qui intégre 15 neurones cachés permet de distinguer 2 visages avec une bonne fiabilité. L'extension de ce système à la reconnaissance d'un nombre plus important de visages (> 10) nécessite une centaine de neurones cachés ce qui entraîne des puissances de calculs supérieures à 10 Gops et ainsi de nouvelles architectures doivent être développées.

Aussi dans le but de décharger le volume important des calculs dans le FPGA, et ceci pour des applications complexes avec de fortes contraintes temporelles (nombre de visages important, multi-échelles, variation d'éclairage ...), nous étudions actuellement la conception d'une rétine artificielle en technologie CMOS (AMS-0,6μm). Cette rétine intégre dans le plan focal (photo-éléments) le pré-traitement de correction d'éclairage et d'extraction des vecteurs d'entrée.

De cette manière nous pensons aussi tester notre système sur d'autres applications de reconnaissance de formes comme la détection et le suivi en temps réel de véhicules par exemple.

BIBLIOGRAPHIE

[1] R. Férand, O.J. Bernier *and al.*, *A fast and accurate face detector based on neural networks*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.23, No.1, pp.42-53, January, 2001.

[2] Intrator, D. Reisfeld and Y. Yeshurnn, *Face recognition using a hybrid supervised/unsupervised neural network*, Pattern Recognition Letter, No.17, pp.67-76, January, 2001.

[3] H.A. Rowley, S. Baluja and T. Kanade, *Neural network-based face detection*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.20, No.1, pp.23-38, January, 1998.

[4] D. Valentin, H. Abdi and B. Edelman, *What represents a face : A Computational Approach for the integration of Physiological and Psychological Data*, Perception, Vol.26, pp.1271-1288, 1997.

[5] M. Slimane, T. Brouard *and al.*, *Apprentissage non-supervisé d'images par hybridation génétique d'une chaîne Markov cachée*, Traitement du Signal, Vol.16, No.6, pp.461-475, 1999.

[6] T.K. Leung, M.C. Burl and P. Perona, *Finding Faces in Cluttered Scenes using random Labeled Graph Matching*, Fifth Intl. Conf on Comp. Vision, Cambridge, MA, 1995.

[7] B. Moghaddam and A. Pentland, *Probabilistic Visual Learning for Object representation*, IEEE Trans. PAMI, Vol.19, pp.696-710, 1997.

[8] H. Abdi, D. Valentin and A. O'Toole, *A generalized auto-associator model for face semantic process*, In Optimization and neural network, edité par D.Levine (Erlbaum, Hillsdale), 1997.

[9] M. Rosenblum, Y. Yacoob and L.S. Davis, *Human expression recognition from motion using a radial basis function network Architecture*, IEEE Trans. On Neural Networks, Vol.7, No.5, pp.1121-1138, 1996.

[10] A.J. O'Toole, S. Edelman and H.H.B lthoff, *Stimulus-specific effects in face recognition over changes in viewpoint*, Vision Research, Vol.38, pp.2351-2363, 1998.

[11] A.J. Howell and H. Buxton, *Learning identity with radial basis function networks*, Neurocomputing, Vol.20, pp.15-34, 1998.

[12] I. Park and I.W. Sandberg, *Universal approximation using radial basis function networks*, Neural Computation , Vol.3, pp.246-257, 1991.

[13] M.T. Musavi,W. Ahmed *and al.*, *On the training of radial basis function classifiers*, Neural Networks, Vol.5, pp.595-603, 1992.

[14] E. Viennet, *Architecture connexionniste multimodulaire : Application à l'analyse de scène*, Thèse de doctorat, Université de Paris Sud, 1993.

[15] T. Sim, R. Sukthankar *and al.*, *Memory-Based face recognition for visitor Identification*, 4th IEEE International Conf. On automatic face and gesture recognition, Grenoble, France, 26-30 March 2000.

[16] J.M. Torres-Moreno, P. Velzquez-Morales and J.G. Meunier, *Classphères : un réseau incrémental pour l'apprentissage non supervisé appliqué à la classification de textes*, JADT 2000, pp 365-372, M. Rajman & J.-C. Chappelier éditeurs, EPFL.

[17] N. Boujemaa, C. Nastar and J. Malki, *Requêtes partielles sans segmentation pour la recherche d'images par le contenu*, 12eme Congres Francophone ARIF-AFIA RFIA (Reconnaissance de Formes et Intelligence Artificielle), Paris, 1-3 Fev. 2000

[18] *http://www.research.att.com/facedatabase.html*.

[19] site web : *www.general-vision.com*.

[20] A. Eide, *An implementation of the zero instruction set computer (zisc036) on a PC/ISA-bus card*, WNN/FNN, Invited Talk, 1994.

[21] K. Madani, G. Trémiolles and P. Tannhof, *Image processing - zisc036 neuro-processor based image processing*, Computer Vision, Vol.2085, pp.200-207, 2001.

[22] *The programmable logic data book*, Xilinx 1998 et site internet : *www.xilinx.com*.

[23] Omnivision, site web : *www.ovt.com*

[24] N. Malasné, *Localisation et reconnaissance de visage en temps réel : Algorithmes et Architectures*, Thèse de doctorat, Université de Bourgogne, 2002.

[25] site web : *www.memec.com*.

[26] M. Porrmann, U. Witkowski *and al.*, *Implementation of artificial neural networks on a reconfugurable hardware accelerator*, 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing, pp.243-250, 2002.

[27] M. Aberbour, *Architecture and design methodology of the RBF-DDA neural network*, Processing of the IEEE International Symposium on Circuits and Systems, Monterey, CA, USA, 1998.

Manuscrit reçu le 28 mai 2002

LES AUTEURS

Fan YANG

Fan Yang est maître de conférences de l'Université de Bourgogne depuis 2000. Elle enseigne actuellement l'informatique industrielle, l'automatisme et la robotique à l'IUT de Dijon. Ses travaux de recherche au sein du groupe Architecture du LE2I-CNRS UMR 5158 s'intéressent aux méthodes de traitement automatique d'images de visages 2D et 3D, et aux systèmes embarqués en temps réel.

Nicolas MALASNÉ

Nicolas Malasné est docteur de l'Université de Bourgogne depuis Novembre 2002. Ses sujets de recherche concernent d'une part la reconnaissance de formes, la classification, et d'autre part les systèmes embarqués en temps réel.

Michel PAINDAVOINE

Michel Paindavoine est professeur à l'Université de Bourgogne. Il enseigne le traitement du signal et des images à l'Ecole d'Ingénieurs ESIREM et à l'IUP Electronique et Image. Il est le directeur du laboratoire LE2I-CNRS UMR 5158 (Laboratoire d'Electronique, d'Informatique et d'Image). Il effectue ses travaux de recherche dans le domaine de l'Adéquation Algorithmes Architectures en traitement d'images.