Implantation temps réel d'un algorithme de localisation et de reconnaissance de visages sur un FPGA

Nicolas Malasné, Fan Yang, Michel Paindavoine, Dominique Ginhac

Laboratoire Electronique, Informatique et Image
Université de Bourgogne, Aile de l'ingénieur, BP 47870, 21078 Dijon Cedex, France
Nicolas.Malasne@u-bourgogne.fr, Fan.Yang@u-bourgogne.fr
Michel.Paindavoine@u-bourgogne.fr, Dominique.Ginhac@u-bourgogne.fr

Résumé –
Nous présentons dans cet article un système de localisation et de reconnaissance de visages sur un FPGA. Dans un premier temps, nous discutons notre choix algorithmique. Ensuite, nous décrivons l'architecture d'un réseau de neurones de type RBF. Puis, nous présentons les résultats obtenus pour la reconnaissance de visages. Après ceci, nous discutons de la complexité du système dans le but d'une implantation sur FPGA. Enfin, nous conclurons par les approches techniques futures nous permettant d'améliorer notre système.

Abstract –
In this paper, we present faces tracking and recognition. In a first time, we discuss the technique we used to realize such an application. Our aim is to elaborate a quite efficient algorithm wich will be able to respect the real time work. In a second part, we present the chosen method. We describe the RBF neural network used to realize to recognize faces. Then, we present our test results. In a third part, we discuss how we can implant this algorithm on an FPGA device. Finally, we talk about the future evolution.

1 Introduction

Un système de localisation et de reconnaissance de visages en temps réel permet de multiples applications dans les domaines des interfaces hommes machines, de la vidéo-conférence ou du contrôle d'accès par exemple.

Nous avons conçu un tel système en utilisant un réseau de neurone de type RBF (Radial Basis Function). Les classifieurs de type RBF appartiennent à la catégorie des classifieurs à noyaux. Une méthode RBF permet de dessiner un réseau neuronal doté d'une bonne capacité de généralisation et d'un nombre minimum de noeuds pour limiter le volume de calcul. L'utilisation d'une RBF est une technique d'interpolation dans un espace de haute dimension. Dans cet espace des caractéristiques, la méthode RBF consiste en un recouvrement de fonctions noyaux simples pour créer des régions de décisions complexes. Mark Rosenblum[1] a développé un système de reconnaissance d'expressions humaines basé sur l'analyse du mouvement en utilisant un réseau de type RBF. Howell et Buxton ont réalisé une identification de visages appris à l'aide d'un tel réseau[2]. Notre but est d'élaborer puis d'implanter un algorithme suffisamment efficace pouvant localiser et reconnaître des visages de tailles différentes dans des séquences vidéo avec un arrière plan quelconque.

2 Description du modèle et résultats de simulation

L'architecture d'un réseau RBF[3] [4] est composée de 3 couches (voir Figure 1). Chaque noeud caché applique une fonction noyau sur les données en entrée et la couche de sortie réalise une somme pondérée de ces fonctions noyaux. Chaque noeud est caractérisé par 2 paramètres associés qui sont le centre et la largeur de la fonction radiale. La valeur de sortie d'un noeud caché est d'autant plus élevée que la donnée à son entrée est proche de son centre. L'estimation de la distance d'une donnée à un centre peut s'évaluer grâce à plusieurs distances différentes mais la plus couramment utilisée est la distance euclidienne. La fonction noyau utilisée est une fonction gaussienne. La configuration complète du réseau est réalisée en déterminant les centres et les largeurs associées à chaque noeud ainsi que les poids des connexions entre la couche cachée et la couche de sortie. Cette dernière couche contient autant de neurones que de personnes à localiser et à reconnaître.

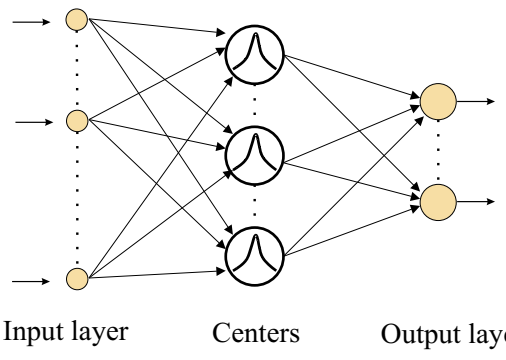

Fig. 1 – Architecture typique d'un réseau de neurone RBF

2.1 Apprentissage

Initialement, nous avons P points d'apprentissage dans un espace à N dimensions. Notre but est de réduire le nombre de points utiles dans cet espace. L'algorithme de *clustering* que nous utilisons est inspiré de celui proposé par Musavi[5].

1. Nous prenons n'importe quel point d'apprentissage P_k de n'importe quelle classe.
2. Nous trouvons le point P_l de cette même classe le plus proche de P_k en utilisant la distance euclidienne.
3. Nous calculons le barycentre entre ces 2 points. A ce nouveau point, nous associons un rayon $r = \frac{\|P_k, P_l\|}{2} + r_k$.
4. Nous calculons la distance D du barycentre au point le plus proche parmi tous les points de toutes les classes.
5. Si $D > \lambda r$, alors nous acceptons la fusion de P_k et P_l puis on recommence en 2. Si cette condition n'est pas satisfaite, nous refusons la fusion. Nous gardons dans ce cas les 2 points originaux et on recommence à partir de l'étape 1.
6. Nous répétons les étapes 1 à 5 jusqu'à ce que tous points de chaque classe aient été testés.

Finalement, nous obtenons les centres c_i et leur rayon associé $r_i (i = 1, ..., I \leq P)$ pour chaque neurone de la couche cachée.

λ est le paramètre de fusion ($\lambda > 0$). Quand λ augmente, la réduction des points est limitée mais la précision augmente. Nous utilisons la valeur $\lambda = 2$ pour notre application. Nous pouvons voir (Figure 2) le résultat après l'utilisation de cet algorithme appliqué à 2 classes dans un espace des caractéristiques à 2 dimensions.

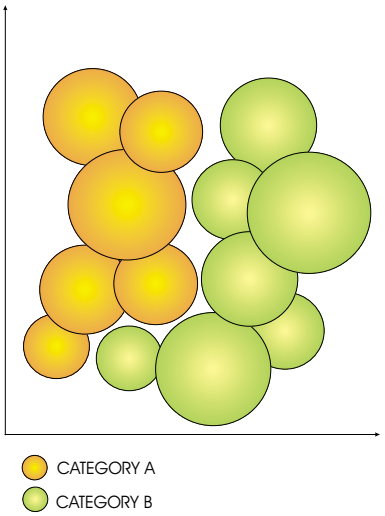

FIG. 2 – Régions de décisions dans un espace 2D

Nous obtenons 2 régions de décisions non linéaires. En effet, une méthode RBF permet d'obtenir des hypervolumes (formés d'hypersphères de dimension N) de décisions de n'importe quelle forme (non linéaire, convexe, parties disjointes).

2.2 simulation et tests

Nous utilisons les vignettes de taille 40×30 comme base d'apprentissage. Un filtre passe-bas est d'abord appliqué sur chaque vignette avant que celle-ci soit sous-échantillonnée. Chaque imagette fournit alors un vecteur d'apprentissage de 200 composantes. L'algorithme de *clustering* nous permet de réduire le nombre de centres utiles à la réalisation du réseau neuronal, de déterminer leur position et leur largeur associée.

FIG. 3 – Visages d'apprentissage

Notre séquence de test est constituée de 74 images (en luminance) de taille 240×320 contenant chacune 2 personnes se deplaçant dans une pièce. Aucun éclairage spécifique n'a été utilisé. Il est à noter que la taille des visages dans la séquence de test varie de 40×30 à 90×68. L'analyse de chaque image se fait par filtrage de l'image entière puis sous-échantillonnage de vignettes 40×30 successives. Ceci nous fournit les vecteurs à tester avec notre réseau de neurones précédemment défini. Les résultats du test sont présentés dans Tab.1, puis quelques images testées sont affichées (voir Figure 4). Les performances du système sont définies ci-dessous :

- Résultat correct : localisation et reconnaissance correcte de la personne.
- Non détection : un visage n'a pas été localisé.
- Fausse détection : un visage a été localisé là où il n'y en a pas.
- Fausse identification : un visage est correctement localisé mais mal reconnu.

TAB. 1 – Tests

nombre de personnes dans la séquence	141	100%
résultat correct	132	93.6 %
non détection	4	2.8 %
fausse détection	5	3.6 %
fausse identification	0	0 %

FIG. 4 – Résultats

3 Implantation du système sur FPGA

Afin de simplifier l'implantation de l'algorithme présenté précédemment, nous allons réduire la complexité de celui-ci. A l'origine, les vecteurs d'apprentissage des visages sont extraits en filtrant les imagettes puis en les sous-échantillonnant. En prenant un pixel sur 6, nous obtenions 200 composantes par vecteur. Nous approximons ce pré-traitement en calculant sur chaque ligne d'une imagette à apprendre 5 sommes de 6 pixels consécutifs (soient $5 \times 40 = 200$ composantes par imagette). L'extraction des vecteurs à analyser dans une image complète par le réseau de neurones se fera de la même manière.

Ensuite, l'implantation d'une fonction d'activation gaussienne pour les neurones est délicate à implanter. Nous approximons alors cette fonction par la fonction *porte* suivante :

$$\Pi_r(d) \begin{cases} = 1 & \text{si } 0 \leq d \leq r, \\ = 0 & \text{sinon.} \end{cases} \tag{1}$$

r : rayon de la fonction variant en fonction de l'apprentissage des visages prototypes.

Cette fonction porte nous permet également de simplifier la prise de décision en sortie du réseau de neurones. En effet, le dernier étage du réseau est alors superflue et la décision du réseau devient :

$$R_k = \Pi_{k_{1_j}} \vee ... \vee \Pi_{k_{j_k}} \vee ... \vee \Pi_{k_{J_k}} \tag{2}$$

R_k : décision binaire liée au visage k reconnu ou non.
$\Pi_{k_{1,...,J_k}}$: réponses binaires des J_k neurones cachés associés au visage k appris.

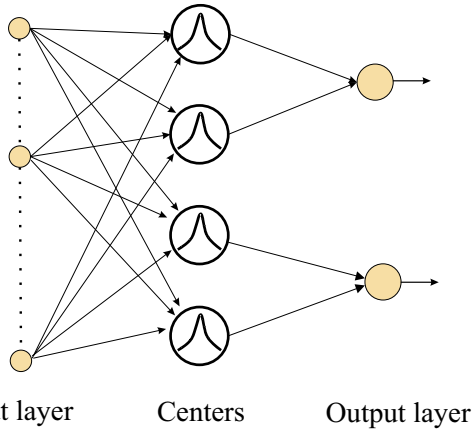

FIG. 5 – Architecture du réseau RBF utilisé

La mesure de similarité d'un vecteur en entrée par rapport à un centre du réseau se fait en utilisant la distance suivante :

$$L1 = \sum_{i=1}^{200} (C_{m_i} - p_{n_i}) \tag{3}$$

C_{m_i} : i^{eme} composante du centre m.
p_{n_i} : i^{eme} composante vecteur n.

Nous analysons des images 240×320 acquises à une cadence de 25 images/sec (une image$\equiv 40ms$). Le nombre de vignettes à analyser dans une image est déterminé par l'équation :

$$N_v = \lfloor \frac{L - L_v}{P_c} + 1 \rfloor \lfloor \frac{C - C_v}{P_l} + 1 \rfloor \tag{4}$$

L **:** nombre de lignes dans l'image.

C **:** nombre de colonnes dans l'image.

L_v **:** nombre de lignes d'une vignette.

C_v **:** nombre de colonnes d'une vignette.

P_c **:** pas de balayage suivant les colonnes de l'image.

P_l **:** pas de balayage suivant les lignes de l'image.

En prenant $L = 240$, $C = 320$, $L_v = 40$, $C_v = 30$ et $P_c = P_l = 2$, on obtient $N_v = 14746$ vignettes à tester en $40ms$. L'extraction des composantes d'un vecteur caractéristique lié à une imagette 40×30 s'effectue en additionnant les pixels consécutifs de B blocs sur chaque ligne. Le nombre d'additions nécessaires au calcul de tous les paramètres de toutes les imagettes est donc :

$$A_1 = (\frac{C_v}{B} - 1) \times B \times L_v \times N_v \tag{5}$$

A_1 : nombre d'additions nécessaires à l'extraction de tous les paramètres d'une image.
B : nombre de blocs sur une ligne.

En prenant $B = 5$, $L_v = 40$ et $C_v = 30$, nous avons $A_1 = 1000$ additions par imagette soit 14.7M additions pour extraire tous les vecteurs caractèristiques à tester dans une image.

Ensuite, chaque neurone donne un résultat binaire en calculant la distance $L1$ entre les I composantes du vecteur caractéristique à son entrée et les I composantes de son centre associé. Cela demande donc I soustractions et $I - 1$ additions par neurone. Nous avons alors :

$$A_2 = (I - 1) \times J \times N_v \tag{6}$$

$$S = I \times J \times N_v \tag{7}$$

A_2 : nombre d'additions nécessaires au calcul de la distance L_1 pour toute une image.
S : nombre de soustractions nécessaires au calcul de la distance L_1 pour toute une image.
I : taille des vecteurs caractéristiques.
J : nombre total de neurones cachés.

En prenant $N_v = 14746$ et $J = 10$, on obtient $A = A_1 + A_2 = 44.1$M additions et $S = 29.5$M soustractions.

La prise de décision n'utilise alors que des fonctions *ou binaire*. D'après l'équation 2, le nombre de comparaison est donné par l'équation suivante :

$$O = N_v \times \sum_{k=1}^{K} (J_k - 1) \tag{8}$$

O : nombre total de comparaison *ou binaire*.
K : nombre de visages appris.
J_k : nombre de neurones cachés associés au visage k appris.

Le nombre maximum d'opérations *ou binaire* est donné pour $K = 1$:

$$O = N_v \times (J - 1) \tag{9}$$

J : nombre total de neurones cachés.

soit 9 comparaisons dans notre cas par vignette analysée. Nous avons alors au total 73.7M opérations par image ou encore 1.84Gops.

L'implantation se fait sur une carte comportant un FPGA *XILINX. Virtex300*[6] fonctionnant à une fréquence maximale d'horloge de 100Mhz. Le temps réel est respecté en parallélisant l'extraction des vecteurs caractéristiques ainsi que les neurones (voir Figure 6).

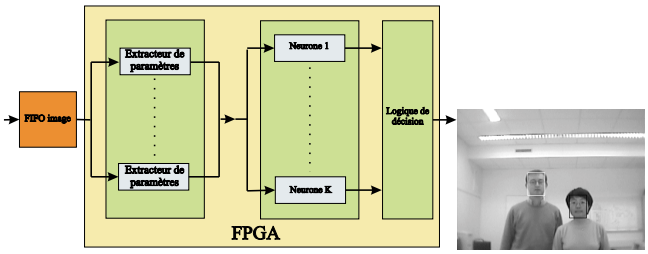

FIG. 6 – Dispositif global

4 Conclusions et perspectives

Notre travail a consisté à valider un algorithme de suivi et de reconnaissance de visages en temps réel afin de réaliser une implantation sur FPGA. Notre perspective à court terme est maintenant de valider notre implantation puis d'intégrer un prétraitement d'égalisation de la luminance des vignettes traitées. De plus, nous allons travailler à plusieurs échelles afin de détecter des visages dans une profondeur de champ plus importante. Une perspective à moyen terme est de réaliser l'extraction des vecteurs à analyser à l'aide d'une rétine artificielle, ce qui déchargerait le FPGA d'un volume de calcul conséquent.

Références

[1] M. Rosenblum, Y. Yacoob and S.V. Larry, "Human expression recognition from motion using a radial basis function network architecture," *IEEE Transaction on neural networks,* Vol.7, No.5, pp. 1121-1138, 1996.

[2] A.J. Howell, Y. Buxton and S.V. Larry, "Learning identity with radial basis function networks," *Neurocomputing,* Elsevier, Vol.20, pp. 15-34, 1998.

[3] M.J.D. Powell, "Radial Basis Functions for multivariate interpolation : A review," *Algorithms for approximation,* Clarendon Press, pp. 143-167, Oxford, 1987.

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

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

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