Détection et localisation de visages dans une scène : implantation parallèle sur un réseau de DSPs

Face detection and localization in a scene : parallel implementation on DSPs

par Fan YANG, Michel PAINDAVOINE

Laboratoire Le2i, aile de l'Ingénieur – Mirande, Université de Bourgogne, BP400, 21011 Dijon cedex
Tél (Fax) : 03 80 39 60 43
e-mail : fanyang@u-bourgogne.fr, paindav@u-bourgogne.fr

résumé et mots clés

Cette étude a été menée dans le cadre de l'élaboration d'une chaîne de traitement automatique d'images de visages dont la première étape est de détecter et localiser des visages dans une image vidéo. Nous avons porté la Transformée de Hough Floue Généralisée (THFG) sur un réseau de DSP TMS320C40 à l'aide du logiciel SynDEx. L'article expose plusieurs schémas de parallélisation de cet algorithme. Nous montrons comment optimiser l'implantation parallèle, en prenant en compte l'algorithme, les contraintes temps réel et en minimisant les composants matériels avec la méthodologie A^3 (Adéquation Algorithme Architecture). L'aspect de la granularité du parallélisme est aussi abordé lors de la spécification de l'algorithme.

Détection et localisation de visages, Transformée de Hough Floue Généralisée, Adéquation Algorithme Architecture, Implantation parallèle, Traitement en temps réel.

abstract and key words

This research concerns the field of development of a set of methods for automatic image processing. The first step is to detect and localize the face in an image. Our approach consists to approximate the face oval shape with an ellipse using the Fuzzy Generalized Hough Transformation. Considering the necessary computation amounts, we realize parallel implementations of this face detection algorithm on DSP using the programming environment SynDEx. In this paper, we present several graphs of parallel implementations. We show how the Algorithm Architecture Adequation methodology may aid to optimize, to distribute the real-time executives, and to minimize the hardware resources. The aspect of parallelism granularity also is discussed for algorithm specification.

Face detection and localization, Fuzzy Generalized Hough Transformation, Algorithm Architecture Adequation, Parallel implementation, Real-time processing.

1. introduction

La détection et la localisation de visages dans des images vidéo est un sujet de plus en plus exploré ces dernières années. Les domaines d'application d'un tel système sont nombreux : identification de visages, interface homme-machine, contrôle d'accès, *etc.* Par exemple, pour des applications concernant la transmission d'images à l'aide de lignes à faible débit, comme les vidéophones, il est très utile de connaître l'existence et la position du visage dans l'image afin d'appliquer un algorithme de compression d'image sélectif. Un autre exemple concerne la gestion de bases de données : la taille grandissante des bases de données de photos utilisées par les agences de presse pose de nombreux problèmes d'indexation et d'identification de photos contenant des visages.

Notre application s'intéresse au contrôle d'accès. La personne à identifier se positionne devant un poste fixe comportant une caméra CCD couplée à un micro-ordinateur. La première étape d'identification consiste à détecter et à localiser des visages présentés dans l'image. Le système doit être exploité dans des conditions générales : le nombre de visages dans l'image est inconnu, l'environnement (le fond, le décor, l'éclairage, ...) est quelconque, les visages sont de profil ou de face, enfin l'algorithme doit être suffisamment simple pour permettre, à l'aide d'une carte électronique insérée dans un micro-ordinateur, la détection et la localisation de visages le plus rapidement possible. Toutes ces contraintes nous ont conduit à utiliser une nouvelle version de la Transformée de Hough : la Transformée de Hough Floue Généralisée (THFG).

Par ailleurs la détection et la localisation de visages doivent être réalisées en un temps raisonnable et pour cela nous avons parallélisé et implanté sur plusieurs DSPs l'algorithme en utilisant une approche Adéquation Algorithme Architecture. La mise en œuvre de cette approche a été réalisée à l'aide du logiciel SynDEx (**Syn**chronous **D**istributed **Ex**ecutive)[1][2] qui apporte un développement de parallélisation plus rapide et, grâce à un module de prédiction de performances, permet de trouver une adéquation optimale entre l'algorithme à implanter et l'architecture disponible dans une optique temps-réel. Notre but à travers cette expérience est d'illustrer comment optimiser l'implantation parallèle, en prenant en compte l'algorithme, les contraintes temps réel, la granularité du parallélisme et en minimisant les composants matériels avec la méthodologie $\mathbf{A}^3$.

Dans la section 2, nous présentons l'algorithme implanté pour détecter et localiser des visages dans une scène, nous exposons les 3 étapes de ce traitement ainsi que les résultats obtenus. Après une description rapide du logiciel SynDEx et de l'architecture utilisée dans la section 3, nous étudions dans les sections 4, 5 et 6 des approches différentes d'implantations de cet algorithme sous forme parallèle à l'aide de SynDEx. Cette étude nous amène à montrer en conclusion qu'une adéquation optimale a été obtenue ce qui a permis d'atteindre une vitesse de traitement de 3,1 images par seconde (pour des images de 143 × 123 pixels).

2. description de l'algorithme : transformée de Hough floue généralisée

La transformée de Hough est largement utilisée en traitement d'images. Cette méthode fut introduite en 1962 par Paul Hough dans le but de détecter les trajectoires rectilignes de particules de haute énergie. Korosec *et al.* [3] sont les premiers à modéliser les contours extérieurs du visage par une ellipse, en appliquant une transformée de Hough standard. Partant de l'approche de Korosec *et al.*, Séguier [4] modélise les contours extérieurs du visage par une ellipse, et propose une nouvelle version de transformée de Hough (THFG) afin de détecter des petits visages dans des séquences d'images vidéo en appliquant le suivi dynamique de mouvement.

Par rapport à d'autres algorithmes de segmentation de visages, un des avantages de cette méthode réside dans la robustesse : elle permet de détecter des visages de petite taille, des visages atypiques (yeux fermés, avec des lunettes, ...), et aussi des visages de profil. En outre, l'utilisation de la transformée de Hough floue généralisée autorise une vitesse de détection assez rapide pour envisager un traitement en temps réel. Nous avons donc décidé d'adapter l'approche de Séguier [4] à une image vidéo en modélisant les contours extérieurs d'un visage par une courbe elliptique, et en recherchant ce type de courbe dans une scène. Cet algorithme s'effectue en trois phases (voir figure 1) :

– Détecter et affiner les contours dans l'image,

– Modéliser des contours avec une courbe elliptique,

– Décider sur la localisation des visages dans l'espace des paramètres.

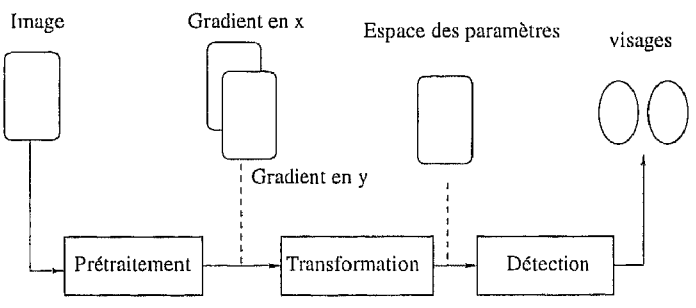


Figure 1. – Le système de segmentation de visages.

2.1. prétraitement : détection et affinage des contours

Pour détecter des formes elliptiques la transformée de Hough se base sur la valeur du gradient des niveaux de gris des contours de l'image dans la direction des lignes (*dLx*) et des colonnes (*dLy*). La qualité de cette information en chaque point de contour est donc extrêmement importante. Le filtre de Shen et Castan [5] est utilisé pour extraire les contours de l'image. Un balayage de l'image en 4 directions (haut, bas, gauche et droite) réalise des calculs récursifs d'ordre 1 ce qui permet d'obtenir directement les dérivées premières D_1 et seconde D_2 de l'image. L'équation 1 montre les calculs à effectuer pour les traitements horizontaux :

$$\begin{aligned} S_{Ld} &= (1-b)x(i) + bS_{Ld}(i-1), & i &= 1,\ldots,N \\ S_{Rd} &= (1-b)x(i) + bS_{Rd}(i+1), & i &= N,\ldots,1 \\ D_1(i) &= S_{Rd}(i+1) - S_{Ld}(i-1) \\ D_2(i) &= S_{Rd}(i+1) + S_{Ld}(i-1) - 2x(i) \end{aligned} \qquad (1)$$

avec $x(i)$: niveau de gris de pixel i, b : coefficient du filtre, compris entre 0 et 1.

Les contours extraits après comparaison de la valeur du gradient de chaque point à un seuil sont très épais. Un test est réalisé sur les signes des dérivées premières et secondes afin d'obtenir les contours d'un pixel d'épaisseur. En présence d'un contour, il existe effectivement une correspondance entre l'image et les signes de ses dérivées premières et secondes (voir figure 2). Si par exemple, en présence du signal représenté sur le graphe (c)

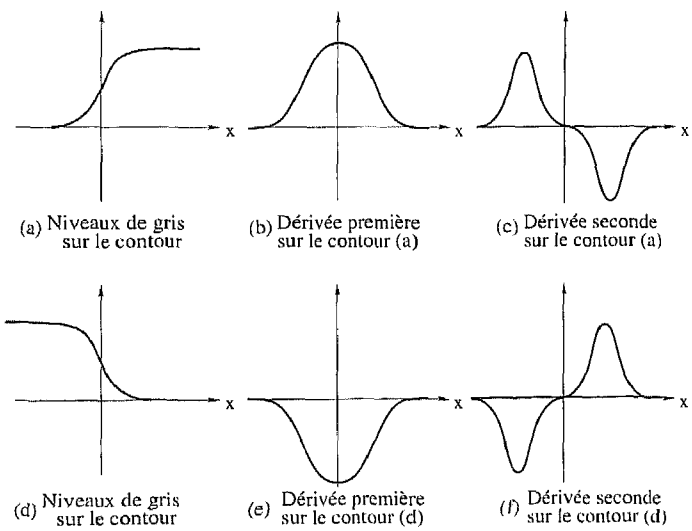


Figure 2. – Correspondance des signes entre le signal et ses dérivées. En haut : lorsque la dérivée première d'un pixel n est positive, si les signes des dérivées secondes D_2 de ses deux voisins satisfont les conditions $D_2(n-1) > 0$ et $D_2(n+1) < 0$, alors le pixel n est un point de contour. En bas : lorsque la dérivée première d'un pixel n est négative, le même test est réalisé avec les conditions $D_2(n-1) < 0$ et $D_2(n+1) > 0$.

de la figure 2 (dérivée seconde sur le contour), on observe une dérivée première négative, le point analysé ne sera pas considéré comme un point de contour.

2.2. modélisation des contours avec une courbe elliptique

La modélisation des contours extérieurs d'un visage par une courbe elliptique est réalisée à l'aide de la Transformée de Hough Floue Généralisée [4][6]. La transformée de Hough est largement utilisée en traitement d'images pour détecter les droites, les cercles et les ellipses. Elle comprend deux étapes principales : dans la première, on effectue une transformation des contours de l'image vers une grille ayant habituellement la même résolution que l'image (l'espace des paramètres). Une cellule de l'espace des paramètres est incrémentée d'une unité lorsqu'elle correspond à l'équation de la transformation (ex. une position possible du centre de l'ellipse). Dans la seconde, on réalise une détection dans cet espace pour localiser la forme recherchée. Le principal inconvénient de la Transformée de Hough Standard (THS) est qu'elle nécessite un énorme espace mémoire. De plus, le nombre d'opérations à effectuer croît exponentiellement avec le nombre de paramètres à détecter.

Un autre inconvénient de la THS, est qu'il faut, pour chaque point de contour, incrémenter un ensemble de cellules dans l'espace des paramètres qui correspondent par exemple aux centres de toutes les ellipses qui pourraient passer par ce point. La Transformée de Hough Généralisée (THG) exploite la valeur et la direction du gradient en chaque point de contour. Dans notre cas, la forme recherchée est une ellipse, donc le gradient en chaque point de contour est perpendiculaire à la tangente de la courbe générée par ce point et son voisinage [4]. Selon que le visage est plus sombre que le fond ou le contraire, un même point de contour aura une direction de gradient inversée. Il est donc nécessaire de considérer les deux directions opposées du gradient lorsqu'on estime la position du centre de l'ellipse qui pourrait passer par ce point. L'avantage de la THG est qu'elle permet de réduire la complexité de l'algorithme (plutôt qu'un ensemble de cellules, deux seulement sont considérées dans l'espace des paramètres) (voir la figure 3).

Dans la majorité des cas, les visages intéressants à détecter sont positionnés selon une direction verticale, donc il n'est pas indispensable de définir l'orientation précise des visages détectés. Une ellipse verticale a pour expression :

$$\frac{X^2}{lh^2} + \frac{Y^2}{lv^2} = 1 \qquad (2)$$

avec :

$$X = x - x_c \text{ et } Y = y - y_c,$$

(x_c, y_c) : le centre de l'ellipse, (x, y) : un point de contour de l'ellipse, lh, lv : demi-hauteur, demi-largeur de l'ellipse.

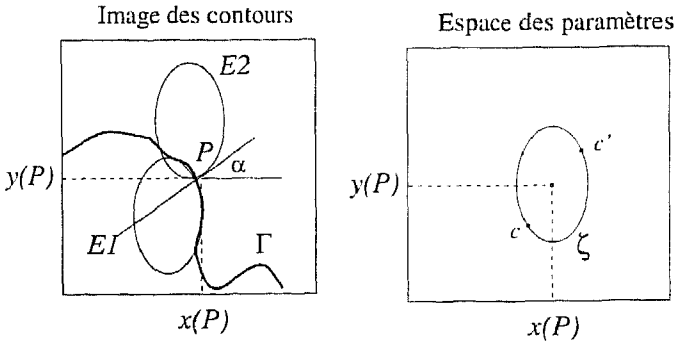


Figure 3. – Exploitation du gradient en chaque point de contour : Sur cet exemple, compte tenu des points voisins de P sur un contour Γ, P peut participer éventuellement à l'élaboration de l'ellipse $E1$, mais certainement pas à celle de l'ellipse $E2$. Dans ce cas il est possible d' incrémenter uniquement dans l'espace des paramètres deux cellules c et c' correspondantes à la présence de P dans l'image de départ (*cf.* [4]).

En exploitant le gradient (évalué sur les niveaux de gris Lx et Ly) en chaque point de contour, nous obtenons deux localisations possibles du centre de l'ellipse qui passe par le point de contour (x, y) :

$$(x_c, y_c) = (x + \Delta x, y + \Delta y) \quad (x_c, y_c) = (x - \Delta x, y - \Delta y) \quad (3)$$

$$\Delta x = \text{sign}(dLx) \frac{lh}{\sqrt{1 + \frac{lv^2}{lh^2}\left(\frac{dLy}{dLx}\right)^2}}$$

$$\Delta y = \text{sign}(dLy) \frac{lv}{\sqrt{1 + \frac{lh^2}{lv^2}\left(\frac{dLx}{dLy}\right)^2}} \quad (4)$$

avec : dLx, dLy : gradient évalué sur les niveaux de gris Lx et Ly.

Pour les formes elliptiques composées par les contours extérieurs des visages, les deux paramètres lh et lv sont dépendants l'un de l'autre : un visage est toujours plus haut que large. D'après la géométrie des visages, le rapport entre lh et lv varie généralement entre 1,2 et 1,7 [7]. Nous avons appliqué une proportion $\frac{lh}{lv} = 1.5$ à l'équation 4. En relation avec notre application de contrôle d'accès, la distance d entre la caméra et la scène à filmer est fixe et la taille des visages varie peu dans l'image. En effet, d'après nos mesures avec un objectif de 8 mm de focale et une distance d égale à 70 cm (représentant la position d'un individu en face d'une caméra positionnée sur un PC), si la distance d varie de ± 10 cm (soit 14 %), la variation de la taille du visage est de l'ordre de 5 %. Ainsi, les paramètres lh et lv peuvent être considérés comme des constantes pour notre application visée. Toutefois, pour augmenter la robustesse de notre système, nous tenons compte de cette faible variation de la taille de visages en introduisant la Transformée de Hough Floue Généralisée (THFG).

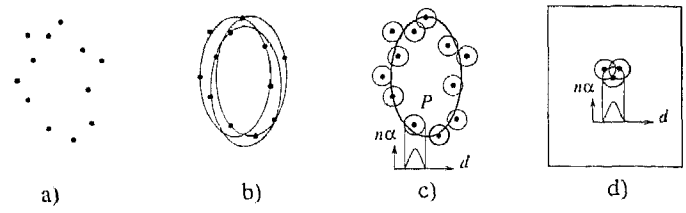


Figure 4. – Détection des courbes ressemblant à des ellipses : a) les points de contours, b) les trois ellipses détectées avec la THG, c) et d) le principe de la THFG et l'ellipse détectée par la THFG.

Les contours d'un visage ne définissent évidemment pas une ellipse parfaite, mais correspondent à une courbe qui se rapproche d'une ellipse. En 1994 est apparue une nouvelle version de la THG en tenant compte de l'imprécision du modèle : la Transformée de Hough Floue Généralisée [6]. Considérons un point de contour **P** dans l'image. On estime que **P** peut participer à l'élaboration d'une ellipse, mais, on lui donne la possibilité de générer un signal dans l'espace des paramètres permettant la détection des ellipses passant à côté de lui. Ceci revient à répartir l'information sur la cellule c (correspondant au point **P** dans l'image de départ) et son voisinage dans l'espace des paramètres. On peut voir cette approche sous l'angle de la logique floue et considérer le voisinage de la cellule c à travers une fonction d'appartenance monotone décroissante par rapport à la distance de la cellule c. La fonction de logique floue consiste à incrémenter d'une unité n la cellule correspondante c dans l'espace des paramètres et les cellules voisines de $n\alpha$, avec α qui décroît avec la distance entre la cellule considérée et la cellule c (voir la figure 4). Cette version de la Transformée de Hough Floue Généralisée nous permet d'augmenter la robustesse du système.

2.3. décision sur la localisation des visages

Cette étape correspond à la recherche des centres d'ellipses dans l'espace des paramètres. Nous détectons dans l'espace des paramètres, les valeurs maximales locales qui correspondent dans l'espace image aux centres des ellipses (visages). Si plusieurs visages sont détectés dans l'image, le système les considère les uns après les autres (les zones précédemment détectées sont remplacées par zéro) en utilisant un seuillage. Ainsi la vitesse d'exécution est d'autant moins rapide que le nombre de personnes présentées dans l'image augmente.

Figure 5. – Une image originale.

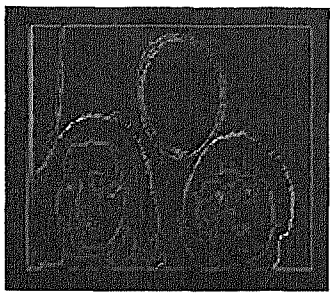


Figure 6. – Contours affinés.

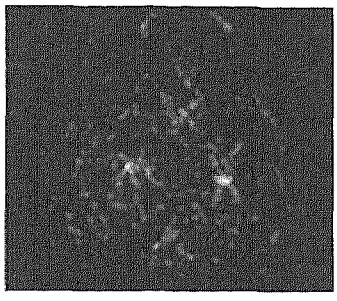


Figure 7. – Espace des paramètres.

Figure 8. – Visages localisés.

2.4. résultats expérimentaux

Les figures 5, 6, 7, 8 illustrent les résultats du traitement de chaque étape en exposant une image contenant 2 visages et une ellipse. Nous pouvons remarquer que dans l'espace des paramètres, nous avons trois points représentatifs : 2 visages plus une ellipse. La THFG ne nous permet pas de distinguer un visage d'une ellipse : une simple ellipse peut générer aussi un signal très fort dans l'espace des paramètres. Un post-traitement qui examine la ressemblance entre une zone d'intérêt et une structure de visage « moyen » [8] est envisagé. L'idée est de calculer la corrélation entre une zone d'intérêt détectée par la THFG et un masque de visage« moyen ». Cela nous permettra d'avoir un système robuste et rapide (la mesure de corrélation ne s'effectue que dans une zone restreinte). La figure 9 montre les résultats de la localisation des visages dans des images de fond complexe. De même, des tests montrent que ce système résiste bien au bruit présent dans la scène [9].

Figure 9. – Images et résultats : localisation de visages dans une scène

3. description de l'outil SynDEx et de l'architecture utilisée pour l'implantation parallèle

3.1. environnement SynDEx

L'adéquation d'un algorithme et d'une architecture consiste à étudier simultanément les aspects algorithmiques et architecturaux en prenant en compte leurs interactions, en vue d'effectuer une implantation optimisée de l'algorithme (minimisation des composants logiciels et matériels) tout en réduisant les temps de développement et les coûts de l'application étudiée. L'outil SynDEx [1] développé à l'INRIA est basé sur une formalisation unifiée des algorithmes, des architectures et des implantations, prenant en compte les contraintes (temps-réel, embarquabilité *etc.*), la nature distribuée des informations à traiter (multi-capteurs et actionneurs, données distribuées *etc.*), le besoin de portabilité. Cette approche formelle permet d'effectuer des vérifications lors de la conception de l'application et de poser des problèmes d'optimisation pour dimensionner au mieux les architectures de circuits spécialisés et/ou de machines. Cette approche permet d'une part d'améliorer les techniques de prototypage rapide, et d'autre part d'aborder de manière plus claire le problème de la conception conjointe logiciel-matériel. Ces deux points sont les grands enjeux du futur dans le domaine des implantations architecturales [1][2].

Les principales fonctionalités de SynDEx sont citées ci-après :

- description de l'algorithme sous forme d'un graphe flot de données,
- description graphique de l'architecture utilisée,
- placement/ordonnancement automatique des tâches sur l'architecture décrite,
- optimisation du temps de réponse de l'algorithme pour une implantation temps-réel,
- prédiction des performances de l'implantation parallèle,
- génération d'un exécutif avec communication sans interblocage.

3.2. architecture utilisée pour l'implantation parallèle

Nous avons utilisé une carte Transtech TDMB408, insérée dans un PC (voir la figure 10) pour implanter l'algorithme de détection automatique de visages. Cette carte contient trois processeurs de traitement du signal TMS320C40 de Texas Instrument connectés en ligne par plusieurs liens de communication. Un module CFG40 contenant un Frame-Grabber permet une acquisition d'image temps réel à partir d'une caméra et éventuellement un traitement de toutes les images acquises à 25 images par seconde. L'image acquise et les résultats sont transmis du C40 **root** au PC qui gère l'affichage.

Le choix de cette architecture a été dicté par la compatibilité entre le DSP TMS320C40 et le code exécutif généré par SynDEx ainsi que par les possibilités d'acquisition d'images temps réel de la carte.

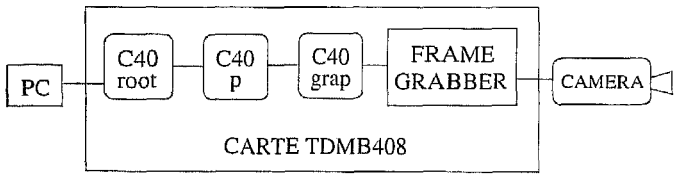


Figure 10. – Carte de 3 DSPs TMS320C40 et 1 module d'acquisition.

4. première implantation parallèle de l'algorithme THFG

En comparaison avec l'algorithme classique de la transformée de Hough [3], l'algorithme de la THFG nécessite moins de calculs. Malgré tout, le nombre de calculs reste relativement important. En effet, pour la phase de détection et d'affinage des contours, l'image originale est parcourue 8 fois (4 balayages d'images dans la direction X et 4 balayages d'images dans la direction Y pour le calcul de la dérivée première et de la dérivée seconde selon l'équation 1). Pour la phase de la transformée de Hough, on applique 2 fois l'équation d'ellipse plus la fonction de logique floue sur chaque point de contour.

A la fin, pour trouver la position des visages, l'espace des paramètres est balayé une fois. Au total, nous avons 10 accès en mémoire d'image.

Pour une image originale de taille $N \times N$, dans la phase de détection de contours, nous effectuons N^2 fois l'équation (1) et N^2 tests sur le signe des dérivées pour la direction X (idem pour la direction Y). Dans la phase de la THFG, nous réalisons d'abord N^2 tests afin de savoir si le pixel considéré appartient aux contours, ensuite pour chaque point de contour, l'équation (4) et la fonction floue sont appliquées. Dans la phase de détection dans l'espace des paramètres, nous cherchons dans tout l'espace les maxima locaux ce qui donne à l'algorithme de THFG une complexité en $O(N^2)$. Quand la résolution de l'image augmente, le temps de traitement d'une image croît rapidement d'où l'idée d'implantation parallèle de cet algorithme.

Dans le but de décrire notre première implantation parallèle, nous résumons ci-dessous l'algorithme séquentiel de la Transformée de Hough Floue Généralisée sous forme algorithmique classique avec le nom de chaque fonction :

Pour une image

DXP et DXM : lisser l'image en direction X (de gauche à droite, de droite à gauche)

DYP et DYM : lisser l'image en direction Y (de haut en bas, de bas en haut)

DX1 et DX2 : calculer les dérivées premières et secondes en direction X

DY1 et DY2 : calculer les dérivées premières et secondes en direction Y

TESTX et TESTY : tester les signes des dérivées selon les directions X et Y

THFG : pour chaque point de contour

* calculer le centre de l'ellipse qui pourrait passer par ce point

* incrémenter la cellule correspondante et son voisinage

DETECT : chercher les maxima locaux dans l'espace des paramètres

POSITION : encadrer les visages localisés en utilisant les maxima

Fin

La première implantation parallèle a été réalisée en mode flot de données à l'aide du logiciel SynDEx. Nous avons choisi un flot d'images, c'est-à-dire que pour chaque élément du flot de données (une image), on calcule d'abord parallèlement les dérivées premières et secondes de l'image suivant les directions X et Y. Ensuite, les 2 images de contours affinés obtenues (directions X et Y) sont divisées en blocs. Nous réalisons simultanément la THFG de chaque bloc avant de constituer l'espace final des paramètres. A la fin, pour décider de la localisation des visages, nous cherchons en même temps les maxima locaux de chaque bloc en suivant le même principe de parallélisme en bloc. Les positions des visages sont obtenues en utilisant les maxima locaux de chaque bloc. Cette spécification de l'algorithme parallèle est représentée en figure 11. Les arcs (dépendances de données) sont orientés de gauche à droite sur cette figure (idem pour toutes les figures suivantes des graphes de SynDEx).

La définition des fonctions est la suivante :

Image : C'est la fonction d'entrée qui donne une image pour laquelle nous allons effectuer une détection de contours avec les fonctions **DXP**, **DXM**, **DYP**, **DYM**, **DX1**, **DX2**, **DY1** et **DY2**.

DXP : Cette fonction prend l'image et effectue un lissage selon un balayage de gauche à droite.

DXM : Cette fonction prend l'image et effectue un lissage selon un balayage de droite à gauche.

DYP, DYM : Ce sont les mêmes fonctions que **DXP** et **DXM** en suivant la direction *Y*.

DX1, DX2, DY1, DY2 : Ce sont les fonctions qui calculent les dérivées premières et secondes de l'image selon les directions *X* et *Y*.

TESTX, TESTY : Ces fonctions de test examinent les signes des dérivées premières et secondes dans le but d'affiner les contours.

edgeX, edgeY : Ces fonctions prennent les contours affinés (en direction *X* et *Y*) et les divisent en trois blocs.

THFG1, THFG2, THFG3 : Chaque fonction réalise la THFG en utilisant un bloc d'image contour.

ACC : Cette fonction accumule les résultats des trois blocs dans l'espace des paramètres, puis divise l'espace des paramètres en trois blocs.

Detect1, Detect2, Detect3 : Ces fonctions cherchent les maxima locaux dans chaque bloc de l'espace des paramètres.

Position : Cette fonction fixe la position des visages en utilisant les maxima locaux de chaque bloc.

D : C'est la fonction de sortie qui découpe les visages présents dans l'image.

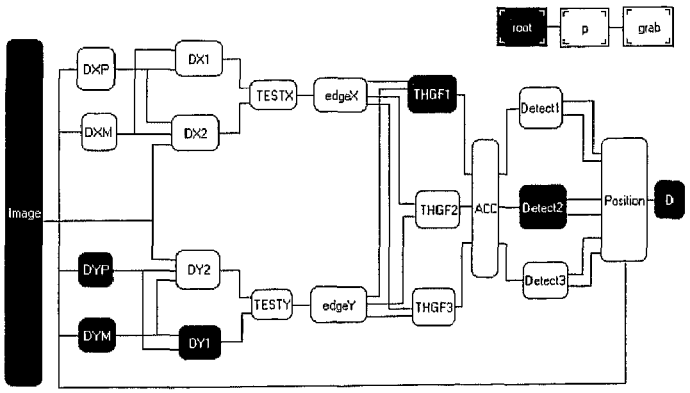


Figure 11. – Graphe logiciel et graphe matériel de l'algorithme : les fonctions en noir sont contraintes à s'exécuter sur le processeur root.

Cette spécification de l'algorithme exprimée sous forme de graphe flot de données utilise un grain de données de type image pour la phase de détection de contours : les données d'entrée et de sortie de chaque tâche ont la taille d'une image. Ce schéma du graphe logiciel restreint d'emblée le parallélisme de données potentiel. Nous verrons dans la section 6 comment surmonter ce problème.

SynDEx gère automatiquement le placement/ordonnancement de ces tâches sur la machine parallèle. Une option du générateur d'exécutif permet d'y insérer des macro-instructions de chronométrage [11]. Nous avons mesuré la durée d'exécution globale de la même application sur un et trois processeurs

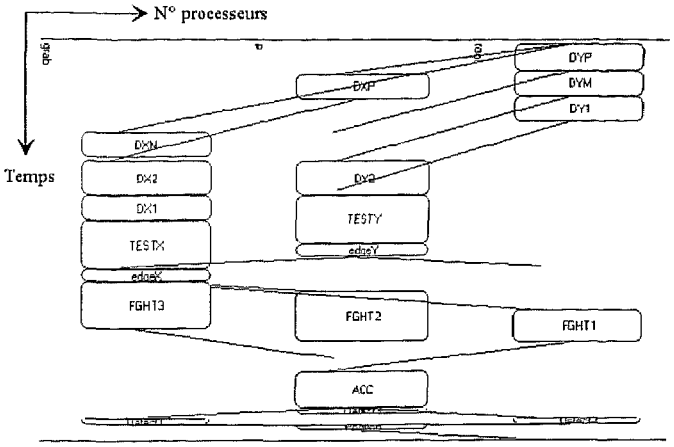


Figure 12. – Diagramme temporel : répartition des fonctions sur les 3 DSPs TMS320C40.

TMS320C40 ainsi que la durée de chaque fonction. La figure 12 montre la répartition des tâches sur 3 DSPs et le déroulement temporel de l'exécution.

L'implantation a été évaluée sur des images de taille 143 × 123. Le choix de cette résolution d'image est lié à la nature de la banque d'images de visages utilisée pour valider notre implantation. Cette banque d'images proposée dans [10] est une référence qui permet de comparer et valider de nombreux algorithmes de traitement automatique de visages.

Les durées d'exécution de chaque fonction sont indiquées dans le tableau 1. Les durées d'exécution globales sont données dans la deuxième colonne du tableau 2. La troisième colonne représente l'accélération effective quand on passe de 1 à 3 processeurs. Enfin, la dernière colonne correspond à l'efficacité de l'architecture parallèle. C'est le rapport, exprimé en pourcentage, entre l'accélération effective et l'accélération optimale.

L'efficacité du système est seulement égale à 0,45 (Accélération = 1,35). Ceci est dû au fait que avec ce graphe, l'exécution sur 3 DSPs nécessite beaucoup de communications (voir figure 12). Chaque traitement commence après que le DSP a obtenu la totalité de l'image traitée dans l'étape précédente ce qui correspond au flot de données images. Donc, le temps de communication est comparable à celui du temps de calcul si

Tableau 1. – chronométrage de l'exécution (*ms*)

T_{DXP}	T_{DX1}	T_{DX2}	T_{TESTX}	T_{THFG}	T_{ACC}	$T_{Detect1}$
25,463	24,611	33,728	46,795	42,137	35,580	6,2465

Tableau 2. – Performances de l'implantation parallèle

	Temps (*ms*)	Accélération	Efficacité
1 DSP	509,291	1	1
3 DSPs	377,252	1,35	0,45

bien que les processeurs restent inactifs pendant un grand pourcentage du temps de l'exécution. Aussi nous proposons dans les sections suivantes des optimisations de l'implantation parallèle pour augmenter les performances du système.

5. deuxième implantation : redimensionnement de l'architecture matérielle

Un avantage de SynDEx est que, selon le placement et l'ordonnancement délivrés par l'heuristique, on peut remettre en cause les résultats obtenus en changeant la spécification de l'algorithme ou de l'architecture, puis réaliser à nouveau le placement et l'ordonnancement des tâches. C'est donc un processus itératif qui se met en place puisque l'on peut continuer à modifier les graphes en vérifiant que cela améliore l'accélération.

Nous avons réalisé la deuxième implantation de l'algorithme de détection de visages en utilisant seulement 2 DSPs. Le graphe logiciel et le graphe matériel sont donnés dans la figure 13. D'après le chronométrage de l'exécution (voir tableau 3) et le diagramme temporel de SynDEx (voir figure 14), une accélération de 1,52 a été obtenue et l'efficacité du système est de 0,76 dans ce cas.

Nous pouvons remarquer qu'il n'y a qu'une seule communication de données pour la phase de détection et d'affinage de contour (la détection et l'affinage de contour dans les directions X et Y peuvent s'effectuer indépendamment). A ce niveau, la perte d'efficacité se situe plutôt au niveau des communications inter-processeurs.

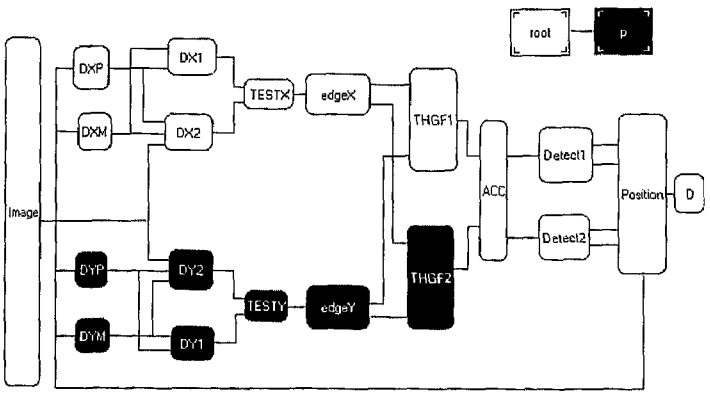


Figure 13. – Graphe logiciel et graphe matériel de la deuxième implantation.

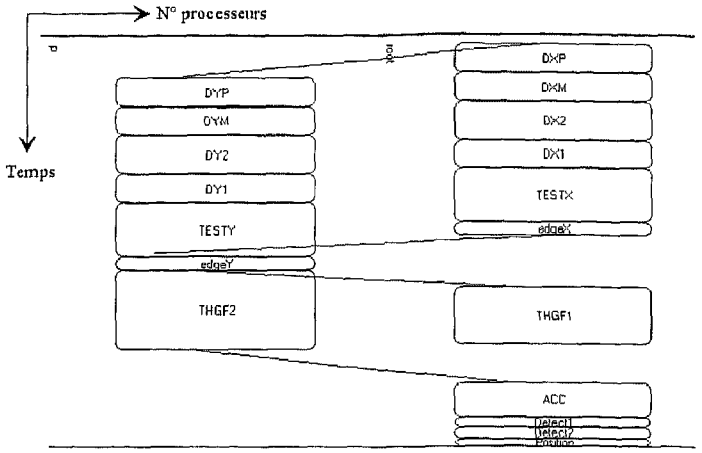


Figure 14. – Diagramme temporel de la deuxième implantation.

Tableau 3. – Chronométrage de l'exécution (*ms*)

$T_{1\ DSP}$	$T_{2\ DSPs}$	Accélération	Efficacité
509,291	334,927	1,52	0,76

6. troisième implantation : étude de la granularité des données

Le logiciel SynDEx spécifie l'application avec un graphe flot de données afin de faire apparaître le maximum de parallélisme. Les traitements d'images bas niveau répètent la même opération appliquée à des données différentes (ici des pixels différents). Evidemment le problème de la granularité des données se pose aussitôt car le grain pixel n'est pas celui qui est le mieux adapté à une implantation efficace [2]. En plus, une représentation parallèle en grain fin est limitée par l'explosion combinatoire du nombre de tâches.

Il faut noter que le choix du grain des données (ou encore le choix du découpage de l'image) est un problème assez difficile car la notion de granularité dépend à la fois de la taille des données à traiter, de la complexité des opérations à effectuer sur ces données et des performances des processeurs composant la machine parallèle. Un des avantages du logiciel SynDEx réside dans la représentation de l'application en « flot de données », nous avons la possibilité d'adapter facilement le grain des données en fonction de la taille de l'application de manière à toujours obtenir un graphe logiciel optimisé pour l'application considérée.

Considérons notre algorithme de détection de visages. Si nous le décrivons au niveau le plus fin de grain (pixel) en explicitant tous les calculs, nous exhibons ainsi à la fois le parallélisme de données et le parallélisme de tâches. Mais dans ce cas, nous sommes confrontés au problème de l'explosion combinatoire du nombre de tâches. Si nous choisissons le grain d'une image, nous ne laissons apparaître que l'enchaînement des fonctionnalités principales. C'est ce que nous avons fait pour la partie de détection et d'affinage de contours. Nous avons ainsi uniquement exhibé le parallélisme de tâches potentiel. Ainsi notre démarche est de choisir le grain des données qui permet de faire apparaître les parallélismes potentiellement exploitables.

Après avoir testé les deux implantations précédentes de l'algorithme, nous choisissons, dans une nouvelle étape, d'améliorer le parallélisme de l'algorithme en partageant l'image de départ en quatre bandes horizontales (verticales) appelées « imagettes » (voir figure 15). Nous faisons alors apparaître un nouveau niveau de grain de parallélisme où le nœud **DXP** est éclaté en quatre afin de mettre en évidence le parallélisme de données (**DXP***i* $i = 1, 2, 3$ et 4 traite chacun une imagette, idem pour **DYP***i*). Nous appliquons en parallèle quatre fois l'opération de calcul récursif d'ordre 1 à chaque imagette. Nous avons ici un parallélisme de données et un parallélisme de tâches à un niveau moins fin que le pixel et plus fin que l'image.

Nous pouvons remarquer l'avantage de cette implantation sur le diagramme temporel de SynDEx (voir figure 16). Au lieu d'attendre la réception d'une image entière pour commencer la fonction **DXP**, le nœud **DXP3** commence ses calculs dès l'arrivée d'un quart d'image. Donc le temps d'attente des données est divisé par 4 si on néglige le surcoût de communication (en réalité, il faut trouver un compromis entre la granularité et le surcoût de communication). De même, nous avons découpé les fonctions d'affinage **TESTX** et **TESTY** en deux sous-tâches pour gagner du temps au niveau des inter-communications. Par contre, même si nous décomposons les fonctions **THFGH** et

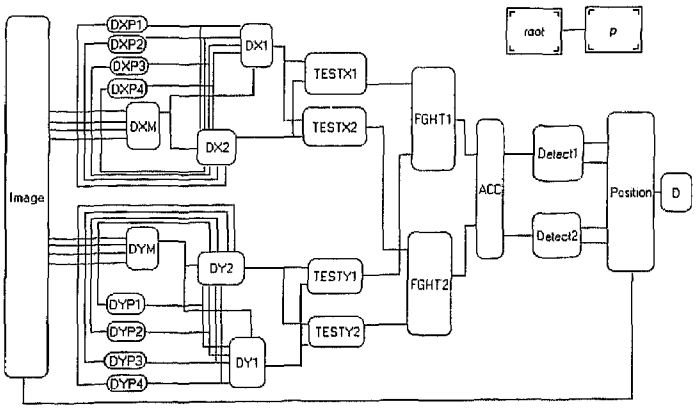


Figure 15. – Parallélisme de données et parallélisme de tâches : l'image de départ est partagée en 4 bandes horizontales (verticales) afin de mettre en évidence le parallélisme de données, l'enchaînement des tâches principales exhibe le parallélisme de tâches potentiel.

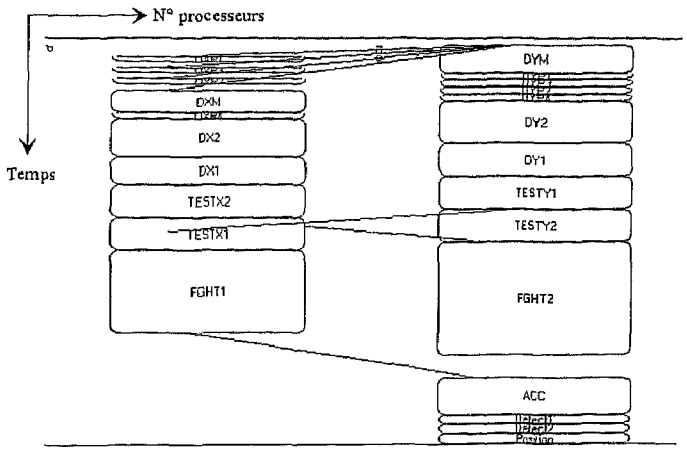


Figure 16. – Diagramme temporel de la troisième implantation.

THFGB, nous ne pouvons pas réduire le temps de communication, puisque le résultat de ces imagettes est une image entière.

Cette étude de granularité nous montre que le niveau de granularité intermédiaire (découpage de l'image en bandes) permet d'exploiter le parallélisme effectif d'une architecture à deux processeurs à partir du parallélisme potentiel de l'algorithme. En effet, dans ce découpage d'image, une efficacité de 0,8 et une accélération de 1,6 sont obtenues (voir le tableau 4).

Dans le but d'affiner notre étude sur l'impact de la granularité des données sur les performances de parallélisation nous avons aussi découpé de la même manière l'image en 1/2 images, 1/3 images, 1/5 images... jusqu'à 1/8 images. Les résultats son résumés dans le tableau 5 et confirment qu'à partir des découpag en 1/4 d'images les performances de parallélisation sont optimu

Tableau 4. – Chronométrage de l'exécution (*ms*)

T_{1DSP}	T_{2DSPs}	Accélération	Efficacité
509,291	318,291	1,6	0,8

Tableau 5. – Performances en fonction du grain de données

Grain	1/2 image	1/3 image	1/4 image	1/5 image	1/6 image	1/7 image	1/8 image
Acc	1.57	1.59	1.6	1.6	1.6	1.61	1.61
Eff	0.78	0.79	0.8	0.8	0.8	0.805	0.805

7. conclusion

Nous avons implanté notre algorithme de détection de visages sur un réseau de DSPs. Le but était d'exploiter le mieux possible les parallélismes potentiels inscrits dans la nature de l'applica-

tion pour se rapprocher du traitement en temps réel. L'approche classique consistant à mettre au point un algorithme en mono-processeur puis à tenter de le porter sur une architecture multi-processeurs est difficile à mettre en œuvre. Non seulement le temps de développement est très long, mais aussi la mise au point est particulièrement complexe en multi-processeurs. Tout au long de cet article, à travers une application de détection de visages dans une scène, nous avons montré qu'il est relativement facile et rapide grâce à une approche « Adéquation Algorithme Architecture » de modifier à la fois l'algorithme et l'architecture pour obtenir les meilleures performances possibles avec les composants que nous avons à notre disposition. Le passage d'une implantation à l'autre, s'effectue sans difficulté. En particulier il n'y a pas eu de mise au point multi-processeurs à effectuer. Tout cela réduit de manière importante le temps de développement de notre application.

A travers cette expérience nous avons pu montrer que la granualarité des données conditionne les performances de la parallélisation. En particulier dans le cadre de notre application de détection et de localisation de visages nous avons montré que le découpage en 1/4 d'images est optimum. Cependant cette étude a été réalisée manuellement à l'aide de l'outil SynDEx et en perspectives nous souhaitons poursuivre les travaux de recherche sur la modélisation de la granularité des données [12][13] pour des applications générales de traitement d'images. Les objectifs de ces recherches doivent permettre de faciliter le choix de la granularité en proposant à l'aide d'un outil de type SynDEx une automatisation de ce choix en spécifiant uniquement l'algorithme initial en représentation flots de données associé à une architecture cible.

Nous avons obtenu une vitesse de traitement de 3,1 images/seconde (pour des images de 143 × 123 pixels). Ceci nous permet de gagner du temps pour l'étape suivante d'identification de visages basée sur une approche neuronale [9].

Ces résultats ont été obtenus avec une architecture à base de DSPs TMS320C40. Compte tenu de notre démarche « Adéquation Algorithme Architecture », il est tout à fait envisageable d'intégrer d'autres types d'architectures (ex. TMS320C6x) dans notre approche d'optimisation de parallélisation qui est indépendante de l'architecture utilisée.

En perspectives, nous envisageons d'utiliser une machine hétérogène (FPGAs + DSPs) pour augmenter les performances du système. L'idée est de distribuer les tâches régulières (détection de contours en particulier) vers les FPGAs afin de profiter de la haute vitesse de ces circuits, et d'effectuer avec les DSPs les tâches de calculs de plus haut niveau.

BIBLIOGRAPHIE

[1] Y. Sorel and C. Lavarenne, *Real-time Embedded Image Processing Application Using the A³ methodology*, IEEE. International Conference on Image Processing, Nov. 1996.

[2] Y. Sorel, C.Lavarenne, *SYnDEx v4.2 INRIA : CAD for the optimized implementation of real-time algorithms on multicomponent architecture*, Technical Report, INRIA, 1996.

[3] J. Korosec, L.Gyergyek *and al.*, *Face contour detection based on theHough transformation*, Automatika, Vol. 31, 1990.

[4] R. Seguier, *Détection et localisation de visages dans des séquences d'images vidéo*, Thèse soutenue à l'Université de Rennes I, 1995.

[5] J. Shen, S. Castan, *An optimal linear operator for step edge detection*, Graphical Models and Image processing , Vol. 54, 1991.

[6] J. Han, L.T. Koczy and T. Poston, *Fuzzy Hough Transform*, Pattern Recognition Letters, Vol. 15, 1994.

[7] V. Govindaraju, S.N. Srihari and D. Sher, *Caption-aoded face location in newspaper photograps*, International Conference on Pattern Recognition, 1992.

[8] M. Turk, A. Pentland, *Eigenface for recognition*, Journal of Cognition Neuro-Science, Vol. 3, 1991.

[9] F. Yang, *Traitement automatique d'images de visages : Algorithmes et Architecture*, Thèse soutenue à l'Université de Bourgogne, Juin, 1998.

[10] D. Valentin, H. Abdi, A.J. O'Toole, *Categorization and identification of human face images by neural networks : A review of the linear autoassociative and principal component approaches*, Journal of Biological Systems, Vol. 2, 1994.

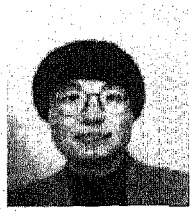
[11] F. Ennesser, C. Lavarenne et Y. Sorel, *Méthode chronomètrique pour l'optimisation du temps de réponse des exécutifs SynDEx*, INRIA Research Report, No.1769, 1992.

[12] L. Haas, F. Yang, M. Paindavoine et C. Milan, *Adéquation de l'algorithme de Canny-Deriche généralisé sur architecture DSP avec l'environnement SynDEx*, Traitement du signal, Vol. 14, No.6, 1997.

[13] C. Aiglon, C. Lavarenne, Y. Sorel *et al.*, *Utilisation de SynDEx pour le traitement d'images temps réel*, INRIA Research Report, No.2968, Sep. 1996.

LES AUTEURS

F. YANG



Fan YANG est docteur de l'Université de Bourgogne depuis 1998. Elle enseigne actuellement l'informatique industrielle, l'automatisme et la robotique à l'IUT de Dijon. Ses travaux de recherche au sein du groupe Electronique du LE2I s'interessent aux méthodes de traitement automatique d'images de visages, et aux systèmes embarqués en temps réel.

M. PAINDAVOINE

Michel PAINDAVOINE est professeur à l'Université de Bourgogne. Il enseigne le traitement du signal et des images à l'Ecole d'Ingenieurs ESIREM et à l'IUP Electronique et Image. Il effectue ses travaux de recherche au LE2I (Laboratoire d'Electronique, d'Informatique et d'Image) dans le domaine de l'Adéquation Algorithmes Architectures en traitement d'images.

LOUIS-JEAN avenue d'Embrun, 05003 GAP cedex Dépôt légal : 830 — Octobre 2000