Adéquation de l'algorithme de Canny-Deriche généralisé sur architecture DSP avec l'environnement SynDEx

Canny-Deriche Optimized Algorithm Adequation on DSP's Architecture Using SynDEx Environment

par Laurent HAAS*, Fan YANG *, Michel PAINDAVOINE*, et Claude MILAN***

* LE2I - Laboratoire d'Electronique, Informatique et Image
** ESIREM Université de Bourgogne
Aile des Sciences de l'ingénieur
BP 400 – 21011 DIJON Cedex.
Membres du GDR ISIS.
Email : Laurent.HAAS@U-Bourgogne.fr

résumé et mots clés

La détection de contour est un processus important de l'analyse d'image. La programmation parallèle, facilitée par l'outil SynDEx, permet d'accélérer l'implantation de ces algorithmes. L'article expose plusieurs schémas de parallèlisation de l'algorithme de Canny-Deriche optimisé. Ces descriptions logicielles sont analysées et l'accélération selon le nombre de processeurs est présentée. Ces schémas sont ensuite portés sur une carte à base de DSP TMS320c40 pour une comparaison des résultats expérimentaux et théoriques.

SynDEx, DSP TMS320C40, filtres récursifs, algorithme de Canny-Deriche.

abstract and key words

The edge detection is an important process of the image analysis. The parallel programming, easier with the SynDEx software, allows to speed up the implementation of these algorithms. This paper shows a few parallelization schemes of the Canny-Deriche optimised algorithm. These software descriptions are analysed and the speed up according to the number of processor is presented. These schemes are implemented on DSP TMS320C40 for a comparison between experimental and theoretical results.

SynDEx, DSP TMS320C40, Recursive Filters, Algorithm of Canny-Deriche.

1. introduction

Les applications de traitement d'images temps réel telles que le contrôle de qualité non destructif ou la vision robotique font appel à des algorithmes qui se décomposent selon trois niveaux :

– le bas niveau concerne les prétraitements d'images qui utilisent des techniques de filtrage linéaire ou non linéaire (élimination de bruit, détection de contours,...). Les données à traiter sont de type image en entrée et image en sortie;

– le moyen niveau s'intéresse à l'extraction des primitives à partir de l'image traitée au premier niveau. Dans ce cas la structure des informations est de type image en entrée et liste en sortie. A titre d'exemple nous pouvons évoquer les algorithmes de codage de contours (code de Freeman, transformée de Hough,...);

– le haut niveau concerne la fusion des primitives (listes) et effectue l'interprétation de la scène observée.

Pour répondre aux exigences de temps réel, le parallélisme est aujourd'hui une bonne solution pour accélérer l'exécution des algorithmes de traitement d'image évoqués précédemment. L'accélération pour une machine composée de P processeurs est au maxi-

mum P, cependant cette programmation en parallèle présente souvent des difficultés.

Le logiciel SynDEx V3.6 (Synchronous Distributed Executive), développé à l'INRIA [1], apporte une programmation parallèle plus facile et, grâce à un module de prédiction de performances, permet de trouver une adéquation entre l'algorithme à implanter et l'architecture à utiliser dans une optique temps-réel. Ce logiciel dédié en particulier aux applications de traitement du signal et des images a déjà montré son intérêt pour la segmentation des images [2][3]. Nous avons utilisé ce logiciel pour décrire différentes implantations d'un algorithme de traitement d'images « bas-niveau » : l'algorithme de détection de contours en utilisant le filtre de Canny-Deriche[4] généralisé[5]. Ce filtre a pour principale caractéristique d'extraire un contour en particulier pour des images fortement bruitées et floues. Sa réalisation numérique apparaît sous forme d'un filtre récursif d'ordre 3.

Des approches d'implantation de type ASIC et FPGA ont déjà été effectuées sur ce type d'algorithme [6][7][8][9][10]. Notre but à travers cet article est d'illustrer une évaluation de SYnDEx à travers l'implantation du filtre de Canny-Deriche sur une machine multi-DSPs.

Après une description rapide dans la section 2 du logiciel SynDEx et de l'architecture utilisée , nous présentons dans la section 3 le filtre de Canny-Deriche généralisé . Le type de granularité des données permet d'envisager différentes implantations possibles. Nous étudions ainsi dans les sections 4, 5, 6 et 7 des approches différentes d'implantation de ce filtre sous forme parallèle à l'aide de SynDEx, ces approches mettant en oeuvre différents modèles de granularité des données. Ceci nous permet de donner en conclusion une comparaison des performances de ces trois implantations obtenues avec SynDEx dans le domaine du traitement d'images « Bas-niveau » avec une orientation temps réel.

2. description de l'environnement SynDEx et de l'architecture utilisée pour l'implantation parallèle

2.1. l'environnement SynDEx

Le logiciel SynDEx V3.6 est développé à l'INRIA dans le cadre du projet SOSSO. C'est un environnement d'aide au développement et à l'implantation d'applications temps réel sur architecture multi-processeurs. Il se rapproche d'autres outils d'aide à la parallélisation et à la modélisation comme ESPION[11], PARSIM[12], RPPT[13] ou TRAPPER[14]. Les principales fonctionnalités de SynDEx sont détaillées ci-après :

– description graphique de l'algorithme sous forme d'un graphe flot de données (les données sont vues comme un flux qui passe d'un nœud du graphe à un autre sans variable statique extérieure aux nœuds);

– description graphique de l'architecture utilisée et placement/ordonnancement automatique des différentes tâches sur l'architecture décrite avec une optimisation du temps de réponse de l'algorithme pour une implantation temps réel. Cela permet aussi de changer d'architecture sans modifier l'algorithme;

– prédiction des performances de l'algorithme avant implantation;

– génération d'un exécutif (en C pour TMS320C40 ou Transputer T800) avec communications sans interblocage et, optionnellement, des instructions de chronométrage permettant de mesurer les performances temps réel de l'implantation.

2.2. architecture utilisée pour l'implantation parallèle

Nous avons utilisé une carte Transtech TDMB408, insérée dans un PC 486DX50. Elle contient trois processeurs de traitement du signal TMS320C40(6 ports de communication chacun) de Texas Instruments connectés en ligne par plusieurs liens 8 bits(canal de communication). Le nombre de liens utilisés pour relier les processeurs, dans cette étude, influera très peu sur les performances observées. 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. La restitution des images se fera par transmission du C40 root au PC qui gérera l'affichage. Le choix de cette architecture a été dicté par la compatibilité entre le TMS320C40 et le code exécutif généré par SynDEx ainsi que par ses possibilités d'acquisition d'images temps réel.

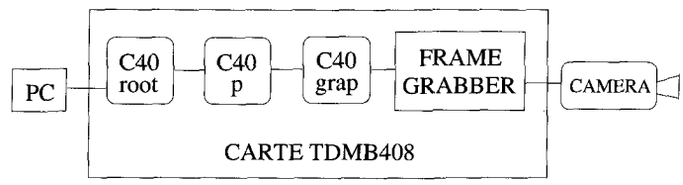


Figure 1. – Carte de 3 DSP TMS320C40 et 1 module d'acquisition.

3. le filtre de Canny-Deriche généralisé

Ce filtre permet de détecter les contours d'une image bruitée et floue en tenant compte de paramètres liés au rapport signal sur

bruit et à la localisation des contours [7]. Ce filtre fonctionne à partir d'un modèle de contour prédéfini donné figure 2, il a pour équation :

$$C(x) = \begin{cases} 1 - \dfrac{e^{-sx}}{2} & x \geq 0 \\ \dfrac{e^{sx}}{2} & x \leq 0 \end{cases} \quad (1)$$

On cherche à maximiser les mêmes critères que ceux de Canny [7] : Le rapport signal sur bruit (RSB), la localisation (L) et la suppression des réponses multiples (MRC) :

$$RSB = \frac{\int_{-\infty}^{+\infty} C(-x)f(x)dx}{\eta_0\sqrt{\int_{-\infty}^{+\infty} f^2(x)dx}} \quad L = \frac{\int_{-\infty}^{+\infty} C'(-x)f'(x)dx}{\eta_0\sqrt{\int_{-\infty}^{+\infty} f'^2(x)dx}} \quad (2)$$

$$MRC = \sqrt{\frac{\left|\int_{-\infty}^{+\infty} f'^2(x)dx\right|}{\int_{-\infty}^{+\infty} f''^2(x)dx}}$$

$f(x)$ étant la fonction du filtre, $C(x)$ le contour prédéfini et η_0 la variance du bruit.

En maximisant le produit RSB $\times$ L sous la contrainte MRC on trouve la forme générale de $f(x)$:

$$f(x) = ke^{\alpha x}\sin(\omega x) + e^{\alpha x}\cos(\omega x) - e^{sx} \text{ avec } x \leq 0 \quad (3)$$

On peut séparer cette fonction en deux fonctions causale et anti-causale :

$$\begin{aligned} f^+(x) &= ke^{-\alpha x}\sin(\omega x) + e^{-\alpha x}\cos(\omega x) - e^{-sx} \\ &\qquad \text{avec } x \succ 0 \\ f^-(x) &= ke^{\alpha x}\sin(\omega x) + e^{\alpha x}\cos(\omega x) - e^{sx} \\ &\qquad \text{avec } x \leq 0 \end{aligned} \quad (4)$$

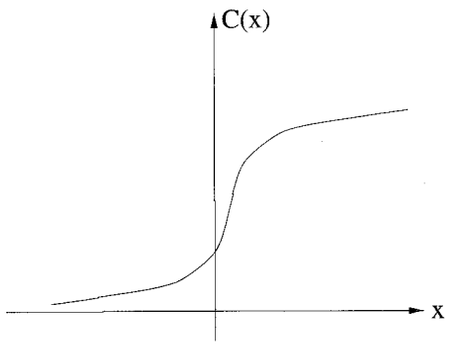


Figure 2. – Contour Prédéfini.

Pour obtenir une implantation numérique de ce filtre(une fois normalisé) sous forme récursive il suffit de calculer la transformée en Z de $f_+(x)$ et $f_-(x)$:

$$\begin{aligned} y^+(i) &= a_1x(i-1) + a_2x(i-2) + a_3y^+(i-1) \\ &\quad -a_4y^+(i-2) + a_5y^+(i-3) \\ y^-(i) &= -a_1x(i+1) - a_2x(i+2) + a_3y^-(i+1) \\ &\quad -a_4y^-(i+2) + a_5y^-(i+3) \end{aligned} \quad (5)$$

C'est un filtre récursif d'ordre 3 qui nécessite 4 lectures de l'image : de gauche à droite, de droite à gauche, de haut en bas et de bas en haut. A chaque lecture une fonction du filtre est calculée. C'est la somme de ces 4 fonctions qui donne le contour final. Il faut encore faire subir à l'image du contour un processus identique mais de lissage, qui se calcule de la même façon que le gradient, pour une détection optimale des contours d'images bruitées.

Ce filtre peut s'implanter sous la forme récursive d'ordre 3 suivante :

$$\begin{aligned} y(n) &= a_1x(n-1) + a_2x(n-2) + a_3y(n-1) \\ &\quad -a_4y(n-2) + a_5y(n-3) \end{aligned} \quad (6)$$

ou sous la forme d'une suite de 3 filtres récursifs d'ordre 1 :

$$\begin{aligned} y_2(n) &= a_1x(n-1) + a_2x(n-2) + \gamma y_2(n-1) \\ y_1(n) &= y_2(n) + \gamma y_1(n-1) \\ y(n) &= y_1(n) + \beta y(n-1) \end{aligned} \quad (7)$$

Les $x(n-1)$ et $x(n-2)$ représentent les 2 pixels qui précédent le pixel courant $x(n)$ pour lequel on calcule le gradient $y(n)$. Les coefficients $a1$ et $a2$ changent de signe lorsqu'on lit l'image de la droite vers la gauche et du bas vers le haut. Les 2 implantations ci-dessus seront utilisées dans 2 algorithmes différents.

Les résultats du filtre de Canny-Deriche généralisé sur une image normale et sur une image bruitée sont donnés en figures 17, 18, 19 et 20. Les niveaux de gris des pixels ont été codés sur un octet (0-255) et les calculs sont faits avec des mots de 32 bits. Ces résultats ont été obtenus en utilisant l'architecture décrite dans le paragraphe (2.2) et l'exécutif a été généré par SynDEx comme celà est détaillé ci-après.

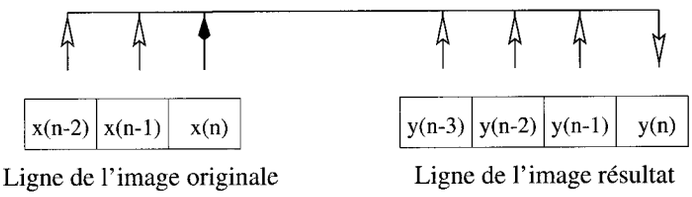


Figure 3. – Schéma de calcul d'un point $y(n)$.

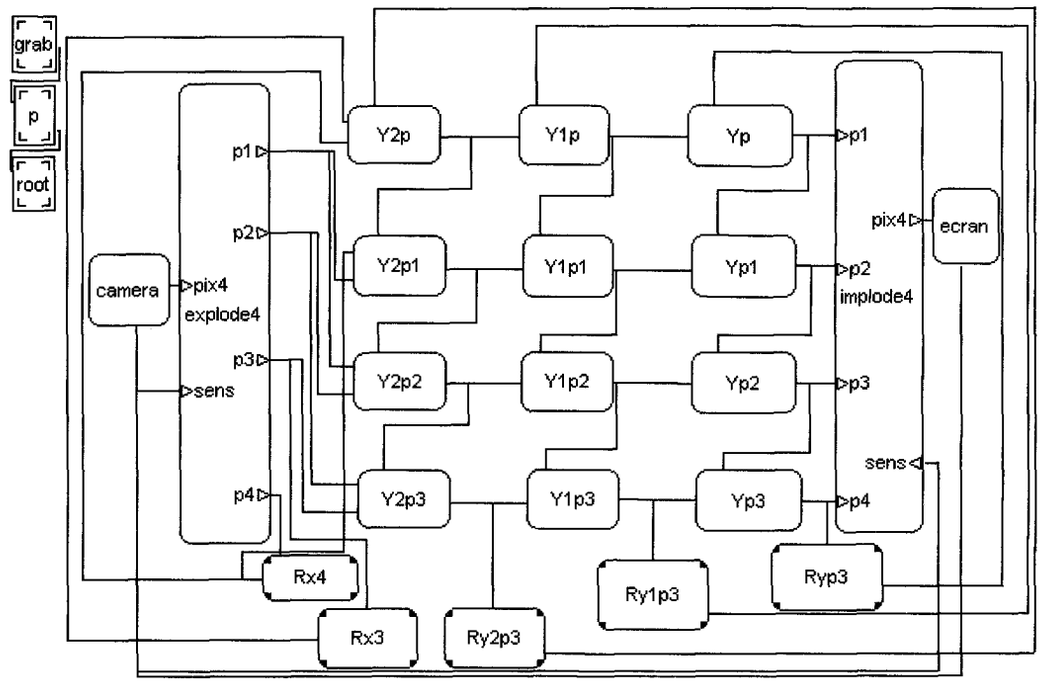


Figure 4. – Graphe flot de données pixel, et architecture tri-processeur (en haut à gauche).

4. implantation par flots de données pixel du filtre de Canny-Deriche généralisé

4.1. principe général

Nous avons choisi un flot de données pixel à travers chaque nœud du graphe. Pour simplifier la présentation du graphe nous ne détaillons que la détection des contours verticaux sans lissage ce qui correspond à une lecture de la gauche vers la droite et de la droite vers la gauche de l'image. Une démarche identique avec des graphes logiciels du même type permettrait de traiter le lissage horizontal ainsi que le gradient vertical et le lissage vertical mais cela compliquerait beaucoup la présentation du graphe dans cet article alors que cette partie a pour but uniquement de tester une implantation à grain fin. Les pixels sont traités successivement par les 3 filtres récursifs d'ordre 1. Le résultat d'un filtre Yipn est présenté au filtre suivant $Y(i-1)pn$. Ce résultat est aussi présenté à un filtre de même nature $Yip(n+1)$ mais traitant le pixel suivant (récursivité). Il peut aussi être affecté d'un retard $(R..)$ qui gardera la valeur jusqu'au cycle suivant d'exécution du graphe et la présentera ensuite à l'entrée du filtre Yip traitant le pixel suivant. Nous avons choisi de présenter en sortie de la fonction d'entrée 4 pixels successifs simultanément. Un mot mémoire de 32 bits du TMS320C40 stockera 4 niveaux de gris de 8 bits et sera donc complètement utilisé. En outre cela permet un parallélisme maximum des 4 tâches de filtrage.

4.2. spécification sous SynDEx

Cette spécification est représentée en figure 4.

La définition des fonctions est la suivante :

Camera : C'est la fonction d'entrée qui donne un mot de 32 bits représentant les niveaux de gris de 4 pixels successifs. Elle indique aussi aux fonctions le sens (gauche => droite ou droite => gauche) dans lequel on traite l'image.

explode4 : Cette fonction prend le mot de 32 bits et en tire les 4 niveaux de gris p1, p2, p3 et p4 correspondants.

Y2pi : Avec $i = '', 1, 2$ ou 3. C'est la fonction $y2(n)$ décrite précédemment qui prend en entrée 2 pixels successifs de l'image et le résultat $y2(n-1)$ de la fonction $Y2p(i-1)$.

Y1pi : Fonction $y1(n)$ déjà décrite. En entrée on a : $Y2pi$ et $Y1p$ $(i-1)$.

Ypi : Fonction $y(n)$. En entrée on a : $Y1pi$ et $Yp(i-1)$.

Rx.. et Ry.. : Ce sont des retards de SynDEx qui permettent en mode flot de données de mémoriser une valeur d'un cycle d'exécution du graphe sur le suivant.

implode4 : Cette fonction prend les 4 pixels calculés au cours de l'exécution du cycle du graphe et forme un mot de 32 bits.

ecran : C'est la fonction de sortie qui récupère le mot de 32 bits calculé et en tire les 4 pixels représentant le contour. Elle affiche ensuite ces 4 pixels à l'écran.

4.3. prédiction de l'accélération et expérimentation pour différentes implantations

Les figures 5 et 6 représentent les diagrammes temporels obtenus après placement automatique avec SynDEx sur deux et trois processeurs.

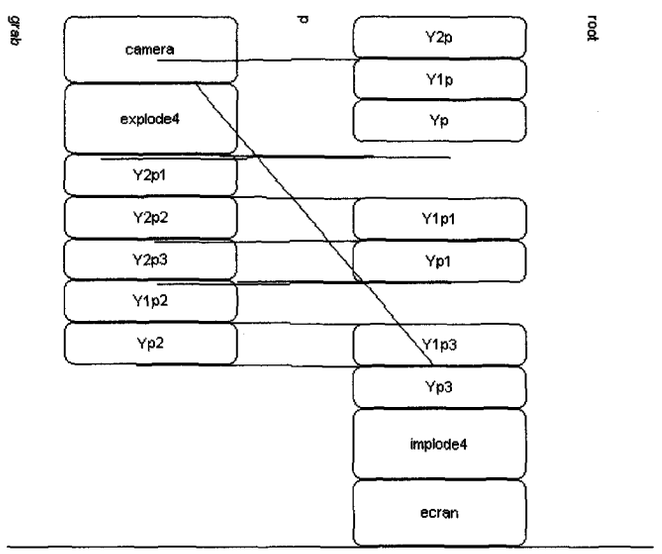


Figure 5. – Diagramme temporel avec 2 processeurs.

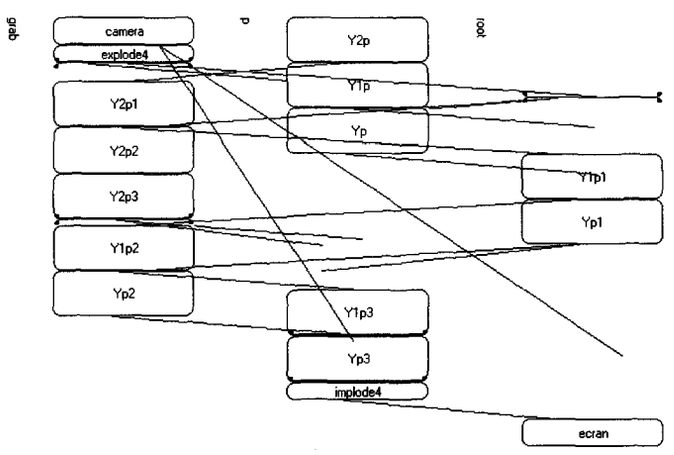


Figure 6. – Diagramme temporel avec 3 processeurs.

L'axe des temps est représenté en vertical sur ces diagrammes. Il permet uniquement d'apprécier la durée relative des tâches les unes par rapport aux autres et par rapport aux durées de communication. Chaque colonne représente la charge de calcul par processeur. Sur deux processeurs, l'accélération prédite obtenue est de 1,47 alors que sur trois processeurs, elle n'est que de 1,26.

Ceci est dû principalement au fait que sous cette forme l'algorithme est très récursif et qu'il faut attendre très souvent la fin d'un pixel pour pouvoir commencer le traitement du pixel suivant. De plus les communications inter-processeurs pour un nombre réduit d'octets sont très consommatrices en temps puisqu'elles nécessitent en plus de l'information utile quelques octets d'information de routage : la longueur du message à transmettre est donc quasiment doublée. C'est un problème qui n'apparaîtrait pas si on transmettait beaucoup plus de valeurs simultanément (La taille de l'information utile serait beaucoup plus grande par rapport à celle de l'information de routage). L'adéquation entre le graphe flot de données et le placement/ordonnancement sur 3 processeurs ne donne donc pas de bons résultats puisque les performances sont meilleures sur 2 processeurs.

L'implantation effective de l'exécutif sur TMS320c40 donne un taux de 1 image/s sur 2 processeurs et de 0.6 image/s en monoprocesseur. De plus le contraste des images traitées est assez faible puisqu'on n'exécute que la moitié du filtre. L'adéquation algorithme-architecture obtenue sur 2 processeurs est satisfaisante en terme d'optimisation mais ne permet pas une exécution de l'algorithme satisfaisante en terme de rapidité.

5. implantation par flot d'images du filtre de Canny-Deriche généralisé

5.1. principe général

Pour cet algorithme, qui correspond à l'équation (6) appliquée à tous les pixels de l'image, nous avons choisi un flot de données images. Ce sont des images de 256×256 points qui sont échangées par les différents noeuds du graphe. On se place donc a un niveau moins fin de description de l'algorithme. On fait subir à l'image 2 traitements : d'une part une détection de contour suivant l'axe X puis un lissage suivant Y et d'autre part un lissage suivant X puis une détection de contour suivant l'axe Y. On somme le résultat de ces 2 traitements pour obtenir l'image finale.

5.2. spécification sous SynDEx

Cette spécification est représentée en figure 7. La définition des fonctions est la suivante :

Acquisition : C'est la fonction d'entrée qui acquiert une image de 256× 256 points en 256 niveaux de gris. Elle fournit ce tableau image de 64 Koctets aux fonctions DX et LX.

DX : Cette fonction effectue sur son image d'entrée $'im'$ le gradient suivant X (détection des contours verticaux) et fournit l'image résultat $'Dim'$ en sortie.

LY : Cette fonction effectue sur l'image des contours verticaux un lissage suivant l'axe Y.

LX et DY : Ces 2 fonctions ont des rôles similaires à LY et DX, seule la direction du traitement change.

SOMME : Cette fonction fait la somme point par point des 2 images en entrée. En sortie on a donc le résultat du filtre de Canny-Deriche optimisé.

Affichage : C'est la fonction de sortie qui sauve l'image sur disque ou l'affiche à l'écran.

5.3. prédiction de l'accélération et expérimentation pour une implantation biprocesseur

L'heuristique de placement/ordonnancement de SynDEx donne le résultat figure 8 sur 2 processeurs et prévoit une accélération de 1,74. Si on augmente le nombre de processeurs on ne produit pas une augmentation de l'accélération du fait de la description de l'algorithme.

L'implantation de l'exécutif sur 2 TMS320C40 donne de bons résultats au niveau des contours obtenus (cf. Figures 19, 20). Le temps d'exécution sur 2 processeurs est de 3019 ms alors que l'implantation séquentielle s'exécute en 5528 ms. L'accélération expérimentalement mesurée est de 1,83 ce qui est légèrement supérieur à l'accélération prédite. La différence vient de la manière dont SynDEx modélise les communications entre les processeurs ce qui donne une durée de communication légèrement différente de l'expérience. En effet SynDEx modélise des communications totalement en parallèle avec les calculs alors que dans la version 3.6 du logiciel les calculs sont interrompus par des paquets de communication.

L'adéquation algorithme architecture est donc optimale dans ce cas. L'utilisation d'un flot de données image a considérablement réduit les échanges entre processeurs et permet de parallèliser presque totalement les tâches mais n'offre aucune possibilité d'amélioration sur plus de 2 processeurs.

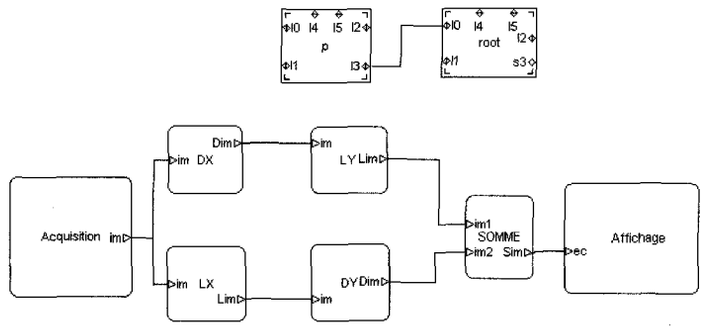


Figure 7. – Graphe Browser de SynDEx pour une implantation flot image.

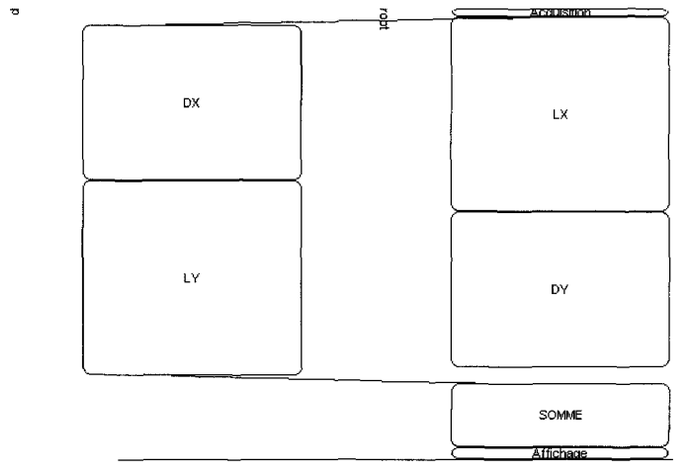


Figure 8. – Diagramme temporel de placement des tâches sur 2 processeurs sous SynDEx.

6. amélioration des performances de l'algorithme avec un flot de demi images

6.1. principe général

Il s'agit de découper l'image en différentes parties sur lesquelles le filtre va pouvoir s'exécuter simultanément et mettre ainsi en place un parallèlisme de type partage de données. Il faut remarquer que du fait de la récursivité du filtre ce découpage ne peut pas être fait de n'importe quelle manière. En effet, par exemple un découpage où une partie intérieure de l'image serait isolée (*cf.* figure 9) ne serait pas intéressant puisque pour pouvoir traiter cette zone intérieure I les processeurs devraient au moins avoir traité une des zones adjacentes A dans le sens indiqué.

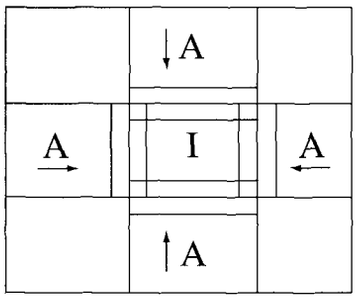


Figure 9. – Découpage de l'image en blocs.

Dans le but d'augmenter le parallélisme potentiel de l'algorithme on va diviser le flot de données par 2 et obtenir un flot principal de demi-images entre les fonctions. Un schéma de type SPMD sera donc mis en oeuvre : « split » de l'image, traitement des différentes zones, communication des données traitées et « merge » des résultats comme expliqué ci-après. La fonction d'entrée fournit donc 2 demi-images représentant la partie haute et la partie basse de l'image. Pour les fonctions de lissage et de gradient suivant X on peut avoir 4 tâches simultanées. Pour les fonctions suivant Y on peut les paralléliser sur 4 niveaux en respectant quelques conditions (*cf.* figure 10) :

– Il faut séparer le traitement de haut en bas du traitement de bas en haut pour pouvoir exécuter simultanément par exemple un lissage haut-bas sur la demi-image haute et un lissage bas-haut sur la demi-image basse (malgré la récursivité du filtre).

– Il faut mettre en place des communications de 3 lignes d'image (formant la frontière des 2 demi-images) entre les tâches du type gradient bas-haut sur l'image basse et gradient bas-haut sur l'image haute, ceci étant dû à la récursivité d'ordre 3 du filtre.

– Pour minimiser les communications des 3 lignes frontières de l'image originale, la fonction d'entrée fournit aux noeuds du graphe non seulement les 2 demi-images mais aussi les 3 lignes frontières.

6.2. spécification sous SynDEx

Le graphe logiciel de l'algorithme est présenté sur la figure 11.

Définition des fonctions :

Acquisition : Fonction d'entrée donnant dans im1 et im2 les 2 moitiés d'image et les 3 lignes de frontière correspondantes.

DX1, LX1, DX2 et LX2 : Ces 4 fonctions ont la même utilisation que les fonctions LX et DX de l'algorithme précédent.

LYbh.. : Lissage suivant Y dans le sens bas-haut sur une demi-image + 3 lignes.

LYhb.. : Lissage suivant Y dans le sens haut-bas sur une demi-image + 3 lignes.

DYbh.. : Gradient suivant Y dans le sens bas-haut sur une demi-image + 3 lignes.

DYhb.. : Gradient suivant Y dans le sens haut-bas sur une demi-image + 3 lignes.

SOMMEL.. : Somme des images lissées de haut en bas et de bas en haut.

SOMMED.. : Somme des images filtrées par le gradient de haut en bas et de bas en haut.

SOMME.. : Somme du gradient et du lissage total sur une demi-image.

Affichage : Récupération des 2 demi-images traitées et affichage à l'écran ou sauvegarde sur disque.

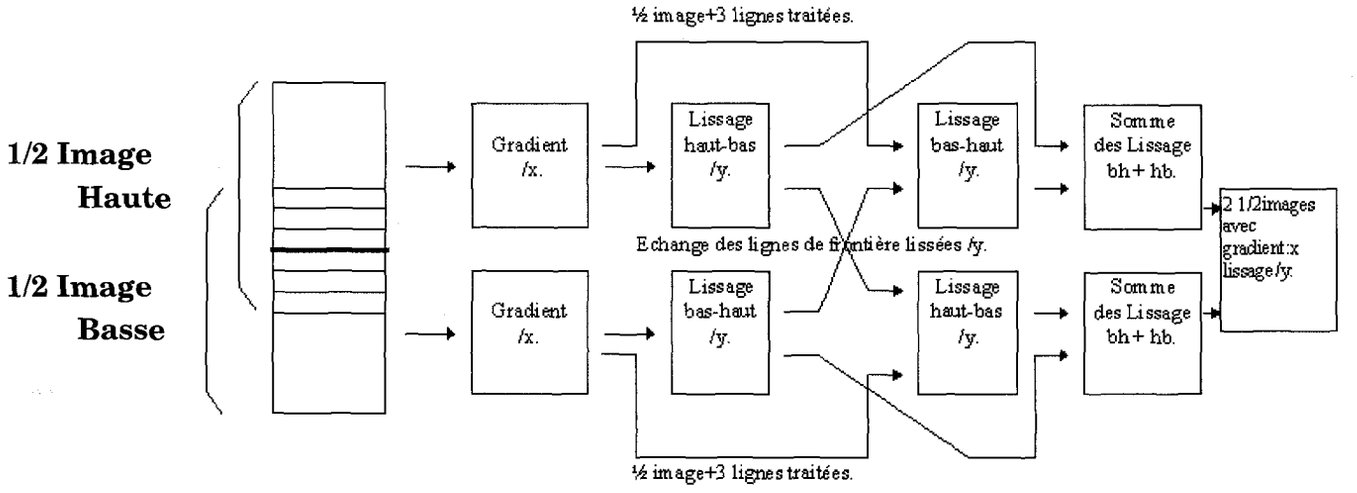


Figure 10. – Schéma des opérations et communications types de l'algorithme avec un flot de demi-image.

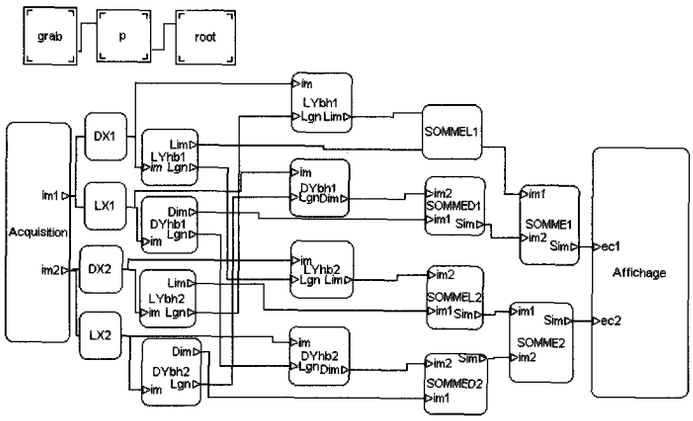


Figure 11. – Description sous SynDEx du graphe logiciel de l'algorithme.

6.3. prédiction de l'accélération et expérimentation pour différentes implantations

Nous avons lancé l'heuristique de placement/ordonnancement de SynDEx pour des configurations matérielles allant de 2 à 5 processeurs. On remarquera que le type de connexion entre les processeurs (en ligne, tore ou connexion totale) joue très peu sur l'accélération dans ce cas.Les performances de ces implantations sont représentées en figures 12, 13 et 14. On obtient des résultats expérimentaux du même ordre pour des configurations de 1, 2 ou 3 processeurs.

Pour pouvoir obtenir une accélération supérieure avec plus de 4 processeurs il faut modifier l'algorithme en introduisant une division par 2 de l'image dans le sens de la largeur et utiliser une méthode similaire à celle utilisée ici pour paralléliser le lissage et le gradient suivant X (on ne peut pas rediviser verticalement du fait de la récursivité du filtre). C'est le but de la dernière partie de cet article.

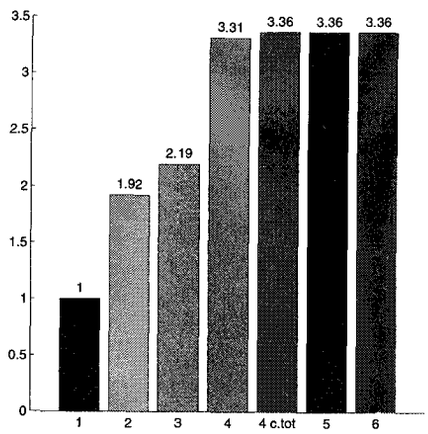


Figure 12. – Courbe récapitulative des accélérations en fonction du nombre de processeurs.

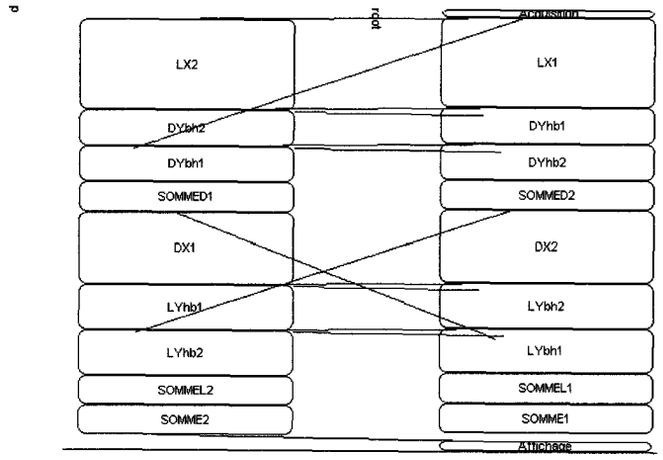


Figure 13. – Diagramme temporel sous SynDEx avec 2 processeurs.

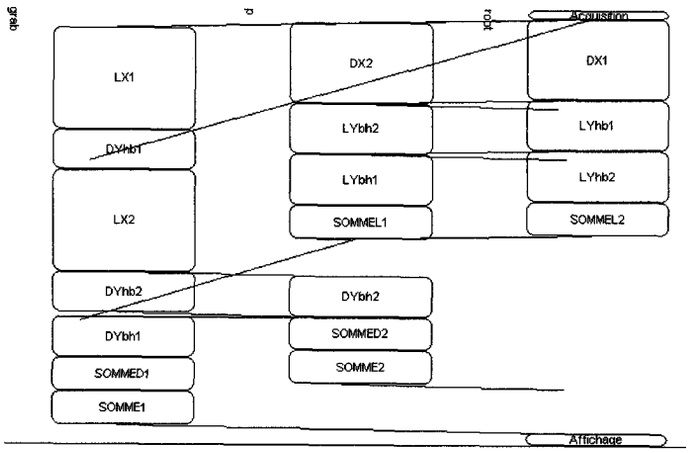


Figure 14. – Diagramme temporel sous SynDEx avec 3 processeurss.

7. amélioration des performances de l'algorithme avec un flot d'un quart d'images

7.1. principe général

Dans le but d'augmenter le parallélisme potentiel de l'algorithme on va diviser le flot de données par 4 et obtenir un flot principal de quarts d'images entre les fonctions. Un découpage en 4 bandes verticales ou horizontales n'autoriserait pas un parallèlisme des tâches sur chacune des bandes du fait de la récursivité du filtre comme expliqué au 6.1. La fonction d'entrée va donc fournir 4 quarts d'images représentant la partie Haute/Gauche(HG),

Haute/Droite(HD), Basse/Gauche(BG) et Basse/Droite(BD) de l'image. Pour les fonctions de lissage et de gradient suivant X et suivant Y on pourra avoir 8 tâches simultanées sous quelques conditions :

– Séparation des traitements causaux et anticausaux pour pouvoir exécuter simultanément par exemple un lissage haut-bas sur le quart-image BG et un lissage bas-haut sur le quart-image HG (malgré la récursivité du filtre).

– Mise en place de communications de 5 lignes d'image (formant la frontière de deux quarts-images) entre les tâches du type gradient bas-haut sur l'image BG et gradient bas-haut sur l'image HG. Ceci étant dû à la récursivité d'ordre 3 du filtre.

7.2. spécification sous SynDEx

La spécification décrite en figure 15 utilise les fonctions suivantes :

Acquisition : Fonction d'entrée donnant les 4 quarts d'image.

LYbh..,LYhb..,LXgd..,LXdg.. : Lissage suivant Y bas-haut ou haut-bas ou suivant X gauche-droite ou droite-gauche sur un quart d'image.

DYbh..,DYhb..,DXgd..,DXdg.. : Gradient suivant Y bas-haut ou haut-bas ou suivant X gauche-droite ou droite-gauche sur un quart d'image.

SOMME.. : Somme des deux quarts d'image en entrée pixels par pixels.

Affichage : Récupération des quatre quarts d'image traités et affichage.

7.3. prédiction de l'accélération et expérimentation pour différentes implantations

Nous avons lancé l'heuristique de placement/ordonnancement de SynDEx pour des configurations matérielles allant de 2 à 8 processeurs. Le type de connexion entre les processeurs joue très peu sur l'accélération. Un tableau donnant les accélérations et les efficacités en fonction du nombre de processeurs est donné ci-après :

Nbr de proc.	Acc.	Efficacité
1	1	1
2	1.96	0.980
3	2.89	0.963
4	3.79	0.947
5	4.53	0.906
6	5.36	0.893
7	5.73	0.818
8	6.89	0.861

La courbe donnant l'efficacité en fonction du nombre de processeurs, figure 16, nous permet de montrer que l'accélération augmente jusqu'à 8 processeurs utilisés. L'efficacité, qui est le rapport de l'accélération sur le nombre de processeurs, est relativement proche de 1 globalement. Elle diminue jusqu'à 7 processeurs et augmente à nouveau pour 8 processeurs. Cela étant dû à la dissymétrie de répartition des tâches sur 7 processeurs d'un graphe logiciel optimum pour 8 processeurs.

Expérimentalement l'implantation de l'algorithme sur 2 ou 3 TMS320C40 donne des résultats conformes à ceux attendus avec des accélérations de respectivement 1.92 et 2.90 légèrement différentes de celles prévues pour des raisons de modélisation des communications comme expliqué précédement.

Une augmentation du parallèlisme par division de l'image en 8 blocs ou plus est en théorie possible mais on se heurte alors au problème des zones isolées décrites au 6.1. Ceci montre l'impact de ce type de découpage sur l'adéquation algorithme architecture car l'algorithme limite lui même, de par sa récursivité, le degré de parallèlisation.

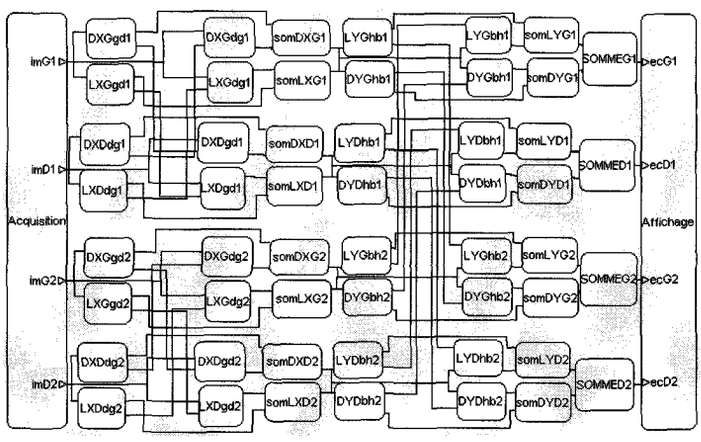


Figure 15. – Graphe Browser de SynDEx pour une implantation avec un flot de quart d'image.

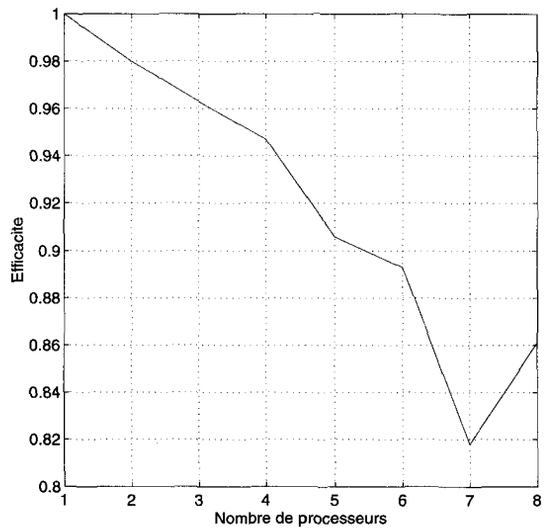


Figure 16. – Efficacité du système.

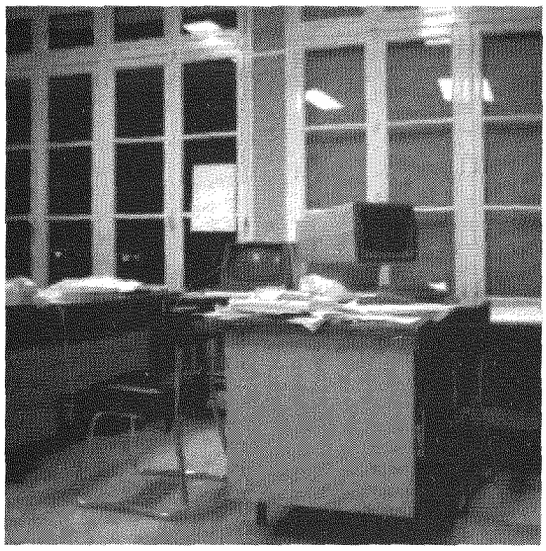


Figure 17. – Image originale

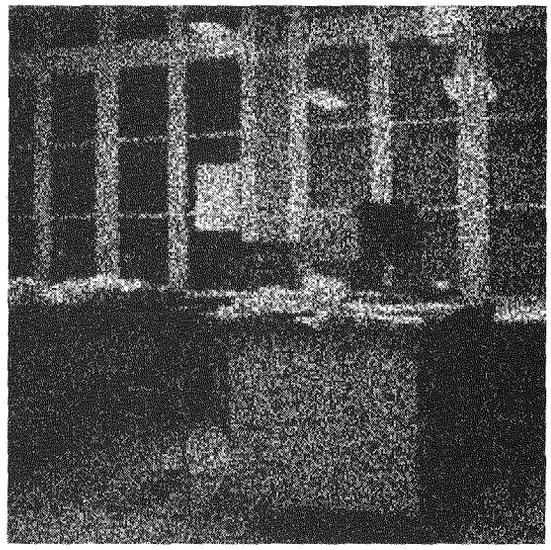


Figure 18. – Image originale bruitée

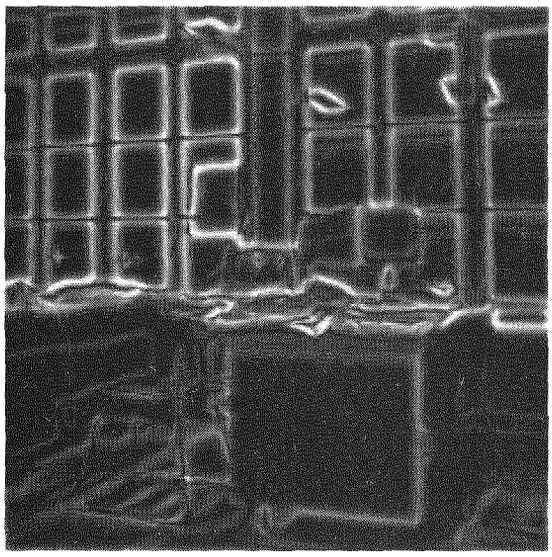


Figure 19. – Contour de l'image bureau

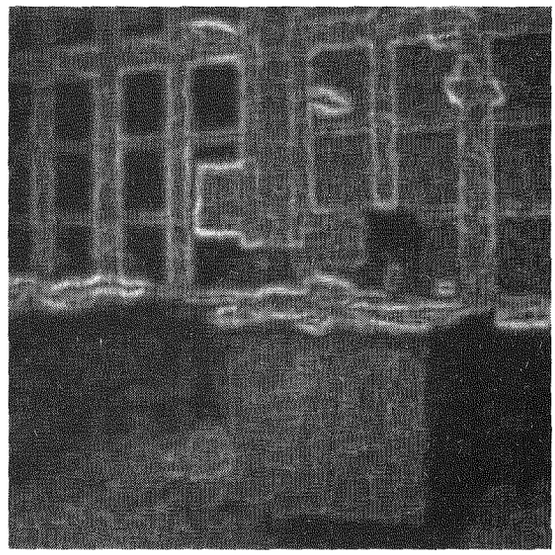


Figure 20. – Contour de l'image bureau bruitée

8. Conclusion

Grâce à l'outil d'aide à la parallèlisation SynDEx nous avons pu tester trois formes d'algorithmes selon le grain de données choisi (niveaux pixel, image et bande), et pour chacune d'elles, plusieurs implantations avec des nombres de processeurs différents ont pu être spécifiées.

Les résultats obtenus ont permis de mettre en évidence l'importance du choix du grain des données : si le grain est « trop fin », la spécification est volumineuse et les communications moins efficaces alors que si ce grain est « trop gros », le manque de parallélisme restreint le nombre de processeurs utilisables. Nos expérimentations ont donc permis de confirmer que le grain « idéal » pour ce type d'applications de traitement d'images bas-niveau est le découpage des images en blocs.

Ainsi nous avons pu montrer que la décomposition par blocs en quarts d'image est la mieux adaptée en terme d'adéquation algorithme architecture. Dans cette étude, l'utilisation de SynDEx nous a aidé à choisir le grain de traitement, grâce au module de prédiction de performances qui permet de comparer les implantations, aussi bien au niveau grain de l'algorithme qu'au niveau du nombre de processeurs.

BIBLIOGRAPHIE

[1] Y.Sorel *Massively Parallel Systems with Real Time Constraints. The "Algorithm Architecture Adequation" Methodology.* Proc. Massively Parallel Computing SYstems, the Challenges of General-Purpose and Special-Purpose Computing Conference, Ischia Italy, May 1994.

[2] C.Lavarenne, Y.Sorel, C.Milan et M.Paindavoine. *Implantation d'un algorithme de segmentation d'image sur une architecture multi-processeur avec l'environnement d'aide à l'implantation SynDEx* WORKSHOP Adéquation Algorithmes Architectures (Grenoble Janvier 1994).

[3] C.Aiglon, C.Lavarenne, Y.Sorel, et A.Vicard. *Utilisation de SynDEx pour le traitment d'images temps-réel* Rapport de Recherche INRIA - n° 2968 - Septembre 1996.

[4] J.Canny. *A computationnal approach to edge detection* Trans. Pattern Anal. Machine Intell Vol.PAMI-8,6 IEEE. (November 1986)

[5] E.Bourennanne. M.Paindavoine. F.Truchetet. *Amélioration du filtre de Canny-Deriche pour la détection des contours sous forme de rampe* Traitement du signal. Vol.10-N4. Recherche 1993.

[6] L.Torres. E.Bourennanne. M.Robert. M.Paindavoine. *Implantation du détecteur de contours Canny-Deriche optimisé sous forme d'un circuit spécifique* Traitement du signal. Vol.14-N1. Recherche 1997.

[7] El-Bay Bourennane. *Conception et implantation d'un détecteur de contours optimisé sous forme d'un circuit asic* Thèse de l'Université de Bourgogne - 1994.

[8] N.Zarka. *Conception d'un Circuit Intégré de Détection Optimale de Contours* Thèse de l'Université de Paris 6 - Décembre 1992.

[9] Lionel Torres. *Intégration de filtres numériques pour le traitement d'image : du silicium au systeme reconfigurable* Thèse de l'Université de Montpellier II - Juillet 1996.

[10] Tawfik Kamle. *Implantation d'algorithmes de traitement des signaux bidimentionnels en flot de données sur ASICS et circuits reconfigurables* Thèse de l'Université de Paris-Sud - Décembre 1994.

[11] H.Dubois. *Analyse des Systemes Multiprocesseurs : Application à la Mise en Oeuvre sous Contraintes d'Algorithmes de Traitement d'Images* Thèse de l'Université de Rennes I, Lannion - Janvier 1991.

[12] S.J.Singh. N.K.Sharda. *Design and Implemantation of PARSIM : An Architecture Specification Language* Parallel Computing and Transputers - ed.D.Arnold et al. - IOS Press 1993.

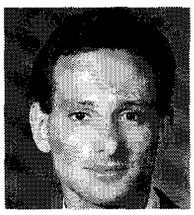
[13] R.C.Covington. S.Madala. V.Metha. J.R.Jump. J.B.Sinclair. *The Rice Parallel Processing Testbed*
Measurement and Modeling of Computer Systems 1988.

[14] C.Scheidler. L.Schäfers. O.Krämer-Fuhrmann.
TRAPPER : A Graphical Programming Environment for Embedded MIMD Computers Transputer Applications and System - ed.R.Grebe et al. - IOS Press 1993.

Manuscrit reçu le 23 juillet 1997.

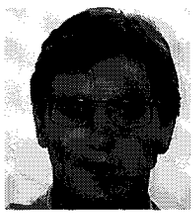
LES AUTEURS

Laurent HAAS



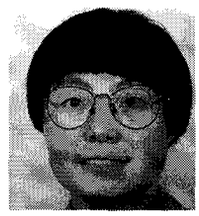
Laurent HAAS, 27 ans, soutiendra sa thèse à l'Université de Bourgogne au sein du LE2I fin 1998. Ses domaines de recherche concernent l'Adéquation Algorithme Architecture en traitement d'Images temps réel pour l'implantation sur des architectures hétérogènes basées sur des DSP et des FPGA. Il enseigne, en qualité d'ATER, dans le domaine des réseaux à l'IUT d'Informatique.

Michel PAINDAVOINE



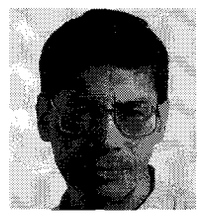
Michel Paindavoine, 42 ans, est professeur à l'Université de Bourgogne. Il enseigne le traitement du Signal et des Images à l'Ecole d'Ingénieurs ESIREM et à l'IUP Electronique et Image. Il effectue ses travaux de recherche au LE2I (Laboratoire d'Electronique, d'Informatique et d'Image) dans le domaine de l'Adéquation Algorithmes Architectures en traitement d'images.

Fan YANG



Fan YANG, 34 ans, prépare un doctorat. Elle va soutenir sa thèse à l'Université de Bourgogne, option Electronique et Informatique, en 1998. Celle-ci porte sur le traitement automatique de visages par réseaux de neurones et plus particulièrement sur les implantations parallèles des algorithmes de localisation, reconnaissance et identification de visages.

Claude MILAN



Claude MILAN, 56 ans, est professeur à l'Université de Bourgogne. Responsable de l'IUP Electronique et Image, il enseigne dans le domaine des DSP à l'IUP et du traitement d'image à l'Ecole d'Ingénieurs ESIREM. Il effectue ses travaux de recherche au laboratoire LE2I, dans les domaines de la vidéo rapide et de la compression d'images, dans le cadre de l'Adéquation Algorithmes Architectures en traitement d'images.