TREIZIÈME COLLOQUE GRETSI - JUAN-LES-PINS DU 16 AU 20 SEPTEMBRE 1991

ARCHITECTURE PIPELINE INTERGREE IMPLEMENTANT DES DETECTEURS OPTIMAUX DE CONTOURS *

Nizar ZARKA , Mohamed AKIL

Laboratoire I.A.A.I., Groupe E.S.I.E.E., 2 Bd Blaise Pascal
B.P.99, 93162 NOISY-LE-GRAND Cedex (FRANCE)

RÉSUMÉ

On présente dans cet article l'étude et la conception d'un circuit intégré en technologie CMOS 1.5μ permettant l'implémentation de détecteurs optimaux de contours. Le circuit qu'on propose a l'avantage d'avoir une architecture pipeline implémentant des filtres récursifs à réponse infinie. Le calcul s'effectue à la cadence vidéo. Toutes les étapes de calcul du gradient d'une image 2D sont intégrées dans cet architecture.

ABSTRACT

We present in this paper the design of an ASIC in 1.5μ CMOS technology intended for optimal edge detection. The advantage of this circuit is its pipeline architecture, implementing infinite response recursive filters. The circuit processes at video rate. All computational steps of 2D image gradient are integrated in the architecture.

1. INTRODUCTION

L'extraction des contours est une étape importante dans un processus d'analyse d'image. Une large variété d'applications (vision robotique, imagerie médicale, image satellite) utilise ce type de traitement comme une des étapes d'un processus de reconnaissance d'objets dans une image. L'efficacité d'un tel processus de reconnaissance dépend, en autre, des résultats obtenus lors de l'extraction des contours. De nombreux travaux lui ont été consacrés [1], [2], [3]. CANNY [4] a défini les critères suivants pour obtenir un détecteur optimal de contours: une bonne détection, une bonne localisation et une réponse unique à un contour. A partir de ses travaux DERICHE [5] a proposé une structure de filtres récursifs calculant le gradient d'une image 2D avec un nombre fixe d'opérations par pixel indépendant de la taille du filtre.

2. ALGORITHME DE DERICHE

Le calcul du gradient proposé par DERICHE [6] consiste à dériver les lignes de l'image par un filtre récursif d'ordre 2 par un balayage de gauche à droite suivi d'un balayage de droite à gauche selon les équations suivantes:

$$X_i^+ = I_{i-1} + 2e^{-\alpha}X_{i-1}^+ - e^{-2\alpha}X_{i-2}^+ \text{ (1) pour } i : 1,..,M$$
$$X_i^- = I_{i+1} + 2e^{-\alpha}X_{i+1}^- - e^{-2\alpha}X_{i+2}^- \text{ (2) pour } i : M,..,1$$
$$S_i = K_1[X_i^+ - X_i^-] \text{ (3) avec } K_1 = (1-e^{-\alpha})^2$$

I_i étant un pixel de l'image à l'instant i, et M la dimension d'une ligne. L'image de la dérivée S_i est ensuite filtrée en traitant les colonnes du haut vers le bas, ensuite du bas vers le haut selon les équations suivantes:

$$R_i^+ = K_2[S_i + e^{-\alpha}(\alpha-1)S_{i-1}] + 2e^{-\alpha}R_{i-1}^+ - e^{-2\alpha}R_{i-2}^+$$
$$\text{pour } i : 1,..,M \text{ (4)}$$

$$R_i^- = K_2[e^{-\alpha}(\alpha+1)S_{i+1} + e^{-2\alpha}S_{i+1}] + 2e^{-\alpha}R_{i+1}^- - e^{-2\alpha}R_{i+2}^-$$
$$\text{pour } i : M,..,1 \text{ (5)}$$
$$R_i = R_i^+ - R_i^- \text{ (6) avec } K_2 = \frac{(1-e^{-\alpha})^2}{1-e^{-2\alpha}+2\alpha e^{-\alpha}}$$

Le paramètre α détermine la forme du filtre. Pour $\alpha < 1$ les filtres sont larges, ce qui constitue un bon détecteur de contours des images bruitées. L'intérêt de l'implémentation de ces filtres est le nombre fixe d'opérations quelque soit la valeur de α. Ce nombre est de 14 multiplications et de 11 additions pour chaque pixel. La figure 1 représente les étapes de calculs pour obtenir le gradient horizontal.

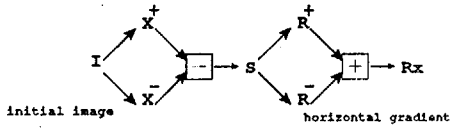

Fig. 1 Etapes de calcul du gradient

Le gradient vertical est obtenu en appliquant sur les colonnes les équations 1,2,3 et sur les lignes les équations 4,5 et 6. L'implémentation matérielle de cet algorithme est réalisée dans le but d'obtenir un traitement en temps réel. L'étude faite au paragraphe suivant montre que le choix d'une arithmétique sur 16 bits permet d'obtenir une fonction de transfert très proche de celle obtenue dans le cas d'une arithmétique flottante. Nous présenterons aussi la conception d'une structure matérielle pour calculer le gradient horizontal et vertical. Il faut noter que les équations 1,2 et 3 nécessitent l'utilisation de deux convolutions, suivies d'une soustraction et d'une multiplication par K_1. La transformation des équations 1 et 2 donne:

$$X_i^+ = K_1 I_{i-1} + 2e^{-\alpha}X_{i-1}^+ - e^{-2\alpha}X_{i-2}^+ \text{ (1')}$$
$$X_i^- = K_1 I_{i+1} + 2e^{-\alpha}X_{i+1}^- - e^{-2\alpha}X_{i+2}^- \text{ (2')}$$
$$S_i = X_i^+ - X_i^- \text{ (3')}$$

*Ce travail est soutenu par l'ANVAR et le PRC-ANM

et permet de diminuer le temps de calcul en un temps égale à deux convolutions et une soustraction.
L'équation (7) réalise le calcul du gradient de l'image:

$$X_i = a_0 I_i + a_1 I_{i-1} + b_1 X_{i-1} + b_2 X_{i-2} \quad (7)$$

Les équations (1') et (2') sont obtenues à partir de l'équation (7) en mettant $a_0 = 0$ et $a_1 = K_1$. Quant à l'équation (4) elle est déduite de (7) avec $a_0 = k_2$ et $a_1 = e^{-\alpha}(\alpha - 1)$. L'équation (5) est réalisée en retardant les pixels à l'entrée de l'équation (7), ce qui donne l'équation suivante:

$$X_i = a_0 I_{i-1} + a_1 I_{i-2} + b_1 X_{i-1} + b_2 X_{i-2} \quad (8)$$

Ainsi l'équation (7) est fondamentale, la réalisation d'un convolueur implémentant ce traitement, permet d'obtenir un filtre récursif d'ordre 2 programmable.

3. ARCHITECTURE

L'architecture qu'on propose permet d'implémenter l'algorithme de DERICHE, ainsi que l'algorithme de détection des lignes de crêtres, des contours de type marche [7], et l'algorithme de Shen [8]. Elle contient les blocs suivants:
1. Le registre des coefficients (*regcoef*) charge les coefficients des filtres.
2. le *convolueur* calcule un filtre récursif d'ordre 2, le *retard* permet la synchronisation des étapes de calcul de la convolution.
3. la mémoie *lifo* est une mémoire temporaire.
4. le bloc *entree* aiguille, soit les pixels en entrée, soit les pixels de la *lifo*.
5. le bloc additionneur/ soustracteur (*add*).
6. Les compteurs d'adresse de la *lifo* et de la mémoire externe (mémoire résultat). Le séquenseur contrôle, après initialisation, du circuit le déroulement temporel des différentes étapes de calculs. La duplication des blocs de calculs: convolueur (*convolueur*1, *convolueur*2), *add*, *lifo*, *entrée*, *retard* permet d'obtenir une structure symétrique relativement régulière, et d'atteindre le traitement à la cadence vidéo. Le convolueur 1 et 2 travaillant en mode maître esclave parallèlise l'ensemble des étapes de calcul. Le schéma des blocs fonctionnels de cette structure est le suivant:

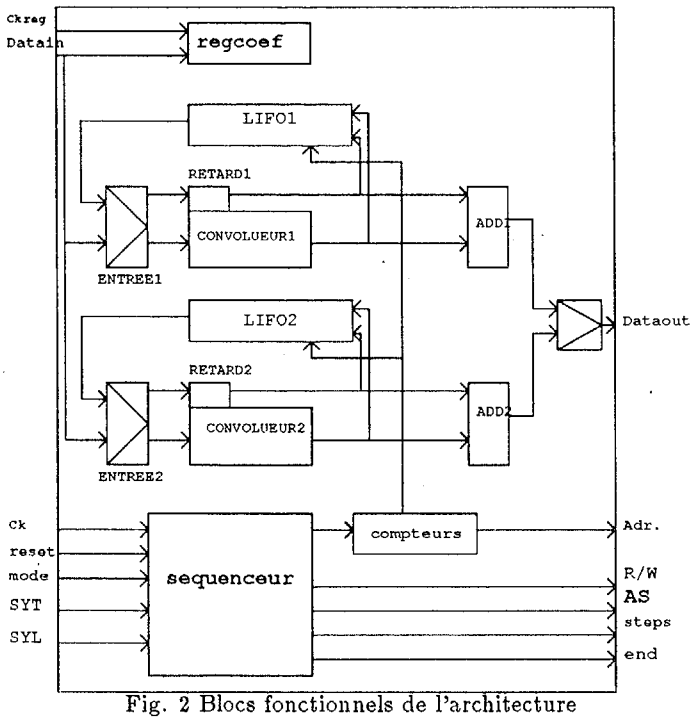

Fig. 2 Blocs fonctionnels de l'architecture

4. FONCTIONNEMENT DE L'ARCHITECTURE

Le circuit possède trois modes de fonctionnement: initialisation, chargement, et calcul.
En mode chargement les coefficients des filtres sont stockés dans le registre des coefficients.
En mode calcul, les étapes sont: tout d'abord, l'initialisation du traitement (mode=1, reset=1),le circuit attend les signaux de synchronisation de trame et ligne (SYT, SYL) pour commencer le calcul. Ensuite les pixels traités sont stockés au fur et à mesure dans la *lifo* réalisant ainsi l'équation (1'). La *lifo* est lue pour traiter les pixels de droite à gauche, le résultat est additionné à l'équation (1'), et stocké dans la mémoire externe: ceci correspond aux traitements réalisés par les équations (2') et (3'). Chaque ligne de la mémoire est ensuite transposée et traitée pour obtenir de la même façon les équations 4,5 et 6. On aboutit ainsi à l'image gradient. Les étapes de calcul sont indiquées par le signal *steps*. Nous allons, reprendre dans ce qui suit le fonctionnement de chacun des blocs constituant la structure du circuit étudié.

5.BLOCS FONCTIONNELS DE L'ARCHITECTURE

5.1. Le registre des coefficients:

Ce registre programmable définit, la taille de l'image (la taille maximale est de 1024×1024), le type du gradient (horizontal, vertical), et comprend les coefficients des filtres de dérivation et de lissage. Chacun de ces coefficients est codé sur 16 bits. Quand $reset = 1$ et $mode = 0$, le circuit charge à chaque top de l'horloge *ckreg* un octet. On commence par le chargement de la taille de l'image et le type du gradient, suivi par les sept coefficients des filtres (soit 14 octets). Afin de minimiser les étapes de calculs, le choix d'une mémoire *lifo*, comme mémoire tampon permet de réduire le nombre d'accés externes.

5.2. La mémoire *lifo*

Cette mémoire de 1024×24, stocke une ligne de pixels (8 bits) et une ligne de résultats intermédiaires de la convolution (16 bits). La figure 3 montre l'intérêt de l'utilisation d'une telle mémoire pour les étapes de calcul

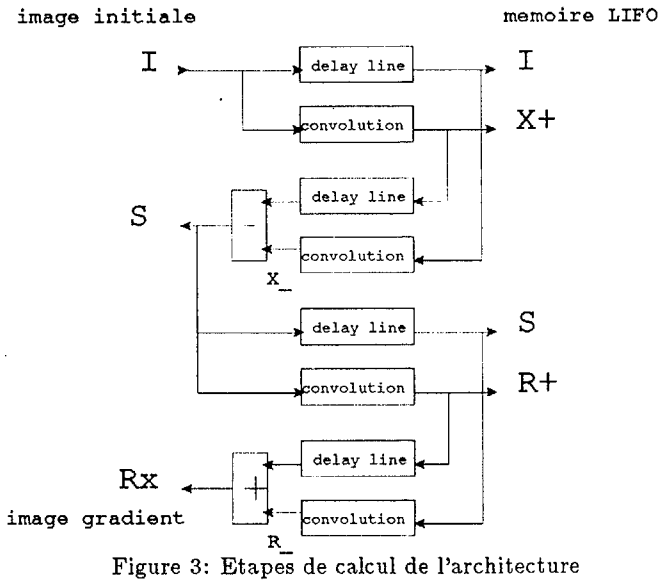

Figure 3: Etapes de calcul de l'architecture

La *lifo* stocke une ligne de l'image I, (ou une colonne de l'image S) et une ligne de convolution X^+ (ou une colonne de convolution R^+). La lecture de cette mémoire dans le sens inverse permet donc de convoluer la ligne (ou la colonne) dans les deux directions.

5.3. Architecture du convoluer

Comme nous l'avons souligné, le convoluer réalisé implémente l'équation (7). Cette équation revient à calculer une convolution discrète. Une structure classique[11] consiste à faire quatre multiplications en parallèle, puis à additionner les quatre résultats. Cette structure est trop lente car le temps de calcul est d'une multiplication et de deux additions. Il existe une structure plus performante que cette première, qui s'inspire d'une architecture systolique[12],[13], c'est celle d'un convoluer pipeline. Elle a l'avantage d'avoir une structure régulière composée de plusieurs étages. Le temps de calcul est égal au temps de propagation dans un seul étage. La structure d'un étage est indépendante de la taille du filtre. Le convoluer qu'on propose possède quatre étages identiques. Chaque étage est formé d'un additinneur 16 bits, d'une cellule de contrôle de débordement, d'un multiplieur 16 bits signé, et deux bascules 16 bits, comme le montre la figure 4.

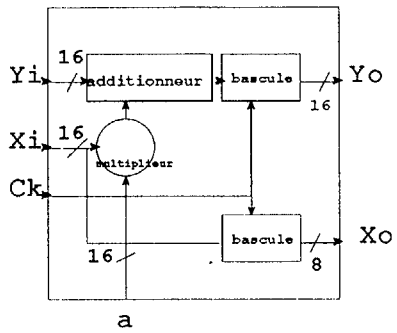

Figure 4: Cellule de base de Convolution

Si on a en entrée de la cellule a, X, Y, on obtient en sortie X_o et Y_o au top d'horloge suivant avec:

$$X_o = X_{i-1} \quad , \quad Y_o = a \times X_{i-1} + Y_{i-1}$$

Nous proposons donc de cascader quatre cellules de ce type pour obtenir un convoluer implémentant les deux équations (7) et (8). Le choix de l'équation est déterminé par l'entrée s comme le montre la figure 5.

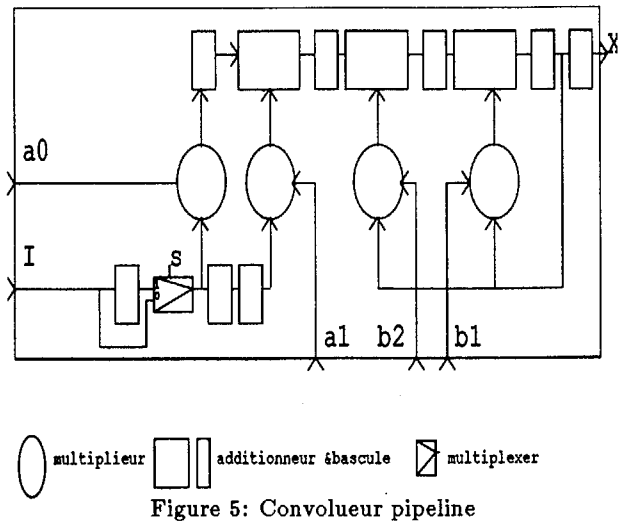

Figure 5: Convoluer pipeline

Quand $s = 0$, les données I sont traitées directement par le convoluer sans passer par la bascule d'entrée; ceci correspond au calcul de l'équation (7). Quand $s = 1$, les données passent à travers la bascule d'entrée retardant ainsi les pixels d'un cycle d'horloge; ceci correspond bien au calcul de l'équation (8). Après cinq top d'horloge le permier résultat est obtenu: temps de chargement du pipeline, puis à chaque top d'horloge un nouveau résultat est obtenu. Les coefficients a_0, a_1, b_1, b_2 sont codés sur 16 bits. L'étude faite dans[5],[9] montre bien que l'écart type entre les fonctions de transfert calculées avec des coefficients flottants et des coefficients codés sur 16 bits est négligeable pour une variation de α entre 0.2 et 1.4. Cette variation de α couvre bien le domaine des images bruitées. L'arrondi après chaque multiplication se fait en prenant les 16 bits de poids fort. L'erreur introduite est acceptable pour α supérieur ou égal à 0.5.

En effet l'erreur d'arrondi après chaque multiplication est considérée comme un bruit blanc[10],[11] de moyenne nulle et de variance $\sigma^2 = \frac{2^{-2b}}{12}, (b+1)$ étant le nombre de bits y compris le bit de signe après arrondi. On peut admettre l'abscence de corrélation entre les divers sources de bruit d'arrondi qui suivent chaque multiplieur. On peut donc obtenir la contribution totale à la sortie de chacune de ces sources par simple superposition. La variance du bruit à la sortie est donnée par:

$$\sigma^2 = \frac{2^{-2b}}{12} \sum_{i=1}^{n} \int H(Z)H(Z^{-1})Z^{-1}dZ$$

n étant le nombre de sources de bruit interne, et H(Z) la fonction de transfert en Z entre la sortie du multipilieur et la sortie du filtre. La structure du convolueur proposée comporte quatre multiplieurs. La fonction de transfert du filtre est:

$$H(Z) = \frac{N(Z)}{D(Z)} = \frac{a_0 + a_1 Z^{-1}}{1 + b_1 Z^{-1} + b_2 Z^{-2}}$$

L'erreur due à la multiplication par a_0 et a_1 traverse tout le filtre, par contre celle provenant de la mutiplication par b_1 et b_2 ne traverse que la partie récursive, d'où l'erreur totale:

$$\sigma^2 = 2\frac{2^{-2b}}{12}[\int H(Z)H(Z^{-1}) + \frac{1}{D(Z)}\frac{1}{D(Z^{-1})}]Z^{-1}dZ$$

$$\text{Selon[9]} \int H(Z)H(Z^{-1})Z^{-1}dZ = \frac{a_0^2(2+a_1^2+a_1^2b_2-4a_1b_1+2b_1^2-2b_2^2)}{(1-b_2)[(1+b_2)^2-b_1^2]}$$

$$\text{et selon[5]} \int \frac{1}{D(Z)}\frac{1}{D(Z^{-1})}Z^{-1}dZ = \frac{1+e^{-2\alpha}}{1-e^{-2\alpha}}\frac{1}{1+e^{-4\alpha}-2e^{-2\alpha}}$$

pour $b = 15$, et α inférieur ou égal à 0.5, la valeur de σ est inférieur à 3×10^{-5}. On déduit que l'erreur d'arrondi sur 16 bits est satisfaisante pour les images bruitées.

6. ENVIRONNEMENT DE L'ARCHITECTURE

La propriété de la commutativité de la convolution entre les filtres de dérivation et de lissage nous permet de calculer les deux gradients en parallèle. On traite ainsi les lignes de l'image par les filtres de dérivation pour obtenir le gradient horizontal et par les filtres de lissage pour calculer le gradient vertical. De ce principe, on déduit l'environnement pour le calcul de l'image gradient à la cadence vidéo (figure6). Les deux gradients sont calculés en parallèle par les deux circuits CI_1 et CI_2. Les étapes sont les suivantes:

1. Les deux circuits font l'acquisition des pixels sur le bus vidéo. Le CI_1 calcule le gradient horizontal en commançant par les filtres de dérivation, pendant que le CI_2 calcule le gradient vertical en commençent par les filtres de lissage. Les pixels sortant de chaque circuit sont stockés au fur et à mesure dans le registre de la mémoire vidéo. La ligne ainsi chargée dans le registre vidéo est transposée et transférée dans la RAM. Ce transfert est effectué par le contrôleur[14].
2. Les deux circuits reçoivent leurs données de deux RAM vidéo. Le CI_1 applique les filtres de lissage sur RAM1 pendant que le CI_2 traite RAM2 avec les filtres de dérivation.

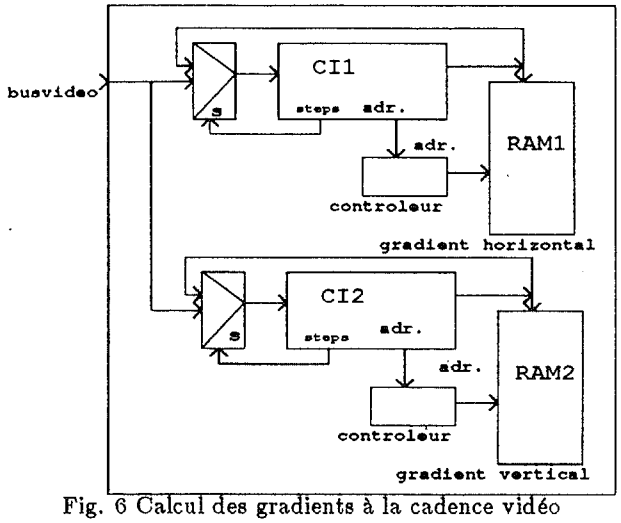

Fig. 6 Calcul des gradients à la cadence vidéo

7. PRESENTATION DU CIRCUIT

On s'est interessé à réaliser une version simplifiée de l'architecture proposée. Cette version ne comporte pas de duplication de blocs, d'où la nécessité de mettre deux circuits pour calculer chaque gradient. Cette solution suppose aussi que l'image est déja stockée dans une mémoire comme le montre la figure suivante:

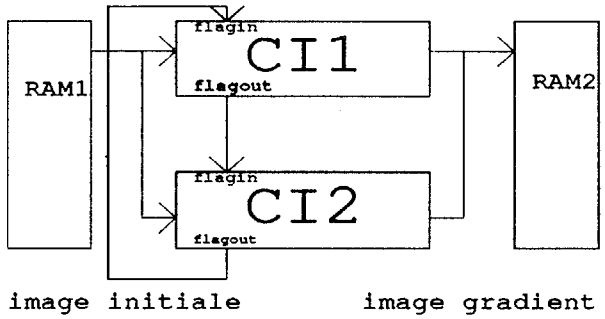

Fig. 7 Environnement du circuit

Le circuit comprend 48 entrées-sorties. On remarque sur la figure 8 que le circuit ne possède plus de signaux de synchronisation de trame et de ligne, puisqu'il adresse directement la RAM1 contenant l'image initiale. Par contre il possède les signaux(flagin, flagout), permettant au circuit de travailler en maître ou esclave.

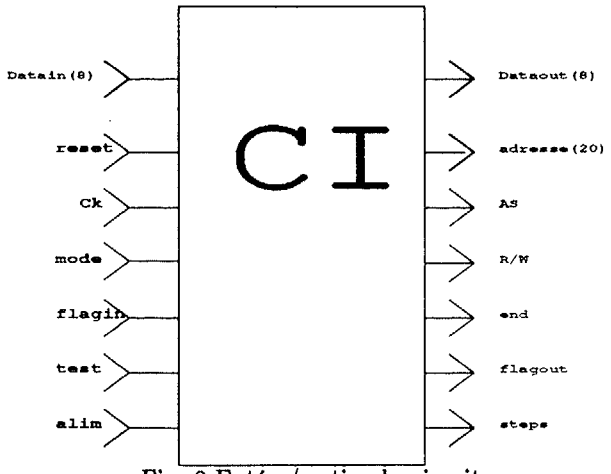

Fig. 8 Entées/sorties du circuit

Le fait d'être maître ou esclave se programme dans le registre des coefficients. Le maître et l'esclave lisent et écrivent dans les RAM externes à tour de rôle. Le maître lit les lignes paires alors que l'esclave lit les lignes impaires. Lors de la première lecture le maître envoie flagout=1, l'esclave commence alors une lecture synchronisée avec l'écriture du maître.
Le circuit a été conçu et simulé avec les outils de VLSI TECHNOLOGY en technologie CMOS 1.5μ. Le circuit est testable (par l'entrée *test*) graçe à la contrôlabilité et à l'observabilité de chaque bloc du circuit. La surface du circuit est de $90mm^2$.

8. CONCLUSION

Nous avons présenté dans cet article une architecture pipeline implémentant des filtres récursifs à réponse infinie, et permettant de traiter à la cadence vidéo les algorithmes de détection optimale des contours des images.

REMERCIEMENTS

Les auteurs tiennent à remercier pour leur aide constante, l'équipe conception du département micro-électronique du groupe ESIEE: Mme H.Trezeguet, Mrs B. Lucazeau et J. Bezamat. Le travail s'effectue dans le cadre d'une collaboration avec le laboratoire CAO-VLSI/MASI.

REFERENCES

[1] J.M. Prewitt, "Object enhancement and extraction" Picture processing and psychopictorics academic press, pp.75-149, 1970.
[2] M. Nagao, T. Matsuyama, "Edge preserving smoothing". Cgip vlo. 9, pp.394-407.
[3] D. O'learny "Some algorithms for approximating convolutions" computer vision, graphic, and image processing 41,333-345 (1988).
[4] J.F. Canny, "Finding edges and lines in images", technical report no 720, M.I.T. 1983.
[5] R. Deriche "using Canny's criteria to derive a recursively implemented optimal edge detector" Inernational Journal of Computer Vision, 167-187 (1987).
[6] R. Deriche "Fast algorithm for low level vision" IEEE 1988.
[7] D.Ziou B. Worbel-Dautcourt "filtres récursifs pour la détection de contours lignes de crêtes et marche. GRETSI 1989.
[8] J. Shen, S. Castan "An optimal linear operator for edge detection" IEEE 1986 pp. 109-114
[9] R. Deriche, H.Guiot, G. Randall "A general recursive filtering structure for early vision and its hardware architecure" IAPR workshope on CV Tokyo 1988.
[10] R. Boite H. Leich "Les fitres numériques Analyse et synthèse des filtres unidimentionnels" MASSON.
[11] M. Bellanger "Traitement numérique du signal" MASSON.
[12] H. Kung "A systolic 2-D convolution chip" multicomputers image processing algorithm and programs 1982.
[13] Y. Robert "Algorithmes et architectures systoliques INPG Octobre 1986.
[14] Texas Instruments TMS34061 User's guide preliminary.