De l'architecture à l'algorithme Un exemple : le détecteur de contours de Deriche

From the Architectural Design to the Algorithm An Example: the Deriche Edge Detector

par Didier DEMIGNY, Frederico GARCIA LORCA, Lounis KESSAL

ETIS, URA CNRS 2235, Equipe Traitement des Images et du Signal
ETIS/ENSEA-UCP, 6 avenue du Ponceau, F-95014 Cergy Pontoise Cedex
demigny@ensea.fr

résumé et mots clés

Au travers de l'exemple du détecteur de contours de Deriche, nous montrons comment des considérations d'optimisation des mémoires et des opérateurs pour une réalisation temps réel conduisent à une modification sans perte de qualité de l'algorithme initial. La nouvelle organisation qui en découle est transposable à tout détecteur de contours linéaire. Par cette expérience, nous souhaitons aussi transmettre aux concepteurs d'outils de CAO un certain nombre d'idées qui doivent à notre avis être exploitées afin que des outils tels que les graphes flots de données ou les langages synchrones assistent efficacement l'architecte dans les problèmes d'ordonnancement, d'allocation et de repliement temporel du graphe vers l' architecture.

Deriche, architecture, contours.

abstract and key words

Through the example of the Deriche edge detector, we show how memories and operators optimizations for a real time implementation lead to modify the initial algorithm without any loss in performances. The new organization can be easily transposed to other linear edge detectors. We also want to transmit to CAD engineers some ideas which have to be exploited to help architect designers in the area of data flow graphs, synchronous languages, allocation and scheduling problems.

Deriche, architecture, edges.

1. introduction

Le filtre de détection de contours proposé par Deriche [1] a été largement utilisé ces dernières années dans la communauté image. La version logicielle la plus efficace en rapidité utilise une implantation récursive. La qualité de ce filtre, la possibilité d'en changer la résolution *dans son implantation récursive* à l'aide d'un unique paramètre ont motivé les architectes à produire une version matérielle temps réel *25 images* 512×512 *par seconde* utilisant des ASICS ou des FPGAS[1]. Son étude fait apparaître différentes solutions d'organisation qui s'accompagnent d'optimisations à tous les niveaux de description : image, ligne, pixel, opérateur, mémoire. Nous proposons ici une organisation architecturale que nous qualifions d'optimale dans le sens où elle minimise le nombre, la taille des mémoires et le nombre des opérateurs nécessaires. Cette assertion sera démontrée en conclusion. Outre

1. Cette étude fait l'objet d'un contrat avec la société Aérospatiale.

le filtre de Deriche, cette architecture permet aussi, sans aucune modification, de réaliser le filtre de Shen [2].

Les travaux actuels du GT6 dans le GDR ISIS sont orientés vers l'A.A.A. généralement interprété comme Adéquation Algorithme Architecture. L'originalité de notre démarche est d'être plutôt du type : Adéquation Architecture Algorithme. A notre sens, l'algorithme n'est que l'expression d'une fonctionnalité. S'il nous semble essentiel de respecter la démarche et les critères de qualité qui ont conduit à son élaboration, nous pensons qu'il est nécessaire de rediscuter l'organisation des calculs ainsi que leur précision afin d'augmenter les possibilités d'optimisation comme nous l'avions déjà montré dans [3] pour le filtre de Nagao. C'est la simplicité et l'efficacité de l'architecture résultante qui nous guide dans les nouvelles organisations des calculs que nous proposons. Nous sommes amenés à distinguer deux types d'optimisation : algorithmique et architecturale. L'optimisation algorithmique consiste à réorganiser les calculs alors que l'optimisation architecturale consiste à augmenter la rapidité ou à réduire la complexité des opérateurs matériels qui les réalisent. Dans le cas du filtre de Deriche, les optimisations algorithmiques conduisent à un gain en surface de silicium identique à celui dû aux optimisations architecturales. D'autre part, il apparait que cette nouvelle conception du filtre se traduit aussi par une implantation logicielle plus rapide et plus compacte en terme de surface silicium. Nous aboutissons finalement à une remise en cause de la manière dont sont perçus les détecteurs de contours par les traiteurs d'images. Notre approche montre que les détecteurs de contours récursifs ou non récursifs à noyaux larges sont équivalents à des filtres de lissage 2D récursifs ou à noyaux larges combinés à de simples dérivateurs locaux.

Nous n'avons utilisé aucun outil de CAO pour la synthèse architecturale. Nous souhaitons cependant transmettre, grâce à cette expérience, un certain nombre d'idées qui doivent à notre avis être exploitées afin que des outils tels que les graphes flots de données ou les langages synchrones assistent efficacement l'architecte dans les problèmes d'ordonnancement, d'allocation, et de repliement temporel du graphe vers l'architecture. Nous pensons que cette étude constitue un exemple pertinent pour le test d'outils de synthèse architecturale.

La seconde section présente les équations de base du filtre de Deriche à deux dimensions (2D) et la structure matérielle qui en découle. Ce filtre 2D se décompose en filtres récursifs 1D. La troisième section présente la nouvelle organisation des calculs. Nous montrons dans un premier temps que la réalisation sous forme cascade plutôt que parallèle des filtres 1D réduit le nombre des mémoires de ligne. Nous définissons ensuite une nouvelle expression du lisseur de Deriche qui conduit à des coefficients et à des structures de réalisation plus simples. Puis nous établissons que la réalisation du détecteur 2D complet se décompose en un filtre de lissage 2D suivi d'un simple opérateur de gradient non récursif utilisant des masques 2×2. Nous montrons alors qu'une étude fine des allocations mémoires permet encore de réduire leur volume d'un facteur deux. La section 4 est consacrée à la réalisation matérielle. Nous étudions d'abord la précision des calculs et des coefficients; précision définie à partir de critères applicatifs concrets que nous décrivons. Puis nous traitons de l'implantation d'un filtre récursif élémentaire du 1er ordre, structure qui est au coeur de la réalisation du filtre complet. Nous montrons que le repliement temporel des calculs permet à la structure proposée de traiter quatre filtres du 1er ordre par cycle pixel et de réduire ainsi la taille des opérateurs d'un facteur quatre. Finalement nous donnons les résultats obtenus pour une réalisation complète du filtre en un unique FPGA. Dans chaque section, sont exposés (*en italique*) les aspects que doivent modéliser des descriptions flots de données pour permettre les différentes optimisations. Dans la conclusion, nous donnons quelques pistes pour lutter contre l'explosion combinatoire qui pourrait résulter de la prise en compte des différents niveaux d'abstraction nécessaires à ces optimisations.

Pour l'aspect matériel, on supposera que l'image arrive en flot de données (balayage vidéo); pour le logiciel on déduira aisément l'organisation des calculs à partir des transformées en z.

2. filtre original de Deriche

2.1. organisation originale 2D

Le gradient de Deriche est obtenu par les calcul des gradients horizontaux G_h et verticaux G_v élaborés tous deux sur le même principe. L'organisation des calculs est illustrée (Fig. 1). Par exemple le gradient vertical est calculé par la combinaison d'un filtre dérivateur vertical DV et d'un filtre de lissage horizontal LH. L'entrée E est une image d'intensité et la sortie G l'image de la norme du gradient. Le calcul de la norme, ne présentant pas de difficultés, n'est pas décrit dans ce papier.

Chaque opérateur LH, LV, DH, DV (Fig. 1) est un filtre du deuxième ordre à réponse impulsionnelle infinie à droite et à gauche *en haut et en bas* : équations (1) et (2). Les opérateurs

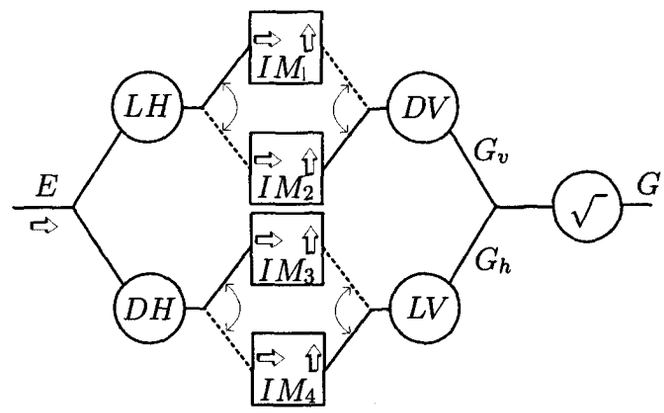


Figure 1. – Organisation classique des calculs du filtre de Deriche.

verticaux sont rigoureusement identiques aux opérateurs horizontaux et travaillent sur des colonnes au lieu des lignes. A cause de la longueur infinie de la réponse impulsionnelle, le calcul en colonne demande la disponibilité de la dernière ligne de l'image traitée horizontalement, d'où la nécessité de stockage des images complètes de résultats horizontaux. Pendant le stockage horizontal des résultats de LH (respectivement DH) dans la mémoire d'image IM_1 (respectivement IM_3), les images des résultats précédents -*flot d'images*- sont lues par colonne dans IM_2 et IM_4. Les mémoires sont donc écrites en lignes et lues en colonnes. IM_1, IM_2 tout comme IM_3, IM_4 fonctionnent en ping-pong et stockent une image sur deux.

Rendre compte de cette utilisation des mémoires dans une description flot de donnée demande de considérer le traitement d'un flot d'images.

$$DH(z) = k_D \cdot \left(\frac{z}{(1-e^{-\alpha}z)^2} - \frac{z^{-1}}{(1-e^{-\alpha}z^{-1})^2} \right) \quad (1)$$

$$LH(z) = k_L \cdot \left(\frac{(\alpha+1)e^{-\alpha}z - e^{-2\alpha}z^2}{(1-e^{-\alpha}z)^2} + \frac{1+(\alpha-1)e^{-\alpha}z^{-1}}{(1-e^{-\alpha}z^{-1})^2} \right) \quad (2)$$

$$avec \quad k_D = (1-e^{-\alpha})^2 \quad et \quad k_L = \frac{(1-e^{-\alpha})^2}{1+2\alpha e^{-\alpha}-e^{-2\alpha}}$$

2.2. organisation parallèle d'un opérateur 1D

Pour simplifier, nous n'exprimons que le cas du dérivateur horizontal DH, la structure étant globalement similaire pour les autres opérateurs. La réalisation matérielle ou logicielle sous forme récursive n'est possible qu'en décomposant le filtre horizontal en deux sous filtres (Fig.2), l'un infini à droite D_d et l'autre à gauche D_g. L'équation (1) conduit naturellement à une forme parallèle.

$$DH(z) = k_D \cdot (D_d(z) - D_g(z^{-1})) \quad (3)$$

$$D_g(z) = \frac{z^{-1}}{(1-e^{-\alpha}z^{-1})^2}$$

$$D_d(z) = \frac{z}{(1-e^{-\alpha}z)^2}$$

Puisque les pixels de l'image arrivent temporellement dans le sens gauche droite, le traitement D_d ne peut débuter qu'après l'arrivée du dernier pixel de la ligne. Les mémoires L_1 et L_2 permettent d'inverser l'ordre temporel des pixels pour l'opérateur D_d. Les mémoires L_3 et L_4 servent à resynchroniser les flots de pixel avant la soustraction. Les mémoires sont donc écrites dans l'ordre croissant des pixels et lues dans l'ordre décroissant des pixels. L_1, L_2 tout comme L_3, L_4 fonctionnent en ping-pong et stockent une ligne sur deux.

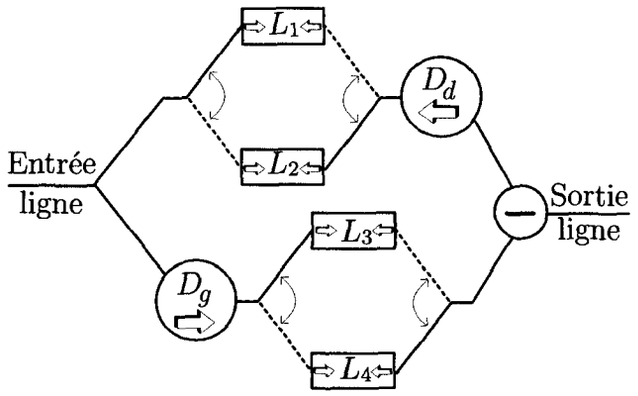


Figure 2. – Forme parallèle de l'opérateur récursif DH.

Rendre compte de cette utilisation des mémoires dans une description flot de donnée demande de considérer le traitement d'un flot de lignes et de rendre compte de l'inversion de l'ordre des évènements (pixels) pour les traitements D_d.

2.3. complexité

L'organisation 2D (Fig.1) utilise 4 mémoires d'images. Chaque opérateur 1D (Fig.2) utilise 4 mémoires de ligne, 16 mémoires de ce type sont donc nécessaires à la réalisation 2D. Des équations (1) et (2) ont déduit aisément les équations aux différences explicitant le nombre total d'opérateurs de calcul; soit au minimum 21 multiplications et 24 additions en regroupant l'ensemble des constantes de normalisation en une seule.

3. optimisations algorithmiques

3.1. forme cascade des opérateurs 1D

Pour réduire d'un facteur deux le nombre des mémoires de ligne par opérateur (tel que DH), il faut renoncer à la structure parallèle (Fig.2) et utiliser une structure où les traitements *gauche* et *droit* sont en cascade (Fig.3). La forme cascade demande alors

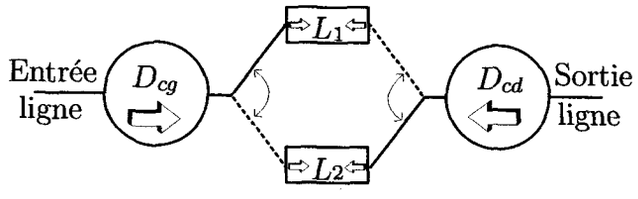


Figure 3. – Forme cascade de l'opérateur récursif DH.

exactement le même nombre de calculs que la forme parallèle initiale.

L'équation (4) donne l'expression de la transformée en z du dérivateur cascade de Deriche équivalente à l'équation (3). On écrit $\gamma = e^{-\alpha}$ le paramètre réglant la largeur du filtre.

$$D_c(z) = D_{cg}(z) \cdot D_{cd}(z) \tag{4}$$

$$D_{cg}(z) = \frac{(1-\gamma)^2 \cdot z^{-1}}{(1-\gamma z^{-1})^2}$$

$$D_{cd}(z) = \frac{(1-\gamma^2) \cdot (z^2-1)}{(1-\gamma z)^2}$$

3.2. nouvelle expression du lisseur

L'optimisation précédente est aussi applicable au filtre de lissage. Cependant l'équation (2) montre que les coefficients et les numérateurs de ces filtres sont moins simples que ceux du dérivateur. Ceci a des répercussions à la fois sur l'architecture globale 2D (voir §3.3) et sur l'uniformisation de l'architecture des filtres élémentaires récursif à concevoir (voir section 4). Afin de simplifier coefficients et numérateurs, plutôt que de calculer le lisseur comme l'intégrale continue de la réponse impulsionnelle du dérivateur, ainsi que l'a établi Deriche, nous proposons de l'obtenir par intégration numérique *-méthode des trapèzes-* de l'équation (4). Nous avons détaillé la construction de ce lisseur [5], et montré par les critères de Canny discrets, que notre approche ne conduisait à aucune différence de qualité par rapport au lisseur initial. Il s'en dégage cependant une expression du filtre et des coefficients plus simples. La méthode des trapèzes donne :

$$L_c(z) = k \cdot D_c(z) \cdot \frac{1+z^{-1}}{1-z^{-1}}$$

où k est une constante de normalisation choisie pour que l'amplification statique du lisseur soit unitaire. L'équation en z pour la forme cascade du lisseur s'en déduit :

$$\begin{aligned} L_c(z) &= L_{cg}(z) \cdot L_{cd}(z) \\ L_{cg}(z) &= \frac{(1-\gamma)^2(1+z^{-1})}{2\,(1-\gamma z^{-1})^2} \quad et \quad L_{cd}(z) = L_{cg}(z^{-1}) \end{aligned} \tag{5}$$

3.3. nouvelle organisation 2D

De façon à généraliser l'utilisation de la transformée en z au traitement 2D, on peut noter que la distance entre deux pixels d'une même colonne de deux lignes consécutives est de N pixels si N est le nombre de pixels par ligne ou colonne. Ainsi pour les traitements verticaux, il suffit de changer l'opérateur retard z^{-1} en l'opérateur z^{-N}. On note $G_h(z)$ le filtre de gradient horizontal et $G_v(z)$ celui de gradient vertical.

$$G_h(z) = L_c(z^N) \cdot D_c(z) \quad et \quad G_v(z) = L_c(z) \cdot D_c(z^N)$$

La similitude des équations (4) et (5), obtenues grâce aux innovations précédentes conduit naturellement à définir un filtre de lissage 2D : $L_2(z)$ qui est bien sûr séparable. L'équation (6) rend compte de la décomposition de ce filtre en cellules élémentaires.

$$L_2(z) = \frac{(1+\gamma)}{(1-\gamma)} \cdot L_1(z) \cdot L_1(z^N)$$

$$L_1(z) = \frac{(1+z^{-1})}{2} \cdot \frac{(1-\gamma)^2}{(1-\gamma z^{-1})^2} \cdot \frac{(1-\gamma)^2}{(1-\gamma z)^2} \tag{6}$$

Avec les filtres de dérivation de taille 2×2 horizontal S_h et vertical S_v :

$$\begin{aligned} S_h(z) &= (z-1) \cdot (1+z^N) \\ S_v(z) &= (z^N-1) \cdot (1+z) \end{aligned}$$

on obtient alors :

$$\begin{aligned} G_h(z) &= L_2(z) \cdot S_h(z) \\ G_v(z) &= L_2(z) \cdot S_v(z) \end{aligned}$$

Ceci montre que le filtre de Deriche n'est autre qu'un filtre de lissage optimal 2D symétrique couplé à un simple opérateur de dérivation 2×2 (filtre de Robert). Le filtre de lissage 2D est lui même décomposé en deux filtres de lissage identiques horizontal $L_1(z)$ et vertical $L_1(z^N)$. Cette présentation a posteriori des équations ne correspond pas à notre démarche qui a été plus intuitive. C'est la volonté de réduire le nombre des mémoires d'image pour une organisation 2D optimale (§3.5) qui nous a amené à renforcer la similitude des expressions des filtres de lissage et de dérivation et donc à utiliser une intégration par la méthode des trapèzes pour déduire l'expression du lisseur de celle du dérivateur. Ce sont bien des considérations architecturales qui ont conduit à un nouvel algorithme !

3.4. optimisations des allocations mémoire

Un facteur deux sur le nombre de mémoires ligne peut encore être gagné en remarquant [4] que lorsqu'un pixel de la ligne précédente est lu dans la mémoire, on peut écrire à sa place un pixel de la ligne courante (Fig. 4). Les lignes consécutives sont alors stockées alternativement dans une même mémoire dans le sens croissant et décroissant des adresses. Une unique mémoire de ligne remplace deux mémoires en ping pong.

Dans le mode ping-pong, on ne réalise sur chaque mémoire qu'un seul accès par cycle pixel, alors qu'une lecture écriture à la même adresse est nécessaire pour la version de la figure 4. Il est possible

de conserver un seul accès en lecture ou en écriture par cycle pixel. Avec une mémoire deux fois plus large, mais possédant deux fois moins d'adresses, on peut lire ou écrire deux pixels en parallèle un cycle pixel sur deux.

Un outil de CAO ne peut réaliser ces optimisations des mémoires en taille et rapidité dans une description flot de donnée qu'à condition de décrire explicitement le traitement d'un flot de lignes de pixels.

Dans le même esprit, chaque organisation en ping-pong des mémoires d'image (Fig. 1) peut aussi être remplacée (Fig.5) par une unique mémoire d'image [6] avec la même démarche qui a conduit à l'optimisation des mémoires de ligne. La première image d'un flot est écrite par ligne horizontalement. Puis la lecture verticale d'une colonne de cette première image libère une zone verticale qui permet l'écriture d'une ligne de la deuxième image, etc. Les images sont donc alternativement stockées horizontalement et verticalement. Comme pour les mémoires de ligne, le nombre d'accès mémoire par cycle pixel peut être réduit en accédant à plusieurs pixels par cycle.

Un outil de CAO ne peut réaliser ces optimisations des mémoires dans une description flot de donnée qu'à condition de décrire explicitement le traitement d'un flot d'images.

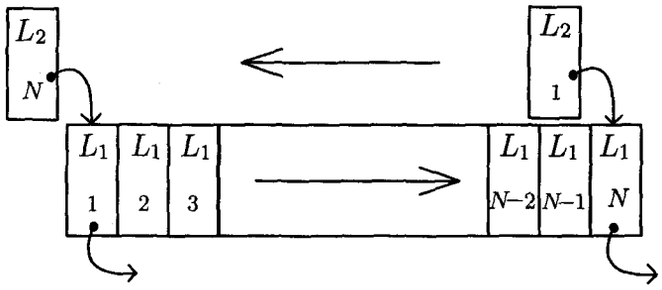


Figure 4. – Optimisation des allocations en mémoires de ligne.

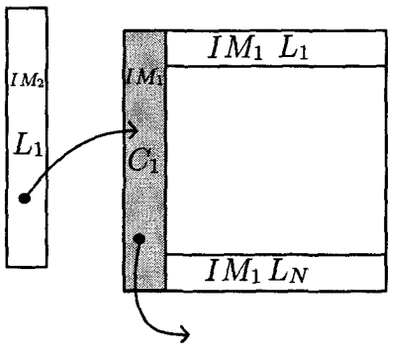


Figure 5. – Optimisation des allocations en mémoires d'image.

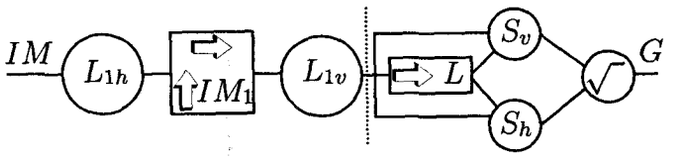


Figure 6. – Nouvelle organisation 2D du filtre de Deriche.

3.5. organisation 2D optimale

L'architecture 2D originale (Fig. 1) est maintenant remplacée par celle de la figure 6. On y retrouve les deux filtres de lissage horizontaux et verticaux séparés par une unique mémoire d'image. La partie à droite du pointillé réalise la détection de contours à l'aide d'une unique mémoire de ligne et des 2 opérateurs S_h et S_v utilisant seulement 4 additionneurs.

Ce schéma est utilisable pour tous les filtres détecteurs RII ou RIF, le lisseur peut être par exemple un filtre gaussien et dans le cas des filtres Laplacien à noyau large, le filtre de dérivation est simplement remplacé par un Laplacien 3×3.

Par exemple, en choisissant $\gamma = 0$ dans l'équation (6), on obtient le filtre de Sobel. On peut noter aussi que chaque partie droite ou gauche du filtre $L_1(z)$ est constituée d'un filtre du deuxième ordre combinant deux premiers ordre identiques. En annulant le coefficient γ d'un des deux filtres premier ordre, on obtient le filtre de Shen-Castan.

Dans un dernier souci de simplification, nous proposons la structure (Fig.7) qui a donné lieu à la réalisation matérielle décrite dans la prochaine section. Par rapport à l'équation (6), le coefficient de normalisation $\frac{1+\gamma}{1-\gamma}$ a été supprimé. Le seuil de détection de contours peut être ajusté en conséquence. D'autre part, dans la réalisation de L_1, le terme $\frac{1+z^{-1}}{2}$ a été supprimé, il en résulte une légère perte de l'amélioration du rapport signal sur bruit pour les faibles valeurs de γ mais une augmentation en proportion de la localisation du contour. Une discussion de ces deux points est donnée dans [7]. On voit nettement Figure 7 la décomposition en cellules du premier ordre identiques. On peut penser que la répartition du numérateur de L_1 sur chaque cellule augmente le nombre de multiplications. Nous montrerons dans la section 4 qu'il n'en est rien. L'homogénéité des cellules peut par ailleurs permettre des optimisations architecturales.

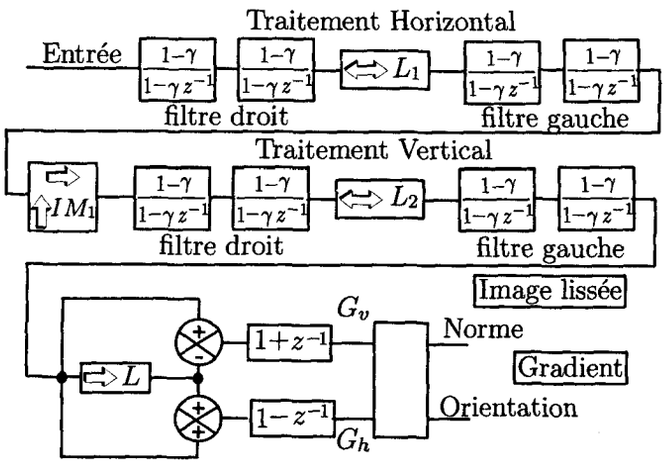


Figure 7. – Détails de la nouvelle organisation 2D du filtre de Deriche.

3.6. complexité, performances logicielles

La nouvelle organisation est optimale en nombre de mémoires. En effet, la nature récursive des filtres impose l'existence d'au moins une mémorisation d'image entre les traitements horizontaux et verticaux et, pour chaque filtre 1D, d'au moins une mémorisation ligne pour le balayage dans le sens contraire du flot pixel. L'organisation proposée atteint la borne minimale du nombre de mémoires nécessaires. Un facteur quatre a été gagné sur le nombre des mémoires d'image et un facteur supérieur à cinq sur les mémoires de ligne. Pour une réalisation logicielle, on a intérêt à regrouper les calculs de deux filtres du premier ordre successifs en un seul filtre du second ordre. Le traitement 2D ne nécessite alors plus que 9 multiplications et 12 additions si la constante de normalisation $(1-\gamma)^4$ est prise en compte en une seule fois. Il faut compter un coût identique en rapidité pour les additions et les multiplications flottantes en logiciel. On passe alors de 45 opérations flottantes par pixel à 21. Le traitement d'une image 512×512 sur une Sun Sparc 5 passe de 8,2 à 4,8 secondes.

4. optimisations architecturales

4.1. précisions des coefficients, des calculs

La connaissance de l'organisation architecturale (Fig.7) nous permet de dimensionner les opérateurs, les chemins de donnée et la largeur des mémoires. Les multiplieurs sont les opérateurs les plus critiques en matière de rapidité et de surface silicium. Les travaux initiaux de Deriche [1] ont montré que données et coefficients devaient être codés sur 19 bits. Des simulations effectuées par L.Torres [8] avec des critères différents ont abouti à des codages sur 12 bits. Compte tenu des technologies actuelles et des vitesses nécessaires (100 ns/pixel), les structures simples de multiplieurs ne sont pas utilisables. D'autre part, le pipeline peut difficilement être utilisé dans les boucles récursives pour augmenter le débit de traitement. L.Torres propose alors deux alternatives : l'optimisation de la structure des multiplieurs ou l'utilisation de techniques d'anticipation de calcul. Ces deux approches ont en commun d'augmenter la surface de silicium. Grâce à une remise en cause des critères permettant de définir les précisions, nous montrons que la taille des coefficients n'influe pas sur la qualité du résultat et que ces derniers peuvent être codés sur 3 bits voire 1 bit!

4.1.1. précision des coefficients

Nous ne détaillons pas les approches utilisées par L.Torres et Deriche. Elles sont basées sur une répartition statistique des erreurs sur les coefficients de la réponse indicielle ou de la réponse en fréquence du filtre. Or le filtre de Deriche n'utilise qu'un unique coefficient γ. Le choix du codage de γ définit uniquement le nombre de filtres différents réalisables, ceux-ci étant alors réalisés parfaitement [9]. L'approche statistique est dans ce cas non fondée. Nous avons choisi de coder γ sur 3 bits.

$$\gamma = \gamma_1 2^{-1} + \gamma_2 2^{-2} + \gamma_3 2^{-3} \tag{7}$$

Il en résulte que sept filtres d'échelles différentes sont réalisables. Les échelles choisies correspondent à une zone de résolution couramment utilisée par les traiteurs d'image. Dans la table suivante, la résolution correspond à la distance entre deux contours pour laquelle le choix du coefficient d'échelle est optimal en matière d'amélioration du rapport signal à bruit [10].

γ	α	résolution
0.125	2.08	1
0.250	1.38	2
0.375	0.98	3
0.500	0.69	4
0.625	0.47	5
0.750	0.29	8
0.875	0.13	18

Remarquez qu'il est aussi possible de définir γ par :

$$\gamma = 1 - 2^{-\varepsilon}$$

La progression des résolutions est alors géométrique et plus adaptée à une décomposition multi-échelle du type ondelette. La multiplication se ramène dans ce cas à une unique addition!

4.1.2. précision des calculs

Chaque filtre du premier ordre de la figure 7 est normalisé, c'est à dire qu'aucun débordement ne se produit si la partie entière du résultat est codée sur autant de bits que l'entrée. Cependant, les filtres récursifs sont par nature imprécis. En effet, l'emploi de coefficients tels que γ inférieur à 1, engendre des résultats de multiplication comportant une partie décimale de longueur croissante au fil des itérations. Le registre qui mémorise la valeur précédente de la sortie étant de taille finie, il en résulte obligatoirement une erreur de troncature ou d'arrondi.

Nous avons retenu de façon pragmatique deux critères pour dimensionner la taille de ces registres et, par là même, les opérateurs et la largeur des mémoires. Nous supposons que le signal d'entrée est codé sur 8 bits et nous associons par convention au poids faible de sa représentation binaire la valeur 2^0.

Premier critère : Le signal d'entrée est lui-même tronqué, la troncature étant modélisée par un bruit uniforme de variance centrée $\sigma_e^2 = \frac{2^0}{12}$. Le filtrage passe bas L_2 améliore le rapport signal bruit d'un facteur asb_γ dépendant de γ. Puisque le filtre

est normalisé, le signal n'est pas amplifié et le bruit de troncature d'entrée ramené en sortie a pour variance σ^2_{es} tel que :

$$\sigma^2_{es} = \frac{\sigma^2_e}{asb^2_\gamma}$$

Les sources d'erreur sont assimilées à des bruits indépendants de densité uniforme. On peut calculer analytiquement, de façon classique, le bruit de calcul ramené en sortie de variance σ^2_{cs}. Le premier critère de dimensionnement est que le bruit de calcul en sortie soit inférieur ou égal au bruit de troncature d'entrée ramené en sortie : $\sigma^2_{cs} \leq \sigma^2_{es}$. Le pire cas correspond à la valeur de γ la plus proche du cercle unité. Si $\gamma = 1 - 2^{-\varepsilon}$, on montre alors [8] que la sortie de L_2 doit être codée sur $1,5\,\varepsilon + 11,68$ bits pour une entrée sur 8 bits. La valeur de $\gamma = 0,875$ conduit à un codage sur 17 bits.

On peut justifier ce critère par le fait que le signal d'entrée sera bruité plus fortement que par σ^2_e à cause du bruit présent dans l'image et dans la chaîne d'acquisition. L'objectif du filtrage étant de réduire ce bruit, il est naturel que le bruit de calcul ne dégrade pas sensiblement l'amélioration obtenue.

Deuxième critère : L'objectif du filtre complet est la détection de contours. le calcul du gradient est suivi d'une suppression des non maxima locaux dans la direction du gradient. Il convient que le maximum soit unique dans les pire cas de contraste envisageables. On peut montrer [7] que le pire cas correspond à un échelon diagonal de hauteur 2^0. Le signal subit alors l'effet du filtre dans les deux directions. L'écart entre le maximum du gradient et le gradient voisin vaut alors : $\Delta \simeq 2^{-(3\varepsilon+4)}$.

La sortie de L_2 doit être codée sur $3\,\varepsilon + 12$ bits pour une entrée sur 8 bits. La valeur de $\gamma = 0,875$ conduit à un codage sur 21 bits.

On note que ce second critère est plus contraignant que le premier. A partir de cette approche analytique, nous avons effectué des simulations et tenté de réduire la largeur des stockages en mémoire en conservant des précisions de calcul plus grandes. Nous avons ainsi montré par essai sur des images synthétiques et des images réelles que les calculs devaient être effectués sur 20 bits et les stockages mémoires sur 12bits. Nous avons ainsi constaté que le bruit de calcul en introduisant des disparités locales aide la détection de maximum sans introduire de délocalisation des contours.

4.2. réalisation du filtre L_1

4.2.1. *filtre du premier ordre combinatoire*

L'opérateur de lissage 1D est constitué de quatre cellules récursives du premier ordre identiques de transformée $P(z)$.

$$P(z) = \frac{1-\gamma}{1-\gamma z^{-1}}$$

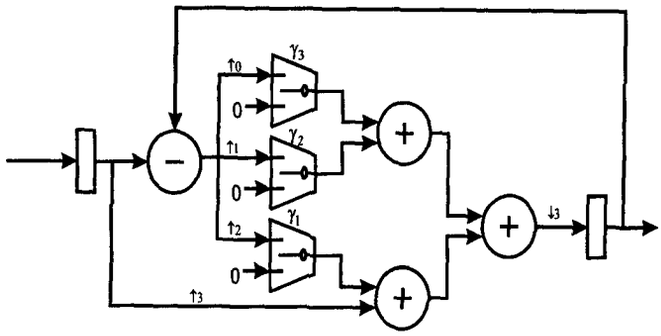


Figure 8. – Structure du premier ordre combinatoire.

Avec le codage sur 3 bits de γ (7), une structure de réalisation combinatoire simple est proposée (Fig.8).

Les symboles $\uparrow i$ et $\downarrow j$ indiquent respectivement des translations des bus données de i bits à gauche, de j bits à droite qui n'occupent aucune surface de silicium. Entrée et sortie sont codées sur 20 bits. On peut noter que seulement quatre additionneurs sont nécessaires, et que la structure peut être rendue transparente en annulant γ. Cette structure a été réalisée avec la famille de FPGA ALTERA flex8000. L'équivalent de 1800 portes est utilisé et la fréquence de traitement atteint 19Mhz.

Remarquons [8] que des choix de codage de coefficients sur 12 ou 16 bits amènent les architectes à étudier des méthodes d'accélération des traitements. Notre choix conduit à une structure si simple qu'on peut envisager avec une unique structure matérielle de réaliser plusieurs traitements dans le même cycle pixel. Il est aussi bien admis que la réalisation de multiplieurs dans les technologies FPGA est un problème difficile car ceux-ci occupent une surface importante. On a alors généralement recours soit à des réécritures de l'algorithme vers du bit série (méthode de Peled et Liu pour les filtres RIF), soit dans le cas des multiplications par des constantes à des optimisations de surface en n'implantant que les additionneurs nécessaires. Dans notre exemple, cela imposerait une reconfiguration du FPGA pour chaque changement d'échelle [9][11].

4.2.2. *pipeline latéral*

Pour optimiser la surface des opérateurs, on peut replier temporellement les traitements sur le même matériel. Une première solution (Fig. 9) consiste à sérialiser les données sur 20 bits en deux blocs de 10 bits. La structure (Fig. 9) traite un premier ordre avec des données sur 10 bits en 50ns et effectue le calcul d'un premier ordre complet *blocs poids faibles (L) et poids forts (H)* en 100ns. Des sauvegardes de retenues intermédiaires et un retiming partiel de certains bits résultats sont nécessaires comme indiqué sur le schéma. Nous avons testé cette structure avec la même famille FPGA que pour la structure combinatoire. Alors que le gain théorique en surface est de 100%, nous n'avons obtenu qu'un gain de 25%. Ceci s'explique d'une part par des problèmes de routage plus difficile pour cette structure et aussi par le fait

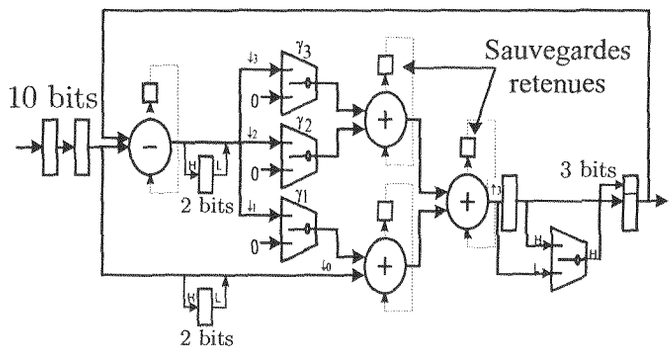


Figure 9. – Structure bloc série d'un premier ordre.

qu'à cause des décalages nécessaires à la multiplication, obtenir 10 bits en sortie impose des additions sur 13 bits et non pas sur 10 bits.

Cette solution ne peut être obtenu sur un outils de CAO qu'en décrivant l'algorithme à la fois au niveau pixel et au niveau bit. Il est nécessaire de prendre en compte bit par bit les délais de calcul des opérateurs et aussi les délais dus aux interconnexions.

4.2.3. pipeline longitudinal

Le pipeline ne peut pas être utilisé à l'intérieur d'une boucle récursive pour accélérer le débit de sortie, par contre il peut être utilisé pour augmenter le débit interne de la structure. Il est par exemple possible (Fig.10) de réaliser de manière entrelacée les quatre calculs du premier ordre du filtre L_1. L'horloge est maintenant de 40Mhz pour un flot pixel d'entrée toujours à 10Mhz. On distingue quatre phases de calcul : (0) 1^{er} ordre 1 du filtre L_g et entrée pixel, (1) $2^{ème}$ ordre 1 du filtre L_g,(2) 1^{er} ordre 1 du filtre L_d,(3) $2^{ème}$ ordre 1 du filtre L_d et sortie du résultat. On constatera aussi qu'en récupérant le résultat à l'issue de la phase (2) on réalise le filtre de Shen! Il faut noter que les registres ajoutés par rapport à la solution combinatoire ne coûtent rien puisqu'ils sont disponibles dans les bloc logiques des FPGA. La rapidité maximale est maintenant fixée par la traversée d'un unique additionneur 20 bits.

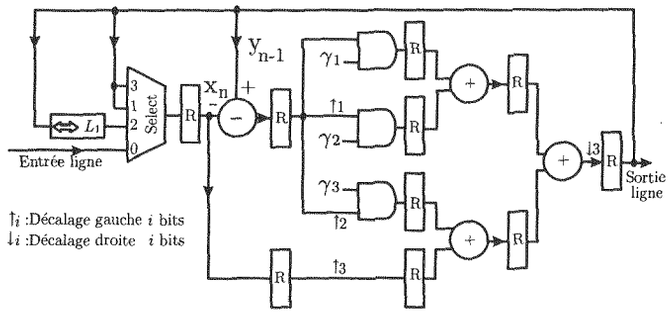


Figure 10. – Entrelacement d'ordre 4 sur une structure du premier ordre.

Cette méthode utilise implicitement le repliement de graphe flot de donnée et peut donc être exploitée avec les outils actuels de synthèse d'architecture.

4.3. résultats

A des fins de validation, nous avons réalisé un filtre de Deriche 1D incluant la mémoire de ligne (256×12) en un unique FPGA XILINX 4010. La synthèse a été réalisée à partir d'une description comportementale VHDL. Aucune intervention manuelle sur le routage n'a été nécessaire. 393 blocs logiques sur 400 disponibles ont été utilisés. Le circuit fonctionne à une cadence pixel de 10Mhz. Le filtre est réalisé par un pipeline longitudinal utilisant un entrelacement d'un facteur deux. La figure 11 montre les images obtenues ainsi que des coupes suivant une ligne de l'image. Nous avons aussi réalisé un filtre 2D complet suivant la structure de la figure 7 en un unique FPGA XILINX 4025. Toutes les mémoires sont cette fois ci placées à l'extérieur. 857 blocs logiques sont utilisés pour la réalisation des opérateurs et la gestion des mémoires.

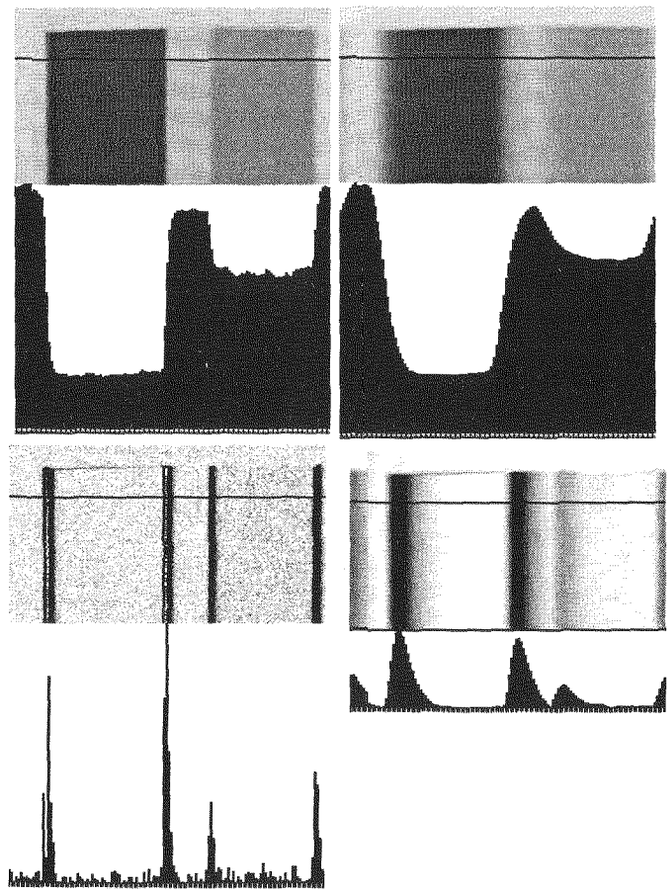


Figure 11. – Lissage (en haut) et gradient (en bas) pour deux valeurs de γ : 0,125 (gauche) et 0,875 (droite).

5. conclusion

Les optimisations algorithmiques ont permis de réduire la taille des mémoires d'un facteur deux et d'homogénéiser les opérateurs à implanter. Les optimisations architecturales ont alors permis de gagner un nouveau facteur deux sur les tailles mémoires et réduit le filtre complet 2D à l'implantation de seulement huit additionneurs 20 bits (entrelacement d'un facteur quatre) pour la partie lissage et quatre pour la partie dérivation locale. Initialement 21 multiplications et 24 additions étaient nécessaires ! Bien que non développé dans cet article, une réalisation ASIC est naturellement envisageable. Compte tenu de la simplicité de réalisation qui découle de notre étude, pour être industriellement justifiée, celle-ci doit inclure une chaine complète de segmentation d'images.

Toutes les optimisations architecturales proposées sont réalisables à court terme à l'aide d'outils de CAO. Il semble peu réaliste d'envisager une description complète allant du flot d'image aux bits élémentaires constituant le pixel à cause de l'explosion combinatoire de la taille du graphe flot de données qui en résulterait. Nous pensons que la prise en compte de seulement deux niveaux de description simultanés est nécessaire :

– flot image, ligne ou colonne pour optimiser la mémoire d'image en taille et rapidité,

– ligne ou colonne et pixel pour optimiser la mémoire ligne en taille et rapidité,

– pixel et bits pour optimiser les opérateurs.

BIBLIOGRAPHIE

[1] R.Deriche, «Fast Algorithms for Low-Level Vision», *IEEE Transactions on Pattern Anal. and Machine Intell.*, vol.PAMI-12, no.1, pp. 78-87, Jan.90

[2] J.Shen and S.Castan, «An Optimal Linear Operator for Step Edge Detection», *Comput. Vision Graphics Image Process.*, vol.54, no.2, pp. 112-133, Mar. 92

[3] D.Demigny, J.Devars, L.Kessal, J.F.Quesne, «Implantations temps réel du filtre de lissage d'images de Nagao», *Revue Traitement du signal*, Vol.10, no 4, 1993

[4] N.Zarka, «Conception d'un circuit intégré de détection optimale de contours» *Thèse de Doctorat en électronique de Paris VI*, Déc. 92

[5] D.Demigny, F.G.Lorca, T.Kamlé, L.Kessal, «Critères de Canny discrets pour la comparaison de filtres détecteurs de contours», *GRETSI Symposium on Signal and Image Processing*, Juan les Pins France, Sept. 95

[6] D.Demigny, F.G.Lorca, J.P.Cocquerez, L.Kessal, «Conceptions nouvelles du détecteur de contours de Deriche», *GRETSI Symposium on Signal and Image Processing*, Juan les Pins France, Sept.95

[7] F. Garcia Lorca, «Filtres récursifs temps réel pour la détection de contours : optimisations algorithmiques et architecturales», *thèse de Doctorat en électronique*, Université d'Orsay, Nov. 96

[8] L. Torres, «Intégration de Filtres Numériques pour le Traitement d'Images : du Silicium au Système Reconfigurable», *thèse de Doctorat en électronique*, Université Montpellier II, Jui.96

[9] D.Demigny, L. Kessal, T.Kamlé, J.P. Cocquerez , «Filtre de Deriche, architectures temps réel pour la multi-résolution», *GRETSI Symposium on Signal and Image Processing*, Juan les Pins France, Sept. 93

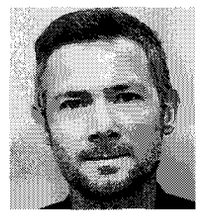
[10] M.Karabernou, D.Demigny, «An effective resolution definition or how to choose an edge detector, its scale parameter and the threshold», *IEEE International Conference on Image Processing*, vol.1, pp.829-832, Lausanne, Oct.96

[11] Sarifuddin, «Implantation sous forme de circuit spécialisé d'un algorithme de détection de contours multi-échelles», *thèse de Doctorat*, Université de Bourgogne, 1995

Manuscrit reçu le 9 octobre 1997.

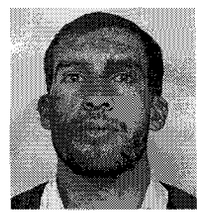
LES AUTEURS

Didier DEMIGNY



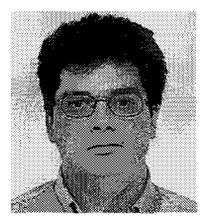
Didier Demigny est professeur d'université dans l'Equipe Traitement des Images et du Signal (ETIS) depuis 1994. Il est actuellement directeur adjoint de cette équipe et responsable du groupe Architecture. Ses activités de recherche sont centrées sur deux thèmes : l'élaboration de critères de qualité pour la comparaison de détecteurs de contours et les architectures temps réel embarquées pour les traitements de bas et moyen niveaux des images.

Lounis KESSAL



Lounis Kessal est maître de conférence à ETIS depuis 1990. Il est responsable des activités CAO numérique de l'équipe. Son expertise en conception et réalisation d'opérateurs temps réel de segmentation, l'amène actuellement à étudier les possibilités offertes par la reconfiguration dynamique des FPGA en vue de séquencer une suite de traitements sur la même structure physique.

Federico GARCIA LORCA



Federico garcia Lorca a mené sa thèse au sein d'ETIS après un DEA d'architecture obtenu à Paris XI. Sa thèse, soutenue en décembre 96 a consisté en l'étude de la conception de filtres détecteurs de contours récursifs et en l'exploitation de ceux-ci pour l'extraction de points caractéristiques (angles) dans les images. Il est actuellement enseignant.