Réalisation récursive temps réel de filtres RIF : filtre de Canny, filtre gaussien et ses dérivées

Didier Demigny, Julien Pons, Nassima Boudouani, Lounis Kessal

Equipe Traitement des Images et du Signal
UPRESA CNRS 8051
ENSEA et Université de Cergy Pontoise
ENSEA, 6 av. du Ponceau, 95014 Cergy Pontoise cedex, France
demigny@ensea.fr

Résumé –
Ce papier présente le filtre POAG (Polynomial Approximation Of Gaussian) remplaçant avantageusement les filtres de Canny, les filtres gaussiens et leurs dérivées première et seconde. Bien qu'il soit à réponse impulsionnelle finie, il est réalisable sous forme récursive. Sous cette forme, ses pôles étant situé sur le cercle unité (en limite de stabilité), on montre que cela conduit à une grande simplicité de calculs, à la suppression des mémoires d'images pour le filtrage 2D et à une utilisation possible pour des signaux temps réel à échantillonnage temporel. Sont discutés : la qualité de l'approximation, les conditions de stabilité de l'implantation, ses performances en rapidité par rapport à une convolution classique, et une implantation matérielle pour ASIC ou FPGA.

Abstract –
This paper presents the POAG filter (Polynomial Approximation Of Gaussian) which can advantageously replace the Canny or Gaussian filters and their first and second derivatives. Even if it is a finite response filter, a recursive implementation is available. In this case, all the poles are located on the unit circle (stability limit). We show that this singularity induces computation simplicity, image memory suppression (2D case) and the usability for real time, time sampled data. We discuss the accuracy of the approximation, stability conditions of the implementation, speed performances compared to classical convolution, and a hardware implementation for ASIC or FPGA.

1 Introduction

Canny [1, 2] a défini un filtre optimal à réponse impulsionnelle finie pour la détection de contours. Il a aussi montré que son filtre pouvait être approximé par un filtre gaussien (en fait par sa dérivée première) par ailleurs largement utilisé que ce soit pour des filtrages 1D ou 2D. L'inconvénient principal de ces filtres est que la complexité de calcul croit linéairement avec le nombre de coefficients significatifs de la réponse impulsionnelle et donc avec la diminution de la résolution. Dans un système multirésolutions, l'architecture est alors nécessairement dimensionnée par le pire cas (résolution la plus faible). Deriche [3] a proposé une extension récursive (à réponse impulsionnelle infinie) des filtres de Canny conduisant à une complexité de calcul indépendante de la résolution. Cette résolution est alors définie par un unique coefficient. L'inconvénient principal du filtre de Deriche est que l'infinité de la réponse impulsionnelle (côté causal et anticausal) impose pour des raisons de stabilité quatre parcours de l'image (gauche vers droite, droite vers gauche, haut vers bas et bas vers haut) et donc une latence image dans le traitement. Cette double infinité de la réponse impulsionnelle interdit aussi l'usage de ce filtre pour des signaux temps réel à échantillonnage temporel. Du point de vue architectural, la présence de multiplieurs dans les boucles récursives (liée aux pôles de la fonction de transfert) tend à limiter les performances de rapidité parce que dans ce cas, le pipeline est très difficilement utilisable sauf au prix d'un surcoût matériel important.

Une extension récente de nos travaux initiaux [4] sur la discrétisation des critères de Canny nous a amené à la définition d'un filtre sous-optimal (-2% par rapport à l'optimal) à réponse impulsionnelle finie dont la demi-réponse impulsionnelle est un polynôme. Il est alors possible d'en déduire une forme récursive dont tous les pôles de la transformée en z sont situés sur le cercle unité et donc en limite de stabilité. Il en résulte une division par deux du nombre de balayages image et l'utilisation possible de ce filtre pour des signaux temps réel à échantillonnage temporel. On bénificie simultanément des avantages des filtres à réponse impulsionnelle finie et des structures récursives sans en avoir les inconvénients. Ce filtre appelé PAOG (Polynomial Approximation Of Gaussian) remplace avantageusement les filtres de Canny et les filtres gaussiens. La même architecture permet sans calcul supplémentaire d'obtenir le résultat du lissage et de ses quatre premières dérivées ! La section 2 présente le filtre PAOG et définit la précision de l'approximation du filtre gaussien. La section 3 propose une réalisation récursive de ce filtre et discute les conditions de stabilité, d'initialisation et les performances de cette version récursive. La dernière section propose une implantation matérielle dans le cas 1D.

2 Le filtre POAG

2.1 Réponse impulsionnelle du filtre POAG

Le nombre de ceofficients de la réponse impulsionnelle RI du filtre (liée à la résolution) vaut $2w+1$. L'expression de la RI h^w du filtre POAG est :

$$\begin{aligned} h_k^w &= C_p.(w+2-|k|)(w+1-|k|) \\ &\quad .(-3k^2+(2w+3)|k|+w(w+3)) \end{aligned} \tag{1}$$

où C_p est un coefficient de normalisation choisi pour que la somme des coefficients soit égale à 1 :

$$C_p = \frac{5}{2w(w+1)(w+2)(w+3)(2w+3)} \tag{2}$$

Pour chaque valeur de w, il existe une valeur de l'écart type σ de la gaussienne pour laquelle h^w est une approximation précise de la RI g^σ du filtre gaussien :

$$g^\sigma = C_g.e^{-\frac{k^2}{2\sigma^2}} \tag{3}$$

En utilisant une minimisation de l'erreur quadratique absolue, on obtient une relation entre g^σ et w :

$$\sigma = 0.3217w + 0.481 \tag{4}$$

Sur la figure 1, on observe la demi RI de h^w et de g^σ pour $w = 20$ et $\sigma = 6.915$. On note que les plus grandes différences apparaissent à l'extémité de la fenêtre pour les plus faibles valeurs des coefficients.

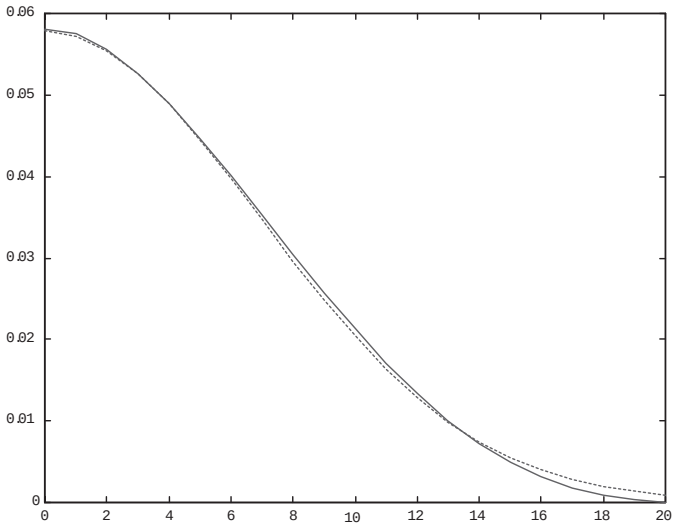

FIG. 1 – Demi réponse impulsionnelle du filtre gaussien (traits pointillés) avec $\sigma = 6.915$ et du filtre POAG (traits continus) avec $w = 20$.

2.2 Précision de l'approximation

On note g_w^σ la restriction à la fenêtre de taille $(2w+1)$ de la fonction g_w. Trois différentes mesures peuvent être utilisées pour définir l'erreur de l'approximation dans la fenêtre .

(a) La moyenne de l'erreur quadratique relative mrq

$$mrq = \frac{1}{2w+1}\sqrt{\sum_{-w}^{w}(1-\frac{h^w}{g_w^\sigma})^2} \tag{5}$$

(b) La moyenne de l'erreur quadratique absolue normalisée par la valeur de $h^w(0)$: maq

$$maq = \frac{1}{h^w(0)(2w+1)}\sqrt{\sum_{-w}^{w}(g_w^\sigma - h^w)^2} \tag{6}$$

(c) Le maximum de l'erreur quadratique absolue normalisée par la valeur de $h^w(0)$: Maq

$$Maq = \frac{\max_k |g_w^\sigma(k) - h^w(k)|}{h^w(0)} \tag{7}$$

La figure 2 montre les erreurs en %pour w variant de 1 à 20 ($0.8 < \sigma < 6.91$). mrq est inférieure à 6%, Maq est approximativement constante (2%) et maq inférieure à 0.5%.

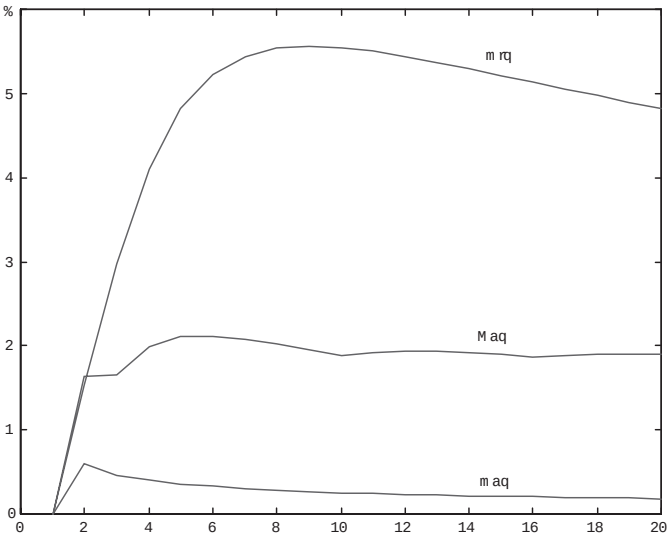

FIG. 2 – Erreurs quadratiques moyenne absolue (maq), maximum absolue (Maq) et moyenne relative (mrq) entre g_w^σ et h^w en % en fonction de w.

Une dernière mesure met en évidence les effets de troncature de la réponse infinie de la gaussienne. Le rapport entre la somme des coefficients de g^σ en dehors de la fenêtre et de la somme des coefficients de g^σ à l'intérieur de la fenêtre est inférieur à 1% pour toutes les valeurs de w.

Tous ces résultats confirment que le filtre POAG est une bonne approximation du filtre gaussien.

3 Implantation récursive

3.1 Transformée en Z

La transformée en z de h^w peut être décomposée en :

$$H(z) = \sum_{k=-w}^{0} h_k z^{-k} + \sum_{k=0}^{w} h_k z^{-k} - h_0 \tag{8}$$

Pour calculer la seconde somme de (8), on commence par développer le polynôme en :

$$\sum_{0}^{w} h_k z^{-k} = \sum_{i=0}^{4} a_i \sum_{k=0}^{w} k^i z^{-k} \tag{9}$$

où les a_i sont des fonctions de w.

La seconde somme de (9) peut être analytiquement évaluée :

$$\sum_{k=0}^{w} k^i z^{-k} = \frac{N_i(z^{-1})}{(1-z^{-1})^i} \tag{10}$$

où $N_i(z^{-1})$ est un polynôme de degré i de la variable z^{-1}.

En introduisant (10) dans (9) et en utilisant la même méthode pour calculer la première somme de (8), on obtient l'expression de $H(z)$:

$$H(z) = C'_p \frac{z^{-2}(1+z^{-1})N(z)}{(1-z^{-1})^5} \tag{11}$$

avec :

$$\begin{aligned} N(z) &= w(z^{(w+2)} - z^{-(w+2)}) \\ &\quad +(w+3)(z^{-(w+1)} - z^{(w+1)}) \\ &\quad +(2w+3)(z - z^{-1}) \end{aligned} \tag{12}$$

et :

$$C'_p = \frac{30}{w(w+1)(w+2)(w+3)(2w+3)} \tag{13}$$

3.2 Stabilité

Parce que les pôles de $H(z)$ sont situés sur le cercle unité, la stabilité du filtre n'est obtenue que si les zéros du numérateur compensent exactement les pôles. Ainsi, toute erreur (approximation) de calcul de $N(z)$ conduit à une instabilité. Ceci interdit l'usage des nombres flottants parce que la somme de deux flottants très différents engendre une erreur sur le résultat. Les calculs doivent donc être impérativement exécutés en entier ! La normalisation (multiplication par C'_p) doit être effectuée en fin de calcul et peut être réalisée en flottant puisque ce calcul est situé en aval de la partie récursive.

Travailler obligatoirement en entier n'est pas restrictif. C'est au contraire un avantage pour l'utilisation des DSP bon marché et pour les réalisations câblés (FPGA, ASIC).

Finalement, pour éviter toute saturation des intégrateurs (partie récursive de $H(z)$), le calcul de $N(z)$ doit être effectué en amont des intégrations.

3.3 Algorithme de calcul

Nous définissons :

$$a_0 = w,\ a_1 = w+3, a_2 = 2w+3 \tag{14}$$

La valeur courante de l'entrée est x_i et on s'intéresse au calcul de y_i. Le calcul débute en $i = 0$.

$$\begin{aligned} n &= a_0(x_{i+w+2} - x_{i-w-2}) \\ &\quad +a_1(x_{i-w-1} - x_{i+w+1}) \\ &\quad +a_2(x_{i+1} - x_{i-1}) \\ t_1 &= t_1 + n \\ t_2 &= t_2 + t_1 \\ t_3 &= t_3 + t_2 \\ t_4 &= t_4 + t_3 \\ t_5 &= t_5 + t_4 \\ y_i &= C'_p(t_6 + t_5) \\ t_6 &= t_5 \end{aligned} \tag{15}$$

Tous les t_j ainsi que n sont de simples variables temporaires.

3.4 Initialisation de l'algorithme

En traitement d'image, parce que le nombre de données sur chaque ligne (chaque colonne) est fini, il est nécessaire de faire attention à l'initialisation près des bords. A cause du traitement en limite de stabilité, une mauvaise initialisation conduit ici à une divergence du filtre (mauvaises conditions initiales sur les dérivées ...).

Près du bord gauche, certaines données ne sont pas disponibles (indices négatifs). Pour éviter une réponse transitoire près de x_0, les données indisponibles (x_{i-w-2}, x_{i-w-1} x_{i-1}) doivent être remplacées dans (15) par la valeur de x_0 jusqu'à ce qu'elles deviennent disponibles. $t_{1..4}$ et t_6 doivent être initialement nulles et t_5 initialisé à $2x_0/C'_p$ qui est toujours une valeur entière quand x_0 est entier.

Du côté du bord droit, si N est l'indice du dernier pixel, les données ($x_{i+w+2}, x_{i+w+1}, x_{i+1}$) doivent être remplacées par x_N quand elles deviennent indisponibles.

3.5 Performances

L'algorithme PAOG utilise seulement 11 additions et 4 multiplications. Un autre avantage est la réduction de la bande passante : seulement 5 accès mémoires sont nécessaires par pixel calculé. Pour comparer les performances, nous avons implanté le filtre gaussien (convolution classique) et le filtre POAG pour une même largeur de fenêtre w. Les programmes ont été écrits en C et exécutés sur PC. Les nombres ont été codés en entier. Le temps de calcul du filtre POAG est bien sûr indépendant de w et donc de σ. le tableau 1 montre le gain en rapidité du filtre POAG par rapport au filtre gaussien réalisé par convolution.

TAB. 1 – Gain en rapidité

w	2	6	10	14	18	20
gain	1.77	4.25	6.71	9.17	11.6	12.9

Le gain croit bien sûr linéairement avec w ou σ. La supériorité liée à la structure récursive du filtre POAG est évidente.

4 Implantation matérielle

4.1 Architecture

La figure 3 montre les détails principaux de l'implantation matérielle. Les données retardées sont obtenues par l'intermédiaire de registres et de deux mémoires A et B de taille w utilisées en mode FIFO. Les opérateurs les plus critiques en temps de calcul sont les multiplieurs. Parce qu'ils sont utilisés uniquement dans la partie non récursive de l'architecture, ils peuvent être aisément pipelinés pour augmenter le débit. La partie récursive n'utilise que des additionneurs. Cette partie peut aussi être accélérée en mettant en œuvre des additionneurs bit-pipelinés ou à sauvegarde de retenue. On remarquera figure 3 que les sorties intermédiaires $y^{(i)}$ correspondent à un coefficient d'amplification près à la ième dérivée de la sortie !

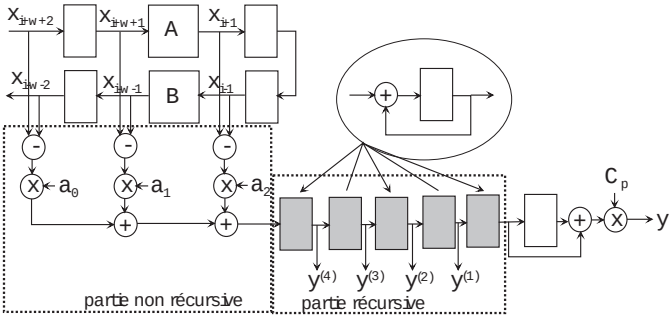

FIG. 3 – Implantation matérielle. Les registres de pipeline ne sont pas précisés sur cette figure.

4.2 Dimensionnement

La présence des intégrateurs dans la boucle récursive complique le dimensionnement des opérateurs et des chemins de données. Or, pour assurer la stabilité, il est nécessaire d'éviter tout phénomène de saturation. Pour la partie non récursive, on suppose les données d'entrées codées sur M bits en non signé. On constate que les multiplieurs sont de faibles dimensions. En effet M est de l'ordre de 8 bits en traitement d'image et le plus fort coefficient vaut $2w+3$. Un codage de ce coefficient sur 8 bits conduit à $w = 126$, soit une réponse impulsionnelle étendue sur 253 pixels ! le format en sortie de la partie non récursive est entier signé et codé sur :

$$M + 3 + \log_2(w + \frac{3}{2}) \quad bits \tag{16}$$

Il est impossible de dimensionner un intégrateur en progressant de l'amont vers l'aval. On procède donc en sens inverse. Puisque le gain global est unitaire, le résulat y de la figure 3 a une partie entière codée sur M bits. Le format en sortie du dernier additionneur est non signé de valeur :

$$M - \log_2 C'_p \tag{17}$$

Compte tenu du gain de 2 du dernier étage d'addition, la sortie du dernier intégrateur utilise un bit de moins que (17). Un intégrateur d'entrée u et de sortie v produit :

$$v_n = u_n + v_{n-1} \tag{18}$$

En inversant cette relation, on obtient :

$$u_n = v_n - v_{n-1} \tag{19}$$

L'entrée de l'intégrateur nécessite un bit supplémentaire par rapport à sa sortie (pire cas). On peut alors établir la valeur d'un majorant pour le dimensionnement qui correspond au stockage en registre de $y^{(4)}$ (figure 3) en entier signé :

$$M + 3 - \log_2 C'_p \simeq M - 1 + 5\log_2(w+3) \tag{20}$$

Les autres dimensionnements découlent aisément de cette valeur.

Pour une implantation DSP avec un codage entier sur 32 bits, et un format d'entrée de 8 bits, la formule (20) conduit à $w_{\max} = 29$.

5 Conclusion

Nous avons montré que le filtre POAG est une bonne approximation du filtre gaussien. Par construction, c'est aussi une excellente approximation du filtre de Canny. Il a une réponse impulsionnelle finie mais peut avantageusement être implanté sous forme récursive. Généralement, l'utilisation des filtres récursifs en image requiert quatre balayages de l'image. Le choix d'une approximation polynomiale de la réponse impulsionnelle conduit à un cas limite où les pôles des parties causale et anticausale coïncident. Il en résulte une réduction d'un facteur deux du nombre des balayages et la suppression de la mémoire d'image pour les applications temps réel. Cette structure est aussi particulièrement bien adapté à un filtrage gaussien 1D (ou à ses dérivées) pour des signaux temps réel à échantillonnage temporel. On pense généralement que la représentation des nombres en flottant donne des résultats plus précis que la représentation entière. Ce filtre exhibe un contre exemple original : la stabilité n'est obtenue qu'en travaillant en représentation entière. Toute la simplicité de la réalisation découle de l'approximation polynomiale. Ce type d'approximation pourrait ainsi conduire à des implantations performantes d'autres types de filtres et notamment des ondelettes.
Cet exemple montre une fois de plus que l'Adéquation Algorithme Architecture profite d'une réflexion approfondie sur les algorithmes. Quantifier la qualité de l'approximation d'un algorithme à l'aide de critères "image" conduit à une architecture qualitativement très différente des architectures généralement utilisées.

Références

[1] J.F. Canny, "Finding edges and lines in images," Tech. Rep., Tech. Rep. AI-TR-720, MIT Artificial Intell. lab., 1983.

[2] J. Canny, "A computational approch to edge detection," *IEEE Transactions on Pattern Analysis And Machine Intelligence*, vol. PAMI-8, pp. 679 – 714, 1986.

[3] R. Deriche, "Fast algorithm for low level vision," *IEEE Pattern Analysis and Machine Intelligence*, vol. 12, no. 1, january 1990.

[4] Didier Demigny and Tawfik Kamleh, "A discrete expression of canny's criteria for step edge detection performances evaluation," *IEEE Pattern Analysis and Machine Intelligence*, vol. 19, no. 11, pp. 1199–1211, november 1997.