Extensions matérielles pour processeurs embarqués de traitement d'images.

Mathieu Thevenin, Michel Paindavoine, Laurent Letellier, Barthélémy Heyrman

RenPar'18 / SympA'2008 / CFSE'6
Fribourg du 11 au 13 Fevrier 2008

Extensions matérielles pour processeurs embarqués de traitement d'images.

Mathieu Thevenin[CEA], Michel Paindavoine[Le2I], Laurent Letellier[CEA], Barthélémy Heyrman[Le2I]

CEA, LIST Laboratoire Calculs Embarqués
Centre de Saclay ; Bt 528 PC 94 ; F-91191 Gif Sur Yvette

Le2I UMR CNRS 5158
9 Av. Alain Savary ; 21000 Dijon

{Mathieu.Thevenin ; Laurent.Letellier}@CEA.fr
{Michel.Paindavoine ; Barthelemy.Heyrman}@U-Bourgogne.fr

Résumé
Le marché des imageurs embarqués est entré dans une ère nouvelle avec l'avènement des téléphones portables munis d'appareils photographiques et de caméras. Il est attendu qu'à l'horizon 2009, leur nombre dépassera celui de l'ensemble des appareils photos vendus depuis l'invention de la photographie, et ce qu'ils soient numériques ou non. Le marché des imageurs électroniques embarqués est donc un secteur porteur, notamment au travers de la téléphonie et de la visiophonie mobile. Les applications ne sont plus limitées à la simple photographie ou la transmission de vidéo ; les lecteurs de codes matrices, la reconnaissance de visages, la biométrie, ou la vision 3D sont des exemples parmi les très nombreuses applications émergentes. L'implémentation de ces applications au sein de dispositifs mobiles requiert une grande flexibilité des composants que les IP dédiées largement utilisées jusqu'alors ne permettent pas. C'est pourquoi des solutions basées sur de processeurs programmables s'avèrent indispensables. Nous proposons dans ce papier des extensions destinées à améliorer les performances des processeurs dédiés au traitement d'image, nous démontrons que ces extensions apportent des améliorations de 60% sur l'ensemble de la chaîne d'acquisition et d'amélioration d'images utilisées derrière le capteur vidéo, ce qui indique le potentiel de ce type d'unité de calcul pour le support des applications à venir.

Mots-clés : processeur, embarqué, image, amélioration, CMOS

---

1. Introduction

Le marché des imageurs embarqués est en forte augmentation, notamment grâce au marché de la téléphonie mobile : en 2009 le nombre de téléphones mobiles munis d'appareils photographiques aura dépassé le nombre des appareils photographiques argentiques et numériques vendus depuis l'invention de la photographie. Les consommateurs sont de plus en plus exigeants tant en terme de qualité d'images, de réactivité du système que d'autonomie. Il est donc crucial pour le marché de concevoir des unités de traitements d'images répondant à ces contraintes tout en gardant un niveau de flexibilité élevé afin de permettre le déploiement d'applications de nouvelle génération. Nous présentons dans ce papier les algorithmes constitutifs de la chaîne d'acquisition d'images en aval d'un capteur CMOS. A partir de ces algorithmes, nous proposons une méthodologie destinée à spécialiser le jeu d'instructions d'un cœur d'un processeur. Nous présentons l'outil mis au point destiné à la quantification de l'impact d'extensions à un processeur de traitement d'image et les résultats obtenus démontrant leur utilité au sein de deux exemples de chaîne de traitement vidéo et d'images à haute résolution destinées aux systèmes nomades. Enfin les résultats obtenus sont situées par rapport aux processeurs du marché de l'embarqué.

2. Chaîne d'acquisition d'images

Tout au long ce papier, nous nous intéresserons plus particulièrement aux systèmes à base de capteurs vidéo CMOS très utilisés en téléphonie mobile. Les imageurs de ce type à faible coût de production, sont largement utilisés dans les produits de grande consommation bien qu'ils souffrent d'une faible qualité d'image inhérente à leur conception. La tendance du marché consistant à augmenter la résolution des capteurs en maintenant ou réduisant

leur taille implique une diminution de la surface des photosites préjudiciable à la qualité d'image [1, 2] que de nombreuses sources de bruits viennent dégrader [3]. Les images sont alors corrigées par traitement numérique [4] après acquisition. La littérature académique ainsi que les bases de brevets comptent un grand nombre de techniques dédiées à cette fin [5]. Ce papier s'appuie sur la chaîne d'acquisition type présentée en figure 1. Cet exemple de chaîne d'acquisition peut être décliné selon de nombreuses variantes pour lesquelles des traitements peuvent être présents ou non. Par exemple le dévignettage[1] est une étape rarement implémentée puisque nécessitant la caractérisation des systèmes optiques.

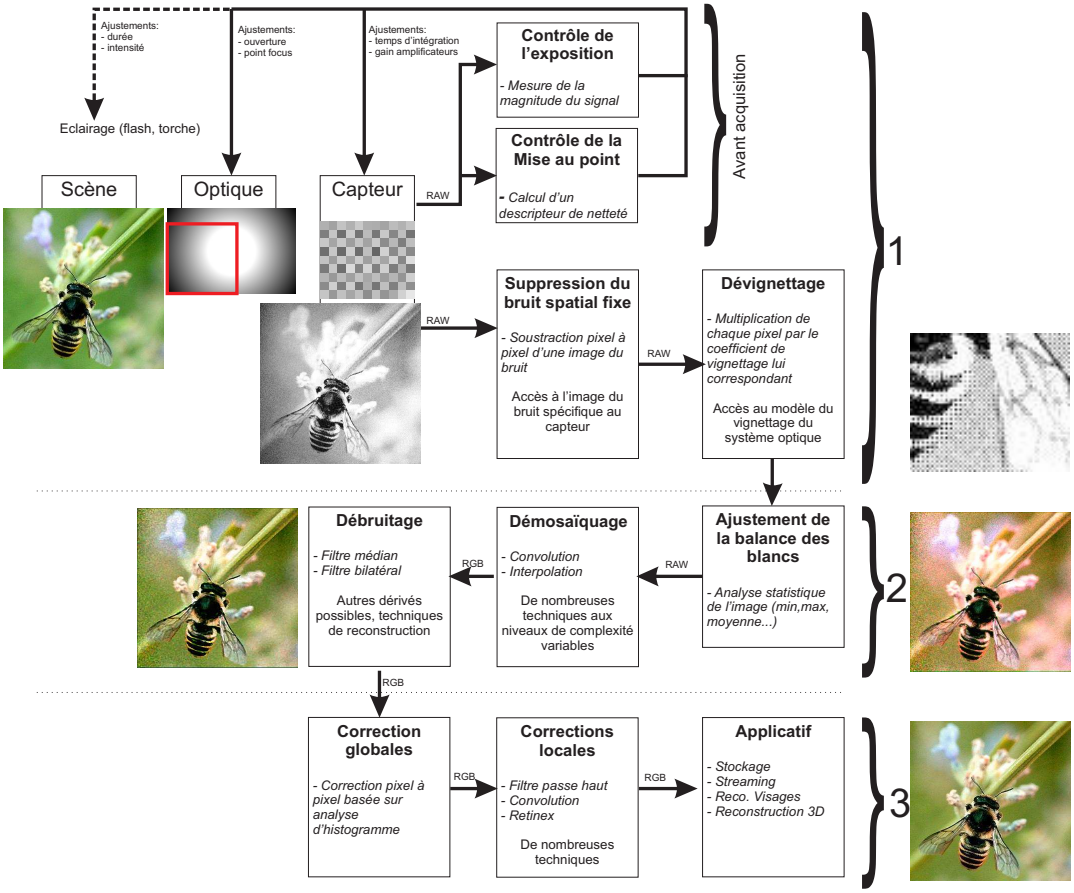

FIG. 1: Exemple d'une chaîne d'acquisition d'images numériques. 1. Traitements avant acquisition et premiers prétraitements. 2. Reconstruction de l'image couleur. 3. Amélioration finale de l'image.

2.1. Chaîne d'acquisition

La chaîne d'acquisition présentée en figure 1 est composée de trois parties. La première partie de la figure 1 a pour objet l'analyse des paramètres de prise de vue jusqu'à l'acquisition de l'image. Il s'agit de la mesure d'exposition, le contrôle de la mise au point suivis d'une première phase de correction de l'image issue du capteur, à savoir la réduction du bruit spatial fixe inhérent à la technologie CMOS, la correction des pixels défectueux, et la suppression du vignettage.

La seconde partie de la figure 1 consiste principalement en la reconstruction de l'information couleur à partir

[1] Le dévignettage consiste en la suppression de l'assombrissement du pourtour d'une image causé par la non uniformité de la réponse du système optique.

de la matrice de Bayer [6]. On trouve un débruitage, une estimation des coefficients de la balance des blancs et finalement le démosaïquage qui consiste à interpoler les plans couleurs à partir des valeurs brutes des pixels.
Enfin la dernière partie de la figure 1 a pour objet des améliorations qualitatives de l'image. Il est important d'attirer l'attention du lecteur sur le fait que le débruitage peut être implémenté à différents niveaux de la chaîne d'acquisition.

2.2. Prétraitements

Après une analyse des différentes chaînes d'acquisition, les algorithmes suivants ont été sélectionnés pour leur représentativité des traitements utilisés à l'heure actuelle. Comme on le verra ensuite, ils sont au cœur de nombreux traitements d'amélioration d'image.
– Calcul d'un critère de netteté ;
– mesure d'exposition ;
– ajustement de la balance des blancs ;
– démosaïquage ;
– débruitage par filtre médian ;
– corrections locales par filtre bilatéral ;
– filtrage par convolution.

2.2.1. Critère de netteté

Ce critère se veut représentatif [7] des algorithmes de mise au point passive utilisés dans les systèmes de prise de vue [8] utilisant uniquement le capteur [9] ce qui exclu les systèmes à mise au point active et à corrélation de phase. Il s'agit généralement de construire la courbe de netteté d'une zone de l'image en fonction de la position de l'optique pour une focale donnée, puis de se placer au maximum de cette courbe. À partir de ce principe, différentes méthodes permettent la recherche de ce maximum en limitant le nombre de prises de vue. Ce critère de netteté est directement utilisable sur les images matricées par le filtre de Bayer. La méthode de calcul du critère consiste en un parcours dans le sens causal et anticausal de la zone dont la netteté est à déterminer, et ce, en ligne comme en colonne. Cette méthode peut être corrélée à des techniques de reconnaissance ou de suivi d'objets afin de préciser la qualité de la mise au point.

2.2.2. Mesure d'exposition

La mesure d'exposition est réalisée par analyse statistique de l'image [10]. Le principe étant d'évaluer la magnitude du signal lumineux afin d'appliquer les corrections nécessaires au capteur ou au système optique afin de produire une image utilisant au mieux la dynamique du capteur. Cette étape pouvant être corrélée au matériel et intégrer des systèmes de reconnaissance de scène. Un histogramme de l'image ou une analyse statistique des valeurs de luminance des pixels de l'image sont des méthodes souvent utilisées.

2.2.3. Balance des blancs

De nombreux travaux concernent les algorithmes d'estimation et d'ajustement de la balance des blancs, certains d'entre eux effectuent une analyse statistique de l'image [11] alors que d'autres plus complexes sont basés sur des techniques d'apprentissage de scène [12] ; certains combinent plusieurs techniques afin d'affiner la mesure de la valeur du point blanc. Enfin, d'autres techniques cherchent à déterminer localement les corrections à appliquer [13] afin de prendre en compte la présence de multiples illuminants. Toutes ces techniques se basent sur la seule image acquise pour déterminer la couleur de l'illuminant. Nous avons choisi d'implémenter la méthode communément appelée << GreyWorld >> .

2.2.4. Démosaïquage

L'étape de démosaïquage est certainement l'étape la plus importante [14] de la chaîne d'acquisition d'images puisqu'elle consiste en l'interpolation des valeurs des pixels pour chaque composante de couleur. La littérature propose de très nombreuses méthodes permettant de reconstruire l'image ; de la simple copie du plus proche voisin connu, à des méthodes fréquentielles [15] en passant par des méthodes locales et spatio-temporelles [16]. La tendance actuelle est de combiner différents traitements lors de l'étape de démosaïquage, comme la réduction de bruit ou des corrections locales adaptatives bio-inspirées [17]. En ce qui concerne les méthodes implémentées pour cette étude, nous avons choisi les deux techniques les plus répandues : l'interpolation bilinéaire et la méthode de constance de teinte [18].

2.2.5. Réduction du bruit : filtre médian

Le filtre médian est largement utilisé pour la réduction du bruit, et de nombreux filtres en dérivent [5, 19]. C'est pourquoi son étude s'avère indispensable. De par sa nature, le filtre médian peut être implémenté selon différentes méthodes. Nous avons choisi une méthode basée sur le tri des données permettant ensuite de sélectionner la valeur médiane ainsi qu'une méthode basée sur la construction d'un histogramme cumulé de la zone à filtrer.

2.2.6. Corrections locales : filtre bilatéral

Le filtre bilatéral [20] a été introduit en 1998, fortement calculatoire il est depuis à la base d'un grand nombre d'algorithmes récents, que ce soit du débruitage à la normalisation Retinex [21], le rendu d'images à large dynamique [22] en passant par le démosaïquage [23] de la matrice de Bayer. Nous avons implémenté la version décrite dans la publication originale en l'adaptant à l'absence d'unité de calcul flottant.

2.2.7. Filtrage par convolution

La convolution est à la base d'un grand nombre de filtres en traitement du signal. Aussi avons-nous implémenté les cas suivants : convolution 3×3, 5×5 et enfin une convolution dont la taille du noyau est paramétrable. Chacune d'entre elle est optimisée afin d'utiliser au mieux le jeu d'instructions.

3. Spécialisation du cœur de calcul

Nous avons décrit la chaîne d'acquisition d'image et les algorithmes associés, puis la méthodologie pour quantifier l'impact liée à la spécialisation d'un cœur de processeur. Pour qu'un dimensionnement précis des unités de calcul de l'architecture de traitement d'image soit réalisable, il est nécessaire de comparer chaque algorithme exécuté à l'aide d'un cœur à jeu d'instructions dit standard avec leur équivalent optimisé pour chaque jeu d'instructions spécialisé que nous avons proposé.

3.1. Techniques de spécialisations

De nombreuses techniques sont destinées à la spécialisation et l'optimisation du jeu d'instructions de cœurs de calculs. Qu'il s'agisse d'approches basées sur une analyse des algorithmes [24] ou sur une analyse du parallèlisme au niveau instruction [25] , ces méthodes permettent de déterminer les sections de programme à accélérer en introduisant des extensions au jeu d'instructions. Des solutions commerciales et des flots de conceptions permettent de générer des cœurs de calculs [26] application-dédiées tel que le traitement vidéo [27]. Ces techniques recherchent les motifs les plus représentatifs du code de calcul et proposent des instructions destinées à les accélérer.

3.2. Le simulateur

Pour notre étude nous avons aussi le besoin d'évaluer l'impact d'unités de gestion des adresses combinées à leurs instructions spécifiques. Nous avons pour celà conçu un simulateur modulaire et comportemental de jeu d'instructions de processeur. Sa conception modulaire permet à l'utilisateur de définir son propre jeu d'instructions et ainsi de décrire les algorithmes à étudier. Dans le cas de ce papier nous avons décrit notre propre jeu d'instructions inspiré de processeurs du marché. Le simulateur permet alors l'exécution rapide réelle du programme, tout en effectuant le décompte des instructions et des accès au banc de registres. L'utilisateur peut introduire du code `C/C++` en ligne au sein de son programme comme dans l'exemple 3 afin de simplifier certaines tâches telles que l'ouverture d'images ou l'écriture de résultats, l'utilisation de librairies externes, ou encore d'extraire des informations supplémentaires sur l'exécution du code. Un code source `C/C++` compilable avec `GCC` puis exécutable sur station de travail est produit à partir du code assembleur introduit par l'utilisateur, ce qui permet de vérifier la validité de la description des algorithmes utilisant le jeu d'instructions spécifié. La table 1 montre le code `C/C++` généré pour l'opérateur d'addition d'une constante à un registre.

3.2.1. Modularité

Le simulateur a été conçu pour faciliter la modification du jeu d'instructions par simple modification d'un fichier de configuration avant appel. Il est possible non seulement d'intervenir sur la taille des registres, leur nombre, mais aussi d'ajouter des instructions, le comportement qui leur est associé, des registres spéciaux ou des espaces mémoires spécifiques.

3.2.2. Jeu d'instructions de base

Le jeu d'instructions de base comprend un ensemble d'opérateurs standards permettant de décrire l'ensemble des algorithmes de la chaîne de traitements. Les résultats ainsi obtenus nous serviront de référence.

– Branchements conditionnels et inconditionnels (`JMP`, `JZ`, `JNZ`...) ;
– accès de registre/constante à registre (`LD`) ;
– lecture et écriture en mémoire externe (`LDA`, `STR`) ;
– opérations au niveau bit (`SHIFTR`, `SHIFTL`, `AND`, `OR` ...) ;
– opérations arithmétique (`ADD`, `CMP`, `SUB`, `MUL`, `INC` ...)

Instruction	Code C/C++ généré	Action correspondante
`ADD R1 0x10`	`nbr.R1+= 1;`	Incrément des compteurs d'accès aux registres
	`instr.ADD+=1;`	Incrément des compteurs d'appels des opérateurs
	`R1 += 0x10;`	Exécution de l'opération
	`Z=R1==0?1 :0;`	Mise à jour du registre spécial `Z` **Z**éro
	`S=R1>=0?1 :0;`	Mise à jour du registre spécial `S` de **S**igne

TAB. 1: Extrait du code compilable généré par le simulateur de jeu d'instructions

3.3. Exemple d'utilisation

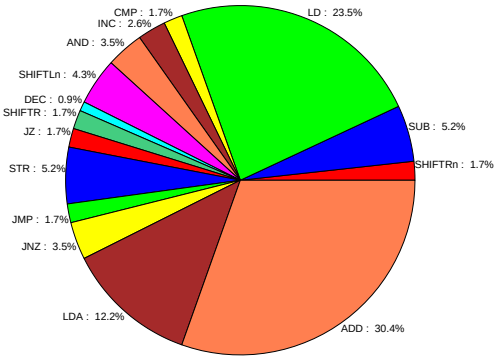

FIG. 2: Taux d'utilisation des différents opérateurs pour le démosaïquage par interpolation bilinéaire.

L'exemple de programme de la figure 3 présente un démosaïquage par interpolation bilinéaire. Cette application utilise deux fonctions externes `LoadPGMFile` et `WritePGMFile` pour l'accès aux données. Pour des raisons de lisibilité seul l'initialisation, l'enregistrement des images et un extrait du calcul de la composante verte sont présentés sur le programme de la figure 3. L'exécution de ce programme produit les résultats visibles en figure 2 et 4 Ces résultats peuvent alors êtres utilisés d'une part pour le dimensionnement d'architectures, d'autre part pour l'exploration architecturale puisqu'ils quantifient précisément l'impact de l'introduction de nouvelles instructions. On voit sur la figure 2, correspondant à la répartition des appels du programme, que 40% des appels sont des instructions de chargement de données et que les additions et soustractions concernent plus du tiers des opérations réalisées. On peut en déduire un jeu d'instructions compact adapté à ce type de traitements, ainsi qu'adapter la taille du banc de registre, ici les 32 registres disponibles sont utilisés.

4. Méthodologie

À partir du simulateur présenté nous avons implémenté un ensemble d'algorithmes représentatifs de la chaîne d'amélioration d'images présentée en section 2 dans toute sa diversité. Ces algorithmes ont été exécutés sur des images à différentes résolutions et de type différent puisque nous différencions les traitements réalisés sur les images brutes[2] des traitements réalisés sur des images couleurs. Nous présentons les extensions au coeur de processeur introduites, ainsi que la méthodologie suivie pour l'obtention des résultats. Dans un premier temps les algorithmes sont décrits et simulés en utilisant le jeu d'instructions standard, puis optimisés pour chaque jeu d'instructions correspondant à ces nouvelles extensions. Les résultats de simulation sont alors comparés et permettent d'évaluer quantitativement l'impact de ces extensions.

4.1. Algorithmes implémentés

Les algorithmes implémentés vont de la mesure de l'exposition de l'image, à des traitements plus complexes comme le filtre bilatéral en passant par la convolution. Pour chacun d'entre eux le programme a été décrit en utilisant le jeu d'instructions standard vu en sous-section 3.2.2. Cette implémentation sert de référence pour les comparaisons futures. À ce jeu d'instructions dit standard ont été ajoutées de nouvelles instructions. Nous cherchons à évaluer leur impact sur les performances des algorithmes.

[2] Les images brutes correspondent aux données enregistrées directement à partir du capteur.

Classes	Instructions	Appels	Taux	Action correspondante
Accès aux données	LD	3 539 463	23.5%	Copie de registre à registre
	LDA	1 835 016	12.2%	Lecture en mémoire externe
	STR	786 432	5.2%	Stockage d'une donnée en mémoire externe
Arithmétique	CMP	262 656	1.7%	Comparaison sans stockage du résultat
	SUB	786 432	5.2%	Soustraction registre à registre ou constante
	ADD	4 587 525	30.4%	Addition à un registre
	INC	393 728	2.6%	Incrément d'un registre
	DEC	131 072	0.9%	Decrément d'un registre
Logique	SHIFTLn	655 360	4.3%	Décalage de *n* bits sur la gauche
	SHIFTRn	262 144	1.7%	Décalage de *n* bits sur la gauche
	SHIFTR	262 144	1.7%	Décalage à droite
	AND	524 288	3.5%	ET logique
Sauts	JMP	262 145	1.7%	Saut inconditionnel
	JNZ	524 800	3.5%	Saut conditionnel sur le registre Z non nul
	JZ	262 144	1.7%	Saut conditionnel sur le registre Z nul
Total		**15 075 349**	**100%**	

TAB. 2: Résultats produits par l'exécution du programme présenté en figure 3 avec pour image d'entrée $\ll$Lena$\gg 512 \times 512$ en utilisant le jeu d'instructions dit standard – sans extensions – spécifié en 3.2.2.

```
 # Extrait du programme assembleur de démosaïquage
 # par interpolation bilinéaire des composantes RGB à partir d'une image brute (RAW)

;(...)
; Calcul    de  la  composante      verte
; G(i,j)    =  (imCFA(i−1,j)  +  imCFA(i+1,j)  +  imCFA(i,j−1)  +  imCFA(i,j+1))/4
; i−1,j   => offset    − W  (∗  8)
LD          R8          R1          ; R8  =W
SHIFTLn     R8          0x3         ; W∗8
LD          R7          R3          ; R7=imCFA
ADD         R7          R12         ; R7=imCFA+offset
SUB         R7          R8          ; R7=imCFA     +  offset    − W∗8
LDA         R14         R7          ; R14=imCFA(i−1-1,j)
; i+1,j    => offset    +  W(∗8)
LD          R7          R3          ; R7=imCFA
ADD         R7          R12         ; R7=imCFA+offset
ADD         R7          R8          ; R7=imCFA     +  offset    +  W∗8
LDA         R13         R7          ; R14=imCFA(i−1,j)
ADD         R14         R13         ; R14=imCFA(i−1,j)+i(i+1,j)
;(...)
; Stockage     de  la  valeur
LD          R7          R5          ;
ADD         R7          R12         ;G(i,j)=R14
STR         R7          R14         ;
;(...)

 # // Code C en ligne
 # WritePPMFile(argv[2],R,G,B,W,H,Depth); // Enregistre une image dans un fichier .PPM
```

FIG. 3: Programme d'exemple : démosaïquage par interpolation bilinéaire ; extrait du calcul de la composante verte ; pour des raisons de clarté seules quelques opérations sont présentées ici.

(a) Image brute (RAW) (b) Détail de l'image brute (c) Image RGB après démosaïquage

FIG. 4: Résultats obtenus par simulation (a)Image brute originale ; (b)Détail de l'image brute ; (c)Image RGB reconstruite par interpolation bilinéaire.

4.2. Instructions additionnelles

Des extensions destinées à accélérer le traitement d'image sont ajoutées au simulateur décrit précédemment. Elles prennent la forme d'instructions additionnelles au jeu d'instructions standard. Le choix de ces extensions a fait l'objet d'une étude préliminaire indiquant qu'une large part des instructions d'un cœur de boucle effectuent le calcul des adresses mémoire correspondant aux positions x,y linéarisées des pixels. Cette même étude montre qu'une autre part des instructions d'une boucle consiste à tester la condition de fin ainsi que l'incrément des compteurs. Par ailleurs nous cherchons à évaluer l'impact d'un opérateur de type Multiplication-ACcumulation (MAC). Il est à noter que ces extensions introduisent des registres spéciaux permettant d'alléger l'utilisation du banc de registres standards.

4.2.1. Multiplication-ACcumulation (MAC)

Il s'agit de l'instruction de multiplication accumulation proposée au sein de la plupart des processeurs de traitement du signal. En effet les opérations de filtrage utilisent massivement la succession d'opérations de multiplication et addition, comme le montre l'équation suivante décrivant un filtre standard d'ordre M :

$$y[n] = \sum_{k=0}^{N} (b_k \times x[n-k]) \left(- \sum_{k=1}^{M} a_k \times y[n-k] \right)$$

$$y[n] = b_0 \times x[n] + b_1 \times x[n-1] + ... + b_N \times x_{n-N} - a_1 \times y[n-1] + a_2 \times x[n-2] + ... + a_M \times x_{n-M}$$

4.2.2. Gestionnaires de boucles (GADDR)

Dans le cadre du traitement d'images, il apparaît que les traitements sont construits autour d'une ou plusieurs boucles comme par exemple le parcours de l'image pixel par pixel ou le parcours d'un voisinage. Nous appellerons les gestionnaires de boucles << GLOOP >> dans les tableaux de résultats.

4.2.3. Gestionnaires d'adresses (GLOOP)

Les images sont des matrices 2D, stockée linéairement en mémoire. Le calcul des adresses correspondantes aux coordonnées x,y se fait par :

$$adresse = (y \times Largeur + x) \times SizeOfWord$$

Où *largeur* est la largeur de l'image en pixels, et *SizeOfWord* est la taille en octet d'un mot nécessaire au stockage des données de l'image, soit 1 *octet* dans le cas d'images sur 256 niveaux. Nous appellerons les gestionnaires d'adresses << GADDR >> dans les tableaux de résultats.

4.3. Déclinaison des algorithmes

Chaque algorithme a été réécrit et optimisé pour chacune des extensions décrites précédemment afin de le comparer à sa version standard. Il s'avère utile de quantifier les différentes associations d'extensions possibles. Ainsi, nous

avons optimisé chaque algorithme manuellement pour chacune des associations possibles, ce qui nous donne les implémentations suivantes :
– Jeu d'instructions standard ;
– jeu d'instructions standard associé à un gestionnaire de boucles ;
– jeu d'instructions standard associé à un gestionnaire d'adresses ;
– jeu d'instructions standard associé à un gestionnaire de boucles ainsi qu'un gestionnaire d'adresses ;
– jeu d'instructions standard associé à un opérateur MAC[3] ;
– jeu d'instructions standard associé à un gestionnaire de boucles ainsi qu'un opérateur MAC ;
– jeu d'instructions standard associé à un gestionnaire d'adresses ainsi qu'un opérateur MAC ;
– jeu d'instructions standard associé à un gestionnaire de boucles, un gestionnaire d'adresses ainsi qu'un opérateur MAC.

5. Résultats

Les résultats obtenus présentés en table 3 expriment l'accélération en pourcentage algorithme par agorithme pour chacune des extensions proposées et leurs associations. L'amélioration des performances globales que leur introduction nous permettent d'atteindre est ensuite présentée et commentée pour deux applications destinées à la téléphonie mobile. La première présente une application de capture vidéo VGA et en Haute Définition (HD), la seconde est une application de photographie dont la résolution des images est de 6 MégaPixels (Mpx).

5.1. Vidéo

La chaîne d'acquisition vidéo dont les résultats pour chaque traitement employé sont présentés en table 4. Les résolutions VGA et HD 1080×720 à 25 images par seconde sont considérées. La chaîne d'acquisition consiste en un module de contrôle d'exposition permettant d'ajuster les paramètres du capteur et de l'optique afin d'assurer une utilisation optimale de la dynamique du capteur. Le bruit spatial fixe inhérent aux capteurs CMOS est supprimé, suivi d'une correction de la balance des blancs. Une mesure de l'illuminant toutes les 10 images a été retenue.
On peut voir un gain de l'ordre de 60% sur l'ensemble de la chaine de traitement. Les résultats sont exprimés en million d'instructions pour une seconde de traitement vidéo qui doit répondre à la contrainte temps réel des 25 images par seconde. Deux sous résultats sont présentés, le premier utilisant la technique de démosaïquage de la constance de teinte, le second utilisant une technique de démosaïquage bilinéaire ; on constate alors l'impact de ce traitement sur les ressources de calcul nécessaires au traitement des images.

5.2. Photographie

L'ensemble des traitements en photographie n'ont pas à répondre à des contraintes temporelles hormis le contrôle de la mise au point et la mesure de l'exposition ; ces deux derniers exemples se déroulent avant capture et traitement de l'image. Dans cet exemple cette phase critique est évaluée séparément et les résultats sont présentés en tableau 5. La chaîne d'acquisition considérée dans cet exemple est celle d'un téléphone mobile, construite autour de la chaîne de traitement représenté en figure 1.

5.2.1. Mesure d'exposition et contrôle de la mise au point

Les algorithmes usuels de mesure d'exposition et de contrôle de la mise au point sont nettement accélérés par nos extensions comme le montre le tableau 5 puisqu'un facteur de 1.70 est atteint. La plupart de ces algorithmes effectuent des opérations proches de la convolution, qui, comme on l'a vu en table 3 est notablement accélérée. Leurs faibles besoins en ressources de calcul $-$ 2.9 million d'instructions pour un système complexe et moins d'un million pour un système plus rudimentaire $-$ ouvre la voie à l'utilisation d'algorithmes plus complexes, notamment pour le contrôle de la mise au point pour lequelle le suivi d'objets peut être envisagé.

5.2.2. Chaîne d'amélioration de l'image

La chaîne de prétraitement, dont les résultats sont présentés table 6, consiste en l'amélioration de l'image capturée. Les algorithmes utilisés sont plus complexes que dans le cas de la capture vidéo, et opèrent à une résolution nettement plus importante. Grâce à nos extensions il est possible de réduire de 54% les ressources nécessaires au traitement.

[3] Tous les algorithmes ne permettant pas l'utilisation de l'opération Multiplication-ACcumulation.

```
# Extrait du programme de démosaïquage avec gestionnaire d'adresses et de boucles
# par interpolation bilinéaire des composantes RGB à partir d'une image brute (RAW)

;(...)
; Calcul    de  la  composante      verte
; G(i,j)    =  (imCFA(i-1,j)   +  imCFA(i+1,j)   +  imCFA(i,j-1)   +  imCFA(i,j+1))/4
; i-1,j
LD          R8          LC0.cpt                         ; R8  =  i
DEC         R8                                          ; R8  =  i-1
IMREAD      IM0         R8          LC1.cpt             ; IM=imCFA(i-1,j)
LD          R14         IM                              ; R14   =  IM
; i+1,j
ADD         R8          2                               ; R8  =  i+1
IMREAD      IM0         R8          LC1.cpt             ; imCFA(i+1,j)
ADD         R14         IM                              ; R14   =  R14   +  IM
;(...)
; Stockage     de  la  valeur
LD          R7          R5                              ;
ADD         R7          R12                             ; G(i,j)=R14
STR         R7          R14                             ;
;(...)
```

FIG. 5: Programme d'exemple : démosaïquage par interpolation bilinéaire ; extrait du calcul de la composante verte en utilisant les instructions gestionnaire de boucles (en bleu) et d'adresses (en rouge).

Algorithmes	MAC	GLOOP	GLOOP MAC	GADDR	GADDR MAC	GADDR GLOOP	GADDR GLOOP MAC
Critère de netteté	NA	7.5%	NA	20%	NA	27.6%	NA
Mesure d'exposition	NA	9.1%	NA	44.5%	NA	54.2%	NA
Suppression du bruit spatial fixe	NA	9.1%	NA	27.1%	NA	36.4%	NA
Suppression de vignettage	NA	9.1%	NA	9.1%	NA	16.7%	NA
Balance des blancs	NA	5.7%	NA	17.4%	NA	23.7%	NA
Démosaïquage bilinéaire	NA	2%	NA	31.1%	NA	32.9%	NA
Démosaïquage constance de teinte	NA	2%	NA	35.5%	NA	42.9%	NA
Filtre médian par tri	NA	9.1%	NA	1%	NA	10%	NA
Filtre médian par histogramme	NA	13%	NA	<1%	NA	13.1%	NA
Filtre bilatéral	3%	3.9%	6.6%	16%	19.4%	19.4%	22.5%
Convolution 5×5	5%	14.6%	20%	27.1%	32.9%	33.8%	39.4%

TAB. 3: Réduction du nombre d'instructions appelées par rapport à la version décrite en utilisant le jeu d'instruction standard : $1 - NbInstr_{Opt}/NbInst_{Std} \times 100$; MAC : optimisé pour utilisation de l'opérateur MAC ; GLOOP : optimisé pour le gestionnaire de boucles ; GADDR : optimisé pour le gestionnaire d'adresses et combinaison des différents opérateurs ; NA lorsque l'utilisation de l'opérateur MAC n'apporte pas d'amélioration.

Traitements	Jeu d'instructions standard Vidéo VGA	Vidéo HD	Jeu d'instructions étendu Vidéo VGA	Vidéo HD	Gain
	(*Million d'instructions*)		(*Million d'instructions*)		
Contrôle d'exposition	0.1	1	<0.1	0.5	2.14
Suppression du bruit spatial fixe	80	210	52	136	1.54
Balance des blancs automatique	15	20	12	16	1.29
Démosaïquage *(constance de teinte)*	1 000	2 800	620	1 740	1.60
ou Démosaïquage *(bilinéaire)*	440	1 125	295	750	1.51
Total : *(constance de teinte)*	**1 095**	**3 031**	**685**	**1895**	**1.59**
(bilinéaire)	**535**	**1 356**	**360**	**903**	**1.49**

TAB. 4: Ressources nécessaires pour les traitements réalisés lors de l'acquisition d'une seconde de vidéo HD et VGA ; exprimées en million d'instructions exécutées.

Traitements	Jeu d'instructions standard (*Million d'instructions)*	Jeu d'instructions étendu (*Million d'instructions)*	Gain
Contrôle d'exposition global	**0.8**	**0.3**	2.24
Contrôle d'exposition central pondérée	0.2	0.1	1.53
Mise au point (mesure spot)	0.2	0.2	1.36
Mise au point centrale pondérée	1.0	0.7	1.46
Mise au point 9 points	**2.1**	**1.4**	1.54
Total (pire cas)	**2.9**	**1.7**	**1.70**

TAB. 5: Ressources nécessaires pour la mesure de l'exposition (6 mesures par seconde) et le calcul du critère de netteté (16 mesures par seconde) ; exprimées en million d'instructions exécutées.

Traitements	Jeu d'instructions standard (*Million instructions)*	Jeu d'instructions étendu (*Million d'instructions)*	Gain
Suppression du bruit spatial fixe	123	78	1.57
Balance des blancs	410	310	1.31
Démosaïquage	1 721	1 078	1.59
Réduction du bruit (médian)	1 284	1 137	1.12
Accentuation	6 378	3 825	1.66
Total	**9 916**	**6 428**	**1.54**

TAB. 6: Ressources nécessaires pour le traitement d'une image $6Mpx$, exprimées en million d'instructions exécutées.

5.3. Interprétation des résultats

Nous avons décrit la chaîne d'acquisition et de traitement des images issues de capteur CMOS, puis nous avons décrit et simulé les algorithmes constitutifs en utilisant un jeu d'instructions standard. Nous avons ensuite introduit des optimisations au niveau du cœur de calcul et du jeu d'instructions, puis nous avons comparé les résultats différents résultats des simulations correspondantes. Cette section explique les résultats obtenus et situe ces travaux par rapport aux cœurs de calcul du marché de l'embarqué.

5.3.1. Gestionnaire d'adresses

Les résultats présentés dans ce papier montrent que l'unité apportant un gain notable est celle de gestion des adresses des images : en effet la plupart des traitements accèdent a un voisinage, ce qui nécessite le calcul systématique de toute ou partie des adresses de ces données. Aussi est-il cohérent de proposer une ou plusieurs unités dédiées. On remarque que deux à trois registres généraux sont libérés, puisque ceux dédiés au calcul des adresses sont remplacés par ceux spécifiques au gestionnaire d'adresse. Le nombre utile de ce type d'unités a été estimé à quatre ; ce nombre permet une programmation souple des traitements étudiés ici. En effet la notion d'images peut être étendue à toute autre donnée sous forme de table qui peut alors être adressée grâce à ces unités. L'économie peut atteindre six registres généraux sur certains algorithmes.

5.3.2. Gestionnaire de boucle

La plupart des algorithmes de traitement d'images décrits séquentiellement mettent en jeu une ou plusieurs boucles. Ces boucles sont généralement du type `for` et utilisent des registres généraux pour les tests de conditions et les compteurs. Aussi un gestionnaire de boucle peut être implémenté efficacement à l'aide d'un compteur et d'un comparateur. On note des gains nettement supérieurs à 5% et pouvant atteindre 15% selon le nombre de boucles du programme. Une économie d'au moins un registre générique par boucle est rapportée.

5.3.3. Multiplication-ACcumulation

La convolution, autour de laquelle de nombreux traitements sont construits, utilise intensément l'instruction de MAC. En ne considérant que l'ajout d'une unité MAC, le gain en nombre d'instruction est de quelques pourcents : en effet cette opération est constituée de deux à trois instructions successives au sein d'un cœur de boucle, ces

dernières sont alors remplacées par la seule MAC. Il s'avère que le cœur de boucle est généralement constituée de nombreuses autres opérations, y compris dans certains cas des sauts conditionnels ce qui réduit dramatiquement le gain. C'est pourquoi le gain induit par l'utilisation de l'opérateur MAC est nettement supérieur lorsqu'il est associé aux opérateurs précédents dont l'objet est de diminuer le nombre d'instructions du cœur de boucle par le calcul d'adresse et de gestion de la boucle.

5.3.4. Puissance de calcul équivalente

Nous avons vu table 4 que l'acquisition en temps réel d'une seconde de vidéo VGA nécessite l'exécution de 600 million d'instructions et de 3 milliard d'instructions pour une vidéo HD. De même, près de 10 milliard d'instructions étaient nécessaires au traitement d'une image photographique de résolution $6Mpx$. Nous avons montré que les extensions présentées dans ce papier permettent d'obtenir des facteurs d'accélération de 60% sur l'ensemble de la chaîne d'acquisition, permettant alors de réduire d'autant les ressources de calcul. On parvient alors à un besoins de $360MIPS$ pour l'acquisition d'une vidéo de qualité VGA et $685MIPS$ pour de la vidéo haute définition à partir des résultats de la table 4. Un processeur embarqué actuel tel qu'un ARM 11 est capable de délivrer environ $1MIPS/MHz$. On voit dès lors qu'un tel processeur muni des extensions présentées ici serait en mesure de réaliser le traitement d'acquisition de vidéo en VGA à une fréquence inférieure à $400MHz$. Un seul de ces processeurs ne peut en revanche réaliser l'acquisition de vidéo HD. Seule la parallélisation de l'algorithme de démosaïquage sur plusieurs unités de calcul permet de répondre à cette problématique ; deux unités de ce type se révèlent alors nécessaires. Le traitement d'images photographiques sur ce type de processeur dépasserait les 10 secondes. Il faut donc parallèliser les algorithmes sur plusieurs unités de calcul afin de réduire ce temps de traitement.

6. Conclusions et perspectives

Les systèmes mobiles ayant la capacité d'acquérir et de traiter des images et de la vidéo sont de plus en plus nombreux, que ce soit des téléphones portables aux baladeurs vidéo en passant par les Personal Digital Assistant (PDA), tous sont contraints par leur autonomie et donc sur la capacité à embarquer des unités de calculs flexibles. C'est pourquoi la plupart des traitements réalisés à ce jour sont effectués par des IP dédiées. Ce papier propose des extensions aux processeurs embarqués afin d'améliorer leurs capacités de traitement. Nous avons présenté et détaillé algorithme par algorithme une chaîne d'acquisition d'images en aval de capteurs CMOS, puis nous avons quantifié l'impact lié à l'introduction de nouvelles extensions (gestionnaire de boucle, le gestionnaire d'adresses...). Nous avons ainsi pu montrer, grâce à l'utilisation d'un simulateur conçu pour cette étude, que le gain en terme de nombre d'instructions exécutées pouvait atteindre 60%. L'application de ces résultats aux processeurs embarqués du marché montre que l'utilisation de telles extensions rend possible des traitements temps réel sur des images vidéo de qualité VGA. Ces travaux ouvrent donc la voie à l'utilisation d'unités programmables et flexibles pour le traitement d'image sur systèmes mobiles. La mise en parallèle de telles unités est en revanche incontournable pour le traitement d'images à résolutions élevées. La suite actuelle de ces travaux consiste à développer un coeur de processeur modulaire capable de supporter un parallélisme de type flot de données et dont les ressources de calcul proposées ici ne sont activées que lorsqu'elles sont utilisées.

Bibliographie

1. P. B. Catrysse and B. A. Wandell. Roadmap for CMOS image sensors : Moore meets Planck and Sommerfeld. In N. Sampat, J. M. DiCarlo, and R. J. Motta, editors, *Digital Photography. Edited by Sampat, Nitin ; DiCarlo, Jeffrey M. ; Motta, Ricardo J. Proceedings of the SPIE, Volume 5678, pp. 1-13 (2005).*, volume 5678 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, pages 1–13, February 2005.
2. Ting Chen, Peter Catrysse, Abbas El Gamal, and Brian Wandell. How small should pixel size be ?, 2000.
3. Hewlett-Packard Components Group. Noise sources in cmos image sensors. Technical report, Hewlett-Packard Company, 1998.
4. Xinqiao Liu. *CMOS Image Sensors Dynamic Range and SNR Enhancement Via Statistical Signal Processing*. PhD thesis, Stanford university, Jun 2002.
5. Alan C. Bovik. *Handbook of Image and Video processing*. Academic Press, 2000.
6. Gunturk, B.K. Glotzbach, J. Altunbasak, Y. Schafer, and R.M. R.W. Mersereau. Demosaicking : color filter array interpolation. *Signal Processing Magazine*, 22 Iss. 1 :44–54, January 2005.
7. L. Shih. Autofocus survey : a comparison of algorithms. In *Digital Photography III. Edited by Martin, Russel A. ; DiCarlo, Jeffrey M. ; Sampat, Nitin. Proceedings of the SPIE, Volume 6502, pp. 65020B (2007).*, volume 6502 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, March 2007.
8. Kazuki Konishi. Autofocus control, apparatus and method. *US. Patent 7,053,350*, B2 :, may 2006.

9. Timothy Alderson. Method and system for using an image based autofocus algorithm. *US. Patent 7,187,413*, B2 :, March 2007.
10. Tetsuro Ashida. Image capturing apparatus and automatic exposure control correcting method. *US. Patent 7,129,980*, B1 :, Oct 2006.
11. Raimondo Schettini Francesca Gasparini. Color balancing of digital photos using simple image statistics, 2004.
12. M. Akamatsu Karungaru, S. Fukumi. Neural networks and genetic algorithms for learning the scene illumination in color images. In *Computational Intelligence in Robotics and Automation, 2003. Proceedings. 2003 IEEE International Symposium*, volume 3, pages 1085– 1089, July16-20 2003.
13. A. Gijsenij and Th. Gevers. Color constancy by local averaging and scale selection. In *Proceedings of the Twelfth Annual Conference of the Advanced School for Computing and Imaging*, pages 141–148, June14-16 2006. Deprecated. Use ICIAP 2007 instead.
14. David Alleysson. 30 ans de démosaïckage - 30 years of demosaicing. *Traitement du signal*, 21 issue 6 :561–581, 2004.
15. E. Dubois. Frequency-domain methods for demosaicking of bayer-sampled color images. *IEEE Signal Processing Letters*, 12(12) :847–850, December 2005.
16. K.N. Lukac, R. Plataniotis. Adaptive spatiotemporal video demosaicking using bidirectional multistage spectral filters. *Consumer Electronics, IEEE Transactions*, 52, Iss.2 :651– 654, May 2006.
17. Laurence Meylan, David Alleysson, and Sabine Süsstrunk. A Model of Retinal Local Adaptation for the Tone Mapping of Color Filter Array Images. *Journal of the Optical Society of America A (JOSA A)*, 24(9) :2807–2816, 2007. Supplementary material available at : http ://ivrg.epfl.ch/supplementary_material/LM_JOSA06/index.html.
18. D. R. Cok. Signal processing method and apparatus for producing interpolated chrominance values in a sampled color image signal, 1987.
19. N.C. Gallagher G.R Arce. Median filters : Theory and application. *Advances in Computer Vision and Image Processing*, 1986.
20. Carlo Tomasi and Roberto Manduchi. Bilateral filtering for gray and color images. In *ICCV*, pages 839–846, 1998.
21. Michael Elad. Retinex by two bilateral filters. In Ron Kimmel, Nir A. Sochen, and Joachim Weickert, editors, *Scale-Space*, volume 3459 of *Lecture Notes in Computer Science*, pages 217–229. Springer, 2005.
22. Frédo Durand and Julie Dorsey. Fast bilateral filtering for the display of high-dynamic-range images. In *SIGGRAPH '02 : Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 257–266, New York, NY, USA, 2002. ACM Press.
23. R. Ramanath and W. Snyder. Adaptive demosaicking. *Electronic Imaging*, 12(4) :633–642, 2003.
24. Daniel Benyamin and William H. Mangione-Smith. Function unit specialization through code analysis. In *ICCAD '99 : Proceedings of the 1999 IEEE/ACM international conference on Computer-aided design*, pages 257–260, Piscataway, NJ, USA, 1999. IEEE Press.
25. Kubilay Atasu, Günhan Dündar, and Can Özturan. An integer linear programming approach for identifying instruction-set extensions. In *CODES+ISSS '05 : Proceedings of the 3rd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, pages 172–177. ACM Press, 2005.
26. R. Leupers, K. Karuri, S. Kraemer, and M. Pandey. A design flow for configurable embedded processors based on optimized instruction set extension synthesis. *Automation & Test in Europe (DATE)*, 1 :6, March 2006. Munich, Germany, March 2006.
27. Stephane Piskorski, Lionel Lacassagne, Samir Bouaziz, and Daniel Etiemble. Customizing cpu instructions for embedded vision systems. *International Workshop on Computer Architecture for Machine Perception and Sensing CAMP*, pages 59–64, 2006.