Conception fonctionnelle d'une caméra intelligente pour le traitement vidéo temps réel

Barthélémy HEYRMAN[1], Michel PAINDAVOINE[1], Renaud SCHMIT[2], Laurent LETELLIER[2]

[1]LE2I, UMR CNRS 5158
Université de Bourgogne Aile des Sciences de l'Ingenieur BP 47870 - 21078 Dijon Cedex, France

[2]CEA/DRT-LIST/DTSI/SARC/LCEI,
CEA-SACLAY, Bat 528, F-91191 Gif-sur-Yvette Cedex

barthelemy.heyrman@u-bourgogne.fr, paindav@u-bourgogne.fr
renaud.schmit@cea.fr, laurent.letellier@cea.fr

Résumé – La vision assistée par odinateur occupe un rôle de plus en plus important dans notre société, comme dans la sécurité des biens et des personnes, la production industrielle, les télécommunications, la robotique, ... Cependant, les développements techniques sont encore timides et ralentis par des facteurs aussi différents que le coût des capteurs, le manque de flexibilité, la difficulté à développer rapidement des applications complexes et robustes ou encore la faiblesse de ces systemes à interagir entre eux ou avec leur environnement. Ce papier présente notre concept de caméra intelligente dotée de capacités de traitement video temps réel. Un capteur CMOS, un processeur et une unité reconfigurable sont associés sur le même chip pour offrir flexibilité et hautes performances.

Abstract – Computer-assisted vision plays more and more a part in our society, in various fields such as personal and goods safety, industrial production, telecommunications, robotics... However, technical developments are still rare and slowed down by various factors linked to sensor cost, lack of system flexibility, difficulty of rapidly developing complex and robust applications, and lack of interaction among these systems themselves, or with their environment.This paper describes our proposal for a smart camera with real-time video processing capabilities. A CMOS sensor, processor and, reconfigurable unit associated in the same chip will allow scalability, flexibility, and high performance.

1 Introduction

Grace aux progrès de la technologie, il est possible aujourd'hui de concevoir des caméras qui intègrent des capacités de traitement temps réel de façon à fournir non seulement la vidéo mais aussi les informations pertinentes de la scène. Essentiellement, deux approches de caméras intelligentes ont été proposées dans la littérature : les rétines et des circuits plus génériques. Les rétines intègrent au niveau du pixel des traitements bas niveau. Ces éléments sont souvents constitués de quelques dizaines de transistors par pixel permettant de réaliser différents traitements tels que :

- l'amélioration d'image, du filtrage [7] [2]
- la détection [11] ou de l'estimation de mouvement
- des opérations morphologiques et logiques [8]

Pour autoriser plus de souplesse, une autre solution consiste à coupler un capteur avec un processeur généraliste. Cela permet l'exécution d'applications plus complexes. Dans [5], chaque pixel contient une ALU bit série, une mémoire RAM de 24 bits et des entrées/sorties sur 8 bits. Ce circuit réalise une détection de contours en $5.6\ \mu s$ et un lissage en $7.7\ \mu s$. L'architecture SCAMP [3] est un réseau de type mesh d'APE (Analog Processing Unit). Un capteur de 21×21 SCAMP a été réalisé et permet d'effectuer un lissage en $5.6\ \mu s$ et une détection de contours en $11.6 \mu s$. Le capteur présenté dans [4] est plus particulierement conçu pour des opérations sur des blocs (DCT) ou des convolutions spatiales en utilisant des noyaux de taille 8×8. D'autres approches, comme celle présentée dans [9], utilisent des réseaux de neurones cellulaires. Ce capteur de taille 128×128 est capable de réaliser des convolutions 3×3, des opérations arithmétiques et booléennes.

Nous avons vu dans cette section que les travaux en ce qui concerne les capteurs CMOS et leurs possibilités sont nombreux. Cependant, les solutions présentées (Rétines et capteurs dotés de possibilités de traitements plus génériques) semblent avoir des difficultés à répondre efficacement à tous les besoins des applications de vision. Les rétines offrent l'avantage de la compacité et sont très efficaces pour les traitements bas niveau, mais pour des raisons de place limitée, leurs fonctionnalités sont souvent pauvres et figées. De leur coté, les capteurs associés à des éléments de calculs programmables apportent plus de souplesse mais leurs performances sont moindres et ils sont moins compacts. Nous présentons, dans la seconde partie de cette article, l'architecture que nous avons proposée afin de répondre aux problèmes soulevés dans l'introduction. La troisième partie présente les applications portées sur ce système et donne les résultats en terme de performances. La conclusion présente les travaux à venir d'un point de vue "outils" et "architecture".

2 Architecture proposée

Comme nous l'avons vu précédemment, les deux approches présentées ne sont pas satisfaisantes pour des applications de vision. En effet, les rétines, même si elles offrent des possibilités de traitements proches du pixel, ne peuvent pas être utilisées seules pour des applications complexes et les performan-

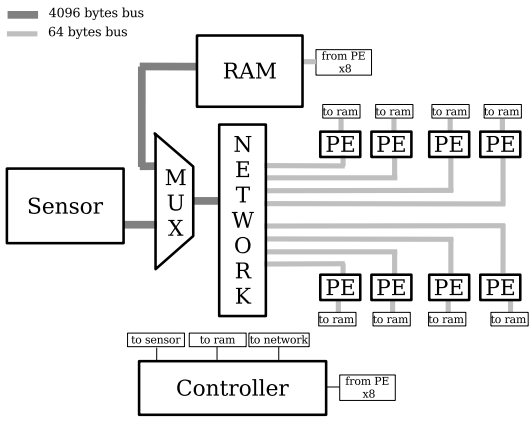

FIG. 1 – Synoptique de la caméra intelligente

ces de processeurs généralistes sont souvent insuffisantes.
La seconde remarque que nous faisons concerne les sorties des capteurs ou rétines. Même si ceux-ci réalisent leurs traitements de manière parallèle, les données pré-traitées sont souvent envoyées de manière séquentielle. Ceci constitue un important goulot d'étranglement dans le chemin de données et réduit considérablement la bande passante.
La troisième remarque s'appuie sur le faible nombre de pixels pertinents dans une image à traiter. La plupart du temps, seules de petites régions d'intérets (quelques dizaines de pixels) sont utiles comme dans le cas du décodage de "data matrix" ou de "détection d'amers".
Pour répondre à toutes ces questions, nous avons choisi de concevoir un capteur avec une sortie massivement parallèle capable de fournir des régions d'une image. Il est couplé à une RAM et un réseau de processeurs capables, respectivement, de stocker/traiter, les données émises. La figure 1 donne le synoptique de ce système.

2.1 Le capteur à sortie massivement parallèle

Le capteur est basé sur une technologie CMOS qui autorise un accès aléatoire aux pixels. Une autre caractéristique plus importante est la possibilité d'accéder à une sous-région de l'image en une seule fois. Pour répondre à ceci, le capteur est conçu pour fournir 4096 pixels en parallèle. En effet, l'évolution des techniques submicroniques permet de réduire la taille des transistors et des liens sur les couches de métal. On peut ainsi disposer en sorties d'un capteur avec beaucoup plus de fils puisqu'on reste dans le même composant. Ainsi, en technologie 90nm, la largeur d'un lien en métal 2 fait $0.30\ \mu m$ et l'écartement entre deux liens est de $0.28\ \mu m$. Il faut donc une longueur d'environ 20mm de silicium pour sortir 4096×8 fils soit un carré de $25mm^2$. Le circuit de lecture autorise en sortie des fenêtres carrées jusqu'à 64×64 pixels. Un autre avantage, de cette sortie en parallèle, est le fonctionnement réduit en fréquence des convertisseurs A/D. Ce concept est aussi présenté dans [10]. Les auteurs présentent un capteur capable de fournir en sortie des ROIs (région d'intéret) et qui permet par ailleurs de réaliser des trackings. Pour limiter la taille de la logique de commande, le déplacement des fenêtres se fait par pas de 32 pixels. Les pixels sont lus 4 par 4.

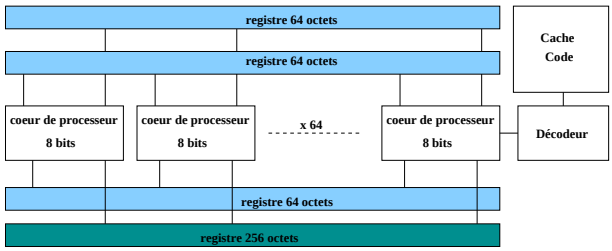

FIG. 2 – Schéma d'un processeur élémentaire

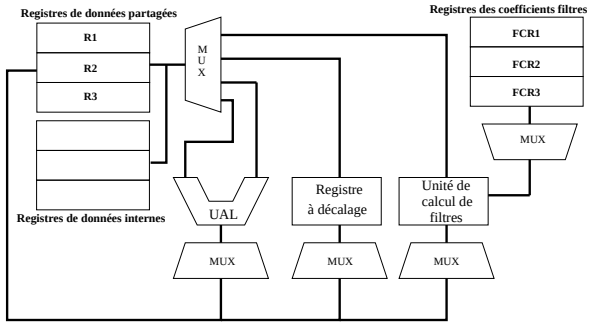

FIG. 3 – Structure détaillée d'un des cœurs 8 bits d'un PE

2.2 Description d'un PE

Ces PEs sont conçus pour exécuter aussi efficacement que possible des opérations simples que l'on retrouve souvent dans des applications de vision. En effet, ces algorithmes ne se concentrent généralement que sur une ou plusieurs petites parties de l'image. Par exemple, la compression JPEG utilise des fenêtres 8×8. C'est pourquoi chaque processeur est capable d'effectuer des opérations sur une fenêtre de 8×8 pixels au maximum. On peut imaginer regrouper des processeurs entre eux pour traiter une plus grande fenêtre. Les fonctionnalités implémentées sont les opérations arithmétiques et logiques, les filtrages par masque, les convolutions et les corrélations mais aussi les calculs d'histogrammes et les seuillages. Une structure de type MAC permet de réaliser efficacement ces traitements. Chaque PE est un groupe de 64 cœurs de processeurs (8 bits en entrée et 16 bits en sortie) qui contiennent des fonctions prédéfinies. Un cœur de processeur contient une ALU qui réalise tous les calculs "classiques" (additions, soustractions, opérations logiques), un registe à décalage qui permet d'effectuer des multiplications/divisions par puissance de deux, et une unité de calcul de filtre qui réalise des opérations de type somme de produits, ou (ET de OU). Les processeurs élémentaires contiennent aussi 3 registres de 64 mots de 16 bits pour le stockage des données entre deux opérations entre pixels, un registre de 256 octets pour les opérations effectuées sur des tables (calcul d'histogramme par exemple), et une mémoire cache pour stocker le code à exécuter. Les applications sont codées en assembleur comme ceci :

- Un mot clé : avr pour moyenne, add pour addition
- l'emplacement de la fenêtre 64×64 que le capteur doit fournir
- la banque mémoire à accéder
- 4 nombres donnant la configuration du réseau

Par exemple :
add r0 m0 0 0 8 8 // addition
hst // histogramme du résultat précédent

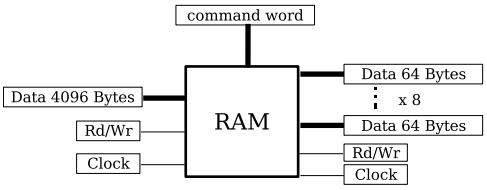

FIG. 4 – Schéma de la RAM

thr 45 // seuillage en fonction de la valeur donnée
va réaliser la somme entre la fenêtre en haut à gauche du capteur et le coin en haut à gauche d'une image stockée en mémoire. Certaines instructions ne requièrent pas de paramètres comme **hst** ou un nombre réduit comme **thr**.

2.3 Choix du réseau

C'est une partie importante de notre système. En effet, si nous ne sommes pas capables de fournir les données efficacement aux éléments de calculs, c'est à dire suffisamment rapidement et en quantité, l'architecture souffrira de pertes de performances importantes. Les réseaux de type cross-bar sont très efficaces pour dispatcher les données mais leur coût en silicium devient vite rédhibitoire quand le nombre d'entrées/sorties grandit. Nous avons choisi un cross-bar partiel qui prend en entrée 4096 pixels et ne fournit en sortie que 512 pixels. Ces derniers sont en fait 8 fenêtres de 8×8 pixels prises n'importe où dans la fenêtre d'entrée.

2.4 Conception de la RAM

Nous avons choisi pour notre système une RAM double port. En effet, elle permet de lire et d'écrire des données simultanément. La capacité de la mémoire est de 128×64 octets. L'un des ports de 4096 octets est relié au capteur ou au réseau via le multiplexeur. Ceci permet de stocker une fenêtre en un coup d'horloge. Son fonctionnement est classique : une horloge, un bus d'adresse et un signal de lecture/écriture. Le second port est différent par son adressage. Il est large de 8×64 octets et il est relié à chaque PE. Il possède une horloge, un signal de lecture/écriture et un mot de commande. Un schéma de cette mémoire est donné figure 4.

2.5 Le contrôleur

Son rôle est de superviser le fonctionnement du système. Il génère les signaux de commandes en fonction des requêtes des PEs. Il gère aussi les acquisitions d'images et leur stockage en mémoire. Il envoie les signaux de démarrage pour les traitements et gère les accès à la mémoire. Il peut être comparé à un contrôleur d'interruption et ne réalise aucun traitement. Ses réactions dépendent de l'application implantée. Une description plus précise de son fonctionnement est donnée dans la section 3.

3 Modélisation et validation de l'architecture

Dans le but d'étudier notre concept, nous avons décidé de le modéliser en SystemC. Les temps de simulation sont grandement améliorés par rapport à une description VHDL grâce à cette librairie C++. Un autre avantage est le haut niveau de description possible. Nous avons fait dans un premier temps un modèle fonctionnel software qui sera par la suite raffiné en utilisant la co-simulation SystemC-VHDL. Afin de valider cette architecture, nous l'avons testée sur trois applications de traitements d'images. Nous présentons rapidement ces application dans une première partie, puis la seconde partie présente les résultats obtenus.

3.1 Les applications implantées sur cette architecture

La première application utilisée pour valider le modèle, détecte le mouvement dans une image par soustraction. Pour cela, une moyenne sur 4 images est réalisée pour réduire le bruit, on soustrait ensuite l'image de référence. Le résultat est seuillé et des opérations morphologiques sont réalisées pour éliminer le bruit résiduel. Le nombre d'objets est comptabilisé et si il est différent du nombre qui apparaît dans l'image de référence, cette image est mise à jour.

La seconde application concerne la localisation et la reconnaissance de visages dans une image. L'algorithme utilisé repose sur les réseaux de neurones RBF. Il s'agit tout d'abord d'extraire de l'image un vecteur de données (moyenne glissante de 4 pixels) que l'on présente au réseau de neurone. Nous avons remplacé l'utilisation d'une fonction gaussienne dans les neurones par une fonction "porte". Cette approximation donne des résultats convenables [6] et permet d'accélérer très efficacement les traitements.

La troisième application réalise une stabilisation d'image [1]. Pour ce faire, des points caractéristiques sont définis dans une première image et recherchés dans une seconde par corrélation. La variations des coordonnées des points retrouvés dans la seconde image permet de retrouver le mouvement. L'application du "mouvement inverse" stabilise les images. Nous n'avons implantés que les deux premières étapes de ce traitement.

3.2 Résulats

Le premier résultat que nous obtenons est l'accélération du stockage des données en mémoire qui se fait par paquets de 4096 pixels. Cela signifie qu'avec une horloge de $10\ ns$ pour le système et un capteur $1K \times 1K$, 256 cycles d'horloges sont nécessaires pour stocker l'image complète et la bande passante théorique maximum entre le capteur et la mémoire est $1K \times 1K/2.56\ \mu s = 409.6\ Goctets/s$. Du point de vue des opérateurs, chacun requiert 3 cycles d'horloge pour traiter les données dans le pire cas (des accès mémoires sont nécessaires pour chaque instruction). Par exemple, une application constituée de 5 tâches différentes requiert 15 cycles d'horloge pour traiter 8×64 pixels. Une image de $1K \times 1K$ est traitée en : $((1K \times 1K)/(8 \times 64)) \times 15 = 30720$ cycles. Si nous choisissons une horloge de $10\ ns$, il faut environ 0.3ms pour effectuer l'opération. Dans ce cas plus de 3000 images peuvent être traitées en une seconde.

Le tableau 1 rassemble les résultats obtenus pour notre architecture. Nous constatons que les performances atteintes sont importantes mais nous pouvons noter que le système est plus efficace sur des traitements correspondant à ses caractéristiques

(fenêtres 64×64, filtrages 8×8) qui exploitent pleinement les ressources disponibles.

TAB. 1 – Performances obtenues

Application	temps (ms)	Accélération	Efficacité
Détection de mouvement	0,52	7,63	0,95
Reconnaissance de visage	2	7,7	0,96
Stabilisation d'image	0,3	7,83	0,98

Le tableau 2 donne un récapitulatif des performances obtenues par notre architecture comparées avec celles d'autres architectures. L'architecture ARM est basée sur un ARM922T qui fonctionne à 100 MHz. Le processeur Athlon est cadencé à 1,43 GHz.

TAB. 2 – Récapitulatif comparatif des performances obtenues (en ms)

Application	Notre archi.	ARM	Athlon	ZISC
Détection de mouvement	0,52	76,9	8	-
Reconnaissance de visage	2	-	-	16
Stabilisation d'image	0,3	-	26,3	-

4 Conclusion et perspectives

Nous avons décrit, dans cet article, une nouvelle architecture de caméra intelligente offrant des possibilités de traitements très rapides. Pour réaliser ceci, un capteur à sorties massivement parallèles qui a la possibilité de lire des régions d'intéret a été proposé. Un réseau de processeurs prenant en compte les spécificités des applications de vision et capable d'effectuer des opérations comme les filtrages par masque ou les convolutions est également proposé. Un modèle fonctionnel a de plus été réalisé pour valider ce concept.
Des travaux futurs vont concerner l'implantation de l'architecture sur silicium. Des améliorations au niveau des processeurs peuvent être apportées notamment en utilisant de la logique reconfigurable afin de rendre l'implantation d'opérateurs spécifiques plus simple. De même, des outils de programmation haut-niveau et automatiques dédiés à l'architecture pour en simplifier l'utilisation sont nécessaires et vont être étudiés.

Références

[1] L. Damez, J.P. Drutin, F. Diaz, and N. Allezard. Stabilisation 3d temps rel d'images : aspects algorithmiques et implantation parallle. In *Journes Francophones sur l'Adquation Algorithme Architecture - JFAAA'05*, pages 203–208, 2005.

[2] T. Delbrück and C.A. Mead. *Analog VLSI phototransduction by continuous-time, adaptive, logarithmic photoreceptor circuits*, pages 139–161. IEEE Computer Society Press, 1995.

[3] P. Dudek and P. J. Hicks. A general-purpose cmos vision chip with a processor-per-pixel simd array. In *Proc. ESSIRC2001*, 2001.

[4] A. Graupner, J. Schreiter, S. Getzlaff, and R. Schüffny. Cmos image sensor with mixed-signal processor array. *IEEE Journal of Solid-State Circuits*, 38(6) :948–957, June 2003.

[5] M. Ishikawa, K. Ogawa, T. Komuro, and I. Ishii. A cmos vision chip with simd processing element array for 1ms image processing. In *Proc. Conf. ISSCC'99*, number TP2.2, 1999.

[6] N. Malasn. *Localisation et reconnaissance de visages en temps rel : algorithmes et architectures*. PhD thesis, Thse de l'universit de Bourgogne, 2002.

[7] Y. Ni, F. Devos, M. Boujrad, and J.H. Guan. A histogram equalization based adaptive image sensor for real-time vision. *IEEE J. of Solid State Circuits*, 32(7) :1027–1036, July 1997.

[8] F. Paillet, D. Mercier, and T. Bernard. Making the most of 15k lambda2 silicon area for a digital retina. In Proc. SPIE, editor, *Advanced Focal Plane Arrays and Elactronic Cameras*, volume 3410, 1998.

[9] A. Rodrigues-Vazquez, G. Linan-Cembrano, L. Carranza, E. Roca-Moreno, R. Carmona-Galan, F. Jimenez-Garrido, R. Domiguez-Castro, and S. Espejo Meana. Ace16k : The third generation of mixed-signal simd-cnn ace chips toward vsocs. *IEEE Transactions on Circuits ans Systems*, 51(5) :851–863, 2004.

[10] O. Schrey, J. Huppertz, G. Filimonovic, A. Bussmann, W. Brockherde, and B.J. Hosticka. A 1kx1k high dynamic range cmos image sensor with on-chip programmable region of interest readout. In *Proc. ESSCIRC2001*, 2001.

[11] S. Shigematsu, H. Morimura, Y. Tanabe, T. Adachi, and Machida K. A single-chip fingerprint sensor and identifier. *IEEE Journal of Solid-State Circuits*, 34(12) :1852–1859, December 1999.