Conception mixte d'un capteur d'images intelligent intégré à traitements locaux massivement parallèles

Juliette Le Hir

université PARIS-SACLAY

CentraleSupélec

NNT : 2018SACLC107

Conception mixte d'un capteur d'images intelligent intégré à traitements locaux massivement parallèles

Thèse de doctorat de l'Université Paris-Saclay

préparée à CentraleSupélec

École doctorale n°575 : electrical, optical, bio : physics and engineering (EOBE)

Spécialité : électronique et optoélectronique, nano- et microtechnologies

Thèse présentée et soutenue à Gif-sur-Yvette, le 14 décembre 2018, par

Juliette LE HIR

Composition du Jury :

Dominique Ginhac Professeur des Universités, Université de Bourgogne (Le2i)	Président
Gilles Sicard Ingénieur de Recherche HDR, CEA-Leti	Rapporteur
Wilfried Uhring Professeur des Universités, Université de Strasbourg (ICube)	Rapporteur
François Rivet Maître de Conférence, Bordeaux INP (IMS)	Examinateur
Jean-Christophe Pesquet Professeur des Universités, CentraleSupélec (CVN)	Examinateur
Jérôme Juillard Professeur, CentraleSupélec (GeePs)	Directeur de thèse
Anthony Kolar Associate Professor, CentraleSupélec (GeePs)	Encadrant

Thèse de doctorat

Remerciements

Cette thèse est le résultat d'un travail de trois ans mené au laboratoire GeePs, dans l'établissement CentraleSupélec. Pour m'avoir permis de mener ces recherches en acceptant de diriger cette thèse, je souhaite remercier Jérôme Juillard. Il a toujours pris le temps d'apporter un éclairage pertinent, de relire mes écrits et d'en proposer des améliorations malgré un sujet différent de son domaine. Merci également à Anthony Kolar d'avoir proposé un sujet si vaste, alliant électronique analogique et électronique numérique, et de m'avoir laissée libre de développer mes idées au fur et à mesure de cette thèse. L'équipe en générale a été très sympathique pendant ces trois ans. Je souhaiterais en particulier remercier Emilie Avignon qui, avec Hervé Mathias du laboratoire C2N, a pris le temps d'apporter son expertise en analogique pour m'aider sur le layout, au cours d'une réunion dont je me souviendrai longtemps pour son efficacité.

Au-delà des enseignants-chercheurs, je tiens à remercier également l'équipe administrative sans qui la thèse se déroulerait bien moins facilement : Alexandra Siebert puis Sophie Gonçalvez au niveau de l'équipe, Laurence Stephen au niveau de l'école doctorale et Anne Batalie au niveau de l'établissement, qui sont toutes remarquables d'efficacité et de gentillesse. Philippe Dessante de l'école doctorale a fait montre de beaucoup de patience également pour répondre aux personnes stressées, et je l'en remercie chaleureusement.

Pour les côtés plus récréatifs de ces trois ans à CentraleSupélec, merci aux collègues amateurs de jeux de sociétés : Laurent, Morgan, Jérôme, mais aussi et surtout aux doctorants. Qu'ils soient de l'équipe Systèmes électroniques, de l'équipe Energie, voire des autres laboratoires de l'école ; qu'ils aient fini leur thèse ou qu'elle soit toujours en cours, ce sont d'innombrables souvenirs qui me viennent à l'esprit. En particulier, les séances de jeux le midi au département Energie faisaient une pause fantastique, que ce soit à 3 doctorants ou à 12 comme les jours fastes ! Peut-on citer tous ces doctorants ? La liste est longue et j'en oublie sans doute : Ludwig, Audrey, Grand Pierre et Petit Pierre, Alexis, Joao, Li, Manon et Fernando en électronique ; Julien, Jad, Paul, Teodor, Mickaël, Paul-Antoine, Eléonore, Marine, Olivier, Simon, Bogdan, Davi, Kelly, Christine, Jean, Ferréol et Guillaume en énergie, Antoine, Arnaud, Andreea, Nolwenn et Ion au L2S, Pedro et Orian à Sondra,…

Un autre projet en parallèle de la recherche et de l'enseignement qui me tenait à cœur est celui de l'association des doctorants (ADSS devenue ATACS). Je tiens donc à remercier les doctorants avec qui nous avons pu remettre sur pied cette association et organiser des évènements, qu'ils soient conviviaux ou de promotion de la recherche (afterworks, Week-end Entre Doctorants, Meet-the-PhD, Cafés Interpol,…) : Ludwig, Eléonore, PA, Andreea, Nolwenn et Ion. Un travail d'équipe très intéressant et des amitiés qui se sont étroitement nouées et qui dureront je l'espère : merci en particulier à ma « grande copine de thèse » Eléonore avec qui j'ai partagé tant de choses et à Paul-Antoine pour les débats enflammés. Merci également aux doctorants des années suivantes qui ont continué et continuent de faire vivre cette association.

Enfin, merci à tous ceux qui ont été présents pendant ces trois ans, de près ou de loin : amis, colocataires, famille…

Table des matières

Introduction

Ce travail s'inscrit dans une action de recherche du Laboratoire GeePs (Génie électrique et électronique de Paris), et en particulier de l'équipe MiSCaS (Mixed-Signal Circuits and Systems). Cette action vise le développement de nouvelles approches dans la conception des systèmes de vision ainsi que l'optimisation du répartitionnement calculatoire dans les systèmes électroniques. Cette thèse se trouve donc à la croisée des chemins entre imagerie et traitements embarqués, il s'agit du domaine des imageurs intelligents. En effet, les systèmes embarqués doivent être capables d'analyser leur environnement, et utilisent pour cela des capteurs. Les imageurs, ou capteurs d'images, en sont un exemple particulier. La vision peut servir à la navigation d'un objet mobile embarqué : repérer son chemin, les ouvertures ou les obstacles. Elle peut permettre l'interaction du système avec son environnement comme par exemple une interaction homme-robot par reconnaissance de gestes. Elle peut aussi permettre d'imager des endroits inaccessibles autrement, par exemple pour détecter des anomalies du tube digestif grâce à des capsules endoscopiques, ou encore repérer depuis le ciel des zones de sinistre, évaluer leur dangerosité pour les sauveteurs et identifier le nombre ou l'état des victimes.

Ces applications n'utilisent pas les images brutes mais l'information contenue dans ces images. Pour bien appréhender les enjeux de cette problématique, il est nécessaire de différencier la prise d'image de rendu et celle de la vision par ordinateur. En effet, la reconnaissance d'objet est le résultat d'une cascade de traitements, comme de la détection de points d'intérêt dans une image, de la classification de ces points d'intérêt, de la comparaison à des bases de données, etc. Tous ces traitements d'images peuvent être réalisés sur des ordinateurs à part entière pour faciliter leur implémentation. En revanche, cela nécessite de transmettre le flux d'images brutes depuis le système embarqué vers l'ordinateur au sol, ce qui représente un grand coût énergétique pour le système.

Les systèmes embarqués ont deux contraintes majeures : une contrainte en volume (on ne veut pas d'un drone énorme) mais aussi et surtout en énergie. En effet ces systèmes doivent être les plus indépendants possibles et fonctionnent donc sur batterie. La capacité de celle-ci étant limitée, l'énergie disponible doit être dédiée le plus possible à la fonction principale du système : par exemple le vol pour un drone. Les fonctions de vision pour appréhender l'environnement sont nécessaires au bon déroulement de la fonction principale mais restent secondaires. C'est pourquoi l'architecture classique d'un système qui embarque un capteur et transmet toutes les données brutes au sol pour recevoir ensuite les informations extraites n'est pas adaptée et peut être remplacée par de l'extraction d'information au plus près du capteur. Cela est possible en rajoutant un circuit de calcul dans le système, voire directement dans le capteur pour éviter des transferts entre puces.

Dans le cas de la vision, une solution peut se trouver dans les imageurs intelligents, qui ne renvoient pas une simple image mais une image prétraitée, et donc de l'information plus pertinente que des données brutes. Cela permet de réduire la quantité de données transmises pour des traitements plus approfondis. Afin de réduire toujours plus la consommation d'énergie, les prétraitements peuvent être réalisés en analogique, ce qui évite des convertisseurs gourmands en surface et énergie. Enfin un système embarqué peut avoir besoin des informations sur son environnement en temps réel (pour la navigation par exemple), les prétraitements doivent donc être très rapides. Dans le cas d'une image, les traitements sont souvent locaux et peuvent donc être parallélisés par répartition spatiale du calcul au sein de la matrice de pixels.

Ce travail explore donc une voie particulière parmi les imageurs intelligents : les circuits travaillant par groupes de pixels, aussi appelés macropixels. Il a pour but de réaliser un imageur intelligent versatile, à cadence vidéo classique, en technologie standard et à faible consommation. Les spécifications données servent à positionner un contexte de réalisation, mais ce travail porte davantage sur le développement d'une méthodologie de conception (capteur intelligent en électronique mixte, répartition par macropixels…) que sur un capteur avec une application stricte. Il s'agit d'une preuve de concept. Les aspects innovants de ce travail sont l'exploration du principe des macropixels, l'adaptation conjointe d'algorithme et d'architecture d'imageur, la proposition d'une nouvelle architecture, la

conception de tous les éléments (matrice, circuits de calculs et de contrôle) avec en particulier des discussions sur le dimensionnement des éléments de calculs pour des compromis entre surface occupée et précision du calcul.

Pour présenter cela, le manuscrit est organisé en six parties. Le chapitre 1 introduit plus en détails le contexte et les prérequis en matière de traitements d'images et imageurs. Il pourra aisément être passé par les lecteurs avertis. Le chapitre 2 quant à lui dresse un état de l'art des imageurs intelligents qui montre le manque d'un bon compromis entre surface dédiée au calcul et versatilité des traitements disponibles. Le chapitre 3 propose donc une nouvelle architecture pour tenter de combler ce manque. Sa structure formée de macropixels asymétriques est étudiée fonctionnellement. Les éléments de calculs nécessaires à cette architecture sont discutés dans le chapitre 4 qui propose donc d'utiliser des circuits analogiques à capacités commutées, contrôlés par un circuit numérique programmable. Les compromis entre surface et précision du calcul sont analysés. Ensuite le chapitre 5 propose une implémentation du système complet : matrice de pixels et d'éléments de calculs, ainsi que contrôle numérique. Le compromis obtenu entre versatilité du système et surface dédiée au calcul dans la matrice est discuté et comparé à l'état de l'art. Dans le chapitre 6, le système est validé par simulations en vues extraites du circuit proposé. Enfin, les conclusions et perspectives d'améliorations de ce travail exploratoire sont discutées.

Chapitre 1 : Traitements d'image et imageurs

Ce travail portant sur la conception d'un imageur intelligent, il se place à la croisée de l'imagerie et du traitement d'image. Pour bien appréhender la problématique de ce travail, ce chapitre introduit les bases des traitements d'image les plus classiquement utilisés dans le domaine de l'embarqué et des capteurs d'image CMOS : photodétecteurs, pixels et architectures. Tout cela permettra de comprendre les enjeux et fonctions des imageurs intelligents.

1.1 Traitements d'image

Dans le cadre des systèmes d'imagerie intelligents, il est important d'analyser les traitements d'image qui peuvent tirer parti de ce genre d'architecture. Il ne s'agit pas ici de faire une liste exhaustive mais bien de comprendre les mécanismes généraux des traitements d'image.

Nous nous intéresserons seulement aux images en niveaux de gris : une image est une matrice dont chaque élément représente le niveau de gris d'un pixel. Une image est donc un signal discret à deux dimensions, souvent numérisé (niveaux de gris quantifiés). Elle peut être destinée à l'affichage ou au traitement pour analyser l'environnement. On peut donc distinguer deux types de traitements d'image : les corrections pour rendre une image agréable à présenter, et les extractions d'informations. Avant la présentation rapide de ces traitements de bas et moyen niveau, les principaux outils de prétraitement d'image de base sont introduits. Plus de détails peuvent être trouvés dans [1 - Bovik 2009][2 - Jähne 2005].

1.1.1 Principaux outils de traitements

1.1.1.1 Filtrage spatial

L'un des traitements les plus utilisés est le filtrage spatial : il s'agit de convoluer l'image par un masque. Selon l'application, le masque -ou kernel- peut changer, c'est une imagette de quelques pixels (typiquement 3x3 pixels ou 5x5 pixels, de taille impaire pour que le masque soit centré sur un pixel) dont les valeurs sont déterminées à l'avance. Convoluer l'image par le masque revient à faire en chaque pixel de l'image la combinaison linéaire de ses voisins pondérés par les coefficients du masque, comme l'illustre la Figure 1. Pour chaque (*i,j*) :

$$R_{i,j} = \sum_{k=i-(M-1)/2}^{i+(M-1)/2} \sum_{l=j-(N-1)/2}^{j+(N-1)/2} m_{k,l} * P_{i+k,j+l} \qquad (1.1)$$

avec : $R_{i,j}$ le pixel (*i,j*) de l'image résultat,

$P_{i,j}$ le pixel (*i,j*) de l'image initiale,

$m_{i,j}$ le pixel (*i,j*) du masque et

M, N la taille (en pixels) du masque.

Donc dans le cas de la Figure 1 :

$$R_{i,j} = 1 * P_{i-1,j-1} + 2 * P_{i-1,j} + 1 * P_{i-1,j+1} + (-1) * P_{i+1,j-1} + (-2) * P_{i+1,j} + (-1) * P_{i+1,j+1}$$

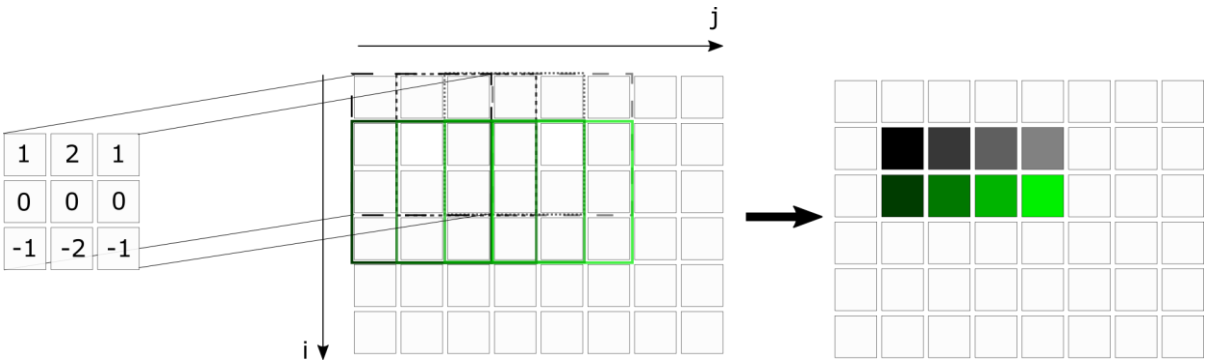


Figure 1 : Principe de la convolution spatiale. Le masque (à gauche) est utilisé pour pondérer la combinaison linéaire des pixels voisins, effectuée en chaque pixel de l'image (au centre).

Le résultat est une image de même taille que l'image de départ, filtrée par le masque. La Figure 2 présente les résultats de différents kernels sur une même image : un masque moyenneur (passe-bas) et un masque de Sobel. Cette technique est donc très versatile. Pour ne pas déformer les niveaux de gris, le masque est en général normalisé.

La Figure 2 montre également les effets de bords dus à la convolution : le bord de l'image de résultat est aberrant (noir ici) puisque la combinaison linéaire des pixels voisins n'a pas de sens sur un pixel de bordure.

1.1.1.2 Transformée de Fourier

En considérant une image comme un signal, on peut utiliser la transformée de Fourier en deux dimensions [1 - Bovik 2009](chapitre 5) [2 - Jähne 2005]. Toute image peut être décomposée en combinaison d'images sinusoïdales (Figure 3). Deux "images" sont obtenues : l'image des magnitudes et l'image des phases (Figure 4). Les détails de l'image spatiale se retrouvent dans les composantes haute fréquence, avec leur direction donnée par leur position dans le plan fréquentiel, tandis que les régions uniformes se retrouvent en basse fréquence. On peut noter que la résolution (pas d'échantillonnage) dans un domaine (spatial ou fréquentiel) détermine la taille de l'image dans l'autre domaine, et vice-versa [2 - Jähne 2005](chapitre 9).

Une fois l'image transformée dans le domaine fréquentiel, il est possible de la traiter, par exemple par des filtrages passe-bas (enlevant les hautes fréquences qui représentent les détails). Cette approche est très souvent utilisée pour la compression d'image.

1.1.1.3 Pyramides

Pour pouvoir travailler à différentes échelles sur une même image, il peut être intéressant d'en former une pyramide. La base est l'image de départ, tandis que les étages supérieurs sont des sous-échantillonnages successifs (cf. Figure 5). Avant de sous-échantillonner, pour éviter les repliements de spectre (théorème de Shannon) l'image est filtrée par un passe-bas. Ainsi, la décomposition en pyramide sert à appliquer un traitement local (utilisant quelques pixels voisins, donc à faible coût calculatoire) à différentes échelles et donc finalement sur de grands voisinages de pixels.

a)

b)

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

c)

1	2	1
0	0	0
-1	-2	-1

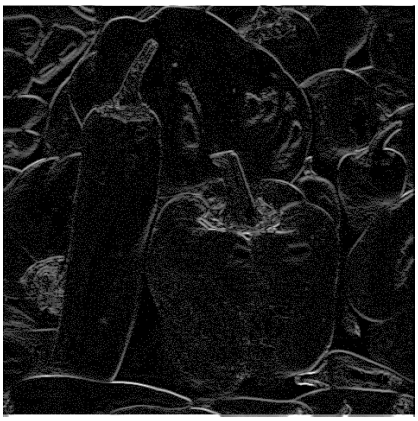


Figure 2 : a) Image initiale, b) masque moyenneur 5x5 et le résultat de la convolution de (a) par ce masque et (c) masque de Sobel pour détecter les contours horizontaux et résultat de (a) convoluée par ce masque.

Figure 3 : Exemples d'images unitaires de la décomposition de Fourier : chacune représente une fréquence spatiale.

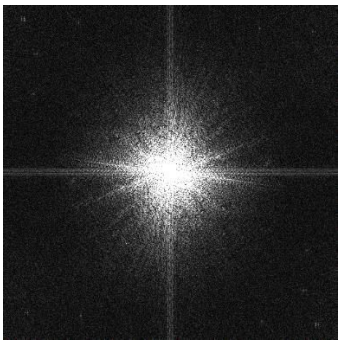
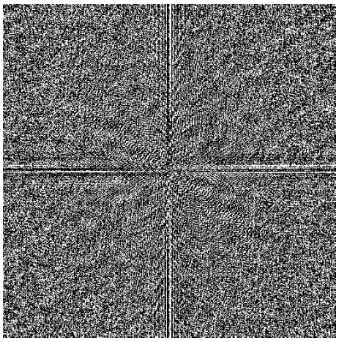
a)  b)  c) 

Figure 4 : Exemple d'image transformée : (a) image originale, (b) l'amplitude et (c) la phase de sa transformée de Fourier.

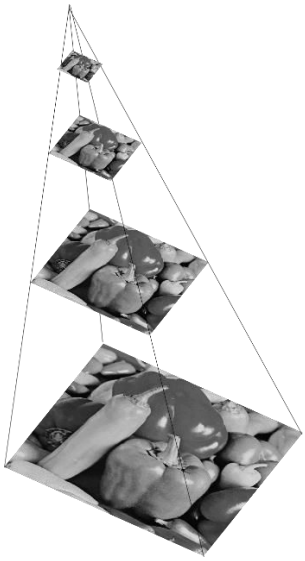


Figure 5 : Exemple de pyramide : la base est l'image originale, et chaque étage est filtré puis sous-échantillonné pour donner l'étage supérieur.

1.1.1.4 Morphologies

Une morphologie est une succession de convolutions spatiales particulières. Il s'agit de dilater puis d'éroder les objets (morphologie fermée) ou d'éroder puis dilater (morphologie ouverte). Une érosion ou une dilatation est obtenue par une convolution de l'image avec un masque unitaire (carré 3x3 par exemple) et suppression (respectivement conservation) de tous les pixels de valeur non maximale (respectivement non nulle) comme l'illustre la Figure 6.

Ces techniques peuvent être étendues aux images avec plus de 2 niveaux de gris, en prenant les minima (érosion) et maxima (dilatation) sur un voisinage [1 - Bovik 2009]. Les morphologies peuvent donc servir à enlever des aberrations de l'image, comme des pixels bruités, de valeur anormale par rapport à leurs voisins.

1.1.1.5 Histogrammes

Les histogrammes permettent de relever la fréquence d'occurrence d'un paramètre. Par défaut, en traitement d'image, l'histogramme d'une image est l'histogramme de ses niveaux de gris comme l'illustre la Figure 7. Il permet de voir la répartition des niveaux de gris.

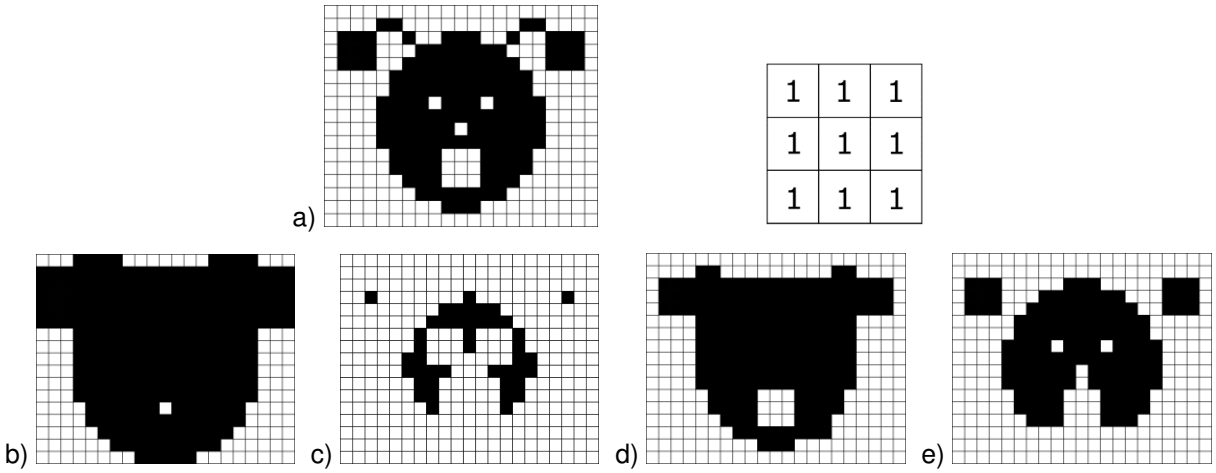


Figure 6 : a) image originale (pixels noirs sur fond blanc) et masque carré 3x3 utilisé pour : b) dilatation, c) érosion, d) fermeture et e) ouverture.

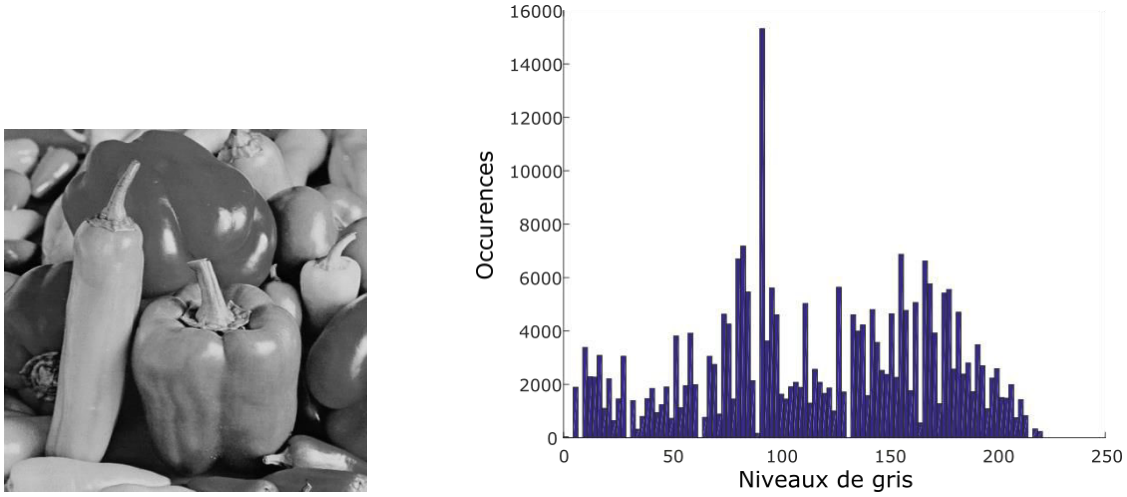


Figure 7 : Image et son histogramme.

1.1.2 Corrections

1.1.2.1 Débruitage

Dans la chaîne d'acquisition d'une image, beaucoup d'altérations de l'image peuvent se produire : distorsion due au système optique, erreur du capteur, dégradation du signal, erreur de quantification, erreur de transmission, etc. Le résultat est une image bruitée. Différents types de bruits existent, avec différentes propriétés et donc différents modèles (voir [1 - Bovik 2009](chapitre7) pour plus de détails).

Tout d'abord, certains bruits varient dans le temps, il est donc intéressant de moyenner des images successives pour les diminuer. Ensuite le bruit est souvent spatialement non-corrélé et est par conséquent une composante haute fréquence de l'image. Un filtrage passe-bas permet de débruiter ; il peut être implémenté directement en filtrage fréquentiel après transformée de Fourier, ou en filtrage spatial avec un kernel adéquat. Cependant, le bruit n'est souvent pas sur une bande de fréquence distincte de celle de l'image : les détails d'une image sont de hautes fréquences spatiales. Un filtrage passe-bas retire donc du bruit mais risque de flouter l'image si la fréquence de coupure est trop basse (Figure 8).

a)  b) 

Figure 8 : a) Image de poivrons bruitée de type poivre et sel, et b) poivrons filtrés passe-bas, débruités mais flous.

Différents filtres passe-bas existent, des plus simples comme la moyenne spatiale de pixels voisins ou la mise à zéro de toutes les hautes fréquences de l'image, aux plus complexes comme le filtrage *frequency neighbourhood weighting filter* (qui préserve les voisinages de composantes fréquentielles puisque les composantes hautes fréquences d'une image non-bruitée ne sont pas isolées). Chacun est un compromis entre efficacité de débruitage et complexité d'implémentation [1 - Bovik 2009](chapitres 10 et 11).

Pour éviter le plus possible de flouter une image en la débruitant, des filtres non-linéaires existent également, comme les filtres médians et leurs variations (l'idée est de calculer le niveau de gris médian sur un voisinage de pixels) ou les filtres morphologiques (érosion, dilatation, ou leurs successions).

1.1.2.2 Améliorations visuelles

Une image destinée à être affichée, visualisée, doit répondre aux attentes du public, notamment en termes de contraste et de netteté.

Un mauvais contraste dans une image peut venir de mauvaises conditions de photographie, comme une faible exposition à la lumière : l'image n'utilise que peu des niveaux de gris disponibles. Pour améliorer le contraste de l'image, on peut utiliser des techniques d'égalisation de l'histogramme. Tout d'abord l'histogramme de l'image peut être transformé linéairement : étiré par multiplication de chaque niveau de gris par une constante, ou étiré jusqu'à utiliser tous les niveaux disponibles (*full scale histogram stretch)*. L'histogramme peut aussi être modifié non linéairement : égalisé sur tous ses niveaux, mis en v pour accentuer les contrastes,… [1 - Bovik 2009](chapitre 3).

La netteté d'une image dépend essentiellement de la qualité de la chaîne optique utilisée, mais elle peut être virtuellement améliorée en accentuant les contours (*image sharpening*). Il s'agit d'ajouter à une image sa version filtrée passe-haut (les détails) par exemple [1 - Bovik 2009](chapitre 12) ou de soustraire sa version filtrée passe-bas (technique du masque flou).

Ces traitements d'améliorations d'images sont parmi les plus basiques, de nombreuses évolutions et dérivés existent.

1.1.3 Extraction d'information

Les images peuvent être destinées à être visualisées, comme nous l'avons vu, mais aussi à être traitées pour en extraire des informations sur l'environnement, comme des points d'intérêt, du mouvement ou encore la profondeur ce qui pourra déboucher sur la détection de la présence d'un piéton, de la survenue d'un accident, de la distance aux objets, etc.

1.1.3.1 Points d'intérêt

Les points d'intérêt dans une image peuvent être de nature variée : des contours, des coins,… Différentes méthodes permettent de les détecter.

Tout d'abord, la détection de contours est l'une des opérations les plus classiques et est souvent utilisée comme point de départ d'un algorithme évolué [3 - Sponton 2015]. Un contour est un changement abrupt de niveau de gris, donc un gradient de forte amplitude comme l'illustre la Figure 9. Calculer le gradient d'une image peut se faire par convolution de masque (Roberts, Sobel,…) [4 - Roberts 1963]. Deux masques sont appliqués, pour calculer chacun une composante du gradient (en général horizontale et verticale). La magnitude du gradient est ensuite calculée à partir de ces deux composantes. Les contours de l'image originale sont donc situés sur les pixels de fortes magnitudes sur l'image des gradients (Figure 9). Un exemple d'une détection de contours par Sobel est donné en Figure 10.

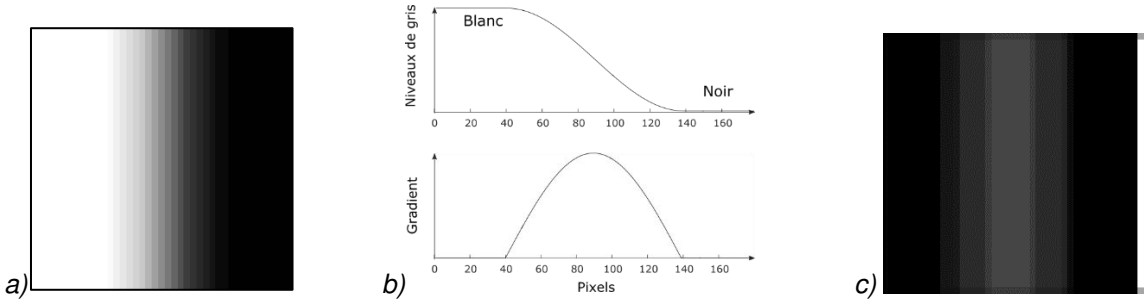


Figure 9 : a) Exemple de contour blanc/noir vertical, b) traduction en une fonction à une dimension et son gradient et c) représentation spatiale du gradient de (a).

Si le gradient donne les contours par ses maxima, la dérivée seconde de l'image originale donne les contours par ses zéros. Elle peut être calculée par convolution de masques également (Marr-Hildreth, Laplacien d'un Gaussien (LoG)) ou en approximant des voisinages de pixels (Haralick). Plus de détails sont expliqués dans [3 - Sponton 2015].

Ces convolutions par masque donneront différents résultats selon la taille du masque utilisée : un petit masque (2 ou 3 pixels de côté) ne verra bien que des contours abrupts : sur la Figure 11 le contour est bien identifié sur (b) lorsque le niveau de gris a changé en très peu de pixels, mais pas sur (d) qui partait d'un contour plus doux. Un grand masque permettra de voir les contours plus doux, au prix de plus de calculs. La décomposition en pyramide peut permettre de détecter simplement tous les types de contours, en utilisant, sur chaque étage, un petit voisinage de pixels. En effet par sous-échantillonnage, les contours doux deviennent des contours abrupts.

D'autres types de détecteurs de contours existent : il est par exemple possible de dilater les objets d'une image puis d'en soustraire l'image originale (détecteur de contour morphologique [1 - Bovik 2009]). Il n'en reste que les contours.

Au-delà des contours, les coins peuvent être repérés par le détecteur de Harris par exemple : il faut calculer les gradients de l'image, les multiplier entre eux, les filtrer (filtrage gaussien) et le résultat est utilisé comme argument d'une fonction dont les maxima sont des points de variation d'intensité lumineuse [5 - Harris 1988]. C'est un algorithme beaucoup plus coûteux en calculs que les traitements cités plus haut, mais très précis et donc largement utilisé pour détecter les coins.

Enfin, ces algorithmes relèvent les contours visibles, mais le cerveau humain est capable de retrouver les contours d'objets partiellement cachés, comme on peut le voir sur la Figure 12. Des algorithmes plus poussés comme la transformée de Hough permettent également de retrouver ces contours en modélisant les droites [6 - Hough 1959][7 - Duda 1972].

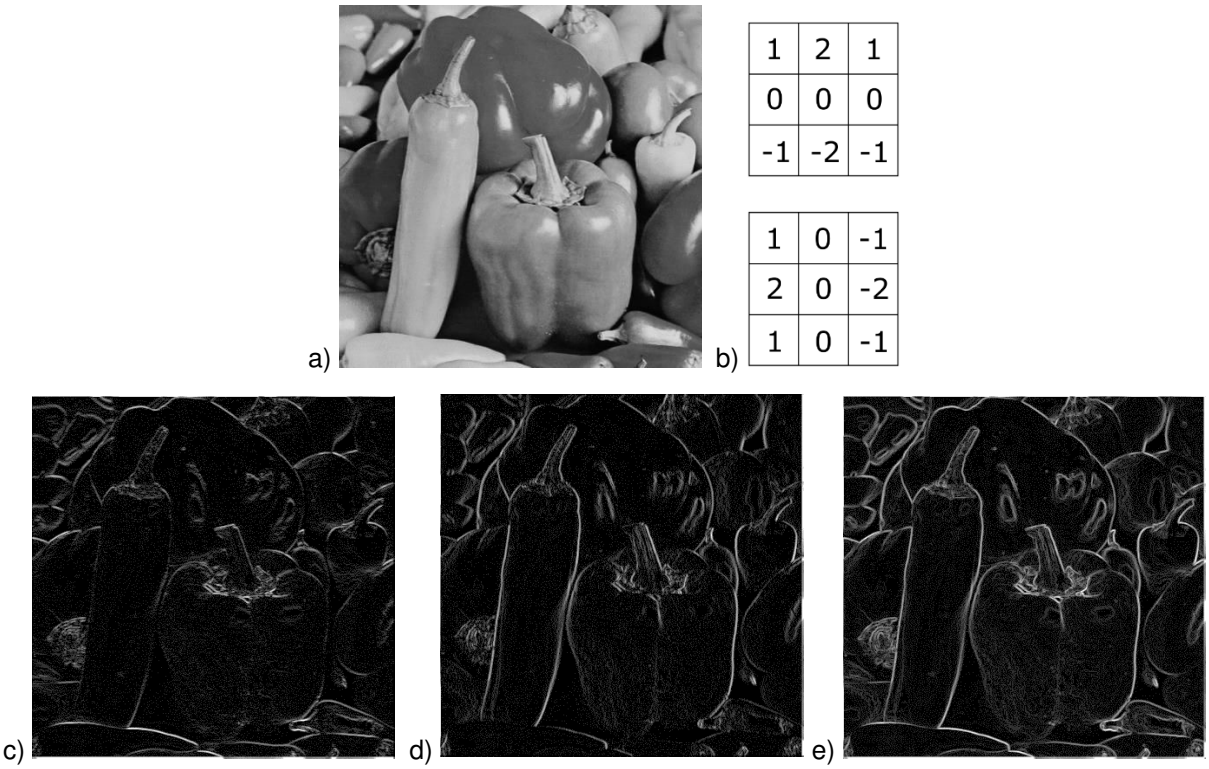


Figure 10 : a) Image originale, b) masques de Sobel (horizontal et vertical), c) contours détectés par le masque horizontal, d) contours détectés par le masque vertical et e) gradient de l'image originale.

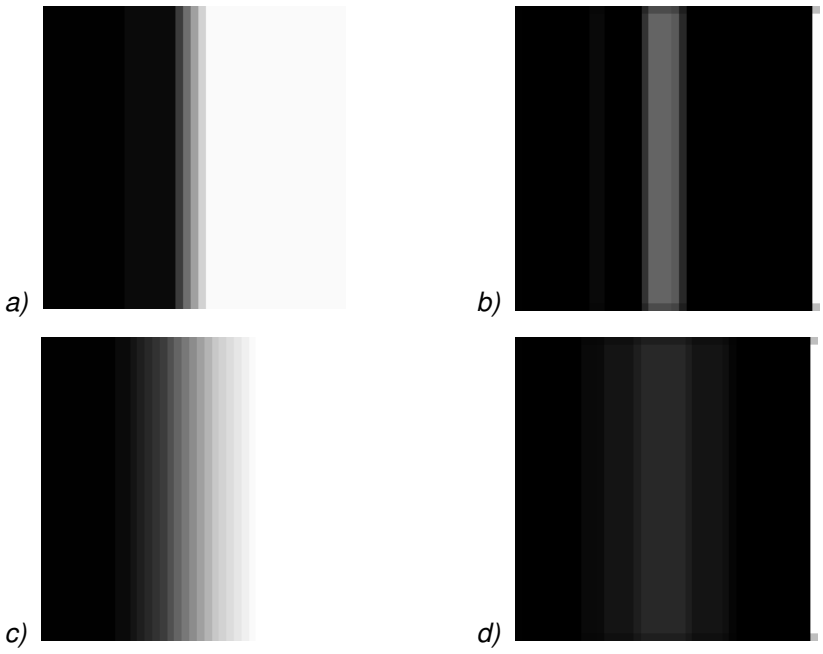


Figure 11 : a) Contour abrupt, c) contour doux et b,d) leur résultat respectif para convolution avec un masque de Sobel vertical 3x3 pixels.

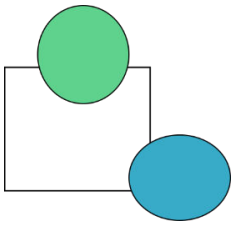


Figure 12 : Exemple de contours partiellement cachés : le carré est en partie caché mais parfaitement identifiable à l'œil.

1.1.3.2 Mouvement

Au-delà de la détection de points d'intérêt dans une image donnée, on peut s'intéresser à la détection de mouvement entre plusieurs images. En première approche, le mouvement peut être obtenu en soustrayant une image à l'image précédente dans un film. Les pixels qui représentaient un objet qui bouge auront changé de niveau de gris. Un exemple est donné en Figure 13. Les pixels clairs dans l'image de résultat sont ceux qui ont changé entre les deux images.

Pour aller plus loin, des algorithmes de plus haut niveau peuvent séparer un objet mouvant de l'arrière-plan. Par exemple, sur la Figure 13(a), on voudrait séparer l'image du personnage de l'image de l'arrière-plan. Ceci peut se faire par soustraction de l'image en cours à un modèle d'arrière-plan mémorisé, fixe ou évolutif au cours du temps. Il peut aussi être intéressant de déduire la vitesse et/ou la direction du mouvement observé pour des applications de suivi (*tracking*) par exemple : la caméra garde l'objet en mouvement dans son champ de vision.

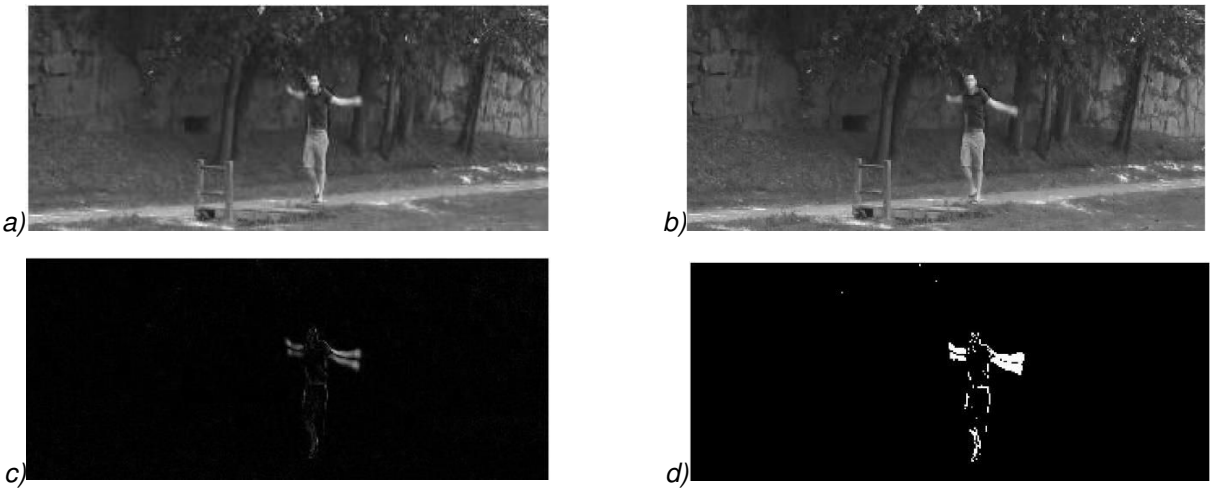


Figure 13 : (a) Image extraite d'un film montrant un homme marchant et agitant les bras, (b) image suivante, (c) leur différence et (d) leur différence seuillée (seuil à 0.1 pour des images converties en double, i.e. entre 0 et 1) pour un meilleur contraste.

1.1.3.3 Profondeur

Les algorithmes les plus simples de détection de profondeur (distance aux objets que l'on observe) font appels à un matériel particulier. En effet, la notion de profondeur dans le cerveau humain vient de la présence de deux yeux, qui donnent chacun un angle de vision différent ce qui permet de calculer le relief. Il est donc possible d'avoir une image de profondeur à partir de plusieurs caméras [8 - Nomura 2016]. Une autre technique courante est d'utiliser une source lumineuse avec l'imageur, et de mesurer le temps de parcours de la lumière entre la source, l'objet sur lequel elle se réfléchit et son arrivée sur l'imageur (*time of flight)*. Enfin, d'autres techniques se développent, comme l'utilisation de la focalisation : lorsqu'un objet est très net, cela signifie que la focalisation est adaptée à sa distance, on peut donc calculer celle-ci. Cela nécessite de faire varier et de connaître avec précision la focale [9 - Kolar 2016].

1.1.4 Exploitation des informations

Détecter des points d'intérêt n'est pas une finalité en soi : une image de contours par exemple n'est pas destinée à être affichée, mais à être utilisée par des algorithmes de plus haut niveau comme la reconnaissance de formes. Détecter du mouvement dans une scène peut permettre de déclencher un appareil (allumer une lampe par exemple) ou un autre algorithme (réveil d'un algorithme plus complexe de reconnaissance de personnes par exemple, dans le cas d'une intrusion). Les applications sont nombreuses, mais les informations extraites des images ou films présentées dans la partie précédente sont rarement utilisées de façon brute. Il faut séparer les formes, les reconnaitre, etc.

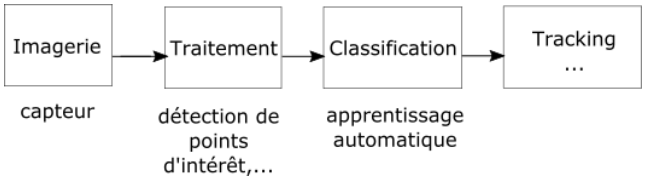


Figure 14 : Exemple de chaîne classique d'utilisation d'une image.

Il existe de nombreux algorithmes de haut niveau, et nous ne cherchons pas à les lister. Un exemple de chaîne de traitement classique est la reconnaissance d'objets. Particulièrement pour les systèmes embarqués navigants, cela permet de connaître son environnement, d'éviter les obstacles, etc. Un exemple particulier, la détection de piéton, sera abordée en chapitre 3.

A partir de points d'intérêt tels que des contours ou des coins, les formes peuvent être retrouvées par segmentation, complétées par morphologie. Les formes sont ensuite identifiées. Une fois les objets intéressants reconnus et classés, une étiquette leur est attribuée. Il est alors possible de les traquer dans les images suivantes pour voir leurs déplacements par exemple.

1.1.4.1 Apprentissage automatique

La reconnaissance d'objets ou leur classification passe en général par de l'apprentissage automatique (ou *machine learning* en anglais).

Une représentation de l'image (en niveaux de gris, en contours, en histogramme, etc.) peut être comparée à une base de données définie pour reconnaitre des objets de la base de données éventuellement présents dans l'image. Ce type de méthode fonctionne lorsque l'objet est toujours vu exactement de la même manière.

Pour détecter dans une image des classes d'objets plus générales, vus sous différents angles, etc., on utilise l'apprentissage automatique. Des algorithmes de haut niveau (SVM, Adaboost, réseaux de neurones) utilisent un descripteur de l'image pour déterminer la classe d'une image (typiquement, si un objet donné y apparait ou pas). Le descripteur est une représentation de l'image souvent déjà traitée. Par exemple une détection de piéton peut utiliser un histogramme des gradients orientés comme descripteur [10 - Dalal 2005]. Une détection de visage peut utiliser des caractéristiques rectangulaires trouvées dans l'image [11 - Viola 2001]. Les algorithmes d'apprentissage automatique créent leur schéma de décision par eux-même, selon les informations dont ils disposent.

L'apprentissage peut être supervisé : la machine est entrainée par des séries d'images avec la réponse (présence ou non de tel objet) avant d'être utilisée comme détecteur. L'apprentissage peut continuer au cours de la détection si un retour est possible : une vérification indique à la machine si elle s'est trompée ou pas, et elle augmente ainsi son entraînement au fur et à mesure. Lorsque les classes n'ont pas été déterminées à l'avance, il s'agit d'apprentissage non supervisé. L'algorithme doit découvrir par lui-même la structure des données.

Les traitements d'images décrits ci-dessus sont généralement réalisés sur des puces voir des ordinateurs extérieurs au capteur d'image. Il en existe beaucoup de dérivés ou de variantes, ils peuvent notamment être simplifiés ou adaptés selon le coût calculatoire recherché. Mais l'utilisation de ressources numériques extérieures implique la conversion analogique-numérique de la sortie des pixels, la transmission entre puces d'un nombre important de données (images de quelques méga pixels, chacun sur 8 bits au minimum en général) et donc une grande consommation d'énergie. C'est

gênant pour les applications nécessitant une optimisation de la consommation en énergie, comme les systèmes embarqués. Il est donc intéressant d'intégrer ces traitements d'image au plus près du capteur. La batterie d'un drone par exemple doit lui permettre de voler, pas se décharger en transmettant des images inutiles au sol. Dans un tel cadre, les traitements d'image qui permettent une réduction de données sont particulièrement intéressants à intégrer, comme la compression ou l'extraction d'information. On ne veut transmettre que des informations pertinentes et nécessaires. Pour intégrer des traitements d'image au plus près des capteurs, il faut d'abord étudier ces capteurs.

1.2 Capteurs d'images

Les capteurs d'images, communément appelés imageurs, appareils photographiques ou caméras, sont des dispositifs qui permettent de convertir de la lumière en une grandeur électrique pour former une image visualisable, transformable, mémorisable. Deux classes d'imageurs se dessinent, selon la technologie utilisée : CCD (pour *charge coupled device*) et CMOS (pour *complementary metal oxide semiconductor*). Les premiers permettent en général d'avoir une meilleure qualité d'image mais les seconds sont fabriqués dans une technologie standard, qui permet d'intégrer sur la même puce de l'électronique classique. Puisque nous cherchons à intégrer des traitements d'image décrits précédemment au plus près du capteur, nous nous concentrons sur les imageurs CMOS.

Un imageur est composé classiquement de photodétecteurs dans des pixels, arrangés en matrice, et de circuits annexes de contrôle et de lecture comme l'indique la Figure 15. Ces différents éléments vont être détaillés.

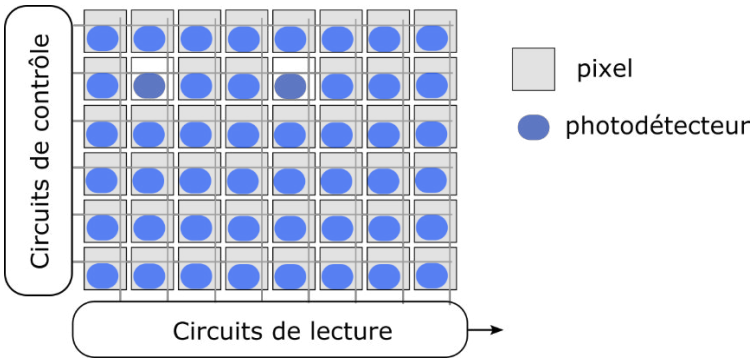


Figure 15 : Structure typique d'un imageur.

1.2.1 Photodétecteurs

Tout d'abord, les photodétecteurs – ou photosites – sont les premiers éléments de la chaîne de conversion de la lumière. Il s'agit de la surface sur laquelle les photons incidents sont reçus et convertis en grandeur électrique, plus facile à manipuler que la lumière. Ils sont de différents types, les principaux vont être présentés.

1.2.1.1 Jonctions PN

Les semiconducteurs comme le silicium peuvent absorber des photons d'énergie supérieure ou égale à leur *bandgap* en créant des paires électron-trou. Ces paires se recombinent naturellement sauf si un champ électrique dans le silicium les sépare. Un tel champ électrique peut être obtenu en polarisant en inverse une jonction PN (voir la Figure 16 : la zone de déplétion de charges possède un champ électrique). Les charges séparées créent un photocourant, c'est-à-dire un courant d'intensité représentative de la lumière captée.

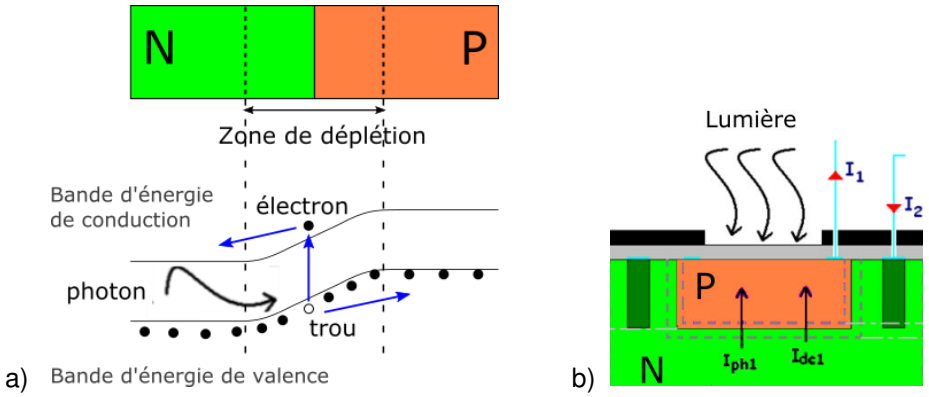


Figure 16 : a) Jonction PN et son diagramme de bandes d'énergie, avec l'absorption d'un photon incident dans la zone de déplétion où le champ électrique déplace les charges b) Schéma en coupe d'un circuit à jonction PN [12 - Feruglio 2008]. *La lumière arrive par le dessus du circuit tandis que la jonction est dans la profondeur. Le photocourant I_{ph1} est créé par les photons incidents.*

Ce courant vient s'ajouter au courant inverse de la photodiode présent sans illumination (*dark current*) comme le montre la Figure 17. Il suffit donc de mesurer le courant délivré pour connaitre la quantité de lumière reçue en temps réel, mais il est très faible, de l'ordre de quelques dizaines de nA, selon la taille de la jonction. Le courant I_{dark} dépend surtout de la température, et prend des valeurs de l'ordre du fA ou pA.

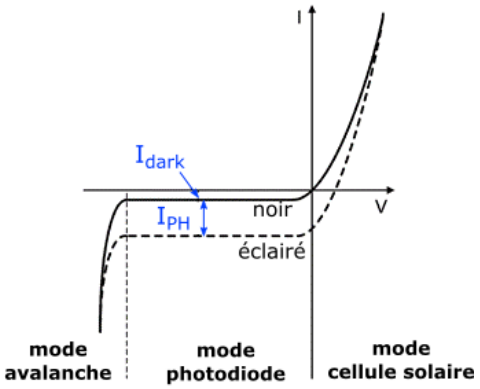


Figure 17 : Caractéristique I-V d'une jonction PN dans le noir ou illuminée [13 - Belbachir 2010].

Il est aussi possible de faire fonctionner les photodiodes en mode intégration [14 - Weckler 1967] : lorsque la diode est en circuit ouvert le courant inverse est intégré sur la capacité parasite de la diode. En première approximation la tension à ses bornes décroit donc proportionnellement à la lumière incidente, et la tension aux bornes de la photodiode dépend du photocourant et du temps d'intégration comme le montre l'équation (1.2) :

$$\Delta V_{PD} = \frac{I_{ph} \cdot T_{int}}{C_{PD}} \qquad (1.2)$$

où C_{pd} est la capacité parasite de jonction, I_{ph} est le photocourant et T_{int} est le temps d'intégration. Pour être plus précis, la capacité parasite d'une jonction PN dépend de la valeur de la tension à ses bornes, une allure réelle de la tension V_{PD} est donc illustrée sur la Figure 18. L'approximation linéaire est couramment faite.

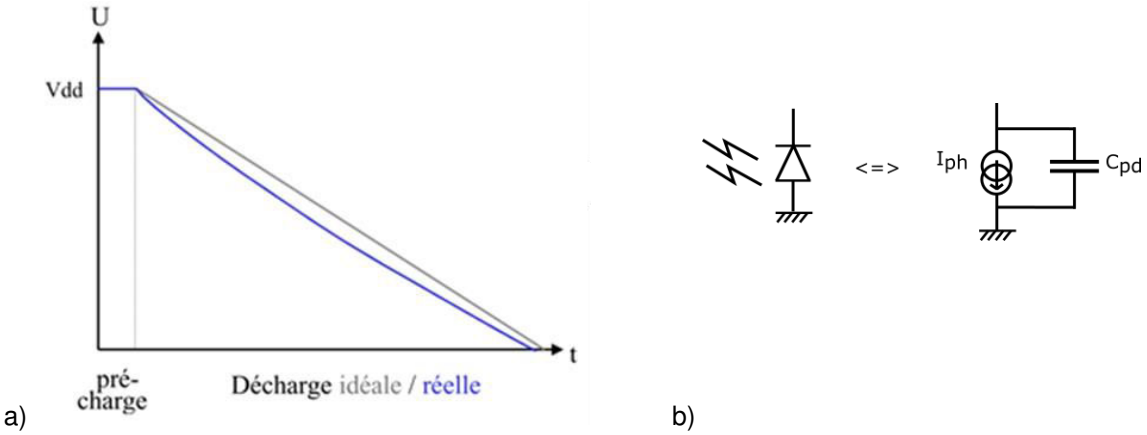


Figure 18 : (a) Evolution temporelle du potentiel d'une photodiode éclairée, comparée à une pente affine [15 - Navarro 2003] *qui correspond (b) au modèle équivalent d'une photodiode éclairée : source de photocourant en parallèle d'une capacité .*

Il existe plusieurs types de jonctions PN dans les technologies classiques selon les couches utilisées : diffusion N / substrat P, diffusion N / puits P, puits N / substrat P en sont des exemples. Elles se différencient par leurs capacités de jonctions et par leur profondeur dans le silicium. Ces paramètres, pour un même type de photodiode, dépendent aussi de la technologie particulière utilisée [15 - Navarro 2003] : se référer aux documentations des fondeurs.

La profondeur d'une photodiode influe sur le type de photons absorbés : les photons de grande longueur d'onde vont statistiquement plus loin dans le silicium avant d'être convertis en paire électron-trou. La Figure 19 montre les taux d'absorption des photons avec la profondeur pour différentes longueurs d'onde. La profondeur de la jonction PN est donc importante pour les couleurs captées : trop proche de la surface, seulement les photons bleus seront absorbés par exemple. Il en découle également la notion d'efficacité quantique du photodétecteur : le rapport des charges générées sur le nombre de photons incidents, qui dépend de la longueur d'onde considérée.

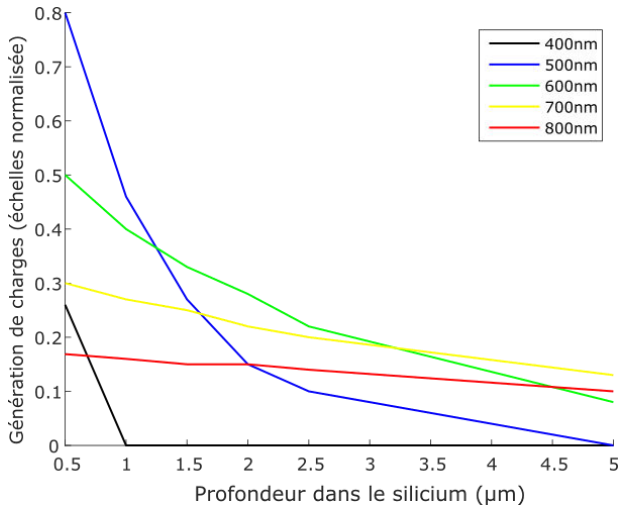


Figure 19 : Représentation de la génération de paires électron-trou en fonction de la profondeur dans le silicium, pour différentes longueurs d'onde (adapté de [12 - Feruglio 2008]).

En plus de se placer à une profondeur adéquate, pour capter un maximum de photons différents, il est intéressant d'avoir une zone de déplétion de charge (la zone qui possède le champ électrique qui empêche la recombinaison des paires électrons-trous) la plus étendue possible, c'est-à-dire qui couvre le plus grand intervalle de profondeur possible. Cette longueur dépend de la valeur de la polarisation inverse de la photodiode ou peut être agrandie par l'insertion d'une couche de silicium intrinsèque entre les deux zones dopées. Cela forme alors une jonction P-I-N, captant mieux la lumière mais de procédé de fabrication plus difficile puisqu'il faut réussir à obtenir cette couche intrinsèque [16 - Kleinman 1956][17 - Caramona Fernandes 2011].

1.2.1.2 Avalanche

Le mode avalanche d'une photodiode (cf. Figure 3) permet d'avoir un signal électrique significatif même pour un faible nombre de photons. Il s'agit alors de photodiode avalanche voire de photodiode avalanche à photon unique (*single photon avalanche diode (SPAD)*). L'avalanche, illustrée en Figure 20, est un phénomène de réaction en chaîne : les paires électron-trou, fortement accélérées dans le champ électrique élevé, créent par collision d'autres paires électron-trou, ce qui amplifie le signal. Une photodiode à avalanche est donc rapide et adaptée aux très faibles flux lumineux. En revanche c'est une structure large et qui doit être très fortement polarisée en inverse [18 - Seitz 2011].

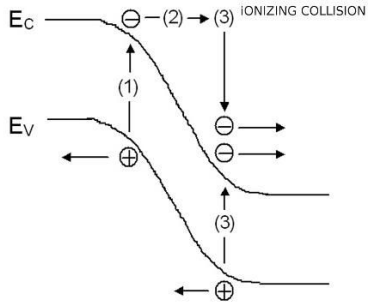


Figure 20 : Phénomène d'avalanche dans une photodiode fortement polarisée en inverse : les électrons sont fortement accélérés par le champ électrique et créent par collisions d'autres paires électron-trou (ionizing collisions). [optique-ingenieur.org]

1.2.1.3 Jonctions enterrées doubles ou triples

Superposer plusieurs jonctions les unes au-dessus des autres forme plusieurs régions de conversion photons/charges avec différentes profondeurs dans le silicium. Il est alors possible de détecter plus de longueurs d'onde différentes, et même de séparer les différentes couleurs en mesurant séparément les photocourants de chaque jonction. La Figure 21 montre un exemple de circuit [12 - Feruglio 2008]. Un exemple de tel photodétecteur a été commercialisé (Foveon), avec 3 jonctions superposées, qui permettent de mesurer simultanément les trois couleurs bleu, vert et rouge.

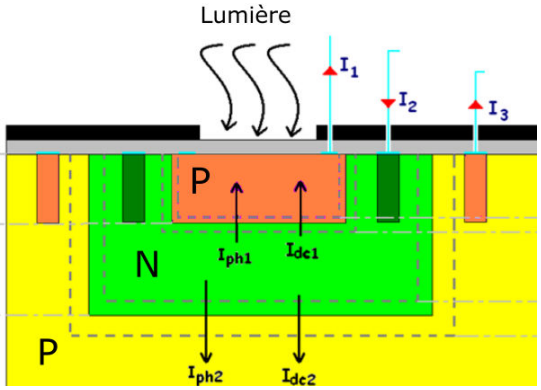


Figure 21 : Schéma en coupe d'une double jonction. Iph1 et Iph2 sont les photocourants générés dans chaque jonction et peuvent être mesurés séparément [12 - Feruglio 2008].

1.2.1.4 Autres photodétecteurs

Au-delà du problème d'absorption différenciée des photons avec la longueur d'onde, la Figure 19 montre aussi que des photons sont captés avant la jonction proprement dite et donc perdus. L'absorption des photons dans le métal étant plus faible que dans le silicium, des diodes Schottky peuvent être utilisées pour capter les photons qui ne descendent que très peu dans le silicium, en particulier les bleus et ultraviolets.

Une photogrille est un autre type de photodétecteur, fabriqué avec une grille de polysilicium sur du silicium dopé. Le champ électrique pour la séparation des paires électron-trou est créé par une

tension de polarisation appliquée sur la grille. Les charges photo-générées sont stockées (plutôt que de créer un photocourant).

Des photoconducteurs peuvent aussi être utilisés comme photodétecteurs : la conductivité d'un semiconducteur sous des températures normales varie avec le nombre de charges photogénérées. Un morceau de semiconducteur polarisé est donc parcouru par un courant représentatif de la lumière incidente. Ces photodétecteurs sont simples mais très bruités, notamment à cause de la sensibilité à la température.

Le Tableau 1 présente un résumé qualitatif des avantages et inconvénients des photodétecteurs. Un bon photodétecteur doit pouvoir être petit, consommer peu, détecter un large spectre, être rapide et facile à implémenter. Ce dernier critère représente, dans le cas des jonctions multiples, PIN ou Schottky, une difficulté de fabrication (dosage des dopages, etc.) ou dans le cas de la SPAD, une difficulté de conception de pixel (voir section 1.2.2.4).

	Taille	Consommation	Largeur de spectre détecté	Vitesse	Facilité d'implémentation
Jonction PN	-	-	-	-	☺
Jonction PIN	-	-	☺	-	☹
SPAD	☹☹	☹☹	-	☺☺☺	☹
Jonction multiple	-	☺☺	☺☺☺	-	☹
Diode Schottky	-	☺	☺	-	☹
Photogrille	-	☺	-	-	-
Photoconducteur	-	☺	☺	☹	-

Tableau 1 : Comparaison qualitative de photodétecteurs.

1.2.2 Pixel

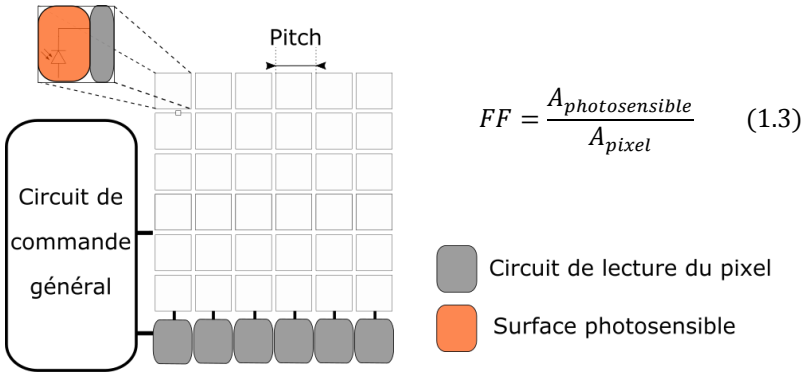
Un photosite donne seulement une information sur la lumière ; pour obtenir une véritable image, les photodétecteurs sont placés en forme de matrice dont chaque élément unitaire est appelé pixel. La première partie de cette section va introduire différentes notions de géométrie ou de paramètres électriques propres aux matrices de pixels, afin de mieux comprendre les intérêts des différentes architectures de pixels décrites dans le reste de la section.

Remarquons qu'on ne cherche pas à détailler toutes les caractéristiques d'un pixel (comme le gain de conversion, etc.), mais à en comprendre les fonctionnements, avantages et inconvénients afin de pouvoir choisir celui qui nous intéressera pour la suite. Pour plus de détails sur leur fonctionnement fin et sur leur dimensionnement, on peut se référer à [19 - Cavadore][15 - Navarro 2003][20 - Ohta 2007]…

1.2.2.1 Notions

Paramètres géométriques

Une matrice de photodétecteurs produit une image échantillonnée spatialement, dans deux directions. Pour avoir une bonne résolution spatiale (i.e. densité de pixels élevée), de petits pixels sont donc préférables. C'est pourquoi la notion de **pitch** a été introduite comme paramètre important d'un imageur : c'est la largeur d'un pixel comme l'illustre la Figure 22 (puisque les images sont souvent visualisées, notamment sur des écrans de pixels espacés régulièrement, les pixels sont généralement carrés).



$$FF = \frac{A_{photosensible}}{A_{pixel}} \qquad (1.3)$$

Figure 22 : Notions de pitch et de facteur de remplissage.

En outre, les photons ne sont captés que sur le photosite (ou surface photosensible) qui n'est pas nécessairement la surface du pixel (à cause de circuits électroniques supplémentaires, de lignes métalliques pour extraire l'information de chaque pixel vers l'extérieur de la matrice, etc. comme la suite l'explicitera). Un autre paramètre a donc été introduit : le **facteur de remplissage** (***FF*** pour *fill factor*) qui est le rapport de la surface photosensible sur la surface totale d'un pixel, souvent exprimé en %. Une illustration en est donnée en Figure 22. Plus le facteur de remplissage est faible, plus on perd de photons.

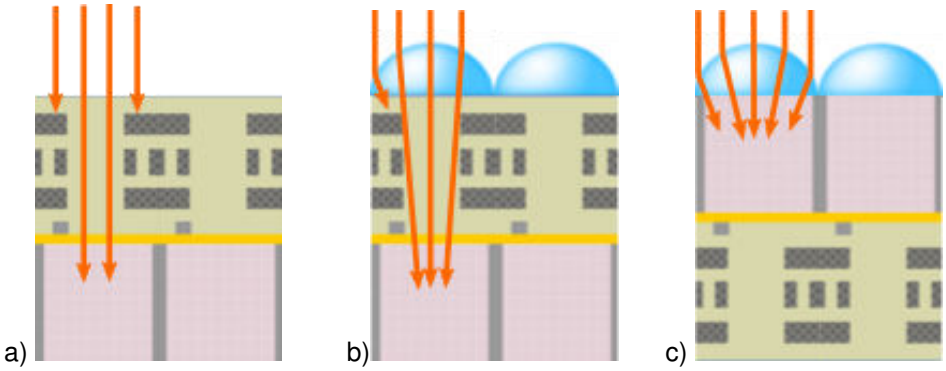


Figure 23 : a) Imageur basique : la lumière est beaucoup arrêtée par les pistes métalliques, (b) imageur à microlentilles : plus de lumière est captée, et (c) imageur BSI : toute la lumière est captée (adapté de [sony.net])

Des techniques permettent de maximiser virtuellement le facteur de remplissage tout en ayant de l'électronique dans le pixel. Les microlentilles sont des lentilles de la taille d'un pixel, placées au-dessus de leur pixel et focalisant les photons incidents sur le photosite de manière à ne pas en perdre comme le montre la Figure 23. Ce pendant cette technique est soit imparfaite (désappariement entre des microlentilles standard et la taille du pixel réalisé) ou très chère (si les lentilles sont dimensionnées spécialement pour l'imageur). Les circuits illuminés par l'arrière (*BSI* pour *backside illuminated*) sont quant à eux des imageurs dont le substrat a été aminci pour que la lumière soit captée par le dessous et non le dessus comme l'explique la Figure 23. Cette technique est aussi très chère puisque l'amincissement et la manipulation de substrats fins ne sont pas encore standards. Enfin, les technologies 3D permettent de séparer la matrice de photosites des circuits de pixels et de lecture en les superposant, ce qui amène encore un facteur de remplissage idéal au prix de techniques pour l'instant chères et en cours d'étude [21 - Knickerbocker 2008].

Paramètres électriques

Pour avoir une image détaillée, la résolution spatiale est essentielle mais la **dynamique** (**DR** pour *dynamique range*) également. Il s'agit de la plage d'intensités lumineuses que l'imageur peut capter, entre le signal maximal atteignable et le niveau de bruit dans le noir. La Figure 24 présente un

exemple d'image avec une faible dynamique et la même image captée avec une grande dynamique, où toutes les nuances sont alors visibles (dans le sombre et dans le clair).

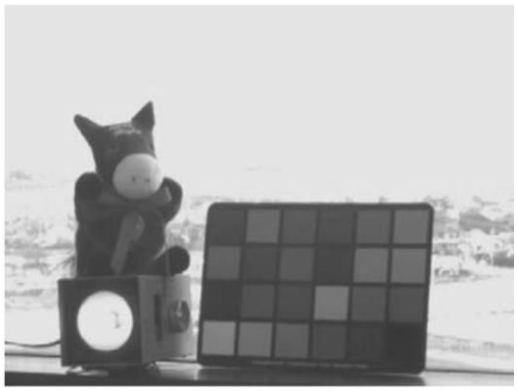
a)  b) 

Figure 24 : Exemple d'image à faible DR (a) et à grande DR : 100dB (b). [22 - Akahane 2006]

Par ailleurs, la plupart des photodétecteurs présentés en section précédente s'utilisent en mode intégration. Ils peuvent être modélisés par une source de courant (la génération de paires électron-trou) en parallèle d'une capacité (*variable, approximée par une constante*) (cf. Figure 18 au-dessus). Si l'intégration dure trop longtemps, les charges générées déchargent complètement la capacité de jonction et débordent. Ce phénomène (***overflow*** en anglais) introduit des charges supplémentaires dans les pixels voisins et peut donc fausser leur valeur (éblouissement). La quantité maximale de charges (notion de *full well capacity* en anglais) avant débordement vers les voisins représente la dynamique intrinsèque (DR) du pixel. On peut faire du HDR (*High Dynamic Range*) en améliorant cette dynamique intrinsèque par modification du circuit (voir la suite de la section) ou en utilisant du post-traitement (concaténation de plusieurs images à temps d'intégration différents ou adaptation du temps d'intégration [23 - Mann 1996][24 - Peizerat 2015] par exemple).

Bruit

Les procédés de fabrication CMOS standard sont bien maîtrisés mais des variations de process persistent. La division spatiale d'un imageur en matrice de pixels censés être identiques crée donc un bruit spatial fixe (FPN pour *fixed pattern noise)* : les pixels ont en réalité des réponses légèrement différentes les uns des autres. Sur une image, des pixels devant avoir la même couleur auraient des nuances légèrement différentes.

Filtres optiques

Devant l'imageur en silicium, des lentilles sont utilisées pour focaliser la lumière, mais il est aussi possible de mettre des filtres optiques pour sélectionner les photons à détecter. Par exemple, on peut filtrer la polarisation de la lumière [25 - Zhao 2008] ou la couleur. Sélectionner la couleur absorbée permet de faire des pixels rouges, bleus ou verts par exemple, qui, agencés en motif régulier de type Bayer [26 - Bayer 1976], permettent d'obtenir une image en couleurs au lieu de simples niveaux de gris.

Plusieurs types de pixels existent. Nous en présenterons les principaux, avec d'abord les pixels linéaires, puis les logarithmiques et enfin des linéaires par morceaux. Plus de détails peuvent être trouvés dans [27 - Fossum 1997][20 - Ohta 2007] par exemple.

1.2.2.2 Pixel passif

Le premier pixel CMOS inventé était le pixel passif (*PPS* pour *passive pixel sensor*) [14 - Weckler 1967]. Il utilise le mode intégration d'une photodiode. Comme le montre le schéma du pixel (Figure 25), il est composé simplement d'une jonction PN et d'un transistor utilisé en interrupteur. Pendant le temps d'intégration de la lumière, le transistor est ouvert ce qui permet à la photodiode de stocker ses charges photogénérées. Ensuite, pour lire la valeur du pixel, l'interrupteur est fermé, et tandis que la photodiode est remise à sa tension haute, la valeur correspondante à la quantité de lumière reçue peut être lue par le courant qui passe dans la piste. L'utilisation du mode intégration permet de lire un courant bien plus élevé que le photocourant lui-même [28 - Noble 1968] au prix d'une latence due au temps d'intégration. La lecture d'un pixel détruit l'information qu'il contenait. Puisque le pixel ne contient qu'un transistor et deux lignes (commande du transistor et sortie), son taux de remplissage est très élevé, autour de 90%.

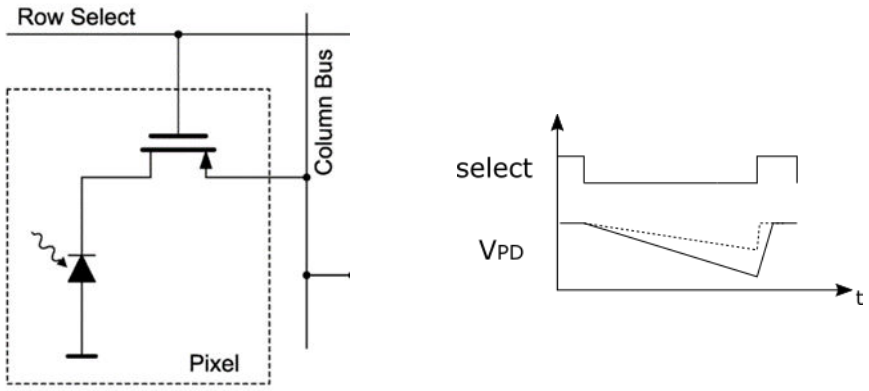


Figure 25 : Pixel passif [13 - Belbachir 2010] *et chronogramme (V_{PD} est le potentiel de cathode de la photodiode) (pointillés = lumière plus faible).*

1.2.2.3 Pixels actifs

Contrairement au pixel passif, les pixels dits actifs (*APS* pour *active pixel sensor*) possèdent au moins un transistor actif.

3T APS

Le plus simple des APS [28 - Noble 1968] est représenté en Figure 26. Connu sous le nom de 3T-APS, il est composé d'une photodiode, d'un transistor de sélection mais aussi d'un transistor branché en drain commun (appelé *SF* pour *source follower*) et d'un transistor de remise à zéro (ou *reset*). Le transistor SF est utilisé pour transmettre le potentiel de la photodiode sur la sortie du pixel lorsque le transistor de sélection est fermé. Ainsi l'information n'est pas détruite par la lecture, on peut la relire si besoin. De plus les charges sont converties en un potentiel. La remise à zéro de la photodiode ne se faisant plus directement par la lecture, le transistor dédié est rajouté (*reset*). Il remet le potentiel de cathode de la photodiode à la tension d'alimentation haute avant l'intégration suivante.

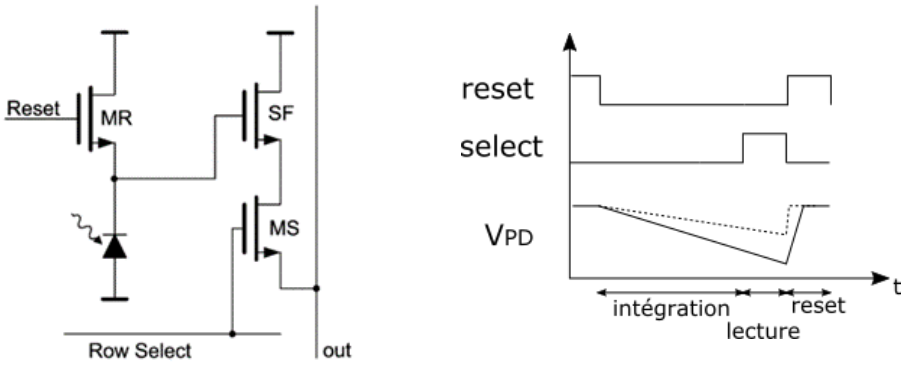


Figure 26 : Pixel 3T [13 - Belbachir 2010] *et exemple de chronogramme (pointillés = lumière plus faible).*

Ce pixel contient seulement 3 transistors et 3 lignes, son facteur de remplissage reste donc élevé même s'il est plus bas que celui du PPS. De plus, il est courant d'utiliser les APS avec seulement des transistors NMOS pour gagner de la place au dessin même si les performances sont moindres (niveau de remise à zéro plus bas par exemple). Le 3T-APS est toujours très utilisé pour sa simplicité de fabrication et de commande.

Le transistor SF donne une sortie en tension pour le pixel. En utilisant à sa place un transistor branché en source commune la sortie peut être en courant.

4T APS

Le pixel 4T ou 4T-APS fonctionne sur le même principe que le 3T mais possède un transistor de plus (Figure 27). Ce transistor (*TX* pour *transfer gate*) transfère les charges stockées dans la photodiode vers le point de diffusion flottante (FD pour *floating diffusion)*. La Figure 27 présente aussi

un chronogramme de la commande du pixel : après l'intégration de la lumière sur la photodiode la FD est remise à zéro via *reset*, le TX se ferme pour transférer les charges de la photodiode vers la FD dont le potentiel est lu par le SF et le transistor de sélection fermé. La remise à zéro de la FD et de la photodiode avant l'intégration suivante se fait via les transistors *reset* et *TX*. La capacité représentée sur la FD dans la Figure 27 permet de recevoir les charges de la photodiode, elle peut être la capacité parasite du transistor SF ou une capacité spécialement ajoutée (pour recueillir plus de charges).

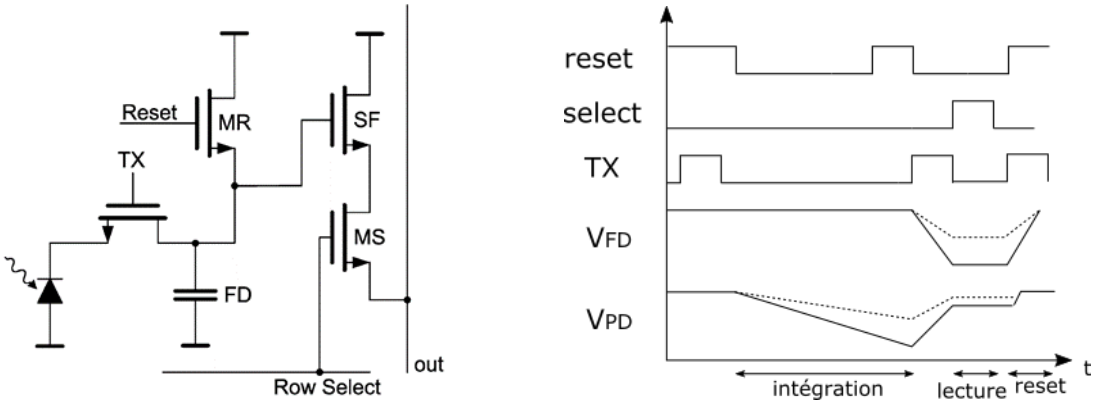


Figure 27 : Pixel 4T [13 - Belbachir 2010] *et exemple de chronogramme.*

Comparé aux pixels susmentionnés, le 4T-APS permet de conserver la valeur du pixel tout en arrêtant l'intégration, pour être lu plus tard (ce qui a des applications comme le *correlated double sampling* (CDS) ou le volet global détaillé dans la section suivante). En revanche le facteur de remplissage est nettement dégradé, surtout lorsque qu'une capacité est rajoutée. Des techniques ont été inventées pour améliorer le FF de ce pixel, comme de partager les transistors de *reset* et *SF* entre plusieurs pixels, au prix d'une commande de lecture plus compliquée [29 - Mori 2004].

La place du reset peut dépendre de l'application. Le placer sur le nœud FD permet d'assurer une valeur de reset sur la FD juste avant de transmettre les charges venant de la PD. En revanche, placer le reset sur la PD permet d'éviter l'éblouissement de la PD si on veut maintenir la valeur du pixel sur la FD pendant longtemps. Et on peut donc profiter des 2 types de reset au prix d'un transistor supplémentaire. Un tel pixel 5T peut aussi permettre d'augmenter la rapidité de l'imageur en commençant une nouvelle intégration avant la fin de la lecture de la FD [30 - Takayanagi 2007].

APS logarithmique

Les transistors présentés ci-dessus ont une réponse linéaire avec l'intensité lumineuse, mais une réponse logarithmique permettrait d'avoir une meilleure dynamique. Pour cela, un transistor est utilisé en faible inversion entre la tension de référence et la photodiode comme le présente la Figure 28. Ainsi la tension sur la grille du transistor *SF* est le logarithme du photocourant. En plus de l'augmentation de la DR, le pixel logarithmique a l'avantage de fournir une représentation de la lumière en temps réel : il n'y a pas de temps d'intégration. En revanche, la réponse non-linéaire nécessite souvent un traitement d'image particulier.

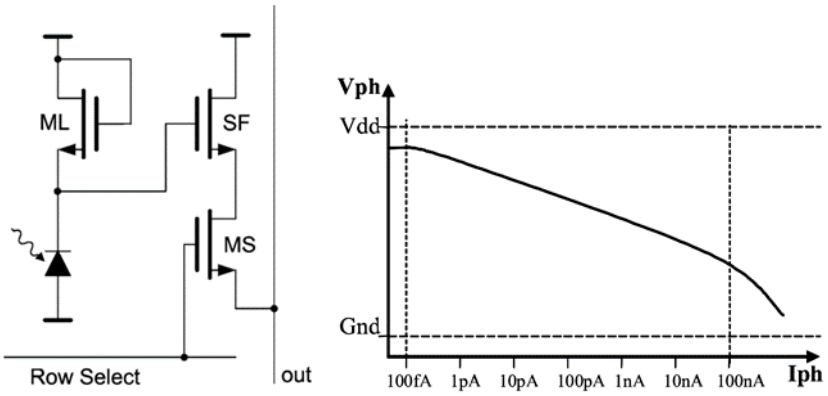


Figure 28 : Pixel logarithmique [13 - Belbachir 2010] *et sa courbe de phototransduction.* [31 - Gogniat 2011]

Le courant dans un transistor en faible inversion est très dépendant des variations de process pendant la fabrication, donc les pixels peuvent avoir des réponses très différentes les uns des autres : un fort bruit FPN dans l'imageur. Une calibration peut le limiter [32 - Kavadias 2000] mais le bruit reste élevé comparé à la faible excursion des signaux dans le transistor en faible inversion ce qui donne un faible rapport signal sur bruit (*SNR* pour *signal to noise ratio*).

Capacité de débordement pour grande dynamique

En plus des pixels logarithmiques, un autre pixel permet d'obtenir une grande dynamique : l'architecture à capacité de débordement (*overflow capacitor* [33 - Sugawa 2005][22 - Akahane 2006]) présentée en Figure 29. Une capacité (CS) et un transistor (M3) sont ajoutés à un pixel 4T. Pendant l'intégration, si la photodiode est éblouie, les charges débordent dans la capacité CS. A la lecture, on mesure les charges sur FD et sur CS, ce qui permet donc un meilleur DR tout en préservant les avantages du pixel 4T comme le maintien de l'information, au prix d'une perte en facteur de remplissage. L'idée peut être étendue à plusieurs capacités de débordement et leurs interrupteurs en parallèle [34 - Marsh 2014].

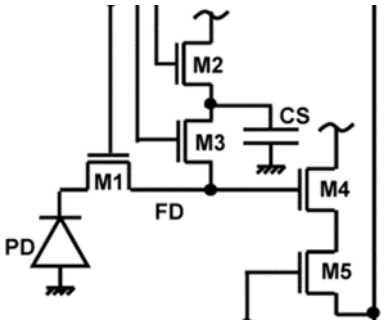


Figure 29 : Pixel 4T avec capacité de débordement. [22 - Akahane 2006]

1.2.2.4 Pixels à SPAD

Les photodétecteurs SPAD ont une sortie intrinsèquement différente de celle des photodiodes plus classiques : à cause du mode avalanche, chaque photon déclenche une impulsion. Les pixels à SPAD ont donc une architecture spéciale : il faut compter les signaux. En général, ils sont composés d'un *quenching circuit*, d'un compteur, et d'une mémoire (Figure 30). Ainsi les photons sont comptés pendant le temps d'intégration, ou le temps d'arrivée du premier photon est mesuré pour des applications de mesures de distance à la cible par exemple.

Les pixels à SPAD contiennent beaucoup de circuits autres que le photosite, lui-même grand. Ces pixels sont donc grands et de faible facteur de remplissage ([35 - Villa 2014]). Ils sont utilisés pour des applications spécifiques à faible flux lumineux.

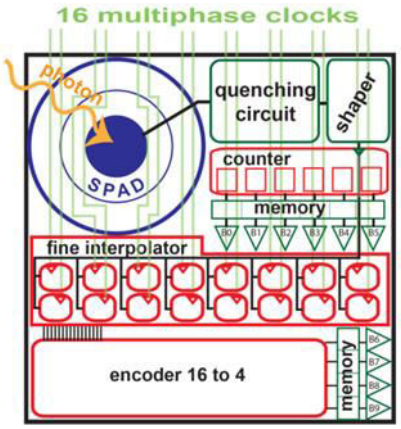


Figure 30 : Exemple de pixel à SPAD. [35 - Villa 2014]

1.2.2.5 Pixels numériques

Des architectures semblables à celles des pixels à SPAD pour des photodiodes classiques existent : les pixels numériques (DPS pour *digital pixel sensors*). Ils peuvent être appelés également pixels à modulation d'impulsions ou à domaine temporel (*time domain*, *time-based pixels*).

Les pixels à modulation d'impulsions ont un signal de sortie fait d'impulsions. Dans une modulation de largeur d'amplitude (PWM pour *pulse width modulation*), il n'y a qu'une impulsion par intégration de lumière et sa largeur est représentative de la quantité de lumière collectée pendant le temps d'intégration (Figure 31(a)). La largeur est obtenue en utilisant un comparateur qui se déclenche lorsque la tension aux bornes de la photodiode passe sous une référence. Un compteur permet ensuite de numériser la largeur de l'impulsion. Puisque le potentiel de PD décroit linéairement avec l'intensité lumineuse, le temps avant le déclenchement du comparateur est inversement proportionnel à l'intensité lumineuse reçue :

$$V_{PD} = V_{reset} - \frac{T_{int} * I_{ph}}{C_{PD}} \quad => \quad T(V_{PD} = V_{ref}) = -\frac{C_{PD}}{I_{ph}} * (V_{ref} - V_{reset}) \qquad (1.4)$$

Un compteur non-linéaire peut être utilisé pour que l'information transmise soit linéaire avec le photocourant. [36 - Wu 2008].

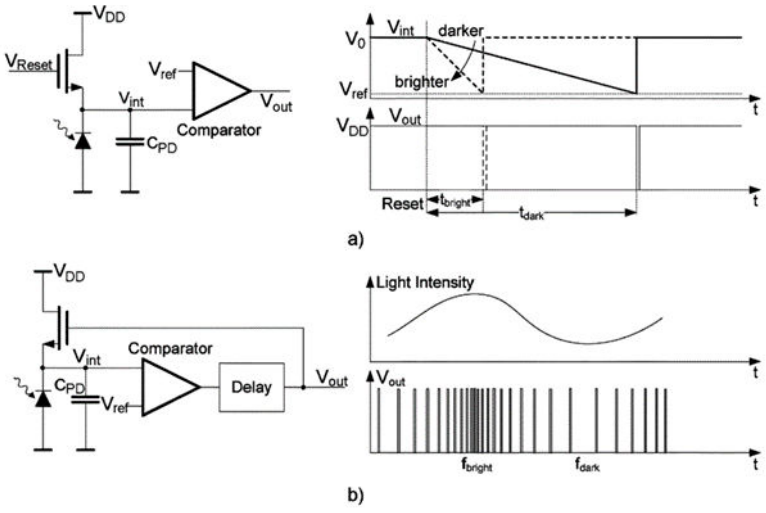


Figure 31 : Pixels (a) PWM et (b) PFM avec les principes respectifs de modulation [37 - Chen 2011].

En modulation de fréquence (PFM pour *pulse frequency modulation*), la fréquence des impulsions est représentative de l'intensité lumineuse reçue. Comme le montre la Figure 31(b), un retard et une boucle de remise à zéro sont ajoutés à un pixel PWM : lorsque le potentiel de photodiode descend sous la référence, la photodiode est remise à zéro et le signal de sortie est un pic. Plus il y a de lumière,

plus le comparateur envoie de pics. Leur fréquence est proportionnelle au photocourant, et un simple compteur permet de numériser l'information sur un temps d'intégration donné. Remettre à zéro la photodiode pendant le temps d'intégration permet d'obtenir de grandes dynamiques d'intensité lumineuse [37 - Chen 2011].

Dans un DPS, la sortie du pixel est numérique (il y a donc peu de bruit dû aux circuits de lecture de la matrice) mais toute l'électronique nécessaire à cela est présente dans le pixel, et donne un faible facteur de remplissage même si différentes implémentations ont essayé de l'améliorer (codage prédictif [38 - Zhang 2008], compromis vitesse-nombre de transistors [39 - Ng 2008],…).

En termes de consommation d'énergie, les principales dépenses d'énergie dans les DPS sont leurs comparateurs et les commutations de compteurs, dont le nombre dépend de l'intensité lumineuse incidente.

Différentes classes de pixels ont été présentées. Chacune possède différents compromis en terme de surface perdue, type de sortie destructive ou non, maintenue ou non, linéarité, excursion dynamique, bruit et facilité d'implémentation (simple ou bien connue), etc. Le Tableau 2 compare synthétiquement ces pixels. Il en ressort que parmi l'ensemble des solutions présentées, le pixel 4T semble être le meilleur concernant la versatilité applicative, ce qui peut expliquer sans doute la raison pour laquelle cette structure est l'une des plus courantes. Mais chaque type de pixel a ses applications pour lesquels il offre un meilleur compromis : par exemple les pixels à SPAD pour détecter de très faibles luminosités, le log pour une large dynamique et une lecture instantanée, le 3T et le passif pour gagner en facteur de remplissage , etc.

	FF/pitch	Non destructif	Retenue	Linéarité	Rapidité	DR	Bruit	Facilité
PPS	☺	☹	☹	☺		-	-	☺
3T APS	☺	☺	☹	☺		-	-	☺
4T APS	-	☺	☺	☺		-	-	☺
Log APS		☺		☹	☺	☺	☹	
APS à débordement	☹	☺	☺			☺	-	☺
Pixel SPAD	☹				☺	☺	☹	☹
DPS	☹	☺				-	-	☹

Tableau 2 : Comparaison qualitative de plusieurs architectures de pixel.

1.2.3 Modes de lecture

1.2.3.1 Lecture courante

Chaque pixel de la matrice fournit une représentation électrique de la lumière qu'il absorbe, et ces signaux doivent être lus pour ensuite être stockés, traités, visualisés sur un écran, etc. L'idée la plus simple peut être de lier chaque pixel par une piste à un étage de sortie hors matrice pour les lire en parallèle, mais cela demande beaucoup de bus dans la matrice. Par conséquent, les pixels sont classiquement lus ligne par ligne : une seule piste de sortie par colonne pour sortir de la matrice, ce qui limite la perte de FF. Un circuit de commande sélectionne la ligne à lire et les sorties de colonne sont lues séquentiellement une à une (Figure 32(a)) ou en parallèle (Figure 32(b)). La sélection de ligne est faite par un circuit numérique, implémenté comme un registre à décalage pour lire les lignes une à une dans l'ordre, ou comme un décodeur d'adresse pour lire n'importe quelle ligne. Dans ce deuxièmmes cas, il est aussi possible de ne lire qu'une partie de l'image (*window* ou imagettes, Figure 33), pour accéder plus rapidement aux zones intéressantes, ou de ne lire qu'une ligne ou colonne sur deux pour faire des lectures plus rapides si des images sous-échantillonnées suffisent.

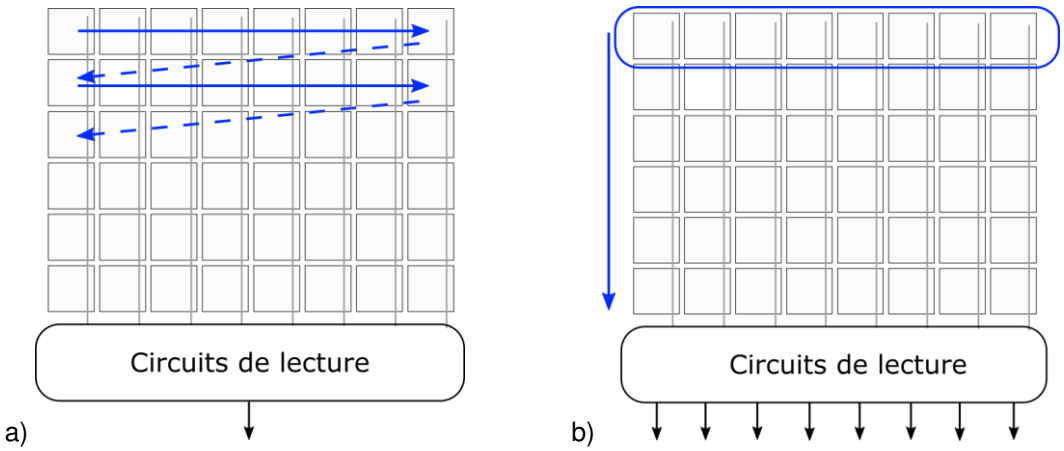


Figure 32 : Principes de lecture (a) pixel par pixel et (b) ligne par ligne.

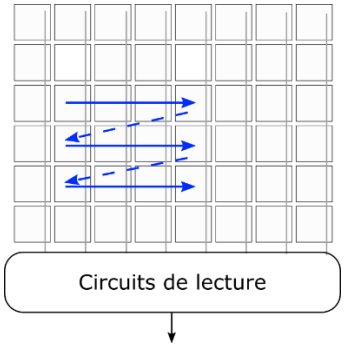


Figure 33 : Mode de lecture par imagette.

Le signal de sortie d'un pixel est donc disponible sur son bus colonne pendant sa lecture. La fonction du bloc de lecture est en général d'amplifier (si nécessaire), d'échantillonner le signal, de supprimer le bruit FPN de pixel et de fournir la sortie du système. Cette sortie du système peut être analogique ou numérique si la puce possède des convertisseurs analogique-numérique (ADC pour *analog to digital converter*). Il peut y en avoir un pour la matrice entière, ou un par colonne voire un par pixel (il s'agit de compromis entre vitesse de conversion et surface occupée).

Les imageurs du commerce étant de plus en plus grands (1920×1080 voire 3840×2160 pixels pour la télévision 4K ultra haute définition, beaucoup plus parfois pour de l'imagerie spécialisée) le temps de lecture d'un mode ligne par ligne (ou pixel par pixel) peut devenir très grand.

1.2.3.2 Volets

Dans le cas d'une lecture classique, image par image et ligne par ligne dans une image, il existe plusieurs méthodes permettant d'enchainer intégration de lumière et lecture pour chaque pixel, appelées volets (*shutter modes*).

Tout d'abord, un pixel peut être remis à zéro dès la fin de la lecture de sa valeur, et capter à nouveau de la lumière immédiatement. A cause de la lecture ligne par ligne, la scène a pu bouger entre le haut de l'image et le bas. Il s'agit d'un volet roulant (*rolling shutter*) et l'image présente de la distorsion et/ou du flou de mouvement (*motion blur*) comme l'illustre la Figure 34(gauche).

Pour avoir une image plus uniforme, il est préférable d'utiliser la méthode du volet global (*global shutter*). Tous les pixels captent la lumière en même temps et la valeur du pixel doit être stockée jusqu'à ce qu'elle soit lue (ligne après ligne). Cette méthode évite le flou de mouvement (Figure 34) mais nécessite un pixel capable de maintenir sa valeur.

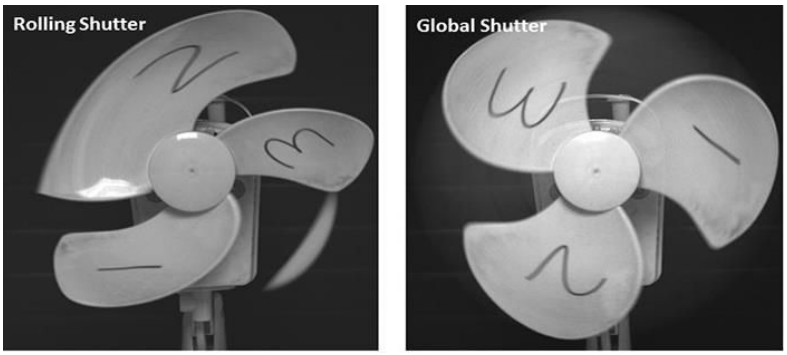


Figure 34 : Comparaison des volets roulant et global ([andor.com]).

1.2.3.3 Autres

Une autre technique de lecture se fait par évènements (AER pour *address event representation*). Une image est captée, mais seules les valeurs des pixels qui ont changé par rapport à l'image précédente sont transmises, avec l'adresse du pixel. Les changements d'intensité lumineuse détectés peuvent être synchrones ou asynchrones. L'intérêt de l'AER est double : détection du mouvement et conversion, transmission de moins de données [40 - Lichtsteiner 2008].

Il est aussi possible de lire les images et de les stocker dans la matrice avant d'en lire plusieurs d'un coup. Cela permet de prendre plusieurs images très rapprochées dans le temps : pas besoin d'attendre le temps de lecture d'une image avant d'en prendre une autre (en anglais *burst mode*). En revanche le stockage prend de la place dans la matrice [41 - Uhring 2018].

Le bloc de lecture est généralement situé en parallèle d'une ligne de la matrice, en bas de colonne : lorsqu'un pixel est sélectionné, il inscrit sa valeur sur une piste dans la colonne (bus de colonne et sa valeur peut être lue en bout de colonne par le circuit de lecture). Pour augmenter la vitesse de lecture, on peut le dupliquer en haut et en bas de la matrice : chaque circuit ne traite qu'une moitié de matrice.

1.3 Traitements dans l'imageur

Dans les systèmes embarqués, les caméras peuvent servir à la prise d'image (drones pour reportages vidéo par exemple) mais aussi très souvent à la navigation (évitement d'obstacles), ou encore à la surveillance (détection d'intrusion). Dans ces derniers cas, le flux vidéo n'a pas besoin en soi d'être envoyé au sol ; extraire l'information au plus proche du capteur semble pertinent. Or on a vu qu'un capteur d'image en technologie CMOS est composé d'une matrice de pixels et de circuits de commande et de lecture. Dans chaque pixel, le photodétecteur est associé à un peu d'électronique classique (transistors, capacités,…). Il semble donc possible d'intégrer des traitements non seulement à côté de la matrice mais aussi dans les pixels.

En terme de techniques de prétraitements, si on veut travailler dans la matrice, il convient d'éviter les tâches qui requièrent toute la matrice, comme la transformée de Fourier. Il vaut mieux des traitements locaux : pixel à pixel (des traitements dits homogènes, comme une différence temporelle, une soustraction d'arrière-plan, une correction de niveaux de gris, …) ou sur des voisinages de pixels (convolution spatiale, histogrammes locaux, filtrages médians,…) voire en pyramide si plusieurs couches de calculs sont disponibles.

Maintenant que nous avons explicité les principaux prétraitements d'images et capteurs, nous allons pouvoir détailler dans la partie suivante les différents types d'intégration de traitements d'image dans la caméra de la littérature des caméras intelligentes.

Chapitre 2 : Imageurs intelligents

Les systèmes embarqués, de plus en plus nombreux dans notre environnement, créent un besoin d'intégration des traitements au plus proche des capteurs pour optimiser la consommation d'énergie. Avant de pouvoir entrer dans la problématique qui nous concerne directement, il est nécessaire de bien appréhender la notion même de l'imageur intelligent, objet qui est au cœur de ce travail. Ce chapitre va donc s'atteler à définir « imageur intelligent » afin de poser des bases communes nécessaires à la compréhension, et à présenter l'état de l'art de ces capteurs intelligents. Ils sont variés aussi bien en termes d'architecture que de performance. La versatilité des traitements implémentés et la qualité de l'imagerie seront en particulier discutées, grâce à l'introduction de deux figures de mérite.

2.1 Imageurs intelligents : définition et notions de base

Les chaînes classiques d'acquisition, de conversion et de traitement d'image comprennent un capteur d'image et un processeur numérique de traitement bien séparés. Cela permet d'optimiser la fonction de chacun, au prix d'un important transfert de données entre les deux. Ceci est d'autant plus vrai que les résolutions d'imageurs (nombre de pixels) et les vitesses (taux d'images par seconde) augmentent constamment. La transmission de données entre imageur et processeur, très consommatrice en énergie et ressources, devient critique pour des systèmes contraints en termes de consommation et donc d'autonomie.

Les systèmes embarqués notamment (tels que les drones par exemple) ont accès à une énergie limitée, et la transmission de données complètes vers un processeur au sol n'est généralement pas la fonction première du système. Il semble donc plus intéressant d'embarquer également des processeurs auprès des capteurs, de traiter ainsi les données brutes et de ne transmettre que les informations pertinentes : il s'agit du traitement proche du capteur ou *near-sensor computing*. Dans le cas d'un imageur, les données brutes sont un flux d'images en niveaux de gris tandis que les informations pertinentes sont plutôt des informations sur l'environnement, une cible détectée, un mouvement repéré, etc. [27 - Fossum 1997].

En premier lieu, il convient de noter une différence entre imageurs intelligents (ou rétines, capteurs d'image intelligents, en anglais *smart imagers*) et caméras intelligentes (en anglais *smart cameras*) : du traitement est dans le premier cas intégré au capteur, et dans l'autre cas au niveau système. Les caméras intelligentes sont souvent composées d'un module d'acquisition couplé à une unité de traitement. L'utilisation de deux puces distinctes permet de les optimiser séparément, notamment en termes de technologie utilisée, mais il reste toujours nécessaire de transférer les données brutes d'une puce à l'autre. Si cela consomme moins d'énergie que d'envoyer les données brutes au sol, on va tout de même préférer les rétines dont la sortie n'est que de l'information déjà traitée. Cette étude va donc se concentrer sur les imageurs intelligents. Des livres traitent également du sujet : par exemple [20 - Ohta 2007] [42 - Zarandy 2011].

Ensuite, il est possible de distinguer différents types de traitements selon leur place dans la chaîne : sur les images brutes des prétraitements ou traitements bas niveau sont d'abord appliqués (présentés en chapitre 1). Cela prépare les images pour des traitements de moyen ou haut niveau. Comme on l'a vu au chapitre 1, les prétraitements permettent de réduire significativement les données à transmettre (par exemple passer d'une image en niveaux de gris à une image binaire de contours) tout en ne requérant que peu de ressources de calcul. Les traitements locaux dans la matrice de pixels semblaient en particuliers adaptés aux imageurs intelligents. La Figure 35 compare synthétiquement les architectures classiques aux imageurs intelligents.

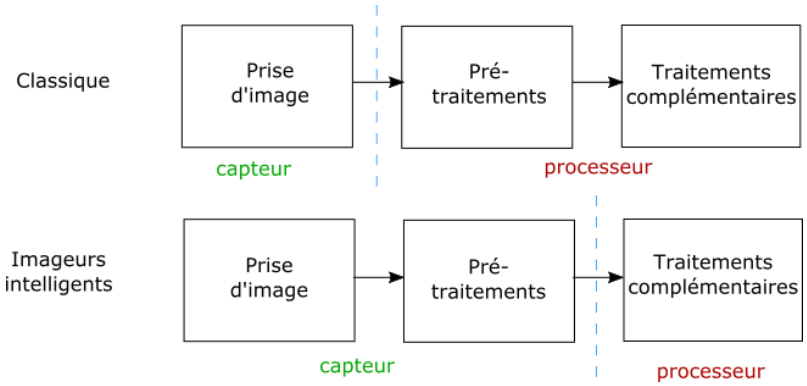


Figure 35 : Comparaison d'une chaîne de traitement classique avec celle d'un imageur intelligent.

Du fait du caractère matriciel d'une image, les traitements sont adaptés à des architectures massivement parallèles : les GPU (*Graphical Processor Unit*) sont préférés aux processeurs généraux. L'utilisation d'un FPGA (*Field Programmable Gate Array*) permet de faire du traitement d'image efficace et en temps réel, grâce à une architecture matérielle massivement parallèle et facilement programmable [43 - Bourrasset 2013]. Même en circuits dédiés, la distribution spatiale du calcul permet d'accélérer et d'améliorer l'efficacité du calcul. Il a été développé des circuits composés de processeurs répartis en matrice avec un processeur correspondant à un pixel pour des traitements locaux de bas niveau, un processeur par colonne pour des traitement de moyen niveau et un processeur global pour le haut niveau [44 - Zhang 2011], voire avec un processeur par équivalent de groupe de pixels (8x8 pixels par exemple) [45 - Yang 2016] [46 - Lindgren 2005] [47 - Wu 2018]. D'une manière générale, il a été démontré qu'une architecture hétérogène comportant une partie de type SIMD ou MIMD (*single ou multiple instruction multiple data*) est bien adaptée pour les applications en imagerie. Ces travaux s'inspirent du traitement d'image dans la nature, par la rétine des mammifères. [48 - Posch 2014] [49 - Thorpe 2010].

Les imageurs intelligents intégrant du calcul au plus près de la matrice peuvent donc aussi se classer par type d'architecture, selon la distribution du calcul, comme l'illustre la Figure 36. Le traitement peut être à côté de la matrice, distribué en bout de colonne, voire distribué dans la matrice. On appelle élément de calcul (PE pour *processing element*) chaque circuit élémentaire servant au traitement d'image. On pourra donc parler d'architecture 1 PE /pixel, 1 PE/ colonne, etc. Chaque architecture a des avantages et des inconvénients, notamment en ce qui concerne la dégradation de facteur de remplissage de l'imageur, les fonctionnalités offertes ou les problèmes de transmissions d'information. La partie suivante détaille les différentes possibilités en s'appuyant sur la littérature.

Insérer du calcul dans la matrice signifie rajouter des circuits électroniques entre les photodiodes. Cela dégrade le facteur de remplissage (FF), la taille du pixel (pitch) et/ou la taille de l'image (et donc la sensibilité et la résolution spatiale de l'imageur). Il apparait un compromis entre complexité des circuits implémentés et qualité de capteur. Les technologies 3D répondent en partie à ce problème en permettant d'implémenter localement des circuits sans empiéter sur la photodiode [50 - Carmona-Galán 2013] [51 - Gruev 2006] [52 - Suarez 2012]. En revanche ces technologies récentes impliquent d'autres problématiques telles que la dissipation thermique, la diaphonie (*crosstalk*) intercouches, le prix,… Ce travail se focalise donc sur les architectures, l'implémentation en technologie 3D étant toujours une amélioration future possible des capteurs présentés.

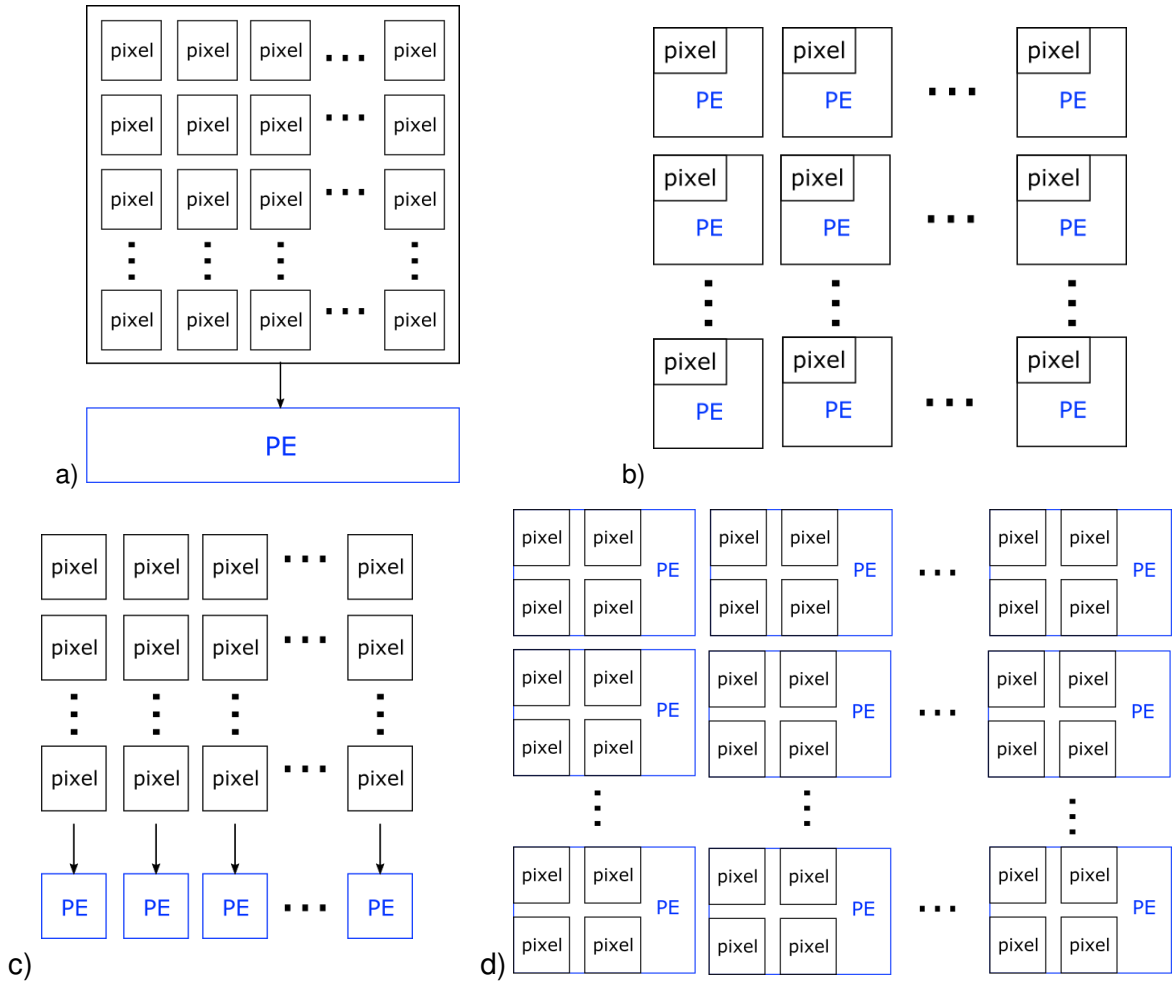


Figure 36 : Différentes possibilités d'architecture globale d'un imageur intelligent, avec un processeur par a) matrice, b) pixel, c) colonne (ou ligne) d) macropixel.

2.2 Imageurs intelligents dans la littérature

Placer le traitement sur la même puce que le capteur mais à côté de la matrice permet d'avoir un imageur aux meilleures caractéristiques (FF, résolution,…) et de minimiser la consommation d'énergie en transmission. L'équipe de R. Etienne-Cummings fait ainsi des puces contenant un imageur juxtaposé à de l'électronique de traitement qui calcule des gradients spatiaux et temporels sur toute la matrice [53 - Gruev 2002][54 - Mehta 2006]. Un circuit à base d'*application specific instruction set processor* a aussi été proposé [55 - Döge 2015] ainsi que des circuits analogiques [56 - Soell 2016] [57 - Takahashi 2010], ou encore des circuits distribués spatialement comme évoqué plus haut ([44 - Zhang 2011] [45 - Yang 2016] [46 - Lindgren 2005] [47 - Wu 2018]).

En revanche, placer le traitement à côté de la matrice nécessite un stockage temporaire de l'image ou d'une partie de l'image (buffers de 9 pixels chez [56 - Soell 2016]), même pour des traitements spatiaux, et n'est pas extensible en terme de vitesse avec l'augmentation de la taille des matrices d'imageurs. Le reste de la partie va donc s'intéresser aux imageurs intelligents intégrant du traitement au plus près des capteurs suivant leur niveau d'intégration du traitement : dans le pixel, en colonne ou encore en macropixels.

2.2.1 Traitement au niveau pixel

La distribution du calcul sous forme d'un PE/pixel (Figure 37) est très populaire. Elle permet d'implémenter de façon massivement parallèle des traitements locaux (s'étendant sur quelques pixels) dans la matrice.

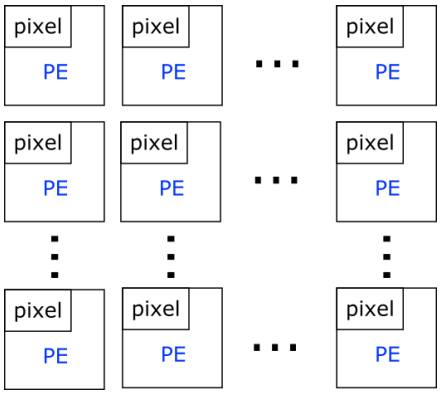


Figure 37 : Architecture 1 PE par pixel.

Comme on l'a vu au chapitre 1, la détection de contours est très utile comme prétraitement d'image. Elle est donc très largement implémentée en imageurs intelligents. Les détections de contours les plus simples sont les détections de différences de couleurs entre pixels voisins, paire par paire. ; Elle a été implémentée par exemple par différence de vitesse de chute de potentiel de la photodiode [58 - Massari 2010] [59 - Gottardi 2018].

La détection de contours peut aussi être simplement implémentée par différence entre l'image et l'image moyennée localement (équivalent à image moins image filtrée passe-bas). Ces deux étapes, moyennisation et différence, ont été implémentées respectivement par une grille commune aux pixels et par miroir de courant [60 - Ikeda 1998], par réseau de diffusion et encodage AER [61 - Costas-Santos 2007] ou encore par encodage PFM [62 - Lenero-Bardallo 2016].

La convolution spatiale par un kernel bien choisi permet de détecter des contours de façon plus complète : contours plus diffus, sens du contour… Certains circuits implémentent la convolution spatiale analogiquement [63 - Dubois 2008] ou numériquement [64 - Zhu 2014]. Chaque pixel récupère la valeur de ses 8 voisins pour calculer la combinaison linéaire de ces valeurs par les coefficients du masque, par multiplieurs ou par accumulations successives.

Pour détecter des contours à plusieurs échelles, des circuits implémentent le calcul de pyramides. Les contours sont détectés par différences d'images moyennées à différentes échelles : par bloc de 2x2 pixels, 4x4, etc. grâce à un réseau de connections par interrupteurs [65 - Katic 2014].

La direction du contour est retrouvée en utilisant deux masques (typiquement verticaux ou horizontaux), ou de façon plus complète avec des filtrages à orientation sélective [66 - Shi 2000] voire en délivrant directement la magnitude et la direction du gradient [67 - Barbaro 2002].

Au-delà des contours, des caractéristiques spatiales précises peuvent être extraites simplement d'une image par comparaison avec un modèle (de coins, de forme particulières, etc.), en chaque pixel [68 - Nishimura 2005]. Une forme peut être caractérisée par ses contours et son barycentre, extractibles dès le pixel [69 - Yin 2013]. Ce circuit a été amélioré par [70 - Yin 2016] (meilleur FF et large dynamique) dont le schéma du pixel est illustré en Figure 38.

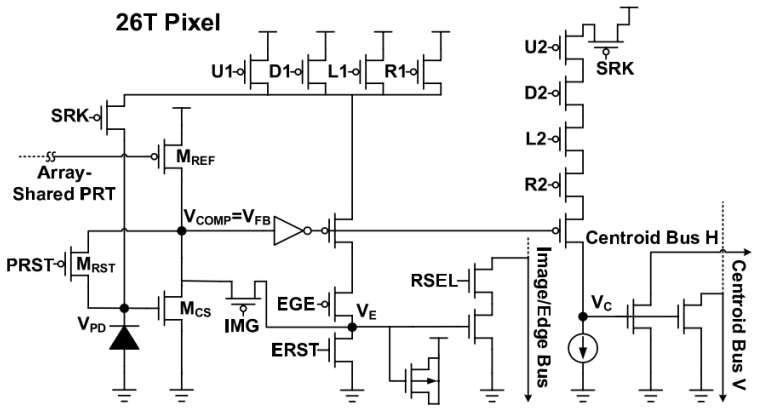


Figure 38 : Exemple de schéma de pixel avec PE pour extraire des barycentres. Il contient 26 transistors seulement et est relié à 2 voisins dans chaque direction (U,D,L,R). [70 - Yin 2016]

En ce qui concerne la détection de mouvement dans le pixel, la façon la plus simple est de soustraire à la valeur actuelle du pixel sa valeur précédente. Il s'agit donc d'avoir dans le pixel une mémoire ([71 - Simoni 1995]) et un circuit de différence. La différence temporelle peut être réalisée image par image [72 - Chi 2007] ou de façon asynchrone avec des pixels de types AER qui se signalent dès qu'un changement apparait ([73 - Huang 2017] pour un exemple récent).

De façon plus complexe, la comparaison de l'image avec un modèle d'arrière-plan permet de détecter les objets en mouvements entrés dans le champ de vision de la caméra. Pour détecter du mouvement récent, le modèle d'arrière-plan doit être mis à jour. Une façon de le coder analogiquement est d'utiliser des seuils pour chaque pixel, mis à jour par filtrage passe-bas de l'histoire récente du pixel [74 - Cottini 2013]. La Figure 39 présente le schéma du pixel et son layout. On peut voir que le facteur de remplissage est moyen (12%) ; des memristors permettraient de l'implémenter de façon compacte [75 - Olumodeji 2015].

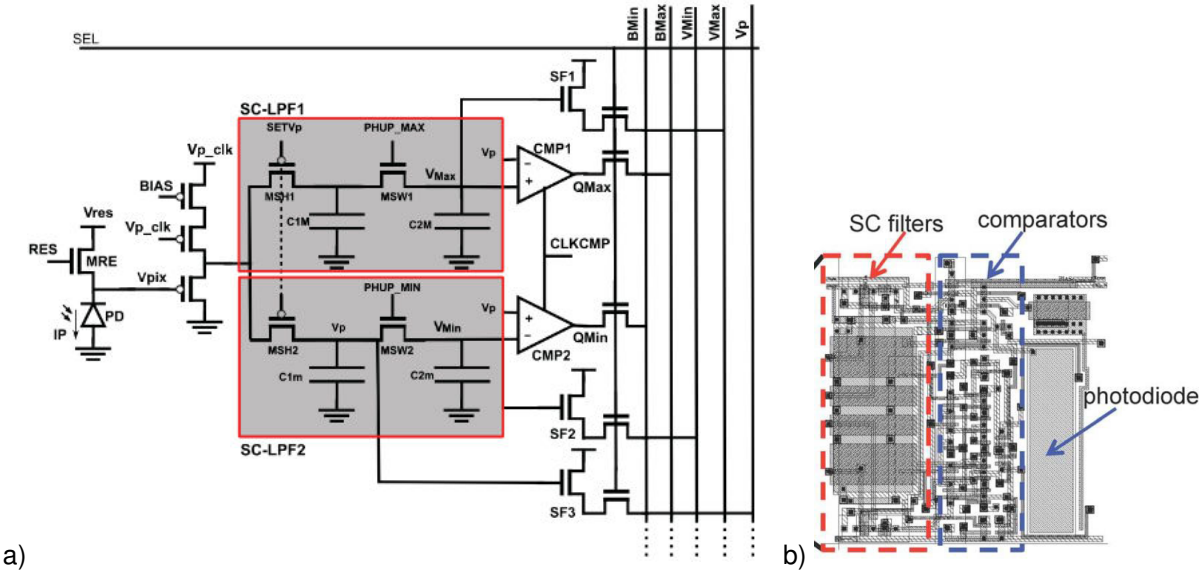


Figure 39 : (a) Schéma de pixel capable de détection de mouvement en comparant (CMP1 et CMP2) la valeur du pixel à un intervalle de son historique récent obtenu par filtrage passe-bas calculé par circuit à capacités commutées (SC-LPF1 et SC-LPF2) et (b) son layout. [74 - Cottini 2013]

Certains imageurs intelligents implémentent plusieurs traitements d'images. Kim *et al.* ont proposé un capteur capable de détecter à la fois des contours et du mouvement. Des circuits de type

winner-take-all et *loser-take-all* calculent soit une différence temporelle, soit une comparaison avec les voisins [76 - Kim 2013]. Un autre capteur offre également la différence temporelle en plus de la détection de mouvement, par kernel cette fois-ci, au moyen de circuits à capacités commutées [77 - Massari 2005] illustrés en Figure 40.

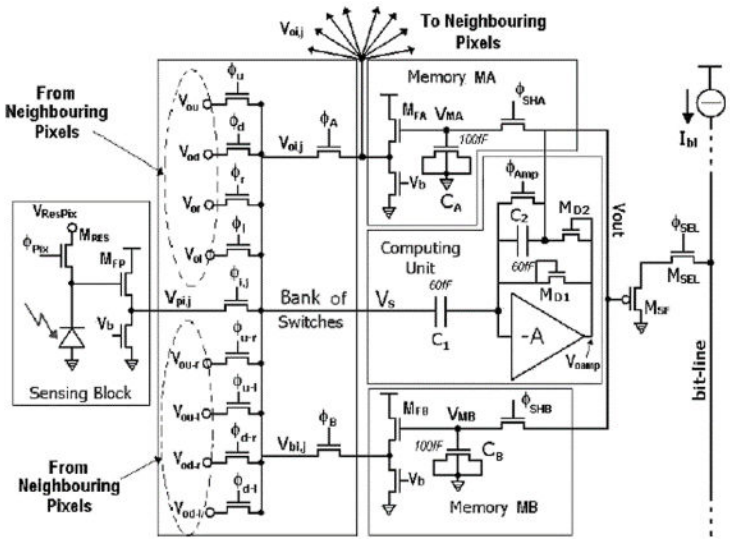


Figure 40 : Schéma de pixel avec PE à capacités commutées, lié à ses 8 voisins pour des calculs spatiaux. [77 - Massari 2005]

Fernandez-Berni *et al.* implémentent dans un même circuit (FLIP-Q) du filtrage par diffusion spatiale, le calcul de l'énergie progressive de l'image, d'une pyramide et de la détection de caractéristiques rectangulaires (un prétraitement de l'algorithme de Viola Jones pour la détection de visages) [78 - Fernández-Berni 2011] [79 - Fernandez-Berni 2012]. Une variante de cette puce implémente deux réseaux de photodiodes pour améliorer la dynamique de l'image tout en proposant le calcul de l'intégrale de l'image pour faciliter l'algorithme de Viola Jones d'une autre manière [80 - Carmona-Galan 2015] [81 - Fernández-Berni 2013]. Le schéma et le layout de ce pixel sont présentés en Figure 41. Plusieurs fonctions sont implémentées mais le FF est faible (5.4%).

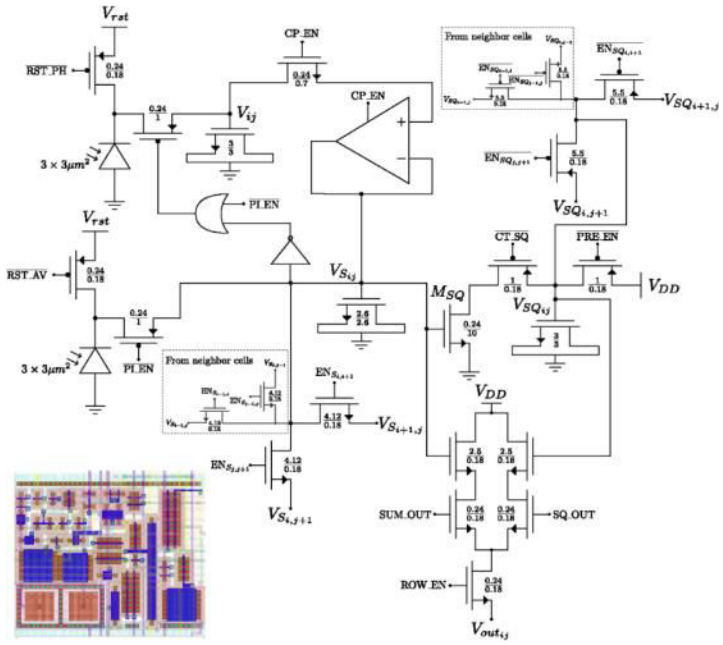


Figure 41 : Schéma et layout d'un pixel avec son PE de [80 - Carmona-Galan 2015] *(PDs en bas à gauche du layout).*

Pour économiser de la surface et de l'énergie, les circuits analogiques sont souvent préférés aux circuits numériques pour des traitements implémentés dans le pixel. Mais le numérique a l'avantage

de proposer une versatilité plus importante et a donc été choisi pour différentes puces : MIPA4k [82 - Poikonen 2009], SCAMP5 [83 - Carey 2013], ASPA [84 - Lopich 2008]. Les PEs sont alors de véritables mini-processeurs, très versatiles mais très grands. Les facteurs de remplissage sont de quelques % seulement. En analogique, un équivalent peut être trouvé dans ACE16k [85 - Rodríguez-Vázquez 2004].

On peut noter aussi la possibilité d'entrelacer différents types de pixels. Pour prendre une image en niveaux de gris, des pixels 3T-APS sont entrelacés avec des pixels plus gros, mais capables de faire de l'AER pour de la détection de mouvement : chacun est optimisé pour sa tâche [86 - Clapp 2002].

Les algorithmes de prétraitement adaptés à l'embarqué tels que la détection de contours ou de mouvement ont donc été implémentés de diverses manières en architecture 1PE par pixel, mais toujours au détriment de la qualité de l'imageur lui-même, en terme de FF ou de densité de pixels.

2.2.2 Traitement au niveau colonne

Pour alléger les contraintes de surface des circuits, des traitements ont été décalés à l'extérieur de la matrice, en bout de colonne (Figure 42).

Tout d'abord, il existe généralement un circuit de lecture d'un imageur (amplification, réduction du bruit par CDS (*Correlated Double Sampling*), conversion analogique numérique, etc.) par colonne pour augmenter la vitesse par rapport à un circuit par matrice entière, tout en facilitant le dessin des pistes dans la matrice. Des imageurs intelligents utilisent ces circuits pour faire du traitement d'image : utiliser le CDS (qui fait normalement la différence entre une valeur de pixel et sa valeur de remise à zéro) pour une différence temporelle [87 - Massari 2007], ou pour de la détection de contours simples [88 - Tabet 2001], ou l'ADC pour de la différence temporelle encore [89 - Mizuno 2003].

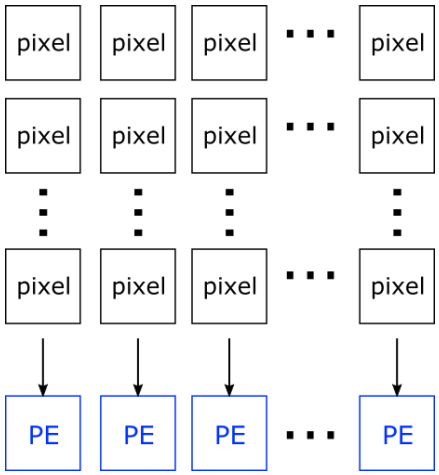


Figure 42 : Architecture 1 PE / colonne.

Certaines puces comportent de véritables circuits de traitement en bout de chaque colonne ou ligne (Figure 43) : traitement d'images binaires (érosion, dilatation, etc.) [90 - Hong 2002], convolution spatiale à coefficients programmables par multiplieurs et accumulateurs en colonne [91 - Basset 2007] ou encore par accumulation en capacités commutées [92 - Dupret 2002], [93 - Angotzi 2008]. Ce type de processeurs en colonne permet d'avoir un imageur de bonne qualité en terme de surface dédiée à la phototransduction, mais n'offre qu'une dimension de parallèlisation : les colonnes. Plus le nombre de lignes de l'imageur augmente, plus le processeur doit travailler vite ou le taux d'image par seconde diminuera [94 - Elouardi 2008]. Le traitement ligne par ligne nécessite aussi la mémorisation des lignes précédemment lues ou calculées, à côté ou dans la matrice [93 - Angotzi 2008].

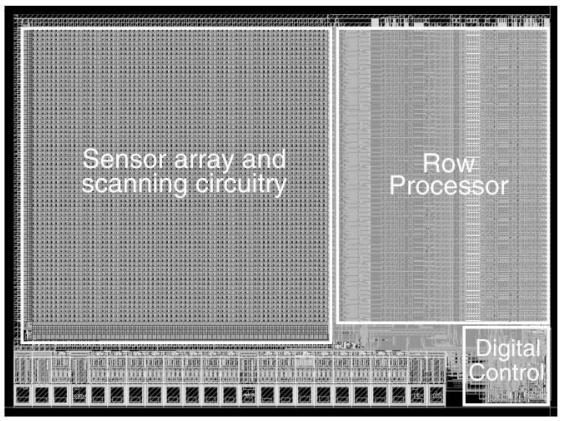


Figure 43 : Exemple de layout d'une puce avec une matrice de pixels et des processeurs en colonnes (row processors). [93 - Angotzi 2008]

Pour tirer parti à la fois de la distribution massivement parallèle du calcul en pixels et du gain de place du calcul en colonne, il est possible de séparer le traitement en plusieurs étapes [95 - Cho 2014]. Pour éviter de répéter un comparateur dans chaque pixel, il peut être placé une seule fois à l'extérieur de la matrice [96 - Zhao 2012] au prix de la vitesse, ou en colonne [97 - Lee 2015], tout en conservant une partie du circuit de traitement dans le pixel. Dans [98 - Benetti 2013], les auteurs améliorent les performances de [74 - Cottini 2013] en transférant la détection du déclenchement d'un pixel du pixel vers la colonne. D'autres puces placent la majorité du traitement dans le pixel (par exemple extraction de contours simple par comparaison avec les voisins) tout en utilisant les colonnes pour encoder le résultat [99 - Gottardi 2009], voire ajouter une caractéristique simple comme l'accélération de la lecture en ignorant les lignes sans contour [100 - Cottini 2011] ou simplement tirer parti du *CDS* [87 - Massari 2007].

2.2.3 Macropixels

Au-delà des répartitions 1 PE par pixel et 1 PE par colonne, voire un mélange des deux pour limiter la perte en surface, les architectures à 1 PE par macropixel se développent. On entend ici par macropixel un groupe de pixels, ou voisinage de pixels. Une architecture par macropixels met donc un PE par groupe de pixels (Figure 44). L'utilisation de macropixels permet d'avoir une architecture massivement parallèle, extensible dans les 2 dimensions de l'imageur, tout en limitant l'impact sur la surface photosensible.

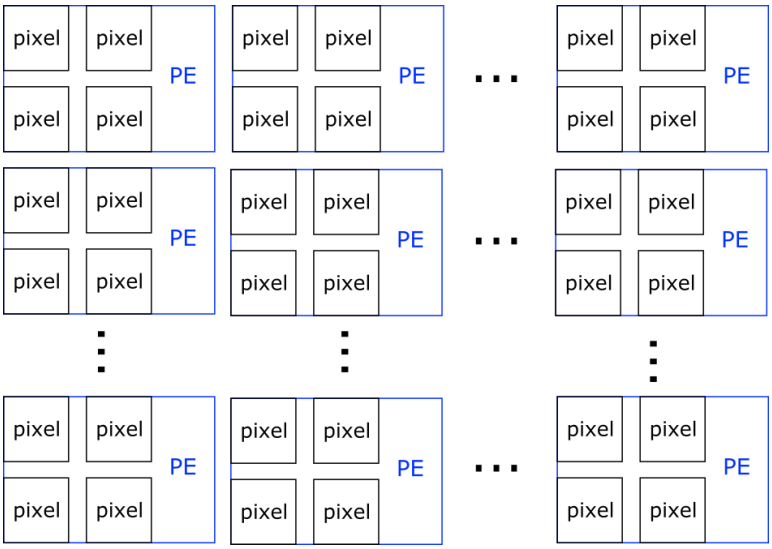


Figure 44 : Architecture 1 PE / macropixel.

Une architecture en macropixels peut être implémentée de plusieurs manières. Tout d'abord certains traitements d'image ne nécessitent l'information que par groupes de pixels et non en chaque pixel. Le circuit ne fait alors qu'un calcul par groupe de pixels. Par exemple, la moyenne de 4 pixels peut être calculée localement sur un macropixel 2x2 et être utilisée ailleurs pour obtenir une grande dynamique ou réduire le nombre de données transmises [101 - Sicard 2014]. Un macropixel peut aussi accueillir plus de circuits de calculs, pour réaliser jusqu'à la détection de mouvement dans la matrice : 1 PE calcule une détection de mouvement sur la moyenne de 4 pixels [102 - Liu 2014].

Ensuite, il est possible d'avoir un PE par macropixel et de réallouer au cours du temps les ressources partagées. Un circuit proposé utilise des groupes de deux pixels pour faire de la détection de mouvement : un pixel sert de mémoire afin d'être comparé à son voisin pour calculer une différence temporelle, avant d'échanger les rôles. Il n'y a qu'un comparateur pour deux pixels, comme l'indique la Figure 45 [103 - Choi 2014].

L'implémentation d'une pyramide bénéficie des deux types d'architecture en macropixel : pour calculer la plus haute résolution, les pixels utilisent tour à tour les ressources du PE (1 PE pour 4 pixels par exemple, soit 4 calculs par PE), tandis que pour les résolutions plus faibles chaque PE ne travaille qu'une fois ou pas du tout (1 calcul par PE pour le second étage, puis 1 calcul par 1 PE sur 4 pour le troisième étage de la pyramide, etc.) [104 - Suarez 2017]. Le schéma et le layout sont illustrés en Figure 46.

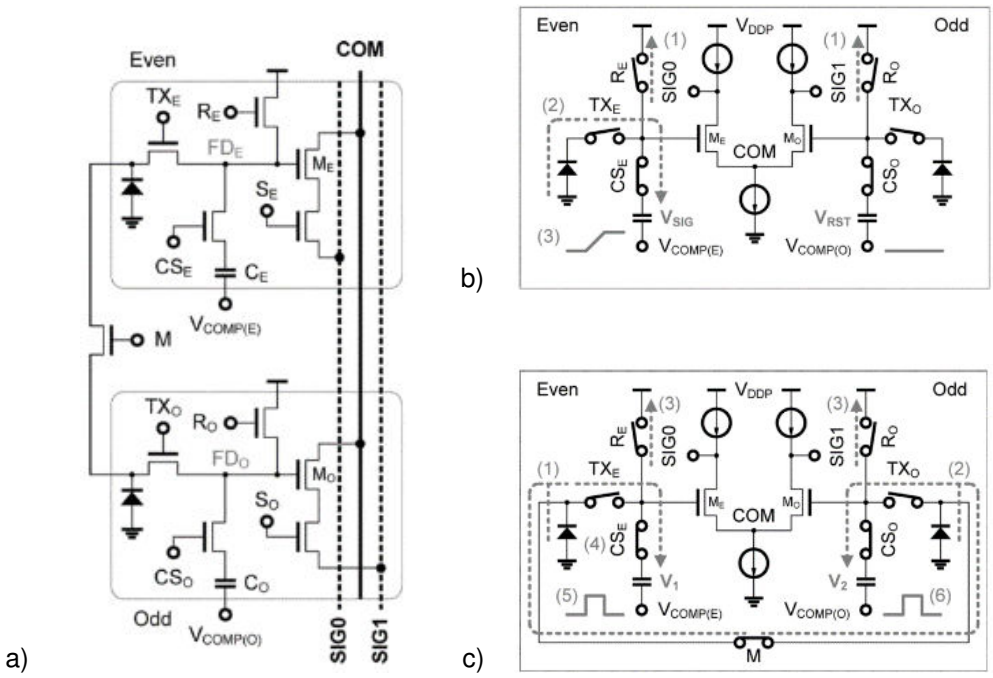


Figure 45 : Schéma d'un double pixel (macropixel 2x1) partageant un comparateur dont les branchements permettent de modifier l'application (a) schéma général, (b) branchements pour basse consommation et (c) branchements pour différence temporelle. [103 - Choi 2014]

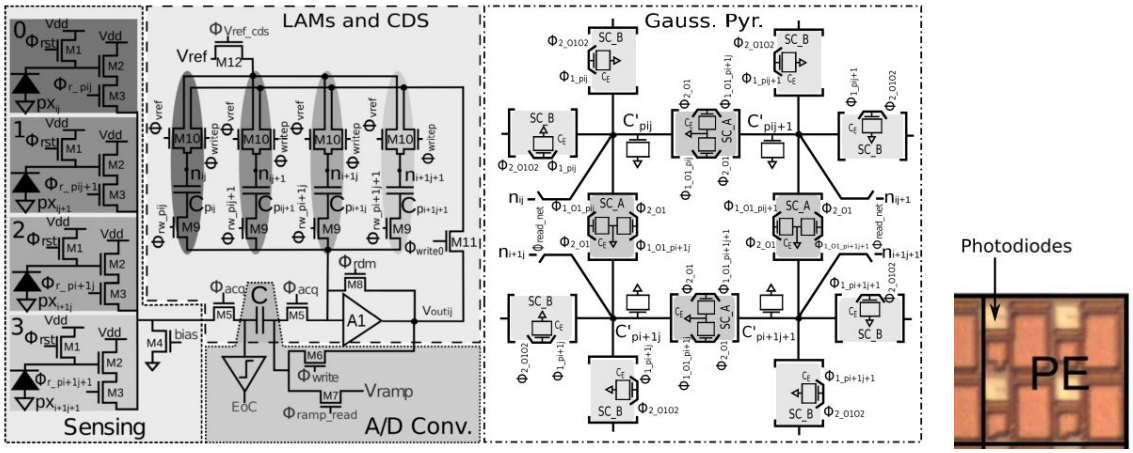


Figure 46 : Schéma et layout d'un macropixel 2x2 utilisé pour de la génération de pyramide gaussienne. [104 - Suarez 2017]

Un macropixel peut-être plus virtuel : Martel *et al.* utilisent la puce SCAMP5 [83 - Carey 2013] pour implémenter des algorithmes demandant plus de mémoire que ce qui est disponible physiquement par pixel : certains pixels sont éteints dynamiquement pour allouer leurs ressources à leurs voisins [105 - Martel 2015]. Ici aussi la résolution est diminuée lorsque des pixels sont inutilisés.

En numérique, Schmitz *et al.* ont proposé récemment une architecture en macropixel 8x8. Ce circuit possède un cœur de calcul complet avec suffisamment de mémoire par groupe de 8x8 pixels DPS, capables de dialoguer entre eux. Le regroupement par macropixel leurs permet d'implémenter de nombreux traitements tout en limitant l'impact sur la surface photosensible contrairement aux puces à traitements similaires par pixel (SCAMP5, ASPA, etc. cf. plus haut) [106 - Schmitz 2017] (Figure 47).

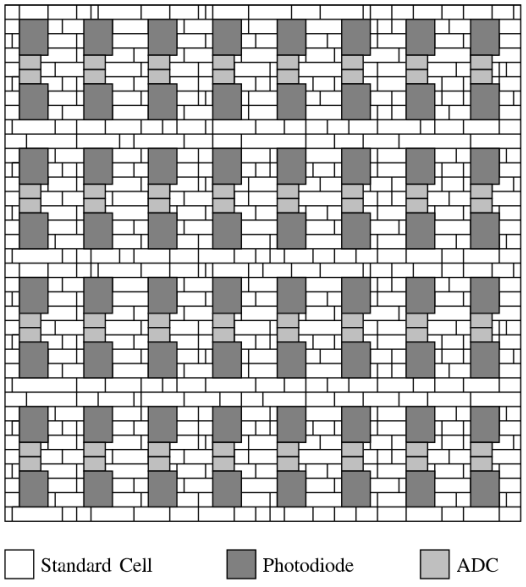


Figure 47 : Représentation spatiale (floorplan) d'un macropixel de 8x8 pixels avec 1 PE numérique de [106 - Schmitz 2017].

2.3 Compromis surface/versatilité

Dans le contexte des systèmes de vision embarqués, les imageurs intelligents permettent de réduire la transmission de données. Cependant, l'avantage d'un flux vidéo brut transmis est la possibilité

offerte d'extraire autant d'informations que souhaité par l'unité de calcul extérieure. Il est donc important qu'un imageur intelligent intègre un maximum de fonctions différentes. Or comme on l'a vu, la plupart des imageurs intelligents se concentrent sur un traitement ou deux (souvent une détection de contours simples ou une différence temporelle), notamment les architectures 1 PE par pixel, pour limiter la perte en surface photosensible. Les puces ayant pour PE des processeurs numériques sont des exceptions notables, qui offrent une large versatilité, mais avec une trop faible surface photosensible.

Pour quantifier la surface photosensible perdue, la littérature utilise le facteur de remplissage (FF, décrit au chapitre 1) : le ratio de la surface du photorécepteur sur la surface totale du pixel. Plus il y a de circuit dans le pixel, plus le FF est faible et plus la captation de la lumière est mauvaise. Cependant, cette métrique ne prend pas en compte la technologie utilisée : la plupart des circuits utilisent des transistors à taille minimale, qui dépend de la technologie. Plus la technologie est récente et petite, plus on peut mettre de circuits dans un pixel à FF donné. En outre, la taille du pixel (ou pitch) est aussi importante pour la qualité de l'image (finesse d'échantillonnage spatial) et influence les circuits implémentables. Plus le pixel est grand, plus on peut mettre de circuits dans le pixel à FF donné, mais plus on dégrade l'image.

2.3.1 Des figures de mérite adaptées

La plupart des publications mentionnent ces trois éléments (FF, technologie, taille du pixel) afin de pouvoir être comparées justement. Pour plus de clarté de comparaison, cette section propose une figure de mérite les prenant tous les trois en compte, le $FoM_{surface}$. Défini par l'équation 2.1, il varie linéairement avec le FF, est inversement proportionnel à la taille du pixel (ou pitch, mais on utilise la racine carrée de l'aire pour les pixels non carrés) et dépend logarithmiquement de la technologie.

$$FoM_{surface} = \frac{FF * \log_{10}(2 + techno)}{\sqrt{A_{pixel}}} \qquad (2.1)$$

avec *techno* le nœud de la technologie exprimé en µm (0.18 par exemple) et

A_{pixel} l'aire du pixel exprimée en µm².

Ce sont bien ces différentes évolutions que l'on retrouve sur la Figure 48, qui représente des courbes obtenues à partir de l'équation 2.1 (l'évolution du $FoM_{surface}$ en fonction de chaque paramètre, pour plusieurs valeurs fixes des autres paramètres). Les courbes ayant en abscisse la technologie sont beaucoup plus plates que les autres grâce au logarithme, ce qui traduit l'influence moindre de la technologie par rapport au facteur de remplissage par exemple. D'autre part, plus la taille du pixel est petite, plus l'effort d'intégration est remarquable, ce qui se note par l'augmentation drastique du $FoM_{surface}$.

La technologie étant exprimée par la valeur du nœud technologique, en µm, un entier est ajouté pour avoir un argument de logarithme supérieur à 1. La valeur 2 est choisie parce qu'elle me semble plus à même de représenter l'influence de la technologie sur la qualité d'une architecture (voir les allures respectives des courbes avec la valeur 1 et avec la valeur 2 sur la Figure 48 d).$FoM_{surface} =$

$$FoM_{surface} = \frac{FF * \log_{10}(2 + techno)}{\sqrt{A_{pixel}}} \qquad (2.1)$$

Cette figure de mérite semble donc être plus représentative de la qualité d'une architecture de circuits dans la matrice, plus détachée d'une implémentation dans un pixel donné, dans une technologie donnée.

En ce qui concerne la versatilité d'un imageur intelligent, une formule ne semble pas pouvoir la représenter. Un indice de versatilité est donc proposé. Chaque fonctionnalité de l'imageur ajoute des points à l'indice en fonction de sa complexité, selon les poids définis dans le Tableau 3.

Par exemple, un imageur qui ne fait que de la différence temporelle sera noté 1. S'il fait de la différence temporelle et une détection de type comparaison à ses voisins, il sera noté 2 (1+1). S'il est capable de faire une convolution spatiale à coefficients programmables, il sera noté 3 (il peut faire une convolution programmable, et donc par extension à kernel fixe, et aussi une détection de contours

simple). Pour les puces intégrant des unités de calcul numérique, un forfait de 8 points (7 points pour l'ASPA qui se dit moins versatile) leur est attribué puisque faire l'inventaire de leurs possibilités semble compliqué.

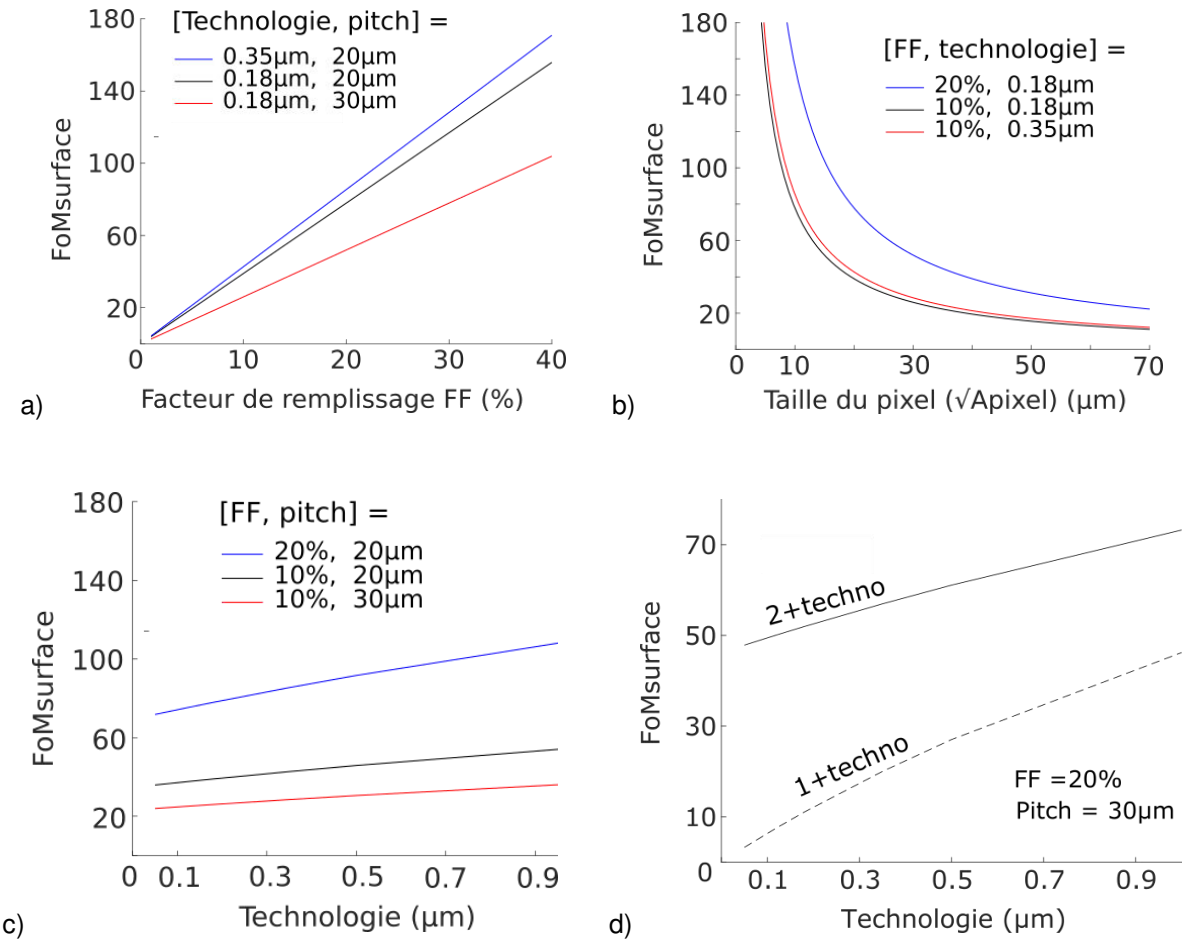


Figure 48 : Evolutions du FoMsurface en fonction de chacun de ses paramètres : (a) en fonction du FF, à technologie et pitch contstants, (b) en fonction du pitch, à technologie et FF constant, (c) en fonction de la technologie, à pitch et FF constants et (d) en fonction de la technologie, pour deux valeurs de constante additive (1 et 2), à pitch et FF constants.

Traitement	indice
Détection de contours basique (comparaison aux voisins)	+1
Détection de contours (convolutions à kernel fixe)	+1
Différence temporelle	+1
Soustraction d'arrière-plan	+1
Filtrage passe bas (diffusion)	+1
Filtrage spatial programmable	+1
Extraction de coins, etc.	+1
Extraction de barycentres	+1
Pyramide	+1
Pyramide Gaussienne (avec filtrage)	+1
Image intégrale (somme progressive)	+1
Mémorisation (indice par image mémorisée)	+0.5
Image normale	+0.5

Tableau 3 : Correspondance fonctionnalité de l'imageur - points ajoutés à son indice de versatilité.

2.3.2 Comparaison des imageurs intelligents

Ces deux figures de mérite permettent de placer les imageurs intelligents décrits plus haut sur un plan $FoM_{surface}$ en fonction de l'indice de versatilité comme le montre la Figure 49. Les puces à processeur numérique sont représentées par un carré, et les autres par une croix. Les couleurs indiquent le type de distribution du calcul. Une version détaillée de cette carte d'état de l'art est présentée en Figure 50 avec les noms de premiers auteurs et année de publication, pour resituer chaque travail présenté plus haut.

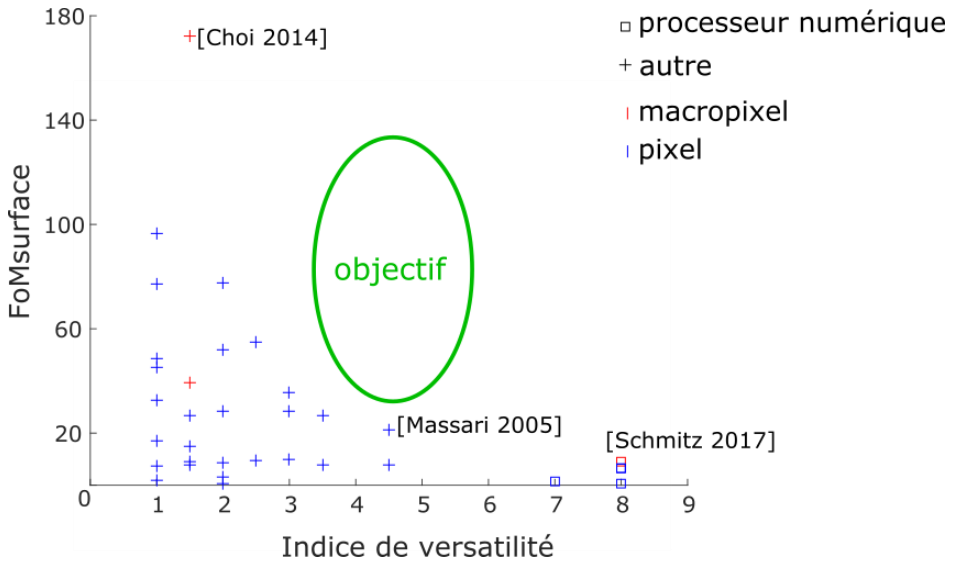


Figure 49 : Classement des imageurs intelligents selon leur FoMsurface en fonction de leur indice de versatilité fourni par les circuits implémentés dans la matrice (et pas en colonne).

En plus des distributions en plusieurs niveaux, les macropixels (en rouge sur la Figure 49) semblent donc effectivement pouvoir apporter des solutions de plus faibles surfaces occupées que les autres circuits à versatilité similaire.

La Figure 49 montre également la difficulté d'obtenir un bon compromis versatilité/surface. Le travail décrit dans la suite de ce mémoire a consisté à chercher une solution permettant d'améliorer ce compromis. Les architectures par macropixels étant récentes, peu explorées et prometteuses comme

le montre cet état de l'art, avec de bonnes possibilités de compromis versatilité/surface et une architecture massivement parallèle dans les deux dimensions, c'est la voie que ce travail a exploré.

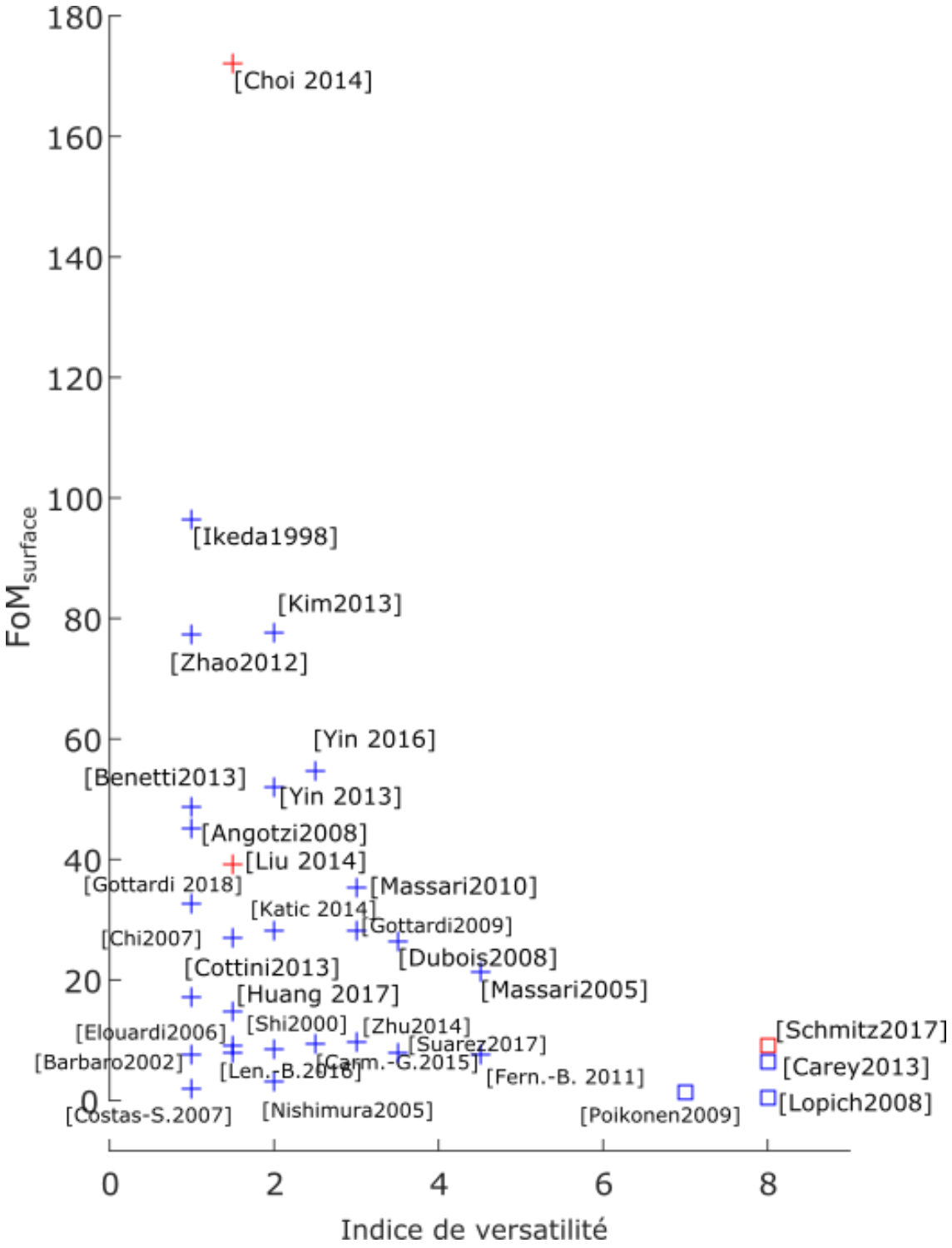


Figure 50 : Version détaillée de la Figure 49.

Chapitre 3 : Exploration architecturale

Dans le contexte choisi - les systèmes embarqués - l'énergie est très limitée, et il est donc souhaitable comme on l'a vu d'implémenter des capteurs intelligents qui extraient les informations pertinentes avant transmission. On a vu que la littérature sur les imageurs intelligents est assez conséquente mais manque d'un bon compromis surface occupée / versatilité des traitements. En effet, plusieurs prétraitements d'images peuvent être intéressants pour les systèmes embarqués, mais il reste essentiel de conserver une surface importante dédiée à la détection de lumière.

Dans ce chapitre, nous souhaitons mettre en avant l'architecture globale du système proposé, c'est-à-dire poser les grands principes algorithmiques et architecturaux qui seront implémentés dans ce travail. Cette exploration architecturale a été faite à haut niveau puis validée par comparaison à des approches plus traditionnelles.

Parmi les traitements d'image, on a vu que ceux qui sont facilement intégrables dans la matrice de pixels sont des traitements locaux, puisqu'il est préférable de limiter les interconnexions. Nous cherchons donc à implémenter efficacement dans la matrice du filtrage spatial (implémenter un opérateur différentiel spatial pour de la détection de contours, du moyennage, etc.) et du filtrage temporel (pour de la détection de mouvement). Ces traitements sont, comme on l'a vu, parmi les plus utiles pour les systèmes de vision embarqués. Afin d'améliorer le compromis entre versatilité et surface occupée, nous co-adapterons des algorithmes existants à des architectures se prêtant à des implémentations efficaces dans une matrice de pixels : il s'agit d'adéquation algorithme-architecture (AAA) [31 - Gogniat 2011].

3.1 Filtrage spatial

Comme on l'a vu au chapitre 1, le filtrage spatial est très utilisé en traitement d'image. Il consiste à convoluer une image avec un masque (Figure 51). Or une convolution implique un calcul centré sur chaque pixel, en utilisant ses voisins : une combinaison linéaire des pixels. Une implémentation matérielle directe est d'insérer dans chaque pixel l'électronique capable de faire ce calcul. Comme on l'a vu au chapitre précédent, cela a été fait de plusieurs manières mais chaque fois au détriment du facteur de remplissage de la matrice de pixels. Pour remédier à cette dégradation, on s'intéresse au principe d'AAA.

En effet, répéter les circuits électroniques dans chaque pixel prend de la place, tandis que l'approche par macropixels permet de globaliser l'électronique pour plusieurs pixels [106 - Schmitz 2017]. Cependant, implémenter une convolution spatiale en groupant les ressources pour plusieurs pixels complexifie la tâche : cela nécessite beaucoup d'interconnexions et de temps de calcul. Il s'agit donc plutôt de co-adapter des algorithmes existants et l'architecture matérielle.

3.1.1 Convolution sous-échantillonnée

La convolution sous-échantillonnée (CSE) décrite en Figure 52 semble ainsi proposer une solution au problème. Le calcul de combinaison linéaire n'est pas fait en chaque pixel (comme dans le cas classique, Figure 51), mais seulement une fois sur neuf. On déplace le masque en le juxtaposant, il ne se recouvre jamais. Chaque pixel intervient une seule fois dans le calcul de convolution sous-échantillonnée d'une image par un masque donné. On peut noter que le résultat est identique à faire une convolution classique puis en sous-échantillonner le résultat. En revanche, ici on ne sous-échantillonne pas a posteriori : on ne fait pas tous les calculs.

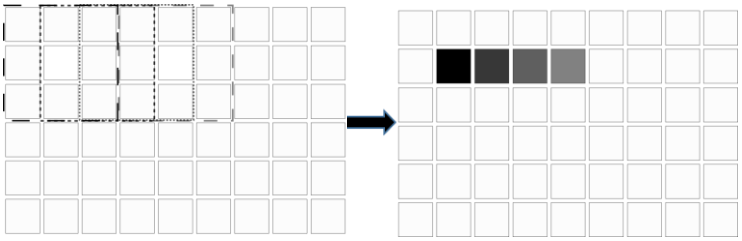


Figure 51 : Principe de la convolution classique.

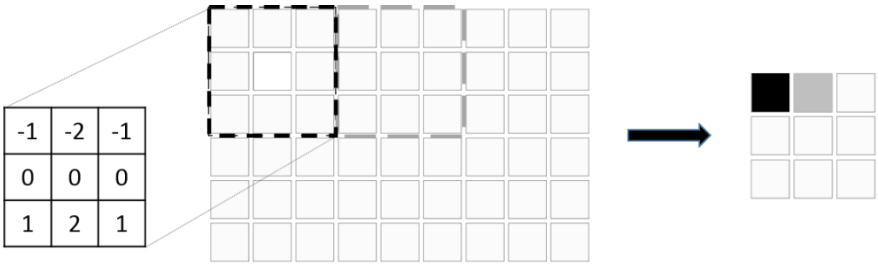


Figure 52 : principe de la convolution sous-échantillonnée (masque de Sobel en exemple).

Cette CSE permet de n'implémenter qu'un circuit de calcul (PE) par groupe de pixels ou macropixel. Le calcul de combinaison linéaire peut être fait en parallèle sur tous les macropixels en même temps comme le montre le chronogramme de la Figure 53.

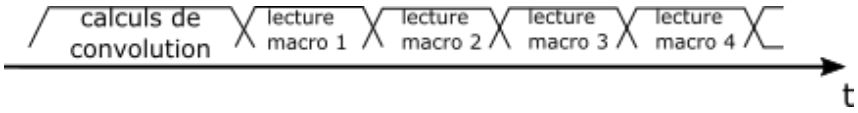
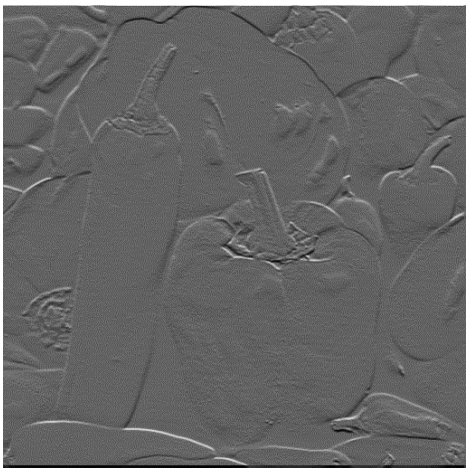


Figure 53 : Chronogramme schématique d'un filtrage spatial avec une lecture série.

Par rapport à une convolution classique, en plus de la facilité d'implémentation dans la matrice, il y a moins de calcul, moins d'informations à transmettre puisqu'il n'y a qu'une seule donnée par macropixel, moins de temps de lecture puisqu'on ne lit que les macropixels et pas tous les pixels.

A titre d'exemple, la Figure 54 présente le résultat d'une convolution classique d'une image par le masque de Sobel horizontal et celui d'une convolution sous-échantillonnée. Il y a une légère perte d'information. Pour compléter cette observation, le résultat du calcul de la magnitude du gradient par les masques de Sobel horizontaux et verticaux est présenté en Figure 55, pour une convolution complète comme pour une CSE. Pour cette dernière, les images ont donc 9 fois moins de pixels, mais sont grossies ici (on peut voir le grain de résolution) pour pouvoir bien comparer les contours entre les deux types de convolution. On observe donc que les contours sont qualitativement conservés, même par sous-échantillonnage. Même sans filtrage, peu d'aberrations sont visibles. Dans l'image du cameraman, le contour d'un bâtiment est presque perdu puisqu'il est parfaitement vertical et tombe sur une zone enlevée par échantillonnage. Mais la plupart des contours ne sont pas parfaitement verticaux ou horizontaux, et une caméra embarquée a une forte probabilité de bouger et donc de décaler ce contour sur une zone conservée dans l'échantillonnage. Les résultats par différents masques de détection de contours ne sont pas représentés car ils sont très similaires entre eux à l'œil, l'impact de la CSE est donc le même. Ces observations montrent bien qu'un critère plus objectif est nécessaire pour valider la CSE.

a) 

b) 

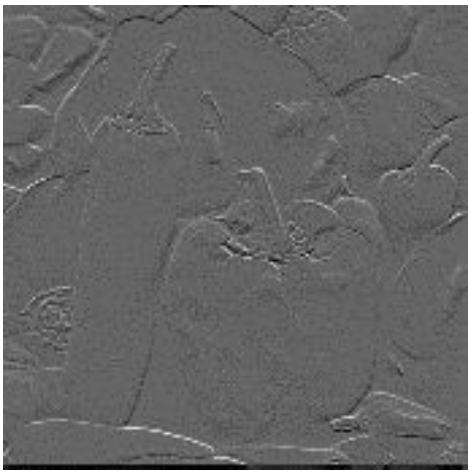
c) 

Figure 54 : Image originale (a) et résultats de convolution par le masque de Sobel horizontal (b) classique (512x512 pixels) et (c) sous-échantillonnée (170x170 pixels).

Figure 55 : Comparaison qualitative des différents prétraitements de contours : (a,b) images originales, (c,d) images traitées par l'algorithme de Sobel classique, (e,f) par Sobel sous-échantillonné. Toutes les images résultant de sous-échantillonnage ont 9 fois moins de pixels mais sont présentées ici à la même taille pour comparer les contours.

3.1.2 Critère d'évaluation du sous-échantillonnage : la détection de piétons

Une convolution sous-échantillonnée permet de réduire la surface dédiée au calcul, mais il faut vérifier que l'efficacité du prétraitement n'est pas dégradée. Pour cela on définit un critère objectif à partir d'un algorithme défini : la détection de piétons. Sur ce cas particulier, on veut garder le même taux de détection de piétons en utilisant en prétraitement une convolution classique ou une convolution sous-

échantillonnée. En effet, la détection de piétons peut se faire en calculant les contours d'une images, puis en extrayant l'histogramme des gradients orientés (HOG) et en utilisant une reconnaissance par *machine learning* [10 - Dalal 2005]. L'algorithme HOG et l'apprentissage automatique sont des algorithmes de plus haut niveau qu'on ne cherche pas à implémenter dans la matrice de pixels mais qu'on cherche à utiliser comme critère de validation de notre convolution sous-échantillonnée (cf. Figure 56). Il existe des algorithmes bien plus performants aujourd'hui [107 - Brunetti 2018] mais ici une solution simple, robuste et reconnue est préférée.

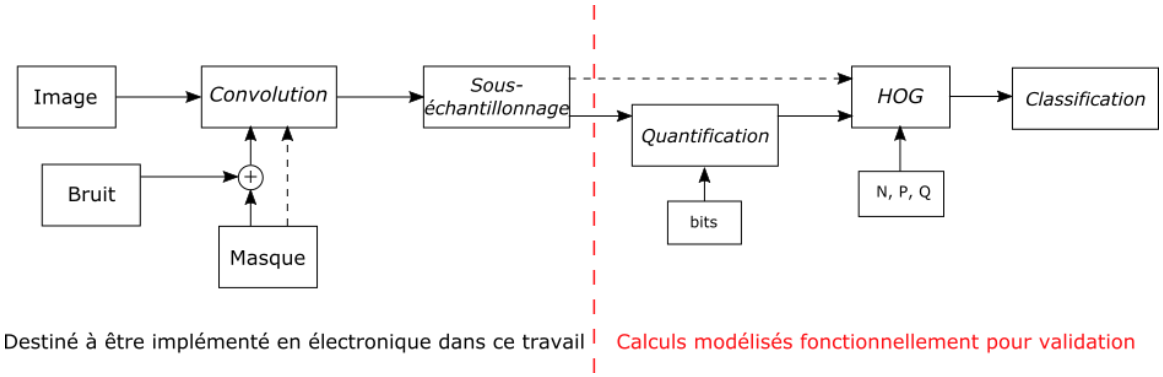


Figure 56 : Schéma bilan de l'utilisation du critère de validation.

On souhaite valider la convolution sous-échantillonnée par une étude fonctionnelle, et si possible spécifier le circuit correspondant. Pour cela, on utilise un logiciel de calcul numérique (Matlab en l'occurrence), dans lequel on modélise :

- la convolution de la matrice de pixels avec un masque,

- l'erreur due au calcul en électronique et au convertisseur,

- le calcul d'histogramme de gradients et enfin

- l'apprentissage automatique.

La suite de cette section présente donc l'environnement de tests pour valider la CSE.

3.1.2.1 Convolution

Tout d'abord, on extrait les contours de chaque image, par convolution spatiale classique ou par CSE. Pour chaque image, on calcule le gradient de niveaux de gris par convolution de l'image avec un ou des masques (ou kernels). La plupart calculent le gradient en coordonnées cartésiennes, on a donc un masque pour les gradients horizontaux et un autre pour le gradient vertical comme l'indique le Tableau 4.

L'algorithme HOG utilisant des gradients orientés, on peut aussi utiliser directement des masques qui donnent le gradient en coordonnées polaires [108 - Maggiani 2015] :

$$\begin{bmatrix} 0 & \sin(\theta_k) & 0 \\ -\cos(\theta_k) & 0 & \cos(\theta_k) \\ 0 & -\sin(\theta_k) & 0 \end{bmatrix}$$

pour θ_k dans {0°; 20°; 40°; 60°; 80°; 100°;120°; 140°; 160°}. Cela évite un traitement supplémentaire de changement de coordonnées, mais impose un plus grand nombre de convolutions pour une seule image (9 contre 2) ce qui ralentit le calcul.

Dans la simulation fonctionnelle, pour chacun de ces kernels, la convolution peut être classique (fonction *conv2* de MATLAB) ou sous-échantillonnée.

Simple	$\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$ et $[1 \quad 0 \quad -1]$
Sobel (3x3)	$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$ et $\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$
Diagonal	$\begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}$ et $\begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{bmatrix}$
Sobel (5x5)	$\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 0 & 0 & 0 & 0 & 0 \\ -4 & -16 & -26 & -16 & -4 \\ -1 & -4 & -7 & -4 & -1 \end{bmatrix}$ et $équivalent\ vertical$

Tableau 4 : Présentation des différents masques utilisées pour de la détection de contours par convolution spatiale.

3.1.2.2 Prise en compte des imperfections de l'électronique de traitement

L'électronique qui calculera la combinaison linéaire introduira une erreur dans le résultat de convolution. Elle est modélisée dans les simulations fonctionnelles par une erreur sur les coefficients du masque, selon l'équation :

$$masque = masque + M * 2 * (rand(k,l) - 0.5) * masque \qquad (3.1)$$

où - *masque* est le kernel de taille (*k*,*l*),

- *rand* est une fonction Matlab retournant une matrice remplie de valeurs pseudo-aléatoires tirées selon une distribution uniforme dans l'intervalle]0 ; 1[, et

- *M* est la fraction maximale du coefficient rajoutée en erreur.

Les coefficients nuls restent à zéro ce qui peut correspondre à la réalité du circuit (absence de calcul, donc d'ajout d'erreur dans le cas d'un coefficient nul). Le masque erroné est le même pour toute une image, ce qui peut correspondre à un circuit intrinsèquement erroné mais pas à des désappariements entre les circuits placés en parallèle. Pour limiter l'impact des tirages de nombres pseudo-aléatoires, plusieurs essais successifs de détections de piétons complètes sont réalisés (15 par exemple) dont les résultats sont ensuite moyennés.

A cette étape, la simulation fonctionnelle a simulé l'équivalent de la sortie de l'imageur intelligent étudié.

Si on choisit de faire le prétraitement en électronique analogique avant des traitements numériques, il peut être intéressant de voir l'impact de la quantification du signal par un ADC. On offre donc dans le code la possibilité de choisir une erreur de quantification. On peut choisir la résolution de l'ADC (*nbMarches*), ainsi que sa plage d'entrée (*inRange*). Le résultat $sortie_{ADC}$ est donc une valeur quantifiée.

$$sortie_{ADC} = \frac{floor\ (sortie_{matrice} * nbMarches/inRange)}{nbMarches/inRange} \qquad (3.2)$$

3.1.2.3 Calcul de l'histogramme des gradients orientés

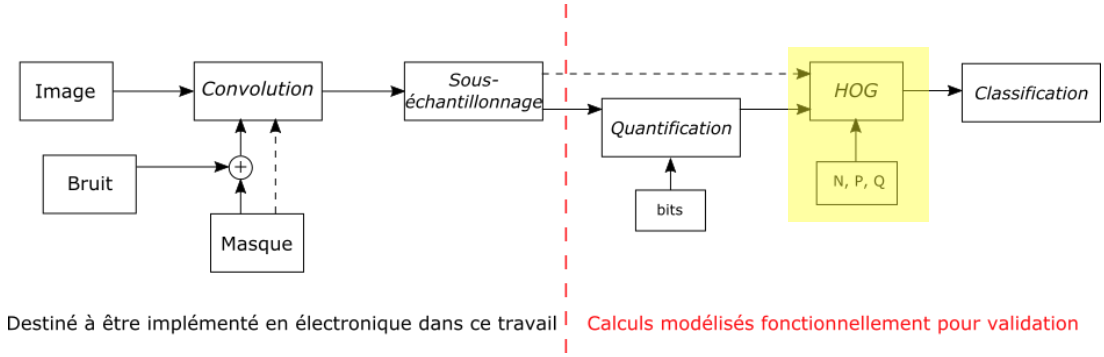


Figure 57 : Rappel de la chaine de traitement.

Grâce à ces valeurs de sortie, qui représentent le gradient de gris en chaque point (pixel ou groupe de pixels) on calcule l'histogramme des gradients orientés (HOG) (cf. Figure 57). Pour cela, on passe le gradient cartésien (si obtenu par masque de type [1 0 -1] ou Sobel par exemple) en gradient polaire en utilisant la fonction *car2pol* de Matlab.

Les points sont ensuite regroupés en cellules de taille *N*x*N* points, dans laquelle l'histogramme des gradients est calculé. La taille *N* doit correspondre environ à la largeur d'un membre de la personne type à détecter [10 - Dalal 2005]. L'histogramme est calculé pour *nBins* classes d'angles entre 0° et 180° (*nBins* = 9 selon [10 - Dalal 2005]). Les cellules sont ensuite regroupées en blocs sur lesquels on normalise l'histogramme. Les blocs contiennent *P*x*P* cellules, et se recouvrent de *Q* cellules (en général, *P* = 2 et *Q* = *P*/2, chaque cellule est normalisée sur 4 blocs, voir Figure 58). On a finalement autant d'histogrammes que le nombre de cellules multiplié par le nombre de blocs auxquels appartient une cellule. Ces histogrammes (Figure 59) mis bout à bout forment le descripteur de l'image qui est utilisé ensuite par l'apprentissage automatique.

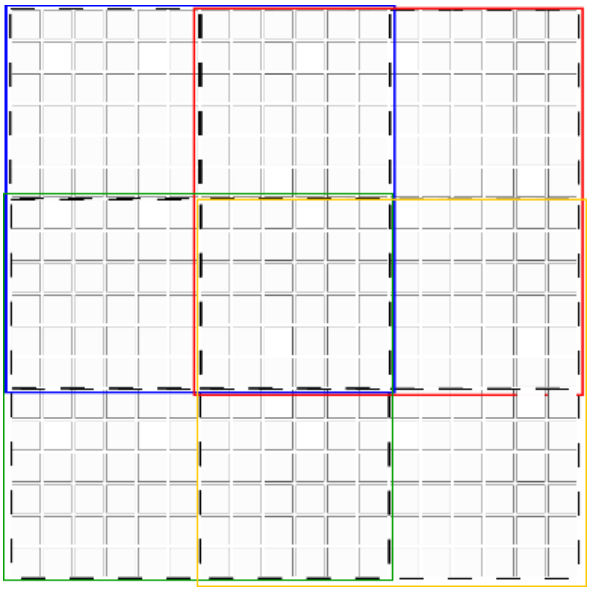


Figure 58 : Représentation schématique du découpage de l'image en cellules (pointillés noirs) et en blocs (couleurs) (N = 6, P = 2 et Q = P/2)

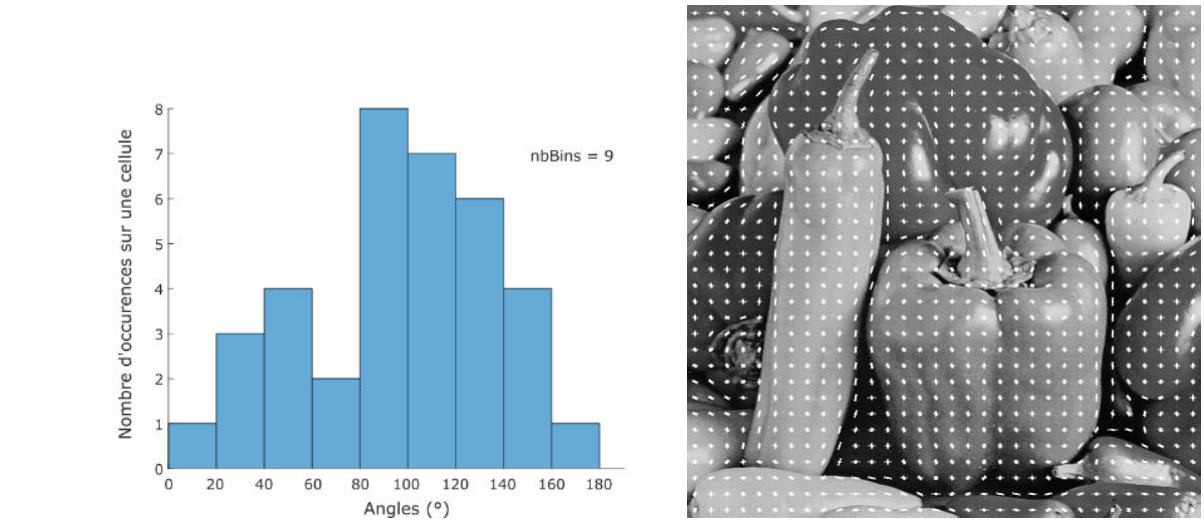


Figure 59 : Exemple d'histogramme sur une cellule (N = 6 donc 6x6 = 36 valeurs de gradient classées selon leur angle dans nBins = 9 classes) et exemple de résultats de l'histogramme des gradients orientés représenté en champ de vecteurs, superposé à l'image originale.

Les valeurs numériques (taille des cellules, taille des blocs, taille de la superposition des blocs, nombre de classes) ont été choisies selon les recommandations de Dalal *et al*. En les faisant varier un peu on n'observe pas de changement radical, et l'optimisation de cet algorithme ne nous intéresse pas ici.

3.1.2.4 Apprentissage automatique

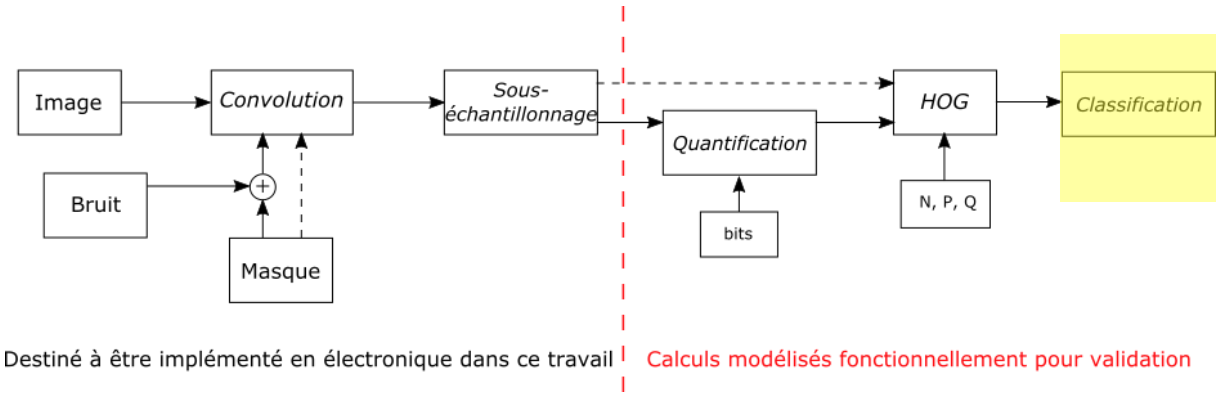


Figure 60 : Rappel de la chaine de traitement.

Le vecteur descriptif d'une image est ensuite utilisé comme argument d'un algorithme d'apprentissage automatique (ou *machine learning* en anglais) : le SVM (*Support Vector Machine* ou machine à vecteurs de support) (cf. Figure 60 pour rappel). Il s'agit de l'algorithme recommandé par [10 - Dalal 2005] pour le HOG. On utilise les fonctions fournies par Matlab comme une boîte noire, puisqu'il ne s'agit pas ici d'étudier, ni d'optimiser l'apprentissage automatique. La phase d'apprentissage est faite par la fonction *fitcsm* qui prend en argument les vecteurs descriptifs d'images et leur classe (présence d'un piéton ou non) pour entrainer la machine. La phase de test est faite par la fonction *predict* qui prend en argument le vecteur descriptif d'une image et renvoie sa classe, que l'on compare ensuite à la réalité.

On utilise des images issues de la base de données de l'INRIA [109 - Dalal], dont quelques exemples sont donnés en Figure 61. L'apprentissage est fait sur 600 images positives, i.e. avec un piéton, et 600 négatives, i.e. sans piéton. Le test est fait sur 100 images positives et 200 négatives. Augmenter le nombre d'images d'entrainement n'améliore pas significativement les résultats, et le temps de calcul étant déjà relativement long, on garde ce nombre d'images.

Figure 61 : Exemples d'images positives (haut) et négatives (bas) extraits de la base de données utilisée [109 - Dalal]. *Les images négatives ont été découpées à partir d'images plus grandes, afin d'avoir le même nombre de pixels que sur les images positives.*

Il convient de noter que nous ne nous intéressons pas à l'optimisation de cet algorithme de haut niveau. Ni les paramètres du HOG (taille des cellules, etc.) ni ceux de l'apprentissage automatique (type, nombre d'images d'entrainement et de test, etc.) n'ont été amplement étudiés. Les résultats obtenus sont loin de l'état de l'art en matière de détection de piétons [107 - Brunetti 2018] mais ce qui nous intéresse ici est seulement de comparer des prétraitements d'images. On applique donc exactement le même algorithme de test (HOG et apprentissage automatique) sur des images obtenues par différentes détection de contours (convolution classique, sous-échantillonnée, etc.) afin de comparer la qualité de l'information transmise par ces prétraitements.

3.1.3 Résultats de l'étude fonctionnelle

Une détection de piéton par le HOG, critère défini ci-dessus a donc été réalisée. Le Tableau 5 présente les résultats obtenus pour ces différents prétraitements d'image : différents masques et les deux types de convolution. De plus le cas « Simple - pixel large » désigne l'utilisation du HOG avec un masque simple en convolution complète mais calculée sur des pixels correspondant chacun à la moyenne de 3x3 pixels. Cela revient à prendre un imageur avec la même densité de circuit électronique qu'un cas CSE par masques 3x3 pixels, mais en utilisant une architecture 1 PE par pixel.

Un faux positif est une erreur de l'algorithme d'apprentissage automatique : il détecte un piéton alors qu'il n'y en avait pas sur l'image. De même, un faux négatif est l'absence de détection de piéton alors qu'il y en avait un. Dans le Tableau 5, la modélisation est faite à bruit nul, sans quantification et avec N=6, P=2, Q=1 sauf mention du contraire.

Vu le nombre restreint d'entraînements et les paramètres non-optimisés des algorithmes de HOG et d'apprentissage automatique, on se limite ici à comparer les ordres de grandeur des résultats. De ce point de vue, tous les prétraitements sont comparables : moins de 15% de faux positifs et moins de 10% de faux négatifs, avec respectivement 0% et 6% dans le cas idéal, le HOG classique avec le masque simple. On observe dans le Tableau 5 que certains traitements donnent en revanche nettement plus de faux positifs que d'autres. Ceci n'est pas forcément critique pour une application telle que la détection de piétons par un véhicule autonome : arrêter le véhicule alors qu'il n'y a pas de piéton (du fait d'un faux positif) est moins dommageable que faire avancer le véhicule alors qu'un piéton traverse...

	Type de masque et convolution pour HOG						
	Simple	Simple CSE (N=2)	Sobel	Sobel CSE (N=2)	Polaire	Polaire CSE (N=2)	Simple - pixel large (N=2)
Faux positifs (%)	0	14 (2)	0	14 (2)	0	4 (5)	13 (3)
Faux négatifs (%)	6	8 (8.5)	5.5	7.5 (7)	5	8.5 (6.5)	10 (6)

Tableau 5 : Résultats de l'algorithme de détection de piéton (HOG et SVM) selon différentes détections de contours, sans bruit ni quantification.

Pour le cas du pixel large, on voit que le résultat est également comparable. En revanche, en utilisant la CSE, on peut avoir accès à une image de plus haute résolution pour d'autres traitements ou de l'imagerie classique.

Dans les colonnes correspondant à des traitements sous-échantillonnés, deux résultats sont donnés selon la valeur du paramètre N, la taille des cellules dans le calcul de l'algorithme HOG. En effet, Dalal *et al.* expliquent que ce paramètre dépend de la taille des membres d'une personne sur les images considérées [10 - Dalal 2005]. Ainsi lorsqu'on sous-échantillonne les images par 3x3 pixels il faudrait également diviser ce paramètre par 3. *N* étant à 6 pour une convolution classique sur ces images, on voit effectivement dans le Tableau 5 que les cas de CSE avec $N = 2$ sont légèrement meilleurs que les cas CSE avec $N = 6$. Cela montre que les algorithmes de plus haut niveau doivent aussi être légèrement adaptés en fonction des couches plus basses : l'AAA doit se faire à plusieurs niveaux.

Le Tableau 5 ne tient pas compte du fait que les sorties des circuits de prétraitement seront inévitablement entachées d'erreur et peut-être quantifiées par un ADC. L'influence de ces non-idéalités est présentée aux Tableaux 6 et 7. Le Tableau 6 présente les résultats lorsque le signal de sortie de la matrice passe par un ADC avant d'être traité par l'algorithme HOG. Il montre que la quantification n'est pas gênante même lorsqu'elle est relativement grossière avec par exemple un ADC de 8 bits seulement.

	Type de masque et convolution pour HOG				
	Simple	Simple CSE	Simple CSE avec ADC		
			12 bits	8 bits	5 bits
Faux positifs (%)	0	2	1	4	5
Faux négatifs (%)	6	8.5	8	7	11.5

Tableau 6 : Résultats de l'algorithme de détection de piéton (HOG et SVM) pour le masque simple avec différentes quantifications (sans bruit, N = 2 pour les cas de CSE).

	Type de masque et convolution pour HOG							
	Simple	Simple CSE	Simple avec erreur			Simple CSE avec erreur		
			M = 0.05	M = 0.1	M = 0.3	M = 0.05	M = 0.1	M = 0.3
Faux positifs (%)	0	2	1.7	2	3.8	7.2	8.8	15.13
Faux négatifs (%)	6	8.5	6.8	6.3	8.1	10.6	11.3	13.2

Tableau 7 : Résultats de l'algorithme de détection de piéton (HOG et SVM) pour le masque simple avec différentes erreurs (sans quantification, N = 2 pour les cas de CSE).

Le Tableau 7 montre qu'une erreur conséquente sur les coefficients d'une convolution, sous-échantillonnée ou pas, dégrade peu les résultats du HOG avec apprentissage automatique. La CSE est certes plus sensible à l'erreur qu'une convolution classique, mais une CSE avec un masque simple qui a jusqu'à 10% d'erreur sur ses coefficients par exemple conserve des taux d'erreur de détection de piétons relativement proche du cas idéal pour les faux négatifs. L'erreur est plus importante pour les faux positifs, mais comme discuté précédemment cela a moins d'impact sur l'application. Cela nous conduit à implémenter le PE pour du calcul approximé (ou *approximate computing* en anglais). C'est-à-dire que le circuit sera très peu contraint en termes de précision du résultat exigé.

Au vu de ces résultats, et en rappelant que l'optimisation de l'algorithme de détection de piétons n'est pas l'objectif de ce travail, nous concluons qu'une approche par convolution sous-échantillonnée pour imager des contours est adéquate pour des systèmes de vision embarqués. De plus une implémentation électronique numérique ou analogique est possible puisque la contrainte de précision n'est pas grande (Tableau 7).

On propose donc pour le filtrage spatial d'utiliser un macropixel avec un circuit capable de calculer une combinaison linéaire de valeurs de ses pixels. Les macropixels sont fixés à la taille 3x3 pixels car les masques les plus utilisés sont de cette taille (cf. chapitre 1).

3.2 Filtrage temporel

De la même manière que pour le filtrage spatial, il faut implémenter une détection de mouvement le plus efficacement possible dans la matrice de pixels. Or la détection de mouvement la plus simple est la différence temporelle : on calcule la différence entre deux images successives d'un flux d'images. Cela permet de détecter du mouvement lorsque la caméra est fixe, dans le cas d'une caméra de surveillance par exemple. Il s'agit d'un traitement local, au niveau du pixel. On souhaiterait l'implémenter avec la même idée de regroupement des ressources à un niveau macropixel afin de conserver de la surface pour la photodiode.

Pour pouvoir comparer qualitativement les résultats de plusieurs solutions, la Figure 62 présente le résultat d'une différence temporelle idéale sur deux images successives d'une vidéo montrant une personne marchant et agitant les bras. Pour bien visualiser le mouvement, on peut également seuiller l'image des différences (le seuil est trouvé à tâtons). L'action de seuiller est simple à implémenter, même dans la matrice (avec un comparateur), et permet de récupérer une image binaire ce qui signifie très peu de données à transmettre et traiter.

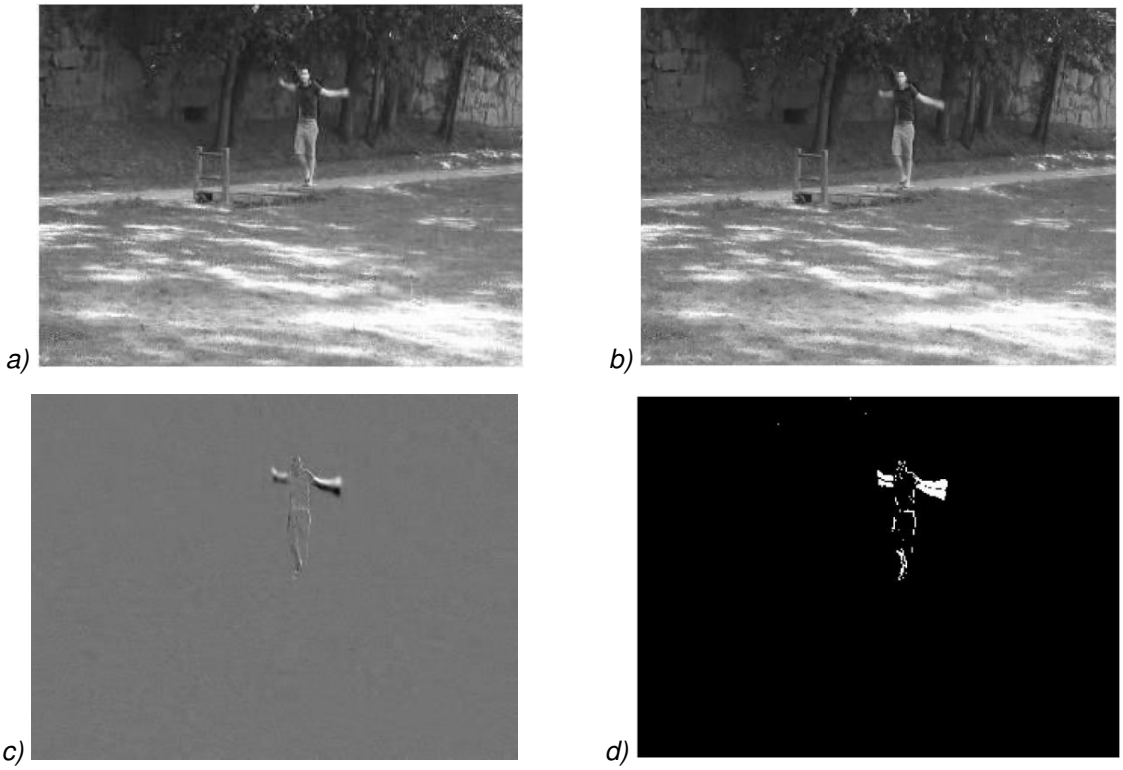


Figure 62 : (a) Image d'un film montrant un homme marchant et balançant les bras, (b) image suivante, (c) leur différence et (d) leur différence seuillée (seuil à 0.1 pour des images converties en double*, i.e. entre 0 et 1) pour un meilleur contraste. Les images ont 240x320 pixels.*

La différence temporelle se calcule normalement pixel à pixel : on mémorise tous les pixels d'une image puis on prend une seconde image et on fait la différence de chaque pixel avec son homologue mémorisé. Une implémentation intra-matricielle imposerait donc une mémoire dans chaque pixel. Pour gagner en surface, nous proposons de n'avoir qu'une mémoire pour plusieurs pixels. Une possible solution est de sous-échantillonner les deux images consécutives,c'est-à-dire qu' une différence temporelle classique est réalisée, pixel à pixel, mais sur une matrice sous-échantillonnée comme le montre la Figure 63. Comme dans le cas spatial, cela engendre naturellement une perte d'information. Plusieurs cas sont alors à distinguer, comme l'illustre la Figure 64 en présentant différents types de flux d'images et les différences temporelles correspondantes, sous-échantillonnées par 2x2 pixels ou non.

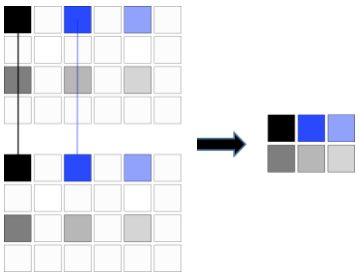


Figure 63 : Principe de la différence temporelle sous-échantillonnée (par 2x2 pixels ici) : les pixels représentés ici avec la même couleur sur deux images successives sont soustraits l'un à l'autre pour donner l'image de résultats.

Tout d'abord, le mouvement dans un film peut être de faible amplitude : un objet ne bouge qu'un peu, se décale d'un pixel par exemple, puis s'arrête. Ce cas est illustré sur la Figure 64(a) et on y voit que l'on peut perdre l'information de mouvement en sous-échantillonnant. Cependant, ce type de mouvement très faible n'est pas essentiel à détecter : un objet qui ne bouge quasiment pas n'est pas une menace par exemple.

Ensuite on peut considérer les mouvements de plus grande amplitude. Ceux-ci peuvent être rapides par rapport au temps entre deux images consécutives, c'est-à-dire que l'objet a bougé de plusieurs pixels entre les deux prises d'images. La Figure 64(b) présente ce cas et montre que l'on ne perd alors pas d'information en sous-échantillonnant. Dans le cas d'un mouvement de grande amplitude mais plus lent (Figure 64(c)), l'objet peut, dans le cas extrême, ne se déplacer que d'une ligne de pixels à la fois. L'information est perdue sur certaines images : dans le cas d'un sous-échantillonnage par 2x2 pixels, on perd l'information une image sur deux seulement.

Les objets présentés jusqu'ici sont larges comme c'est souvent le cas sur des images, mais certains objets sont beaucoup plus fins, ce qui amène d'autres cas. Par exemple, la Figure 64(d) présente le cas d'un objet fin (1 ligne de pixels) qui se déplace lentement. Grâce au fait qu'une ligne de pixel passe d'une couleur foncée à claire et qu'une autre passe de claire à foncée, le sous-échantillonnage par 2x2 pixels n'engendre pas de perte de mouvement détecté (contrairement au cas de la Figure 64(c)). En revanche, dans le cas d'un objet fin, sur une seule ligne droite, en mouvement synchronisé avec le temps entre deux images (Figure 64(e)), l'information peut être complètement perdue, mais il s'agit d'un cas très précis. Il suffit que l'objet soit en biais ou à peine plus épais pour que des pixels changeant de couleur soient conservés par sous-échantillonnage.

Plus on sous-échantillonne la différence temporelle, plus on réduit le nombre de calcul à mener sur un couple d'images, mais les pertes d'information dépendent fortement de la taille du sous-échantillonnage. En effet, sur le cas de la Figure 64(c) par exemple, l'information est perdue dans 1 image sur 2 pour un sous-échantillonnage par 2x2 pixels, mais dans 2 images sur 3 pour un sous-échantillonnage par 3x3 pixels. De même la taille de « l'objet fin » dans le cas de la Figure 64(e) devient 2 pixels au lieu de 1.

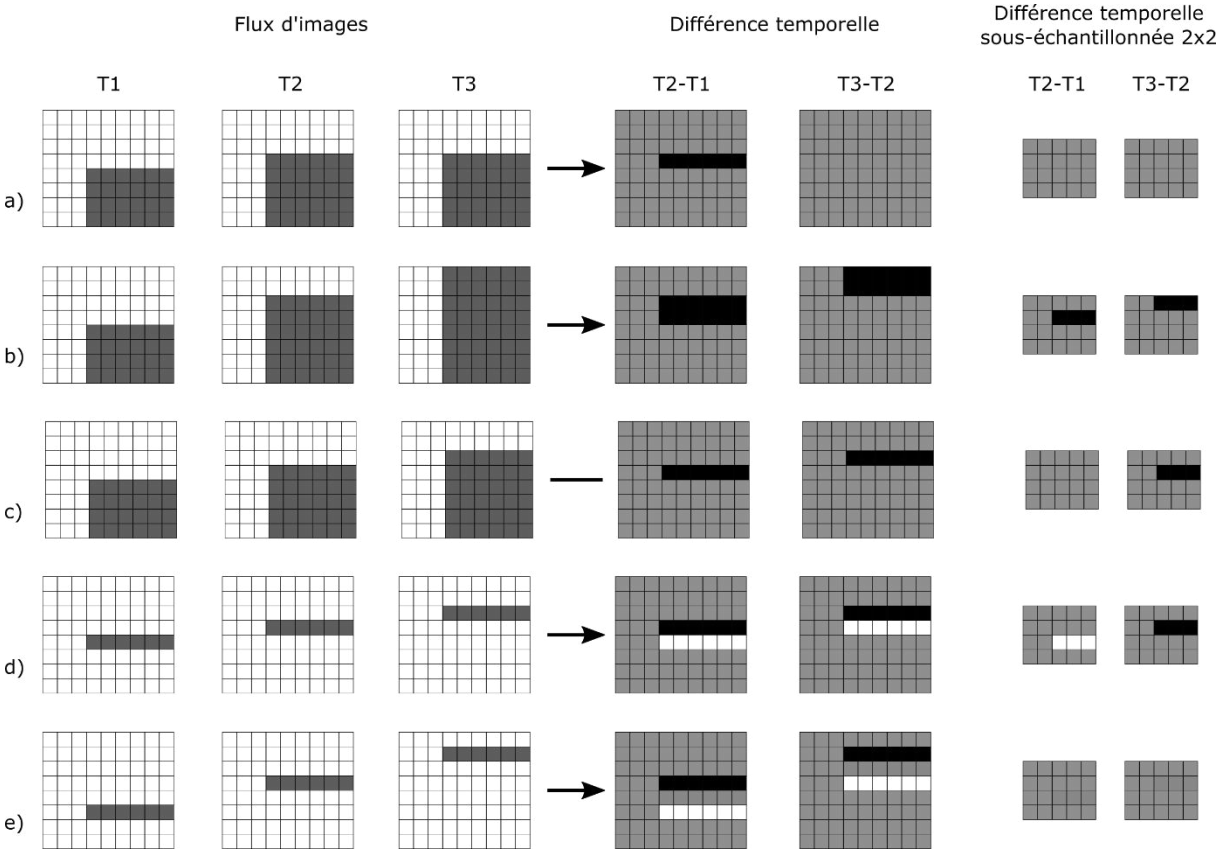


Figure 64 : Différents cas de flux d'images, avec leurs résultats en différence temporelle et en différence temporelle sous-échantillonnée par 2x2 pixels.

Pour résumer, en théorie le sous-échantillonnage de la différence temporelle engendre peu d'erreur, en tout cas sur du sous-échantillonnage par 2x2 pixels. Des résultats sur des images extraites de la vidéo de l'homme marchant et remuant les bras sont présentés en Figure 65 et Figure 66 pour des différences temporelles sous-échantillonnées respectivement par 2x2 ou 3x3 pixels. On peut voir qualitativement que le résultat de ces solutions se rapproche bien du résultat idéal de la Figure 62. En

particulier, le sous-échantillonnage par 2x2 pixels donne bien un très bon résultat et est donc la technique que nous allons adopter pour implémenter la différence temporelle.

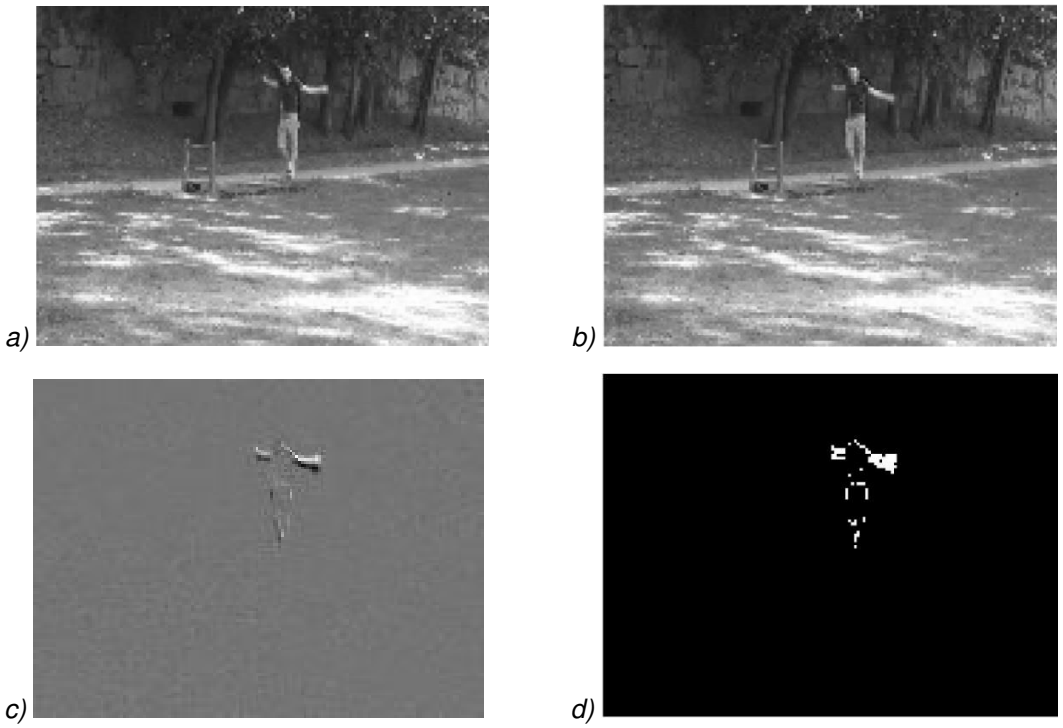


Figure 65 : (a, resp. b) Première (resp. deuxième) image sous-échantillonnée par 2x2 pixels (c) différence a-b et (d) a-b seuillée à 0.1. Ces images ont donc 120x160 pixels.

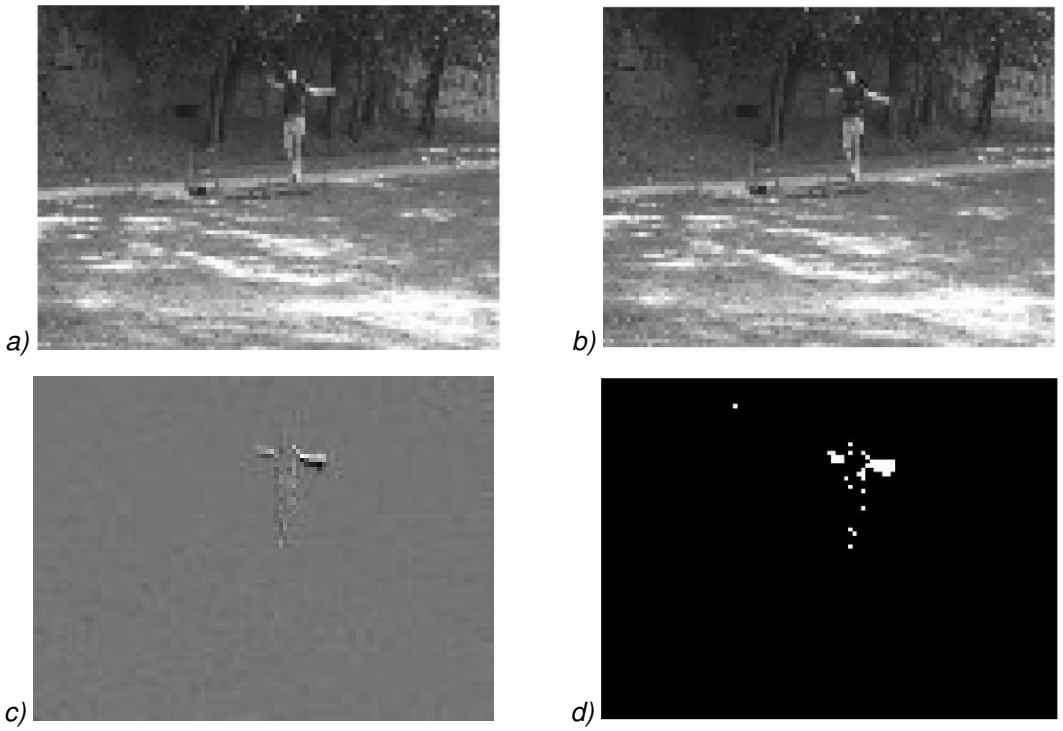


Figure 66 : (a, resp. b) Première (resp. deuxième) image sous-échantillonnée par 3x3 pixels (c) différence a-b et (d) a-b seuillée à 0.1. Ces images ont donc 80x106 pixels.

Un calcul de PSNR peut être fait en comparant l'image de résultat idéale et celle sous-échantillonnée dont on répète plusieurs fois chaque pixel (4 fois ou 9 fois pour des sous-échantillonnage par 2x2 ou 3x3 pixels) afin de retrouver la résolution originale. On obtient ainsi des PSNR de 40.2dB et 39.3dB respectivement pour les sous-échantillonnages par 2x2 et 3x3 pixels. Ces valeurs sont appréciables et très proches entre elles, mais ne reflètent pas forcément l'impact du sous-échantillonnage sur le résultat d'un algorithme dans un cas d'utilisation. Sans critère plus objectif pour juger la qualité du résultat, l'impact de l'erreur introduite par le circuit électronique qui calculerait la différence temporelle n'est pas non plus quantifié (contrairement au cas de la convolution spatiale, avec le critère de détection de piétons).

3.3 Co-intégration des traitements

On a proposé une méthode très simple pour calculer une détection de mouvement par différence temporelle avec un bon compromis information/surface ou complication des calculs. En réutilisant les mêmes ressources que pour le filtrage spatial, on peut atteindre un compromis surface/versatilité encore meilleur.

En effet, le filtrage spatial a pour opération de base une combinaison linéaire de valeurs de pixels, ce qui est aussi le cas pour la différence temporelle. On peut donc utiliser le même PE pour le filtrage spatial et pour le filtrage temporel.

Le PE étant capable de faire une combinaison linéaire à coefficients programmables, des différences temporelles pondérées peuvent aussi être réalisées, suivant la formule suivante :

$$Diff_{pondérée} = coeff_1 * pix - coeff_2 * mem \quad (3.3)$$

avec *pix* la valeur d'un pixel et *mem* la valeur du même pixel pour l'image précédente. En particulier, si on a besoin d'étaler les résultats sur une plage plus grande pour mieux classer les résultats de différence temporelle, on peut utiliser $coeff_1$ égal à $coeff_2$: c'est la différence temporelle amplifiée. Cela ne modifie pas les contraintes sur le PE puisque les filtrages spatiaux restent plus demandeurs en calculs (combinaison linéaire à 9 termes au lieu de 2).

3.4 Architecture proposée

3.4.1 Architecture globale

On propose donc une nouvelle architecture d'imageur intelligent fondée sur le principe de macropixels. Le kernel de calcul pour le filtrage spatiale étant de 3x3 pixels, et celui pour le filtrage temporel étant de 2x2 pixels, la matrice est organisée en :

- un circuit de calcul (PE) pour chaque groupe de 3x3 pixels
- une mémoire pour chaque groupe de 2x2 pixels.

Cette organisation, issue d'une adéquation algorithme-architecture et de réutilisation des ressources de calcul, a pour avantage de minimiser la surface globale dédiée au calcul et aux interconnexions tout en assurant une versatilité dans les prétraitements.

Cette architecture résulte en un motif de 6x6 pixels (PPCM(2,3) = 6) présenté en Figure 67. Ainsi la réalisation d'une matrice de résolution aussi grande que souhaitée est possible par la simple répétition de ce motif.

3.4.2 Optimisation de la répartition de la charge de calcul

Un filtrage spatial se faisant par convolution sous-échantillonnée par un masque de taille 3x3 pixels en général, chaque calcul de combinaison linéaire se fait en parallèle sur les PEs. Puisqu'il y a un PE pour 3x3 pixels, il y a un calcul par PE : la répartition de la charge de calcul est simple et directe. En revanche, le filtrage temporel est calculé sur une base de 2x2 pixels mais doit toujours être calculé par les mêmes PEs, qui sont sur une base de 3x3 pixels. Puisqu'il y a plus de mémoires que de PE, un même PE doit calculer plusieurs différences temporelles pour un seul couple d'image. Il s'agit donc de faire une différence dans chaque PE, lire la sortie des macropixels, faire une autre différence, lire les résultats, etc. pour un seul couple d'images à traiter.

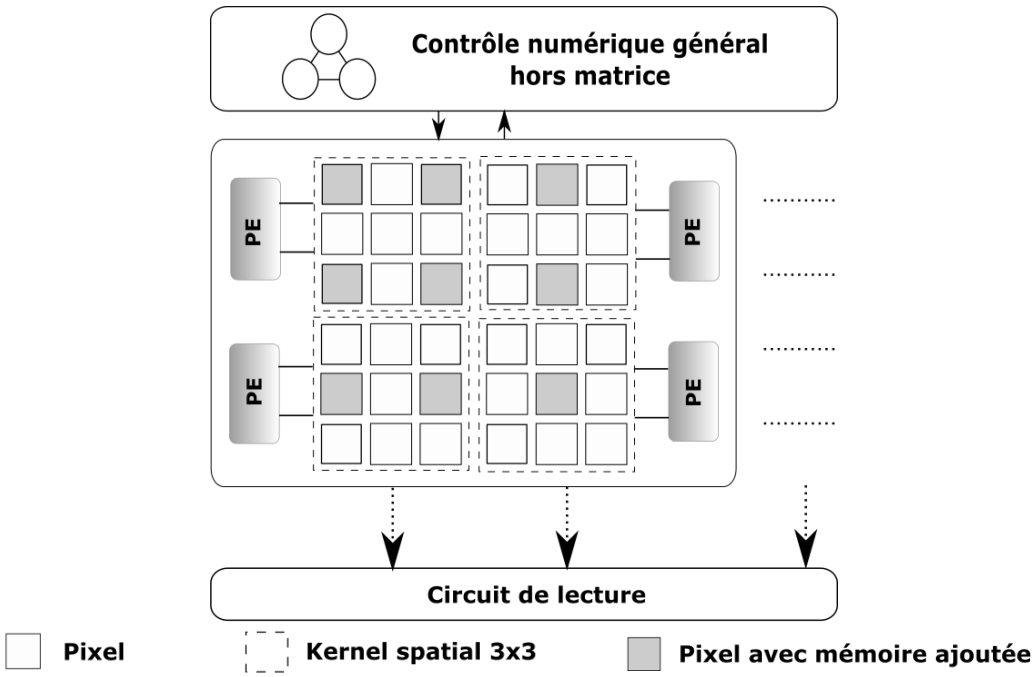
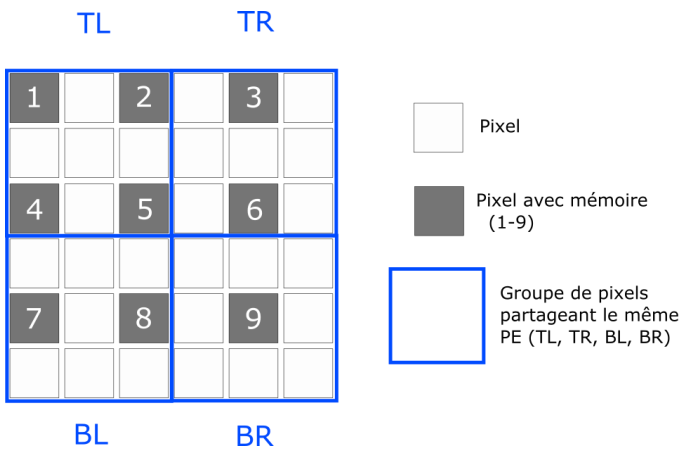


Figure 67 : Structure proposée pour un nouvel imageur intelligent.



PE	TL (*Top Left*)	TR (*Top Right*)	BL (*Bottom Left*)	BR (*Bottom Right*)
Pixels avec mémoire dont le PE s'occupe	1 ; 2 ; 4 ; 5	3 ; 6	7 ; 8	9

Figure 68 : Correspondance entre PE et pixel avec mémoire dont ils s'occupent si l'on fait la répartition spatiale de la charge de calcul la plus simple.

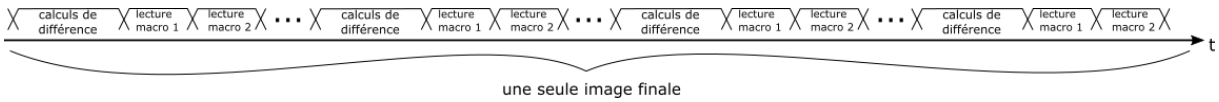


Figure 69 : Chronogramme simplifié du calcul d'une image de différence temporelle, pour la répartition correspondante à la Figure 68.

Si chaque circuit de calcul s'occupe des mémoires de son groupe de 3x3 pixels, alors certains ont 4 cycles de calcul et lectures à faire, tandis que d'autres n'en ont qu'un (voir Figure 68 et Figure 69). Pour gagner du temps, la mémoire du milieu (n°5) du motif répétable de 6x6 pixels est reliée au PE du macropixel BR comme l'indique le Tableau 8.

PE	TL (*Top Left*)	TR (*Top Right*)	BL (*Bottom Left*)	BR (*Bottom Right*)
Pixels avec mémoire dont le PE s'occupe	1 ; 2 ; 4	3 ; 6	7 ; 8	5 ; 9

Tableau 8 : Correspondance entre PE et pixel avec mémoire dont ils s'occupent si l'on fait la répartition spatiale de la charge de calcul qui permet de minimiser le temps total pour un couple d'images.

Les PEs calculent donc en parallèle les différences temporelles en pixels n°1,3,7 et 9, puis la matrice de résultats est lue, ensuite ils calculent pour les pixels n°2, 6, 8 et 5, la matrice est lue, et enfin le pixel n°4 est calculé par le PE TL et il est lu. On a alors seulement une image de mouvement sous-échantillonnée par 2, comme l'illustre la Figure 70.

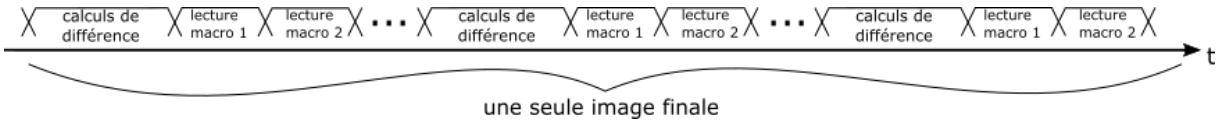


Figure 70 : Chronogramme simplifié du calcul d'une image de différence temporelle, pour la répartition correspondante au Tableau 8.

Cette architecture possède donc un PE pour 9 pixels et une mémoire pour 4 pixels, et permet de faire une convolution spatiale sous-échantillonnée par 3x3 pixels ainsi qu'une différence temporelle sous-échantillonnée par 2x2 pixels. Les circuits de calculs doivent réaliser des combinaisons linéaires de valeurs de pixels. Pour que le système soit qualifié de versatile, les coefficients de combinaison linéaire doivent être programmables : ainsi le système pourra réaliser n'importe quel filtrage spatial ou différence temporelle.

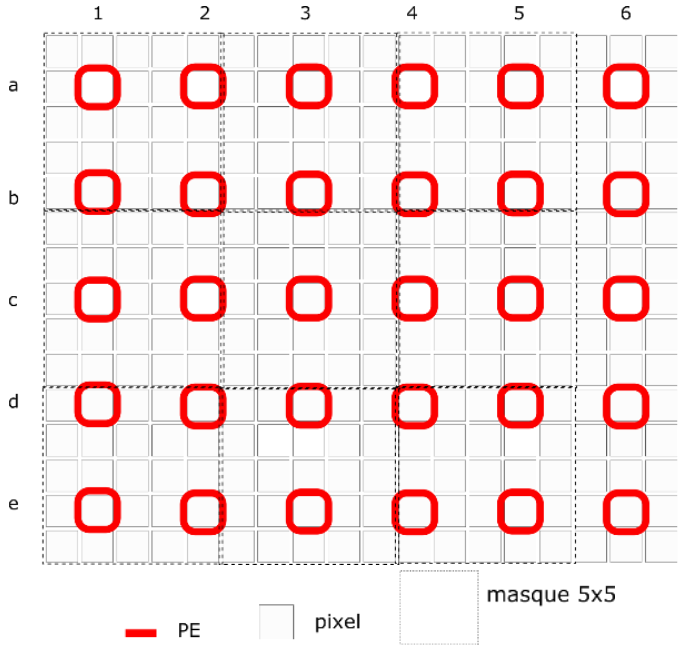
Si on interconnecte les macropixels à leurs voisins, il est toujours possible de faire d'autres tailles de masques, au prix d'un séquencement de calculs et d'un adressage plus compliqués comme le montre la Figure 71. De la même manière, connecter les macropixels voisins entre eux peut aussi permettre de calculer une convolution complète si besoin.

3.4.3 Conclusion et spécifications

Il a été montré qu'on peut espérer un meilleur compromis entre surface et versatilité grâce à l'adéquation algorithme-architecture, le principe de macropixels et la réutilisation des ressources. L'implémentation d'une convolution sous-échantillonnée pour réaliser du filtrage spatial est proposée et il a été montré que cela n'engendrait pas de perte d'information significative pour des algorithmes de plus haut niveau de type détection de piétons. Pour la détection de mouvement, il a été proposé d'utiliser le même circuit de calcul intra-macropixel pour réaliser un calcul de différence temporelle également sous-échantillonné.

Il reste à étudier les implémentations possibles du circuit de calcul, qui a pour spécification une taille la plus faible possible. La tolérance à l'erreur étant élevée, le calcul peut être approximé. Cela permet d'envisager une solution analogique peu contrainte en précision. Pour éviter les ADCs et limiter

la consommation d'énergie, nous allons donc étudier dans le chapitre suivant les implémentations possibles en électronique analogique de notre élément de calcul (PE pour *processing element*).



Masques 3x3 par macropixels pour calculer la CSE par un masque 5x5 en 1 point (en haut à gauche sur la matrice)

1a : $\begin{bmatrix} x & x & x \\ x & x & x \\ x & x & x \end{bmatrix}$ 2a : $\begin{bmatrix} x & x & 0 \\ x & x & 0 \\ x & x & 0 \end{bmatrix}$

1b : $\begin{bmatrix} x & x & x \\ x & x & x \\ 0 & 0 & 0 \end{bmatrix}$ 2b : $\begin{bmatrix} x & x & 0 \\ x & x & 0 \\ 0 & 0 & 0 \end{bmatrix}$

Figure 71 : Convolution avec un masque 5x5 sur des macropixels de 3x3 pixels. Pour calculer le masque tout en haut à gauche, on utilise les macropixels 1a, 2a, 1b et 2b avec les masques indiqués, puis on somme leurs résultats intermédiaires sur un seul macropixel et on lit le résultat. Le masque en bas à gauche par exemple peut être calculé en même temps, mais pour les masques milieu-gauche, haut-milieu et milieu-milieu, il faut attendre que les macropixels partagés soient libérés. L'ensemble de macropixels 1a-5e est le motif élémentaire de calcul (ppcm(3,5) = 15 pixels).

Chapitre 4 : Architecture de traitement

Le chapitre précédent a proposé une architecture d'imageur intelligent basée sur un calcul distribué dans la matrice à l'échelle macropixel. La conception du circuit électronique de calcul répété dans chaque macropixel est l'objet de ce chapitre. Ce circuit doit permettre de réaliser un imageur intelligent pour systèmes embarqués, versatile, à cadence vidéo classique et en technologie standard. Après avoir détaillé les spécifications, nous présenterons quelques architectures possibles afin de choisir la meilleure à implémenter pour notre cas. Enfin nous discuterons le dimensionnement du circuit retenu et l'étudierons en détail pour des applications très contraintes en surface.

4.1 Spécifications de l'élément de calcul

Les implémentations numériques d'éléments de calculs dans la matrice (PEs) offrent une large programmabilité, intrinsèque à l'électronique numérique. Un exemple de l'état de l'art [106 - Schmitz 2017] présente un imageur intelligent capable de détecter des contours, de faire un filtrage médian, des histogrammes et du suivi de cible. Mais une implémentation numérique dans la matrice de pixels consomme énormément de puissance et de surface, en particulier à cause des convertisseurs analogique-numérique [42 - Zarandy 2011] [106 - Schmitz 2017]. D'autre part, les calculs dans le domaine analogique peuvent être plus rapides et moins consommer, sur des tâches spécialisées. Puisque ce travail vise des applications de vision embarquées, très contraintes en consommation d'énergie pour le système, et en surface occupée dans la matrice, les implémentations analogiques semblent plus appropriées. De plus, il a été montré dans le chapitre précédent que la précision recherchée sur le résultat est assez faible. Ceci permet de relâcher les contraintes sur la précision du circuit analogique, et donc d'en optimiser la surface.

Le but ici est donc d'étudier différentes architectures de processeur analogique et d'en déduire laquelle est la plus adaptée pour répondre à nos spécifications. En effet, ce système doit calculer la combinaison linéaire des valeurs de pixels, sa sortie Out_{PE} peut donc s'écrire :

$$Out_{PE} = \sum_{i=1}^{N_{pixels}} k_i \, . pixel_i \quad , \qquad k_i \in \mathbb{R} \qquad (4.1)$$

où $pixel_i$ est la valeur du pixel *i,*

N_{pixels} est le nombre de pixels concernés par la convolution en ce point et

k_i le coefficient correspondant au pixel *i*.

Pour pouvoir varier les filtres appliqués au cours de l'utilisation du système (détection de contours verticaux, horizontaux, lissage, etc.), les poids doivent pouvoir être programmables sur le terrain. Typiquement, les filtres utilisés ont des coefficients entiers relatifs assez faibles [3 - Sponton 2015].

En plus du filtrage spatial, on a spécifié au chapitre précédent que le même PE doit pouvoir également calculer une différence temporelle, c'est-à-dire la différence entre deux valeurs successives d'un pixel :

$$Out_{PE} = pixel_i(t) - pixel_i(t-1) \qquad (4.2)$$

où $pixel_i(t)$ représente la valeur du pixel *i* à l'instant *t*.

Il s'agit donc d'un cas particulier de l'équation (4.1) avec N_{pixels} = 2, k_1 = 1 et k_2 = -1, et *pixel*(*t*-1) la valeur d'une mémoire plutôt que d'un autre pixel. Le filtrage spatial est donc le plus contraignant (9 entrées, k_i programmables dans $\mathbb{R}$). Mais comme vu au chapitre 1, la plupart des masques utilisés dans la littérature sont à coefficients entiers relatifs (dans $\mathbb{Z}$). Il n'est donc pas très restrictif de se limiter, au besoin, à des coefficients dans $\mathbb{Z}$.

Spatialement, le circuit doit être implémenté au milieu du groupe de 3x3 pixels, en dégradant le moins possible leur qualité (en termes de facteur de remplissage et de taille totale). La géométrie finale est libre, dépendante du circuit choisi. La technologie utilisée est standard, donc planaire.

En termes de vitesse, le système complet doit travailler au moins à cadence vidéo (25 images par seconde). Par image, pour détecter des contours, il faut appliquer au moins 2 filtres (vertical et horizontal). Pour calculer la combinaison linéaire par un filtre, on a donc au maximum jusqu'à 1/25/2=20ms (50Hz). Selon le type de pixel et le type de lecture, il faut rajouter les temps d'intégration de la lumière et de lecture des macropixels. On peut spécifier un maximum de 10ms de calcul d'une convolution, pour avoir de la marge. En particulier, si la lecture de la matrice de pixels est faite en série (un macropixel à la fois / ligne par ligne) et que la matrice est importante, le temps de lecture peut devenir critique.

En termes de précision du calcul, pour notre application-test (la détection de piéton) l'algorithme de plus haut niveau tolère jusqu'à 10% d'erreur sur les coefficients de la combinaison linéaire pour pouvoir donner un résultat correct, comme l'a montré la section 1.3 du chapitre 3.

4.2 Exploration architecturale de circuit de PE analogique

Le PE doit calculer des combinaisons linéaires. Deux types d'architecture se dessinent : à calculs parallèles ou à calculs séquentiels. Dans le premier cas, toutes les multiplications sont effectuées en même temps (parallèlement) et les résultats sont sommés, tandis que dans le second cas une multiplication est faite, puis une deuxième sommée à la précédente, etc. (voir Figure 72). Structurellement, les calculateurs parallèles semblent plus rapides que les séquentiels, mais nécessitent plus de surface (plusieurs multiplieurs). Les deux approches vont être détaillées.

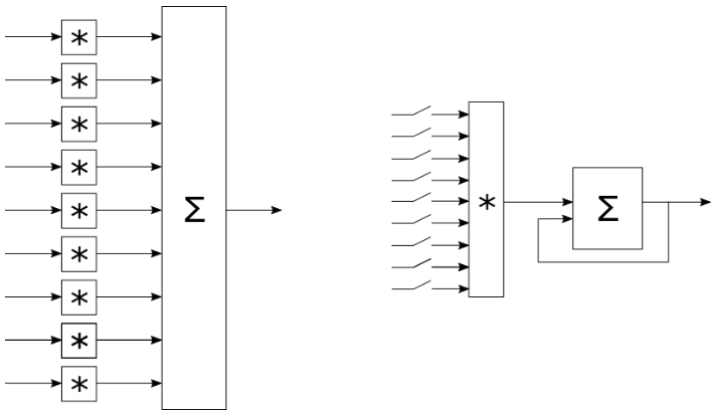


Figure 72 : (a) Calculs parallèles : toutes les entrées sont multipliées par leur coefficient simultanément et sommées ; (b) calculs séquentiels : on sélectionne l'entrée du multiplieur et le coefficient correspondant, puis on accumule les résultats.

4.2.1 Calculs parallèles

Au vu des lois de Kirchhoff, le plus simple pour sommer simultanément plusieurs grandeurs électriques est de travailler en courant. La somme correspond alors à un nœud du circuit. Dans notre cas, chacun des courants à sommer doit donc correspondre à la valeur d'un pixel multipliée par son coefficient. La somme étant en courant sur un nœud, les multiplications doivent être simultanées, c'est-à-dire qu'il faut 9 multiplieurs programmables.

Pour multiplier un courant par une constante, un circuit peu complexe est un miroir de courant avec pour rapport de tailles cette constante, comme l'illustre la Figure 73. C'est une solution utilisée par exemple dans les calculs de transformée de Fourier discrète analogique [110 - Sadeghi 2009]. Pour rendre cela programmable, on peut se limiter à des coefficients k_i faibles (valable pour la plupart des filtres utilisés) et donc utiliser plusieurs miroirs sélectionnés selon la valeur de k_i souhaitée.

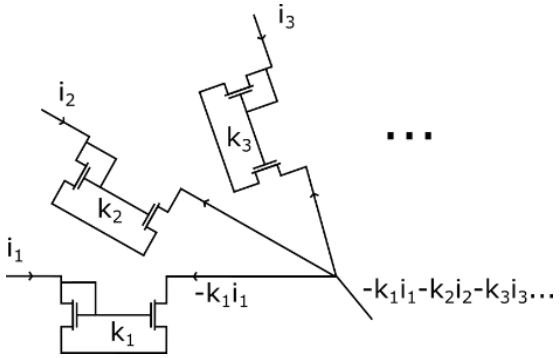


Figure 73 : Combinaison linéaire de courants en utilisant des miroirs de courant à coefficients fixes.

D'autres types de multiplieurs peuvent être utilisés, comme dans [63 - Dubois 2008] évoqué au chapitre précédent, dont les multiplieurs sont représentés en Figure 74. Ces solutions en courant sont envisageables pour des masques petits (2x2 par exemple) mais génèrent des circuits bien trop conséquents lorsqu'il faut répéter 9 fois le multiplieur pour un filtre 3x3.

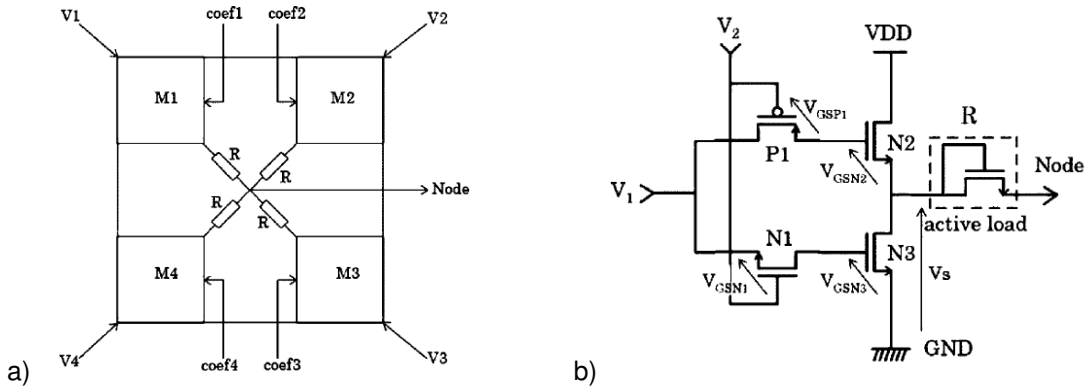


Figure 74 : (a) Combinaison linéaire de 4 valeurs de pixels et (b) détail d'un multiplieur [63 - Dubois 2008].

Pour des calculs en tension, il existe bien des additionneurs et multiplieurs en mode tension ([111 - Chaoui 1995], [112 - Gilbert 1968], …) mais au vu de leur complexité et de leur taille ils ne sont pas implémentés dans les imageurs intelligents.

En conclusion, quelle que soit la grandeur qui porte le signal, les 9 répétitions du circuit de multiplication impliquent intrinsèquement une forte consommation surfacique difficilement compatible avec les besoins de notre problématique. De plus, si le masque possède des coefficients nuls (filtres de Sobel par exemple), des circuits sont inutiles. Il s'agit d'une approche peu utilisée en imageurs intelligents. C'est pourquoi la section suivante étudie plus en détail l'approche séquentielle de la combinaison linéaire, c'est dire faire les additions les unes après les autres, en accumulant.

4.2.2 Calcul séquentiel

Une combinaison linéaire en calcul séquentiel est une accumulation de N résultats de multiplications réalisées successivement avec un seul multiplieur (dont les entrées sont $pixel_i$ et k_i pour $i = 1,\ldots,N$). Il y a donc un seul multiplieur programmable quel que soit le nombre d'entrées et un accumulateur comme le montre la Figure 72(b). Cela peut se faire avec des entrées disponibles les

unes à la suite des autres temporellement (mode intégrateur) ou disponibles simultanément comme dans notre cas (9 pixels distincts fonctionnant en parallèle) et sélectionnées une à la fois.

Dans un calcul séquentiel, la multiplication par des entiers peut être faite en accumulant plusieurs fois la même valeur de pixel (Figure 75). Ainsi pour multiplier une entrée par un entier programmable, il suffit de programmer le nombre d'accumulations faites pour un même signal d'entrée. L'inconvénient de ce type de calcul est de prendre potentiellement plus de temps pour faire plusieurs accumulations de la même valeur qu'en utilisant un multiplieur. Nos spécifications sont de faire une convolution en moins de 10ms, et comme la somme des valeurs absolues des coefficients d'un masque de taille 3x3 dépasse rarement 20 (8 pour Sobel, 9 pour un moyenneur, 16 pour un gaussien, etc.), cela laisse 0.5ms pour une seule accumulation. Cette contrainte reste facile à respecter. La suite de cette section va donc étudier les circuits d'accumulateurs.

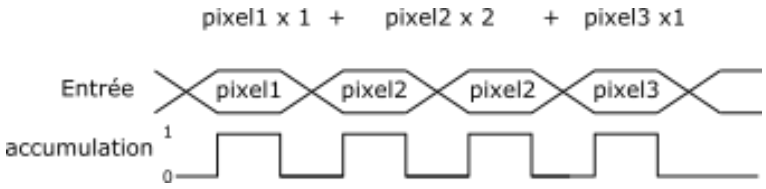


Figure 75 : Principe de combinaison linéaire par accumulations successives.

Une accumulation est une suite d'additions et de mémorisations des valeurs intermédiaires puis finale. Puisque des grandeurs en courant se somment facilement, considérons des mémoires de courants. Elles peuvent être composées d'un transistor (cascodé pour plus de stabilité) et d'une source de courant. Trois mémoires permettent de faire des combinaisons linéaires à coefficients entiers : une mémoire charge la nouvelle valeur, une mémoire charge la valeur accumulée, et la troisième reçoit la somme des deux premières. Un exemple de circuit est présenté en Figure 76. Ces circuits de type commutation de courant (*Switched-Current* ou *SI)* peuvent être rapides et petits [113 - Dudek 2000] mais consomment beaucoup à cause des sources de courants de référence. De plus les photocourants étant très faibles, il faut les amplifier pour limiter l'impact du bruit dans ces circuits.

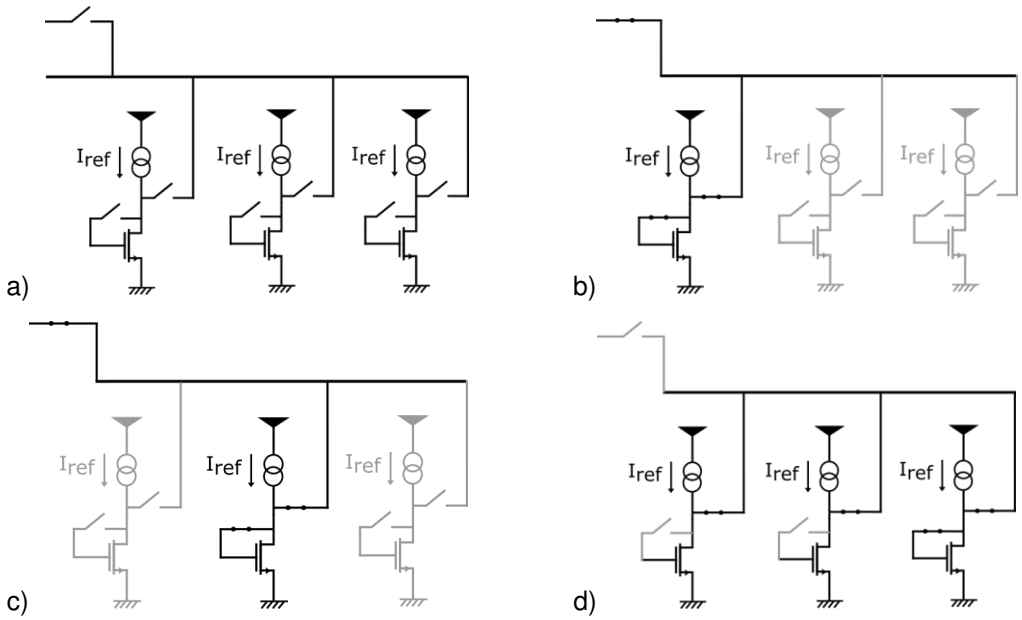


Figure 76: (a) Circuit de calcul séquentiel en courant, sur 3 mémoires, et ses différents états pour la somme de deux valeurs d'entrée consécutives : (b) écriture de la première mémoire, (c) écriture de la deuxième mémoire et (d) lecture des deux premières mémoires et écriture de la somme sur la troisième. Ensuite la phase (b) est remplacée par une écriture sur la première mémoire du résultat intermédiaire stocké sur la dernière mémoire et ainsi de suite.

Une autre grandeur facilement accumulée est la charge. C'est pourquoi au-delà des circuits à commutation de courant, on trouve les circuits à capacités commutées (SC pour *switched capacitors)* [114 - Lam 1983].

Pour accumuler des charges, il suffit de charger une capacité, puis de transférer ses charges vers une seconde capacité. La Figure 77 présente un schéma simple : les phases φ_1 et φ_2 sont opposées ; sur φ_1 on charge la capacité d'entrée C_{in} à la valeur voulue tout en isolant la capacité de sortie C_{out}, puis sur φ_2 on isole l'entrée du circuit et on transfère les charges sur C_{out}. Cependant, il est plus exacte de parler d'équilibrer les charges entre C_{in} et C_{out} pendant la phase 2 : toutes les charges ne sont pas transférées. Un exemple d'utilisation peut être trouvé dans les ADCs à distribution de charges (*passive charge sharing SAR)* [115 - McCreary 1975].

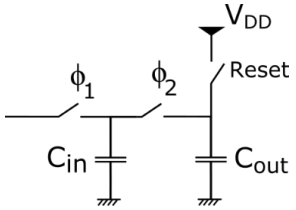


Figure 77 : Transfert de charges passif.

Pour pallier le problème de transfert partiel, il faut une masse virtuelle qui pousse toutes les charges de C_{in} vers C_{out}. Elle peut être fournie par un amplificateur opérationnel. La tension d'entrée est d'abord chargée sur C_{in} (phase 1) puis accumulée sur C_{out} (phase2) de la Figure 78. Pendant la phase 2, la capacité C_{in} est bien entre une masse et une masse virtuelle (entrée - de l'amplificateur opérationnel) donc toutes ses charges sont transférées vers C_{out}. La capacité d'entrée peut être montée entre 2 interrupteurs dans chaque phase (Figure 78(b)) ou directement à la masse comme dans le cas passif (Figure 78(a)), selon que l'on veuille inverser les charges ou pas.

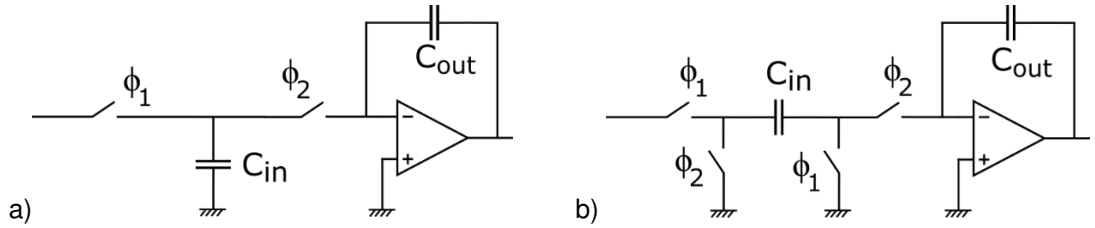


Figure 78 : Intégrateur classique à capacités commutées, (a) non-inverseur ou (b) inverseur.

Ces circuits permettent bien d'accumuler des charges représentatives des valeurs d'entrées successives. En gardant la même entrée pendant plusieurs accumulations, cela revient à la multiplier par un entier positif.

Au vu de cette étude rapide des différentes architectures capables de réaliser des combinaisons linéaires, la dernière, le circuit à capacités commutées, semble la plus appropriée pour notre application.

4.3 Analyse et définition du circuit SC retenu

Les circuits à capacités commutées ont été largement discutés dans la littérature ([114 - Lam 1983][116 - Nagaraj 1986][117 - Nauta 1993][118 - Razavi 2017],…). Une version adaptée à nos spécifications en est déduite et exposée dans cette section. Son dimensionnement est discuté pour obtenir la plus petite surface pour des spécifications de précision données. Il s'agira dans ce chapitre de considérations de dimensionnement générales, applicables à d'autres cas similaires.

4.3.1 Architecture adaptée

L'architecture de circuit SC utilisée doit être adaptée à nos contraintes : une faible surface et le calcul d'une combinaison linéaire à coefficients entiers relatifs.

Tout d'abord, pour pouvoir multiplier par un entier négatif, il suffit de charger la capacité d'entrée à l'envers. Pour cela seulement 2 interrupteurs et 2 signaux de commande sont rajoutés au circuit inverseur comme indiqué sur la Figure 79. Un chronogramme possible est présenté en Figure 80.

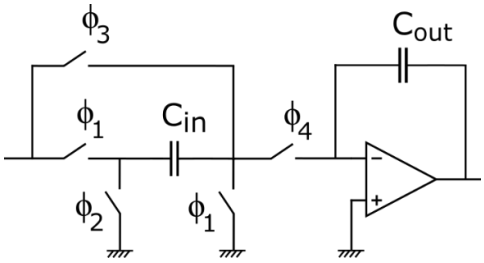


Figure 79 : Accumulateur à capacités commutées avec possibilité de soustraction.

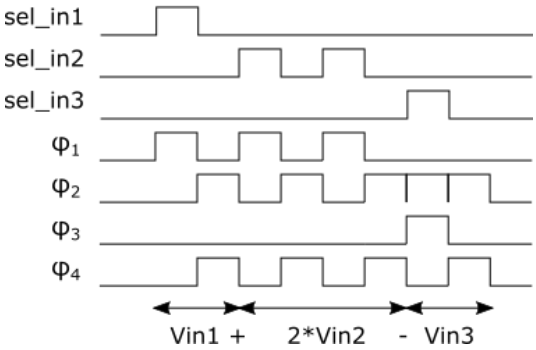


Figure 80 : Exemple de chronogramme de l'additionneur - soustracteur à capacités commutées. Les signaux sel_in *sont les sélections des entrées, i.e. les sélections des pixels dans notre application ; les signaux* φ *sont les commandes des interrupteurs du circuit SC de la Figure 79.*

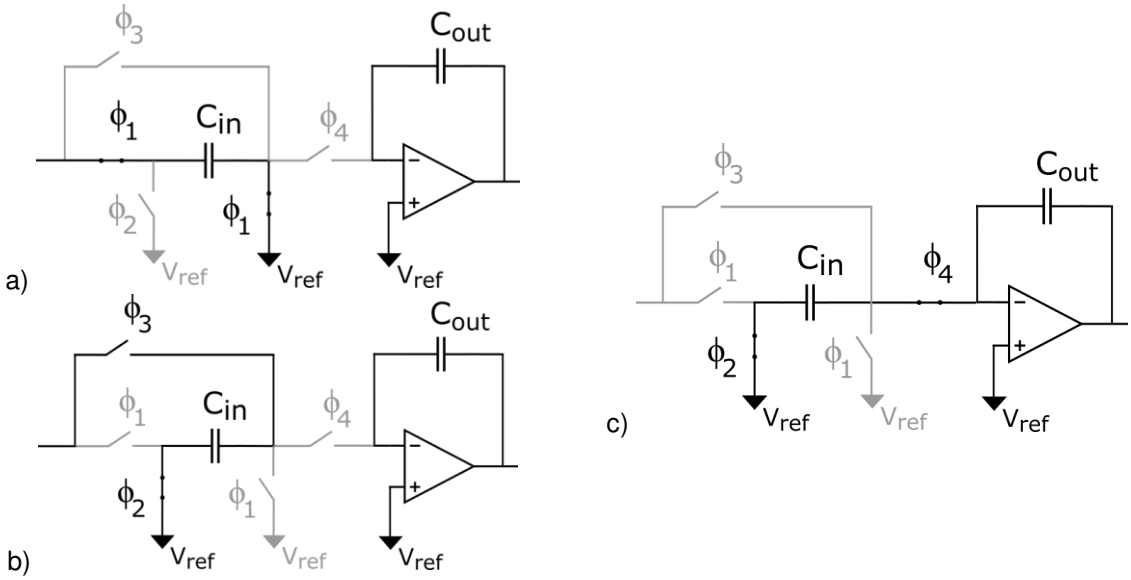


Figure 81 : Schémas des différentes phases de l'accumulateur : (a) chargement direct pour addition, (b) chargement inverse pour soustraction et (c) transfert de charges pour effectuer chaque opération.

En supposant l'amplificateur idéal, pendant la première phase φ_1 (Figure 81(a)) on a :

$$Q_{in1} = C_{in}\left(-V_{in1} + V_{ref}\right) \qquad \text{et} \qquad Q_{out1} = 0$$

avec Q_{in} et Q_{out} les charges sur C_{in} et C_{out} respectivement. L'indice numérique indique le numéro d'opération. V_{ref} est la tension de référence (par exemple le milieu de l'excursion des alimentations), V_{in} est le potentiel d'entrée et V_{out} de sortie.

Pendant la phase φ_4/φ_2 (Figure 81(c)) :

$$Q_{in1bis} = C_{in}\left(V_{ref} - V_{ref}\right) = 0 \quad \text{et} \quad Q_{out1bis} = Q_{out1} - (Q_{in1} - Q_{in1bis}) = -Q_{in1}$$

avec l'indice *bis* indiquant la phase de transfert φ_4/φ_2 .

Et d'autre part $\quad Q_{out1bis} = C_{out}\left(V_{out1bis} - V_{ref}\right)$

Donc $\quad C_{out}\left(V_{out1bis} - V_{ref}\right) = C_{in}\left(V_{in1} - V_{ref}\right)$

D'où $\quad V_{out1bis} = \frac{C_{in}}{C_{out}}\left(V_{in1} - V_{ref}\right) + V_{ref}$

Par itération, pour N accumulations :

$$V_{outNbis} = \frac{C_{in}}{C_{out}}\left(\sum_{K=1}^{N}\left(V_{inK} - V_{ref}\right)\right) + V_{ref}$$

Et pendant la phase φ_3/φ_2 (Figure 81(b)) :

$$Q_{in1} = C_{in}\left(V_{in1} - V_{ref}\right) \text{ et} \qquad Q_{out1} = 0$$

Pendant la phase φ_4/φ_2, de même que précédemment :

$$Q_{in1bis} = 0 \quad et \quad Q_{out1bis} = -Q_{in1}$$

Donc $\quad C_{out}\left(V_{out1bis} - V_{ref}\right) = Q_{out1bis} = -C_{in}\left(V_{in1} - V_{ref}\right)$

D'où $\quad V_{out1bis} = \frac{C_{in}}{C_{out}}(-1) * \left(V_{in1} - V_{ref}\right) + V_{ref}$

Par itération, pour N accumulations :

$$V_{outNbis} = \frac{C_{in}}{C_{out}}\left(\sum_{K=1}^{N} s_K\left(V_{inK} - V_{ref}\right)\right) + V_{ref} \qquad (4.3)$$

avec s_K le signe de l'accumulation K, c'est-à-dire 1 si φ_1 est utilisée et -1 si φ_3 est utilisée. Si le même pixel est utilisé plusieurs fois successivement comme entrée de l'accumulateur, on peut réécrire l'équation ainsi :

$$V_{outNbis} = \frac{C_{in}}{C_{out}}\left(\sum_{i=1}^{N_{pixels}}\ \sum_{K=1}^{coef_i} s_i\left(V_{in\,i} - V_{ref}\right)\right) + V_{ref}$$

Ou encore

$$V_{outNbis} = \frac{C_{in}}{C_{out}}\left(\sum_{i=1}^{N_{pixels}} s_i * coef_i * \left(V_{in\,i} - V_{ref}\right)\right) + V_{ref} \qquad (4.4)$$

On retrouve alors l'équation (4.1) recherchée, à multiplication et addition de constantes près (C_{in}, C_{out} et V_{ref}).

Pour limiter la surface occupée, un simple inverseur de point de repos V_{ref} est utilisé comme amplificateur à la place d'un amplificateur opérationnel [117 - Nauta 1993] [119 - Chae 2009]. Pendant

la phase de charge de la capacité d'entrée, l'inverseur est rebouclé sur lui-même, isolé de la capacité de sortie C_{out}, comme le montre la Figure 82.

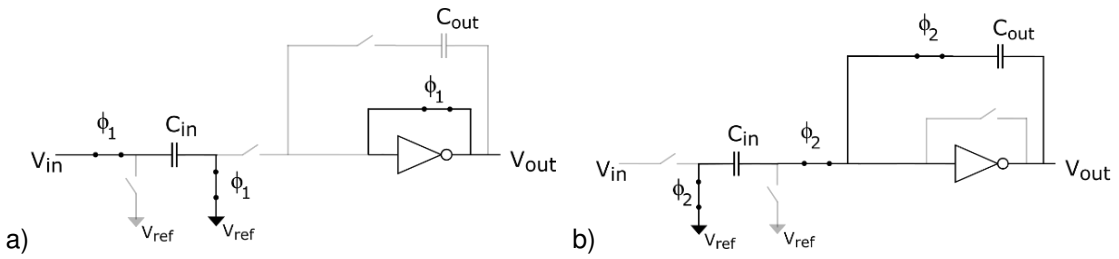


Figure 82 : Accumulateur avec un inverseur comme amplificateur avec (a) phase de charge et (b) phase de transfert.

On a alors

$$C_{out}(V_{out1bis} - V_{OFF}) = Q_{out1bis} = C_{in}\left(V_{in1} - 2V_{ref} + V_{OFF}\right)$$

donc $$V_{out1bis} = \frac{C_{in}}{C_{out}}\left(V_{in1} - 2V_{ref} + V_{OFF}\right) + V_{OFF}$$

et en itérant :

$$V_{outNbis} = \frac{C_{in}}{C_{out}}\left(\sum_{K=1}^{N}\left(V_{inK} - 2.V_{ref} + V_{OFF}\right)\right) + V_{OFF} \qquad (4.5)$$

où V_{OFF} est la tension de repos (*bias point*) de l'inverseur.

Ce point de repos varie beaucoup en fabrication, comme le vérifiera le chapitre 5 pour notre cas particulier. Par conséquent, une capacité de compensation C_C est rajoutée afin d'obtenir une « masse virtuelle » sur le passage des charges en transfert (Figure 83) [119 - Chae 2009]. Pour pouvoir faire des soustractions tout en conservant cette masse virtuelle, le schéma de la Figure 84 est obtenu. Les différentes phases de travail de cet accumulateur sont présentées en Figure 85.

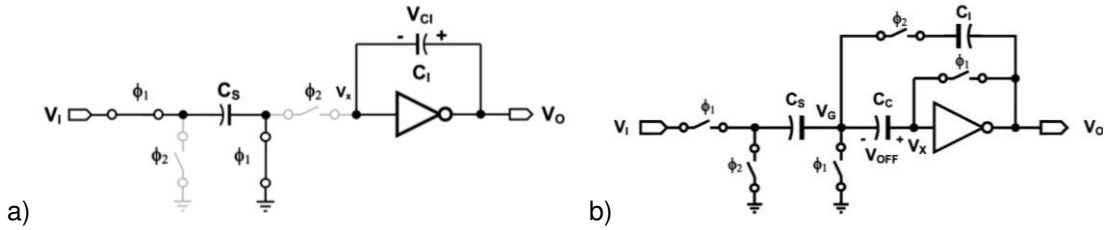


Figure 83 : Intégrateur à capacités commutées à base d'inverseur : (a) sans et (b) avec compensation [119 - Chae 2009].

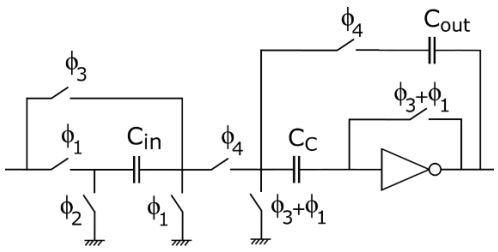


Figure 84 : Accumulateur-soustracteur à capacités commutées à base d'inverseur, avec compensation.

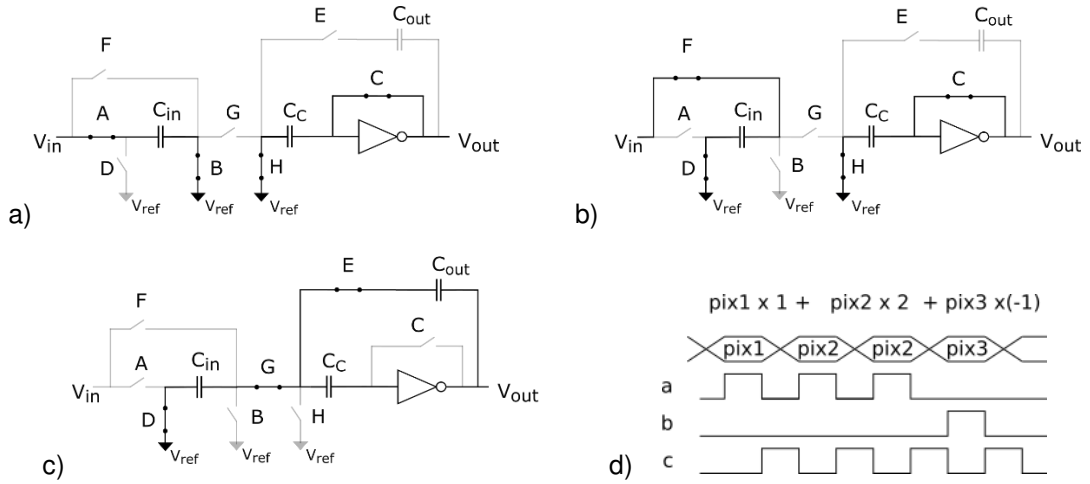


Figure 85 : Phases de travail : a) charge directe, b) charge inversée pour soustraction et c) transfert des charges pour accumulation. d) Exemple de chronogramme pour utiliser ces différentes phases a, b et c.

Ce circuit permet de remplir les spécifications du PE : un circuit réalisant le calcul d'une combinaison linéaire à coefficients programmables, sur une surface a priori faible, avec une précision et un temps raisonnables. Nous allons l'analyser plus en détails pour le vérifier.

4.3.2 Dimensionnement

Lorsqu'on s'intéresse à l'accumulateur avec un gain infini on a (rappel de l'équation 4.3) :

$$V_{out(N)} = \frac{C_{in}}{C_{out}} \left(\sum_{K=1}^{N_{accu}} s_K \left(V_{in(K)} - V_{ref} \right) \right) + V_{ref}$$

C_{in} et V_{in} dépendent du pixel choisi, et N_{accu}, le nombre d'accumulations souhaitées est une spécification (la somme des coefficients du masque, 8 pour faire des convolutions Sobel par exemple). Or on souhaite que la sortie reste dans la plage de linéarité de sortie de l'inverseur, centrée sur V_{ref} :

$$V_{ref} - \frac{Range}{2} \leq V_{out(N_{accu})} \leq V_{ref} + \frac{Range}{2}$$

$$-\frac{Range}{2} \leq V_{out(N_{accu})} - V_{ref} \leq \frac{Range}{2}$$

$$-\frac{Range}{2} \leq \frac{C_{in}}{C_{out}} \left(\sum_{K=1}^{N_{accu}} s_K \left(V_{in(K)} - V_{ref} \right) \right) \leq \frac{Range}{2}$$

avec *Range* l'excursion de sortie de l'inverseur (par exemple, si sa plage de sortie est [0.5V-2.5V] autour de V_{ref} = 1.5V, alors *Range* = 2V). Le pire cas est : toutes les entrées $V_{in(K)}$ ont la valeur extrême, i.e. la plus éloignée possible de V_{ref}, et sont toutes associées au même signe s_K. D'où :

$$\frac{2 \times N_{accu} \times \max(|V_{in} - V_{ref}|)}{Range} \leq \frac{C_{out}}{C_{in}} \qquad (4.6)$$

Pour avoir un accumulateur de faible surface, il faut C_{out} le plus petit possible, donc max($|V_{in}$-$V_{ref}|$) le plus petit possible. V_{ref} est donc choisie comme la tension milieu de l'excursion de sortie du pixel. On a alors par la formule (equation 4.6) le minimum du rapport C_{out}/C_{in}, ce qui permet de dimensionner C_{out}.

La compensation de la variation du point de repos de l'inverseur est réalisée par la capacité C_C quel que soit son dimensionnement. En revanche, celui-ci va influencer sa taille mais aussi l'erreur

introduite. Le dimensionnement de C_C sera fait par analyse paramétrique de l'erreur introduite dans le circuit.

Jusqu'ici le circuit a été étudié selon son modèle idéal, bien loin de la réalité. Nous allons donc étudier ici l'influence de ses non-idéalités.

4.3.3 Influence du gain fini

La plus grosse approximation est de considérer le gain de l'inverseur comme infini. Notons A le gain réel de l'inverseur (A<0) et modifions les équations de fonctionnement du circuit à capacités commutées. Avec les notations précédentes, le calcul développé en annexe 2 permet d'aboutir à :

$$V_{out(N)bis} = \frac{C_{in}}{C_{out}} \sum_{k=1}^{N} s_k \frac{\left(V_{in(k)} - V_{ref} - \frac{V_{OFF}}{A}\right)}{\left(1 - \frac{1}{A} - \frac{C_{in}}{AC_{out}}\right)^{N-k+1}} \left(1 - \frac{1}{A}\right)^{N-k} + V_{ref} \qquad (4.7)$$

Cette équation donne l'influence de la valeur du gain sur le résultat d'une combinaison linéaire dans le circuit choisi. L'erreur introduite pour une valeur de gain donnée dépend de la tension de repos de l'inverseur (V_{OFF}), des signes successifs (s_k) et des tensions d'entrées ($V_{in(k)}$), paramètres qui varient selon le process ou au cours du temps. Le gain apparait au dénominateur de fractions avec numérateur inférieur ou égal à 1, fractions qui sont donc négligeables devant 1 dès que |A| vaut quelques centaines. De même V_{OFF} est du même ordre que V_{ref} donc un gain de quelques centaines suffirait à négliger son influence. Par exemple, les résultats présentés dans le chapitre 6 sont obtenus avec un gain en valeur absolue maximal de 300 seulement.

4.3.4 Influence du désappariement des capacités

4.3.4.1 Cas gain infini

$$V_{outNbis} = \frac{C_{in}}{C_{out}} \left(\sum_{K=1}^{N} s_K \left(V_{inK} - V_{ref}\right) \right) + V_{ref} \qquad (4.3)$$

Dans l'équation (4.3), rappelée ci-dessus, les capacités C_{in} et C_{out} n'interviennent que sous forme du rapport C_{in}/C_{out}. Donc l'erreur sur le résultat due à la variation ΔR de ce rapport est :

$$\Delta V_{outNbis} = \Delta \mathrm{R} \left(\sum_{K=1}^{N} s_K \left(V_{inK} - V_{ref}\right) \right) \quad \text{avec} \quad \frac{C_{in}}{C_{out}} = \left(\frac{C_{in}}{C_{out}}\right)_{ideal} + \Delta R \qquad (4.8)$$

Le pire cas ($|\Delta V_{outNbis}|$ maximum) a lieu lorsque tous les signes de $s_K(V_{inK} - V_{ref})$ sont égaux (les erreurs ne se compensent pas) et V_{inK} le plus loin possible de V_{ref}. On peut calculer cette erreur sur la sortie pour un tel cas. La Figure 86 présente les variations possibles du ratio C_{out}/C_{in} autour de sa valeur idéale, pour une erreur relative donnée en sortie, lorsque 8 accumulations (ce qui correspond à un masque de Sobel par exemple) ont été calculées dans un pire cas selon les paramètres choisis (voir la légende de la figure). On peut y lire par exemple que pour une erreur maximale de 20% sur la sortie, le ratio peut varier entre 8.5 et 12.5 environ. Pour nos spécifications de précision, cela implique de très légères contraintes sur le ratio de capacités, et en particulier sur leur dessin. Pour un autre circuit avec d'autres spécifications de précision, on peut voir qu'il est possible de définir la précision requise sur le ratio pour une erreur maximale en sortie facilement.

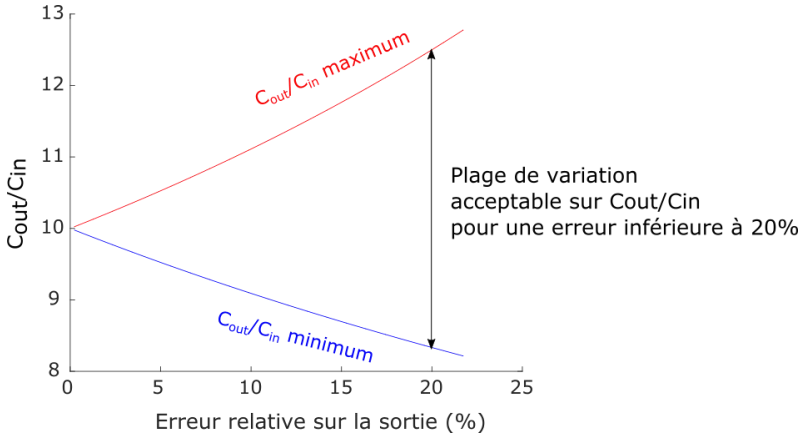


Figure 86 : Maximum et minimum requis sur le rapport C_{out}/C_{in} en fonction de l'erreur relative sur la sortie, pour des paramètres réalistes (voir chapitre 5) : V_{DD} = 3.3V, V_{ref} = V_{DD}/4, 0.3V<V_{in}<1.4V donc V_{in} = 1.4V dans la simulation, N = 8 accumulations, $(C_{out}/C_{in})_{idéal}$ =10.

4.3.4.2 Cas gain fini

En prenant en compte le gain fini *A*<0 de l'inverseur on peut trouver l'équation itérative :

$$V_{out(N)bis}\left(1-\frac{1}{A}-\frac{C_{in}}{AC_{out}}\right)=V_{out(N-1)bis}\left(1-\frac{1}{A}\right)+\frac{C_{in}}{C_{out}}\left(V_{in(N)}-V_{ref}-\frac{V_{OFF}}{A}\right)$$

i.e.

$$V_{outNbis}\left(1-\frac{1}{A}-\frac{1}{A}\left(\frac{C_{in}}{C_{out}}\right)_{id}+\frac{\Delta R}{A}\right)=V_{out(N-1)bis}\left(1-\frac{1}{A}\right)+\left[\left(\frac{C_{in}}{C_{out}}\right)_{id}+\Delta R\right]\left(V_{inN}-V_{ref}-\frac{V_{OFF}}{A}\right) \qquad (4.9)$$

Ou encore d'après l'équation (4.7), en négligeant ΔR/A devant 1/A (0.03<ΔR<0.03 pour 8<C_{out}/C_{in}<13) au dénominateur :

$$\Delta V_{outNbis}=\Delta R\sum_{k=1}^{N}s_k\frac{\left(V_{in(k)}-V_{ref}-\frac{V_{OFF}}{A}\right)}{\left(1-\frac{1}{A}-\frac{C_{in}}{AC_{out}}\right)^{N-k+1}}\left(1-\frac{1}{A}\right)^{N-k}+V_{ref} \qquad (4.10)$$

En utilisant l'équation (4.9), les mêmes courbes que précédemment sont tracées en Figure 87. L'influence de la valeur du gain est faible. A partir d'un gain |*A*|>50 environ, ce qui est facilement atteignable même pour un inverseur, on retrouve la même conclusion que pour l'approximation de gain infini.

4.3.5 Influence des interrupteurs

Les interrupteurs doivent être réalisés en CMOS si l'excursion du signal qu'ils transmettent est trop proche de l'excursion des alimentations. Cela dépend de la dynamique de la tension d'entrée. Il s'agit de garder en tête que si le CMOS n'est pas nécessaire, on peut gagner de la place, en ne mettant qu'un seul transistor, idéalement un NMOS en technologie à substrat P. Dans le cas où on réalise un système complet, comme ici l'imageur entier, il peut être intéressant de co-réaliser le pixel et l'accumulateur pour fournir à ce dernier une tension d'excursion adéquate.

La taille des interrupteurs influe sur le courant qu'ils peuvent laisser passer et donc sur les temps de montée des signaux. Elle est un compromis entre surface occupée et rapidité du circuit.

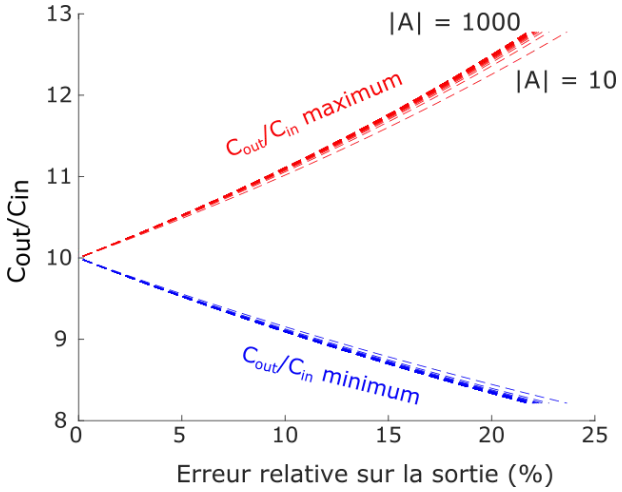


Figure 87 : Maximum et minimum requis sur le rapport C_{out}/C_{in} en fonction de l'erreur relative sur la sortie, pour différentes valeurs de gain fini et les mêmes paramètres que pour la Figure 86.

Dans un circuit à capacités commutées il y a des injections de charges : lorsqu'un interrupteur s'ouvre, les charges formant son canal doivent être évacuées, tout comme elles doivent être acheminées lorsqu'il se ferme. Cela injecte des charges dans le circuit SC en plus des charges qui représentent le signal. Il existe différentes techniques pour limiter ces injections [120 - Razavi 2000]. On peut les séparer en deux catégories : celles qui rajoutent des éléments au circuit et celles qui n'en rajoutent pas.

Les premières, comme l'utilisation d'interrupteurs CMOS ou de transistors fantômes (*dummy*) impliquent un compromis précision-surface occupée par le circuit. La nécessité de les utiliser dépend des spécifications du système ; leur utilité peut être mesurée en simulation. Par exemple, vu nos spécifications, nous nous contentons dans ce travail de l'utilisation de commutations lentes (technique de la deuxième catégorie) pour limiter du mieux possible les injections de charges. L'impact de l'injection de charges résiduelles dans le circuit est jugé en simulation.

Dans ce chapitre, nous avons discuté quelques possibilités d'implémentation analogique de la combinaison linéaire de pixels. Il en ressort que pour les systèmes dont la contrainte majeure est la limitation de surface, un calcul séquentiel est préférable : les différentes entrées sont chacune successivement multipliées par leur coefficient puis accumulées. Cela permet également une combinaison à coefficients entiers sans aucun multiplieur, ce qui réduit la surface occupée. Une accumulation de charges est choisie pour notre circuit plutôt qu'une accumulation de courants vus les faibles courants de photodiodes. Un circuit à capacités commutées à base d'inverseur a donc été défini pour notre application. Ce circuit a été analysé dans un cas général pour discuter de son dimensionnement du point de vue des compromis entre surface et précision. En particulier le ratio entre les capacités d'entrée et de sortie a une valeur minimale qui dépend du nombre d'accumulations souhaitées par l'application. Mais si une erreur sur le résultat est tolérée, alors le désappariement n'est pas problématique et le dessin simplifié.

Maintenant que nous avons étudié le cœur analogique du circuit, c'est-à-dire le PE, nous allons nous intéresser au système entier. Il est composé de ce PE analogique, interfacé d'une part à la matrice de pixels dans laquelle il sera implémenté, et d'autre part à une commande numérique qui permet de le faire fonctionner de façon programmable.

Chapitre 5 : Implémentation du système

Ce travail porte sur la réalisation d'un imageur intelligent dont l'architecture, les spécifications et des considérations de dimensionnement ont été discutées dans les chapitres précédents. Ce chapitre s'attache à l'implémentation du système. Nous allons présenter à nouveau rapidement l'architecture du système avant de détailler l'ensemble de ses éléments, leurs interactions et leur codimensionnement. Nous présenterons donc les éléments primaires du système d'imagerie : la photodiode et le pixel, puis l'élément de calcul et la commande numérique. Ensuite nous discuterons des interconnexions entre circuits numériques et analogiques, avant de conclure par le dessin de la matrice.

5.1 Vue d'ensemble du système

Le système proposé contient une matrice de pixels, le processeur analogique étudié dans le chapitre 4 et la commande numérique de l'ensemble comme le montre la Figure 88. La matrice est divisée en motifs de 4 macropixels, groupant chacun 3x3 pixels et un accumulateur qui sert de PE. Les pixels sont de deux types : avec ou sans mémoire. Un pixel sur 4 possède une mémoire tandis que les PE sont au nombre de 1 pour 9 pixels. Le motif de 6x6 pixels représenté en Figure 88 est donc le motif élémentaire de la matrice.

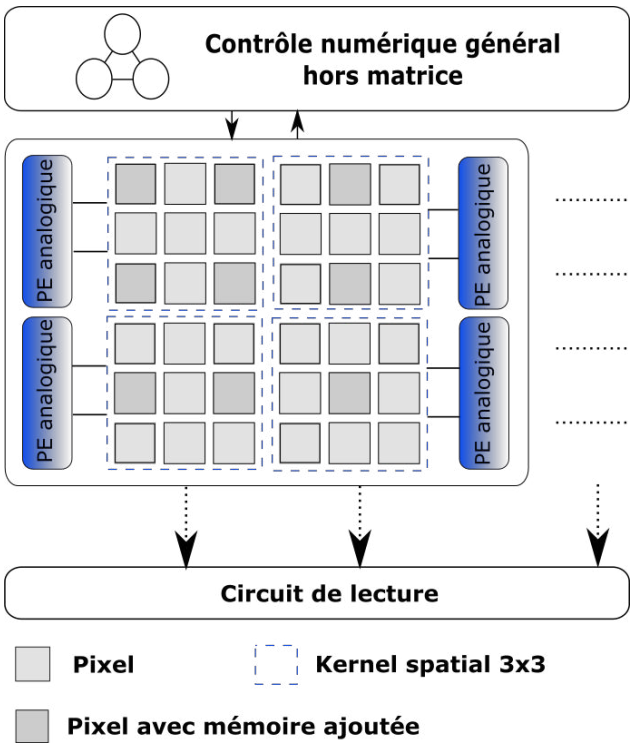


Figure 88 : Vue d'ensemble de l'imageur intelligent proposé.

Les spécifications du système se portant sur une technologie standard, la technologie AMS 0,35µm (C35B4C3) est choisie pour cette preuve de concept. Cette technologie comporte 4 niveaux de

métal et ne présente pas de particularité propre à certaines technologies dites « optiques » (comme des dopages particuliers induisant une meilleure efficacité quantique des photosites).

5.2 Photodiode

En considérant les photodétecteurs décrits dans le chapitre 1, les jonctions PN sont choisies parce qu'elles offrent des performances adéquates pour notre application et sont fabriquées par des procédés bien connus par les fondeurs (rappelons que l'objectif dans ce travail n'est pas l'optimisation du photosite ni du pixel, mais bien une preuve de concept algorithmique et architecturale au niveau système). Dans la technologie AMS 0,35µm, une jonction PN peut être réalisée par une jonction substrat-nWell, substrat-nDiff ou pWell-nDiff. Pour capter le plus de photons possible, on préfère prendre la jonction la plus enterrée : substrat-nWell. Il s'agit donc d'un carré de puits dopé N, avec la cathode en anneau sur tous les côtés du carré. Un anneau de garde (*guardring*) en nWell branché à V_{DD} est ajouté pour éviter la propagation des charges entre les photodiodes et/ou vers les circuits de lecture et de calcul (Figure 89). La photodiode est de taille 30x30µm², ce qui permet d'assurer un bon captage des photons et un impact limité de l'anneau de garde sur la surface perdue (l'espace minimal entre l'anneau et la cathode est fixée par la technologie). La photodiode est donc conçue pour être robuste, mais n'est pas encore optimisée dans le cadre de ce travail.

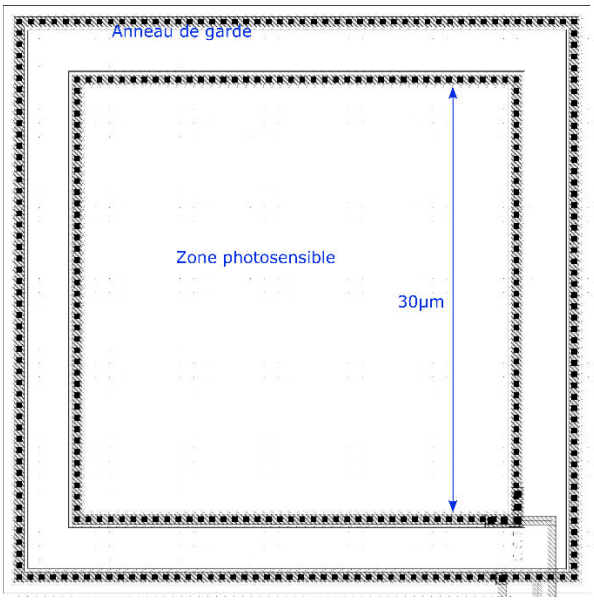


Figure 89 : Dessin de la photodiode avec son anneau de garde.

Une fois dessinée, les parasites résistifs et capacitifs sont extraits de la photodiode pour pouvoir la caractériser. En simulation la photodiode se comporte alors environ comme une capacité de 90fF comme l'illustre la Figure 90 représentant le potentiel de la cathode en fonction du temps pendant l'intégration de la lumière.

La pente de la photodiode dessinée n'est pas parfaitement droite : la capacité varie selon la tension. Dans la suite du dimensionnement on utilise donc cette photodiode dessinée et non simplement modélisée par une capacité en parallèle d'une source de courant. Pour un temps d'intégration de 300ns la photodiode peut aller jusqu'à 1000nA avant d'être éblouie. Le nombre de charges maximum stockables est donné par l'équation :

$$Charges_{max} = \max(I_{ph}) * T_{int} = 300\ fC \qquad (5.1)$$

Les caractéristiques de la photodiode sont résumées dans le Tableau 9. La photodiode est utilisée au sein d'un pixel, qui va être détaillé dans la partie suivante.

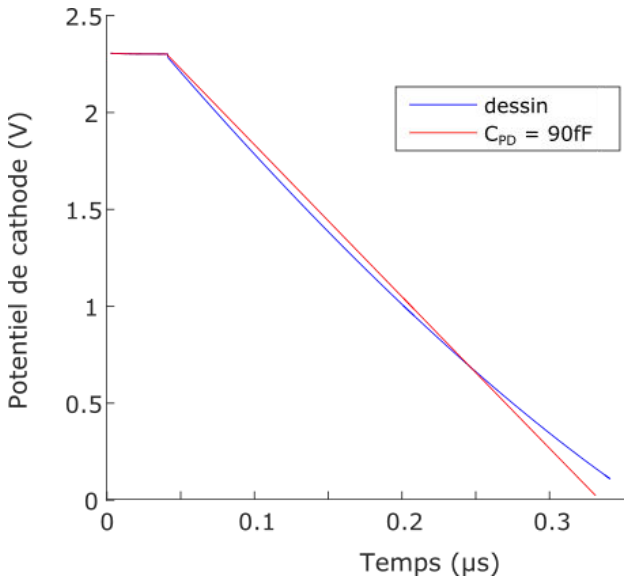


Figure 90 : Potentiel de cathode (V_{PD}) en fonction du temps : (bleu) avec la photodiode dessinée et (rouge) avec une capacité équivalente de 90fF.

Type de photodiode	Puits N - substrat P
Surface de photodiode	30x30µm²
Capacité de photodiode	90fF
Excursion d'entrée	324×10^{-15} C

Tableau 9 : Caractéristiques de la photodiode.

5.3 Pixel

5.3.1 Architecture

Le circuit de PE choisi, à capacités commutées (voir chapitre 4), prend en entrée une tension. Il faut donc un pixel dont la sortie est une tension. Les pixels linéaires à sortie en tension sont les pixels APS présentés au chapitre 1. Pour pouvoir conserver la valeur du pixel disponible pendant toute l'opération analogique, le pixel 4T est choisi. Un transistor de reset supplémentaire est utilisé sur la photodiode afin d'éviter son éblouissement pendant les opérations analogiques (ses charges supplémentaires déborderaient à travers le transistor de transfert *TX* ouvert). Le pixel utilisé est donc présenté en Figure 91. La photodiode intègre la lumière, puis à la fin du temps d'intégration la charge est transférée sur la capacité de diffusion C_{diff} via le transistor *TX*. Un transistor en drain commun (SF pour *source follower*) suivi d'un interrupteur de sélection permet d'avoir accès quand on le souhaite à la valeur de la tension de la capacité.

Pour réaliser une différence temporelle, il faut pouvoir stocker le résultat de 2 intégrations de lumière successives. On insère donc dans les « pixels à mémoire » une seconde chaîne de lecture (TX, capacité, drain commun et transistor de sélection) comme le montre la Figure 92. Cela ne permet pas d'avoir un calcul de convolution spatiale en même temps qu'une mémorisation d'image, mais avec la cadence visée ça ne devrait pas être un problème (25fps demandé seulement).

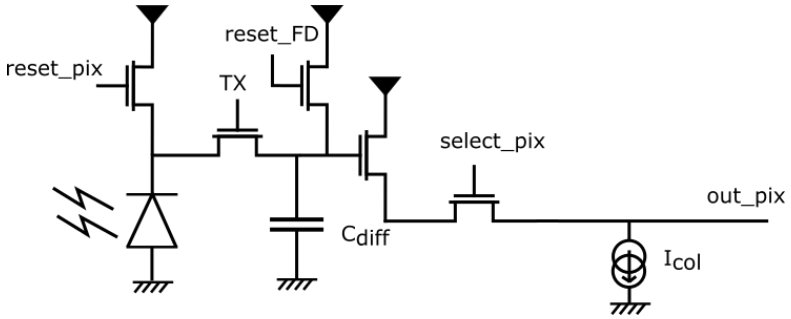


Figure 91 : Schéma du pixel utilisé.

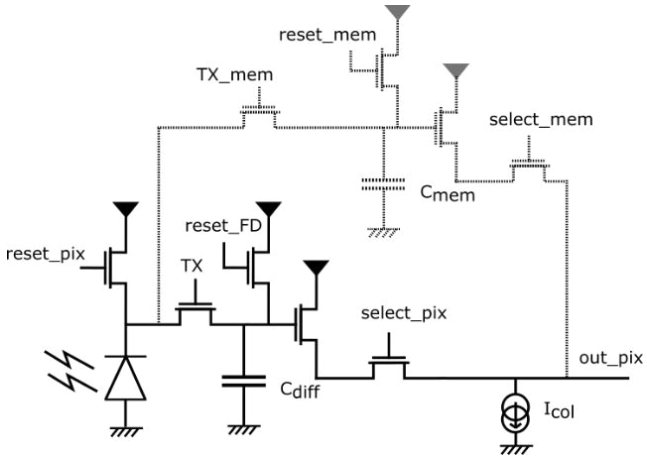


Figure 92 : Pixel avec possibilité de mémorisation supplémentaire.

Dans les pixels sans mémoire, on ajoute un transistor *TXmem* fantôme (*dummy*) : un interrupteur connecté d'un côté à la photodiode, et à la masse de l'autre, avec toujours 0 en commande. Cela permet d'obtenir la même réponse sur un pixel classique comme sur un pixel à mémoire.

La source de courant I_{col} représentée sur les figures ci-dessus permet de fournir du courant au transistor en drain commun afin qu'il puisse reporter sa tension de grille sur sa source, i.e. il s'agit de la source de polarisation du pixel, une source de courant constant. Classiquement elle est commune à toute une colonne de pixels : puisque les lignes sont lues les unes après les autres, la même source sert à tous les pixels d'une même colonne. Dans l'architecture d'imageur proposée dans ce travail, les pixels sont lus les uns après les autres au sein d'un macropixel, mais tous les macropixels travaillent en parallèle (chaque PE sélectionne ses pixels tour à tour pour en accumuler les valeurs). Il s'agit donc ici d'une source commune à un macropixel, et non à une colonne. On l'appellera donc I_{mac}.

La répartition des sources de courant en macropixels plutôt qu'en colonne implique un nombre plus important de sources, proportionnel à M² (avec M le nombre de colonnes ou lignes) et non M. Cependant, la consommation d'énergie n'est pas influencé de la même manière à tout moment. En effet une source de polarisation consomme du courant seulement lorsque son pixel est sélectionné. Pour une fonction de prise d'image uniquement, la répartition des sources en colonne ou en macropixel donne donc la même consommation d'énergie. Le seul surplus d'énergie consommée vient des calculs effectués dans la matrice.

5.3.2 Dimensionnement

Avec le dimensionnement de la photodiode déjà fixé, il reste à définir les transistors, les capacités de diffusion et la source de courant de macropixel. Pour des raisons de surface, comme dans la plupart des imageurs de la littérature, les interrupteurs et le SF sont de simples transistors NMOS de taille minimale. Les transistors de reset, de TX et de sélection fonctionnent comme des interrupteurs.

Le transistor SF quant à lui est actif, il doit travailler en inversion modérée (la faible inversion provoque beaucoup de désappariement et la forte inversion consomme beaucoup d'énergie). La source de courant I_{mac} doit donc fournir un courant pour garantir l'inversion modérée du SF, c'est-à-dire une tension d'*overdrive* V_{OV} d'environ 200mV. On prend pour cela une source de 5µA :

$$I_{mac} = I_D = \frac{\mu CW}{2L}(V_{GS} - V_{th})^2 = \frac{\mu CW}{2L}(V_{OV})^2 \qquad (5.2)$$

La source de courant doit être en saturation, donc sa tension de grille V_{GImac} doit vérifier les inéquations suivantes :

$$V_{GI_{mac}} > V_{th} \quad \text{et} \quad V_{GI_{mac}} < V_{sourceSF_{min}} + V_{th}$$

$$\text{i.e.} \quad V_{th} < V_{GI_{mac}} < V_{sourceSF_{min}} + V_{th}$$

avec V_{th} la tension de seuil, égale à 0.7V environ dans cette technologie. On prend donc V_{GImac} = 825mV = $V_{DD}/4$ et $V_{sourceSFmin}$ = 300mV pour la facilité d'implémentation et une bonne marge face au seuil des deux côtés :

$$0.7V = V_{th} < V_{GI_{mac}} = 0.825V < V_{sourceSF_{min}} + V_{th} = 1V \qquad (5.3).$$

La marge entre V_g et V_{th} est donc de 125mV et celle entre V_g et $V_{sourceFmin}+V_{th}$ est de 175mV, avec une alimentation à un quart de l'alimentation principale, facilement implémentable. Pour fournir 5µA, avec des tailles pas trop petites pour limiter le désappariement (i.e. supérieures au micromètre), on peut alors prendre W_{source} = 2.25µm et L_{source} = 3µm.

La tension minimale de la source du SF a donc été fixée à 300mV pour garantir le fonctionnement de la source de courant, mais on veut pour le pixel la plus grande plage de sortie possible, pour étaler l'information. Le potentiel de sortie du pixel est le potentiel du point FD décalé par le SF. Or la lumière est convertie en charges, elles-mêmes transférées sur la FD à la fin de l'intégration. L'excursion du potentiel de la FD dépend donc de la capacité C_{diff}. La Figure 93 présente la sortie du pixel en fonction du photocourant (1000nA correspond à l'éblouissement de la photodiode), pour différentes C_{diff}. La plus grande excursion de sortie, tout en respectant la valeur minimale pour le fonctionnement de la source de courant, est donc donnée par C_{diff} = 200fF.

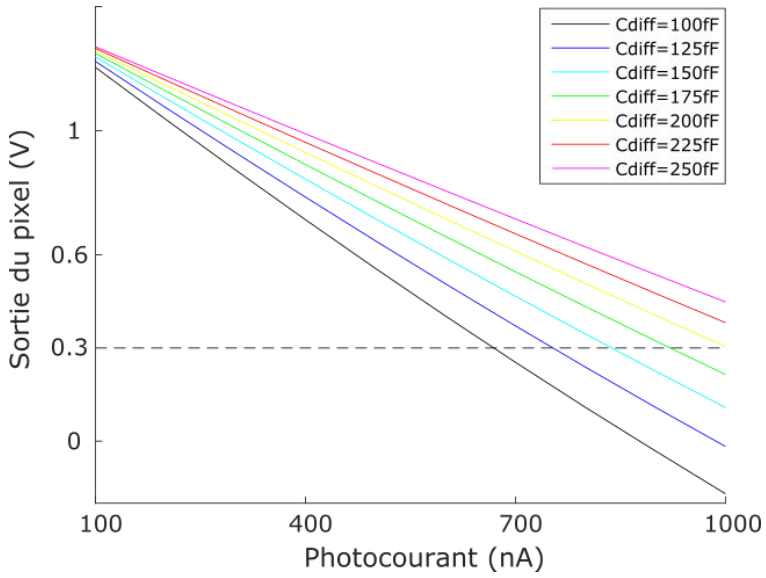


Figure 93 : Sortie du pixel (i.e. source du SF) en fonction du photocourant pour différentes capacités C_{diff}.

La Figure 94 montre que la source de courant délivre bien 5µA sur la plage d'excursion de sortie du pixel (pixel output > 0.3V) et que celle-ci est bien linéaire avec la tension du nœud FD. La Figure 95 résume ceci en montrant la sortie du pixel en fonction du photocourant. L'excursion du pixel est [300mV ; 1.4V]. Le dimensionnement et les caractéristiques de ce pixel sont résumés dans le Tableau 10.

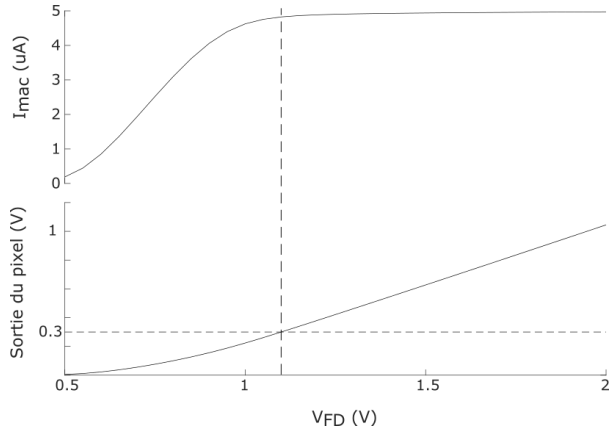


Figure 94 : Sortie du pixel et courant de colonne en fonction de la tension du nœud FD (représentatif de la sortie de photodiode). Pour une sortie de pixel supérieure à 300mV, la source de courant délivre 5µA.

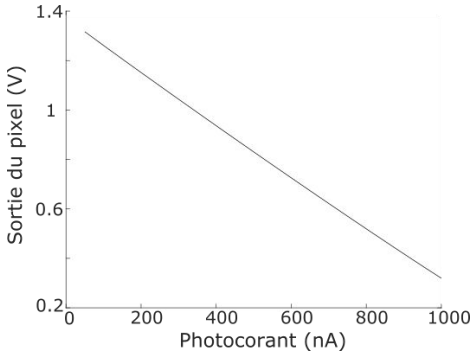


Figure 95 : Sortie du pixel (i.e. source du SF) en fonction du photocourant (1000nA est la limite d'éblouissement de la photodiode pour Tint = 300ns).

Nombre de transistors	6 (pixel sans mémoire) 9 (pixel avec mémoire)
Taille des transistors	0.4µm/0.35µm (min)
Capacité de diffusion	200fF
Courant de polarisation	5µA
Taille de la source de courant	2.25µm/3µm
Excursion de sortie	[0.3 ; 1.4] V

Tableau 10 : Caractéristiques du pixel et de sa photodiode.

5.4 Accumulateur à capacités commutées : dimensionnement

Après la discussion du chapitre 4, cette section donne un dimensionnement de l'accumulateur (rappelé en Figure 96) adapté à notre système.

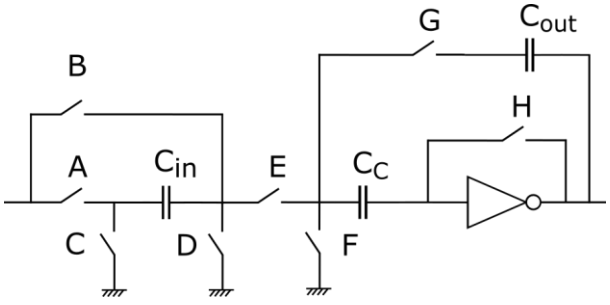


Figure 96 : Rappel de la structure de l'accumulateur.

5.4.1 Inverseur

Conformément au chapitre précédent, pour minimiser C_{out}/C_{in}, le point de polarisation de l'inverseur est choisi comme étant le milieu de l'excursion de sortie du pixel, soit 850mV. On préférera

$V_{DD}/4$ = 825mV qui est plus facile à implémenter au prix d'une très légère perte d'excursion utile comme on va le vérifier ci-dessous.

Le gain A de l'inverseur en valeur absolue doit être grand devant 1 pour que l'accumulateur fonctionne et plus il est grand moins il y aura d'erreur introduite (chapitre 4). On se place donc en faible inversion : on a un fort gain mais un faible courant. Cela a pour avantage de limiter la consommation, et pour désavantage de réduire la vitesse du circuit. On peut avoir un inverseur de polarisation $V_{DD}/4$ avec les dimensions W_n = 6.45µm, W_p = 3µm, L_n = 2µm, L_p = 9µm. Sa plage de sortie est environ [0.3 ; 1.8V] et son gain maximum de 300 comme le montre la Figure 97 (gain statique, au vu des fréquences qui seront utilisées (voir le chapitre 6)). Cela reste faible par rapport à un amplificateur traditionnel puisqu'il s'agit d'un simple inverseur.

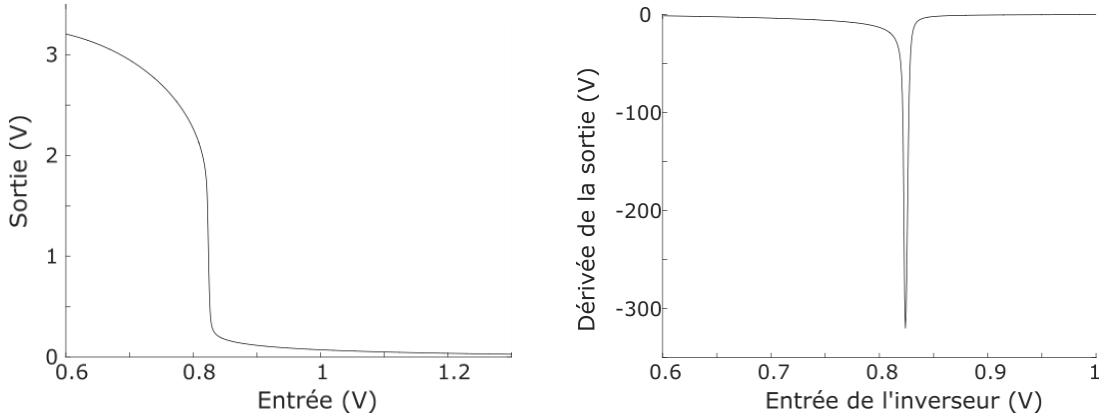


Figure 97 : Relation entrée/sortie et gain de l'inverseur (Wn = 6.45u, Wp = 3u, Ln = 2u, Lp = 9u).

5.4.2 Capacités C_{in} et C_{out}

Comme on l'a vu au chapitre 4, c'est le rapport C_{out}/C_{in} qui est important pour la fonction d'accumulation. Son minimum dépend du nombre d'accumulations souhaitées, de l'excursion du pixel et de celle de l'inverseur selon l'inéquation 4.6 rappelée en 5.4 ci-dessous.

$$\frac{2 \times N_{Accu} \times \max(|V_{in} - V_{ref}|)}{Range} \leq \frac{C_{out}}{C_{in}} \qquad (5.4)$$

Ici la plage d'excursion du pixel est [0.3V ; 1.4V] et celle de l'inverseur [0.3V ; 1.8V] ce qui donne un *Range* égal à 1.05V pour V_{ref} = 0.825V selon l'équation ci-dessous.

$$Range = 2 * \min(|V_{AOmax} - V_{ref}|) = 2 * (0.825 - 0.3) = 1.05\text{V} \qquad (5.5)$$

Le nombre d'accumulations N_{Accu} dépend de l'application visée. En convolution spatiale il s'agit de la somme des valeurs absolues des coefficients du masque (voir le Tableau 11 pour des exemples) tandis qu'une différence temporelle nécessite au minimum 2 accumulations (plus si on souhaite amplifier le signal de sortie). Le Tableau 12 donne le rapport C_{out}/C_{in} minimal selon l'inéquation 5.4 pour différentes valeurs de V_{ref} et du nombre d'accumulations maximum.

On vérifie tout d'abord que le changement de V_{ref}, de sa valeur idéale à $V_{DD}/4$, n'influence pas beaucoup la valeur du rapport minimum. Ensuite, la différence entre 8 accumulations autorisées ou 9 est plus significative, mais pour être suffisamment versatile, on choisit 9 accumulations autorisées. Pour avoir une marge (notamment pour être sûr d'éviter la zone non-linéaire de l'inverseur), un rapport C_{out}/C_{in} égal à 10 est retenu.

Le rapport C_{out}/C_{in} étant dimensionné pour 9 accumulations, les résultats de différences temporelles (2 accumulations) n'utiliseront pas toute la plage d'excursion de sortie de l'inverseur. La différence temporelle amplifiée permet donc d'exploiter mieux cette excursion.

	Masque ou séquence d'accumulations	Nombres d'accumulations
Simple	[1 0 -1]	2
Sobel	$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$	8
Moyenneur	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	9
Différence temporelle	[-1 ; 1]	2
Différence temporelle amplifiée (par 2)	[-1 -1 ; 1 1]	4

Tableau 11 : Nombre d'accumulations nécessaires pour différents exemples de masques de convolution spatiale ou de différences temporelles.

Range	V_{ref}	max($\lvert V_{in}$-$V_{ref}\rvert$)	N_{accu}	C_{out}/C_{in} minimal
1,05	0,85 (idéal)	0,55	8	8,4
			9	9,4
	0,825 (V_{DD}/4)	0,575	8	8,8
			9	9,8

Tableau 12 : Calcul du rapport de capacités minimal pour différents cas de tension de référence ou nombre d'accumulations.

Le dimensionnement de la capacité C_{in} résulte d'un compromis entre temps d'établissement et surface d'une part, et précision d'autre part. En effet, une valeur élevée de capacité induira non seulement une surface importante mais aussi un temps de chargement long selon la valeur de la source de courant I_{mac}. Une valeur faible au contraire convertira la tension de sortie du pixel en une quantité de charges (la grandeur portant le signal dans l'accumulateur) faible devant le bruit dû à l'injection de charges. Pour déterminer le bon compromis, une analyse paramétrique a été menée.

L'accumulateur est simulé sous forme schématique, avec des interrupteurs en taille minimale, une capacité de compensation de 500fF (choisie très grande pour limiter son influence sur cette étude paramètrique, cf. section suivante) et une capacité de sortie égale à 10 fois la capacité d'entrée. Les commandes des interrupteurs sont celles d'une convolution spatiale par le masque de Sobel. L'entrée de l'accumulateur est la sortie de pixels recevant des photocourants donnés. Les simulations sont réalisées pour un ensemble représentatif des illuminations de macropixels possibles (voir Figure 98, sans les cas d'illumination uniforme pour éviter une erreur relative infinie (résultat attendu nul sur un Sobel)), pour chaque valeur testée de C_{in}. Chaque résultat ($Vout_{sim}$) est comparé à celui d'un accumulateur idéal ($Vout_{idéal}$), et les erreurs en valeur absolue sont moyennées sur l'ensemble des sets d'illuminations testés. L'erreur relative moyenne présentée sur la Figure 99 en fonction de la valeur de C_{in} est donc calculée par :

$$Err_{relat,moy} = \text{moy}\left(100 \times \left|\frac{V_{out_{idéal}} - V_{out_{sim}}}{V_{out_{idéal}} - V_{ref}}\right|\right) \quad (5.6).$$

Cette erreur est une fonction décroissante de C_{in}. Or la surface occupée par une capacité est proportionnelle à sa valeur. La Figure 99 montre que 20fF est un bon compromis. Le temps d'établissement est convenable comme le montrent les résultats transitoires au chapitre 6. On prend donc C_{in} = 20fF et C_{out} = 10xC_{in} = 200fF.

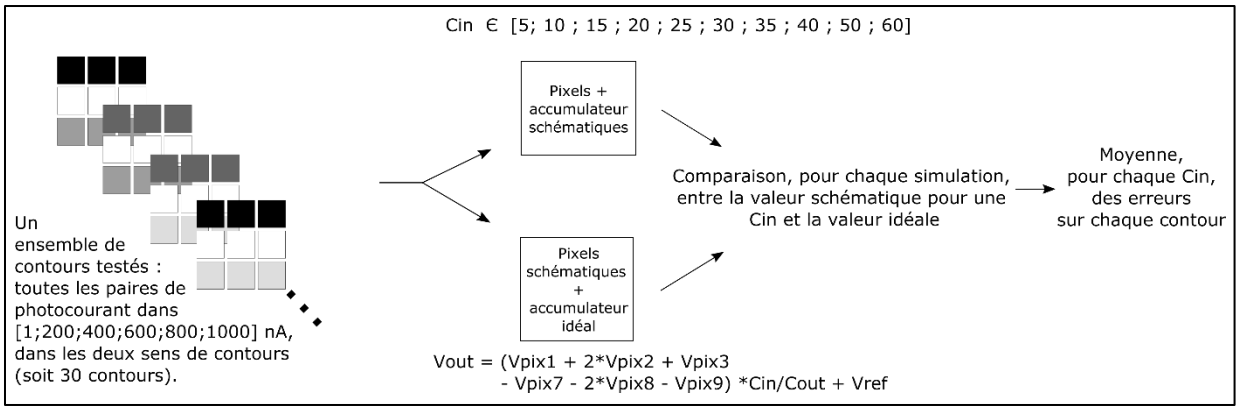


Figure 98 : Procédure de test pour déterminer C_{in}.

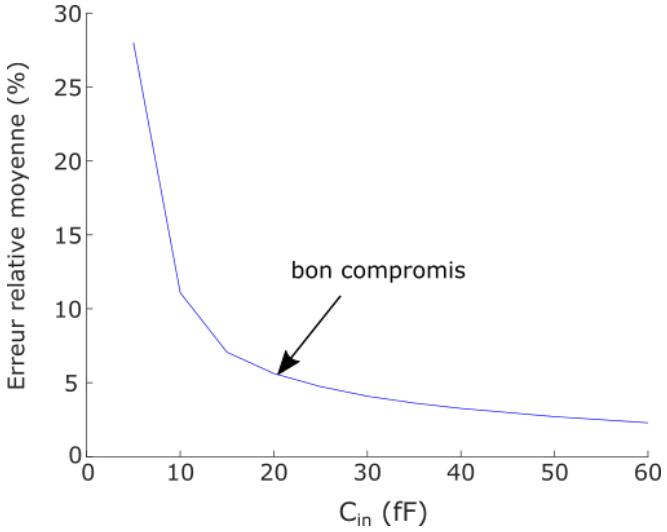


Figure 99 : Erreur moyenne en sortie de l'accumulateur relativement à un accumulateur idéal, sur un set représentatif de contours pour une simulation d'un masque de Sobel, en fonction de C_{in}.

5.4.3 Capacité C_C

Tout d'abord, il convient de vérifier que la capacité de compensation C_C est bien nécessaire dans notre cas. Pour cela, la Figure 100 présente les résultats d'une simulation de Monte-Carlo de l'inverseur dimensionné ci-dessus dans la technologie choisie (AMS 0.35µm). Le point de repos d'inverseur, de valeur nominale 0.825V, peut en fait varier entre 0.73 et 0.93 V. Or d'après l'équation 4.4bis on a une erreur sur la sortie de l'accumulateur de :

$$Erreur_{off} = N_{accu} \times \frac{C_{in}}{C_{out}}\left(V_{off} - V_{ref}\right) + V_{off} - V_{ref} = \left(\frac{N_{accu}}{10} + 1\right) \times \left(V_{off} - V_{ref}\right) \quad (5.7)$$

$$soit \quad -170mV < Erreur_{off} < 190mV \quad pour\ N_{accu} = 8$$

Cette erreur est trop grande par rapport à l'excursion de la sortie [0.3V ; 1.8V], l'ajout de la capacité C_C est donc bien nécessaire.

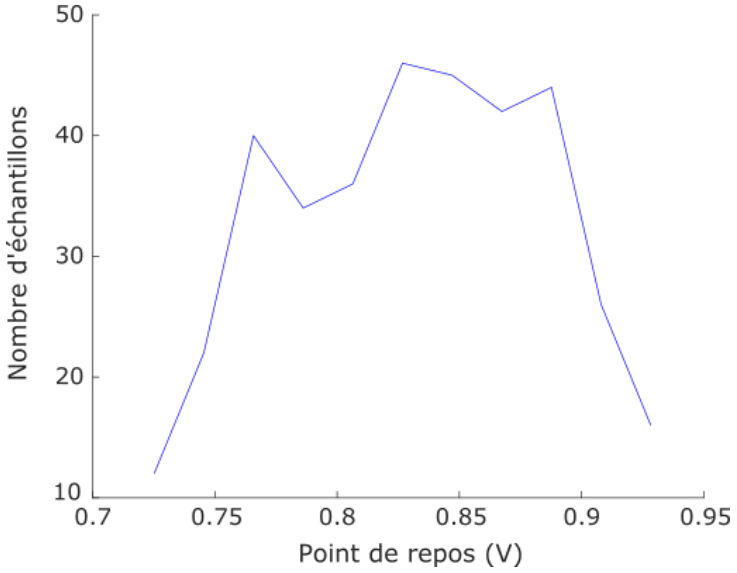


Figure 100 : Résultat de simulation MonteCarlo de la tension de repos de l'inverseur dimensionné en AMS 035 (300 points simulés ; tracé en les regroupant sur 10 intervalles).

Pour déterminer la valeur de la capacité C_c on simule un masque de Sobel horizontal sur un macropixel (TR) recevant différentes images de contours puis on procède à une analyse paramétrique du comportement du système. Celle-ci est conduite dans les mêmes conditions que celle sur la capacité d'entrée (cf. Figure 98) en prenant C_{in} = 20fF et en faisant varier C_C entre 10fF et 210fF. L'impact sur l'erreur moyenne relative est donné en Figure 101. On choisit 70fF pour minimiser l'erreur introduite tout en limitant la surface occupée par C_C.

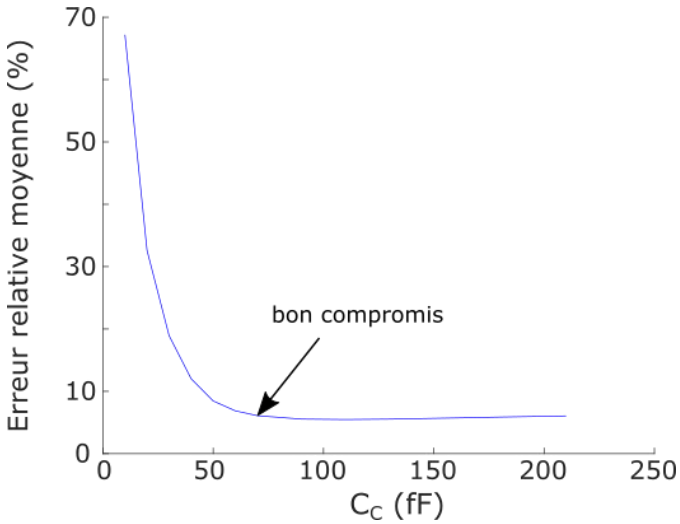


Figure 101 : Erreur moyenne en sortie de l'accumulateur relativement à un accumulateur idéal, sur un set représentatif de contours pour une simulation d'un masque de Sobel, en fonction de C_C.

5.4.4 Interrupteurs

Les derniers éléments à dimensionner de l'accumulateur sont les interrupteurs. Pour des raisons de surface, ils doivent être les plus petits possibles, et notamment en NMOS si les excursions de signaux le permettent. Il faut vérifier que le signal ne s'approche pas trop de la tension haute V_{DD} qui servirait à commander la grille des interrupteurs :

$$V_{GS} > V_{th} = 0.7V \quad \text{lorsque} \quad V_G = V_{DD}$$

i.e. pour le signal sur la source : $V_S < V_{DD} - V_{th}$ (5.8)

Or l'excursion du pixel est [0.3V ; 1.4V], V_{ref} est 0.825V et l'inverseur travaille entre 0.3V et 1.8V. V_{DD} étant 3.3V, tous les interrupteurs de l'accumulateur sont réalisés en simples transistors NMOS.

Pour leur dimensionnement, comme on l'a vu il faut considérer la vitesse requise pour le système. Ici pour une application de type détection de contours avec deux masques Sobel (horizontaux et verticaux), 2x8 accumulations doivent pouvoir être réalisées entre deux images, à une cadence de 25 images par seconde au minimum (flux vidéo). Une accumulation étant composée de seulement deux phases (chargement sur C_{in} puis transfert sur C_{out}), le circuit n'est pas très contraint en temps. On choisit donc les tailles minimales de la technologie pour les interrupteurs afin de limiter la surface et les injections de charges.

L'ensemble des dimensionnements du PE est synthétisé dans le Tableau 13.

Inverseur		Capacités			Interrupteurs	
W_n/L_n	W_p/L_p	C_{in}	C_C	C_{out}	W_n	L_n
6.45µm / 2µm	3µm / 9µm	20fF	70fF	200fF	0.4µm (min)	0.35µm (min)

Tableau 13 : Dimensionnement des éléments du PE.

5.5 Commande numérique

La matrice de pixels et de processeurs analogiques qui a été conçue sont des parties opératives ; elles doivent être commandées par un circuit numérique. Les commandes liées à notre dispositif peuvent être regroupées en 3 catégories :

- la commande du pixel : acquisition, transfert et remise à zéro,

- la commande du PE : remise à zéro, chargement d'entrée et transfert de charges, et

- la commande de l'adressage : sélection de pixels dans le macropixel et sélection de lignes et colonnes de macropixels.

5.5.1 Spécifications et fonctionnement

En premier lieu, ce circuit doit commander les pixels : signaux de commandes des interrupteurs de *reset* de la photodiode, de transfert (TX) et de *reset* de la diffusion (voir la Figure 102 pour un rappel du circuit schématique). Les temps d'intégration, de reset ou de transfert sont tous programmables par l'extérieur pour assurer la versatilité du système. La Figure 103 présente les diagrammes temporels des signaux que doit fournir le circuit numérique pour la commande des pixels dans le cas d'une image classique et d'une image sous-échantillonnée à mémoriser.

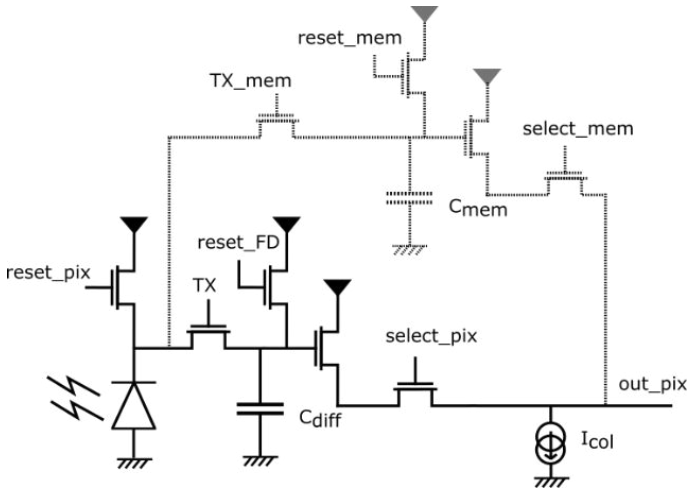


Figure 102 : Rappel du schéma du pixel.

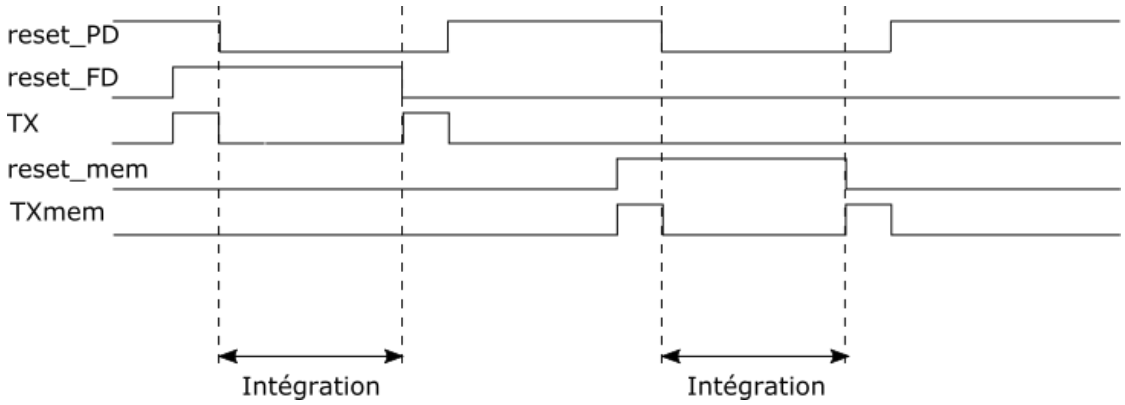


Figure 103 : Diagramme temporel de commandes des pixels (classique et mémoire).

Ensuite, le circuit numérique doit également commander l'accumulateur analogique, c'est-à-dire commander ses interrupteurs pour que les opérations souhaitées soient réalisées. Il y a 3 signaux de commande différents comme l'indique la Figure 104. L'entrée de l'accumulateur change au court d'une opération : la commande numérique doit sélectionner la bonne sortie de pixel (ou la bonne mémoire) au bon moment. Chaque macropixel est relié à 9 pixels et à 2 ou 3 mémoires. Les mémoires sont associées à des pixels différents selon le type de macropixel. Il y a donc 9 signaux de sélection de pixel, 3 de sélection de pixel pour une différence temporelle, et 3 de sélection de mémoire. Les Figure 105 et Figure 106 présentent respectivement un exemple de chronogramme pour une convolution spatiale et pour une différence temporelle. Celle-ci est représentée ici en faisant la différence entre la valeur d'un pixel et la valeur de sa mémoire correspondante. Or l'accumulateur étant capable, pour une convolution spatiale, de faire plus d'accumulations, une différence temporelle peut être multipliée par un coefficient entier comme 3 ou 4 pour amplifier l'écart entre deux valeurs de différences temporelles.

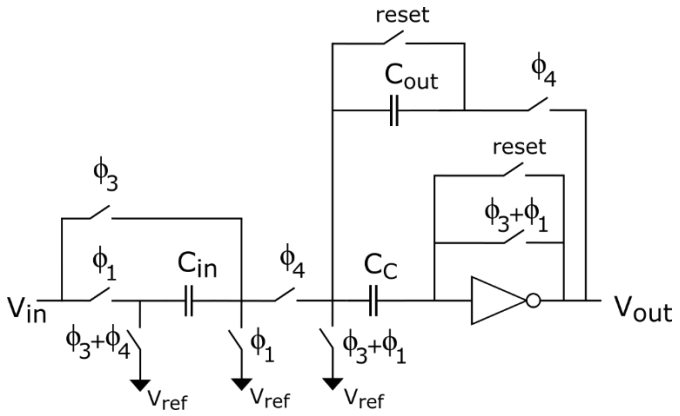


Figure 104 : Rappel du schéma de l'accumulateur avec les commandes.

Afin de limiter le nombre de broches de sortie, et donc la connectique, le prototype est voulu avec une lecture série des sorties. Celles-ci sont les sorties de macropixels dans le cas d'un prétraitement ou des pixels (et/ou mémoires) dans le cas d'une lecture de l'image. Pour lire une image en pleine résolution, puisque les sorties colonnes sont au nombre de 1 par colonne de macropixels, il faut lire 9 pixels lorsqu'un macropixel est sélectionné. L'image pleine résolution doit être lue dans le bon ordre de pixels (ligne par ligne de pixels). L'adressage des pixels dans les macropixels est donc 1-2-3, 1-2-3, … pour la première ligne de macropixels, puis 4-5-6, 4-5-6,… de nouveau pour la première ligne de macropixels, etc., comme le présente le chronogramme de la Figure 107.

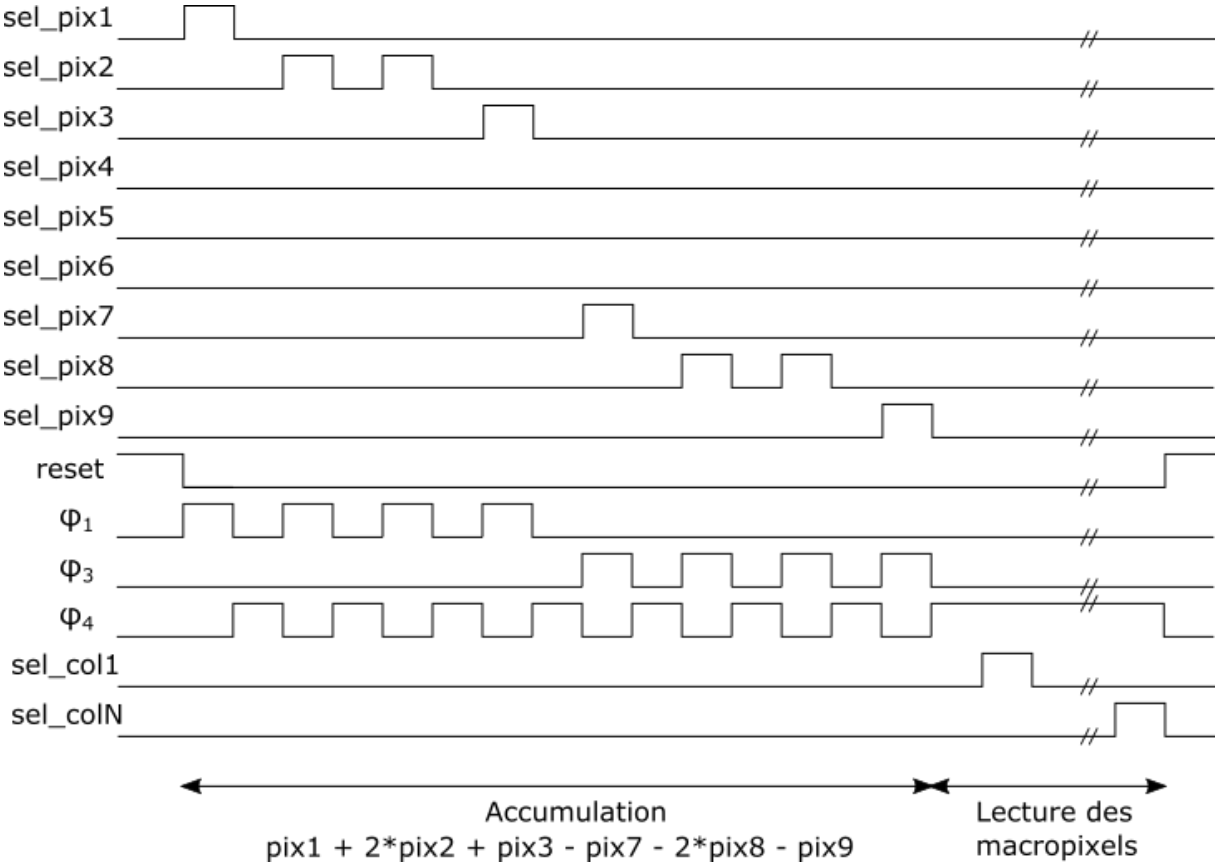


Figure 105 : Diagramme temporel de commande de l'accumulateur pour une convolution spatiale de type Sobel horizontal (masque [1 2 1 ; 0 0 0 ; -1 -2 -1]).

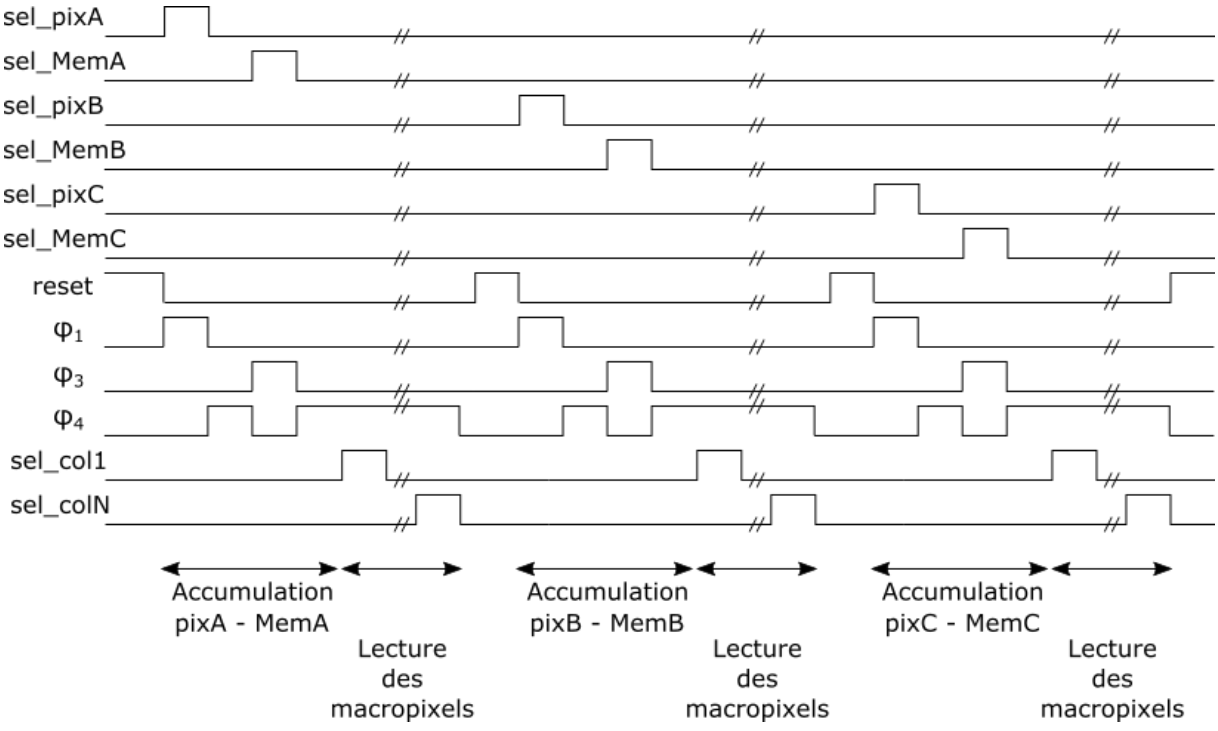


Figure 106 : Diagramme temporel de commande de l'accumulateur pour une différence temporelle (les macropixels, donc les accumulateurs, doivent gérer chacun jusqu'à 3 cycles différence-lecture car ils sont liés à 2 ou 3 mémoires, notées A, B et C).

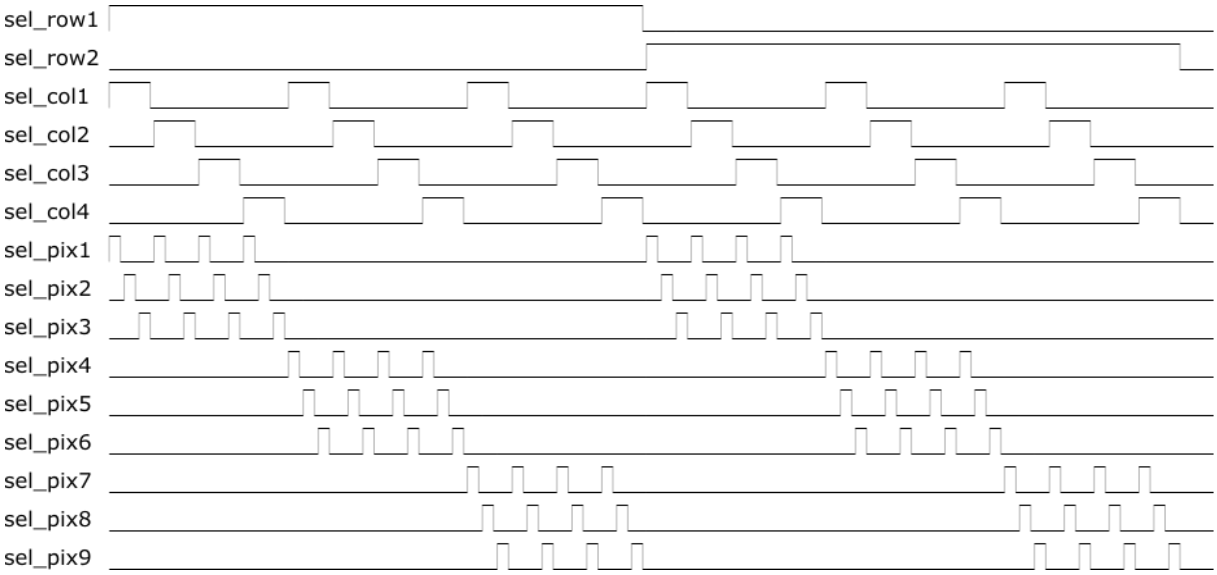


Figure 107 : Diagramme temporel de commande de sélection de ligne, colonne et pixel pour une lecture plein résolution d'un imageur 2(row)x4(col) macropixels, i.e. 6x12 pixels.

Le circuit numérique doit pouvoir être configuré pendant l'utilisation afin d'assurer la programmabilité : sélectionner une convolution spatiale ou une différence temporelle, les coefficients du masque spatial comme le coefficient multiplicatif de différence temporelle, et les temps (d'intégration, de transfert, de lecture,…).

Cette commande numérique a été codée en VHDL selon ces spécifications. L'ensemble du circuit est géré par une machine à états principale. Selon sa commande externe, elle déclenche l'une des actions suivantes :

- une prise d'image,
- une prise d'image à mémoriser,
- une convolution spatiale,
- une différence temporelle,
- une lecture de tous les pixels,
- une lecture des mémoires,
- une configuration pour charger les registres.

Chaque action est gérée par une machine à état dédiée. Un système de signalement de fin de tâche (*acknowkedge*) entre machines à états et avec l'extérieur permet le bon enchainement des tâches. Les paramètres de types coefficients de masque ou de différence, et temps (d'intégration de la lumière, de reset du pixel, de transfert de charges, etc.) sont stockés dans des registres configurables par l'extérieur, via une entrée série. L'ensemble est codé en machine de Moore sauf pour la gestion des sélections de lignes, colonnes ou pixels car cela génèrerait trop d'états.

Les détails de l'implémentation en VHDL sont donnés en annexe 3. Le circuit décrit en VHDL a été testé et est conforme aux spécifications. Il a été synthétisé et co-simulé avec le circuit analogique, il est donc une "preuve de concept" que commander l'accumulateur et un imageur asymétrique ne pose pas de problème.

5.5.2 Place dans la matrice

Puisque les processeurs analogiques sont distribués à travers la matrice de pixels (1 processeur par macropixel de 3x3 pixels), des bus de signaux de commande doivent traverser la matrice en long et en large pour atteindre chaque PE. Il pourrait donc être intéressant de distribuer également les circuits de contrôle numérique pour réduire le nombre de pistes de commandes. Deux aspects majeurs sont alors à prendre en compte.

L'ensemble des lignes de commande représente une trentaine de pistes par macropixel (5 commandes de pixels, 6 commandes d'accumulateur, 15 commandes de sélection, …). L'ensemble du circuit numérique est bien plus grand que cela, mais il pourrait être intéressant d'en mettre une partie dans les macropixels. Or puisque le système est programmable, beaucoup de paramètres sont configurables et stockés dans des registres, comme les temps (d'intégration, de transfert, etc.). Il faudrait donc les transmettre à toute la matrice, ce qui génère plus de pistes que les commandes finales (b>d sur la Figure 108).

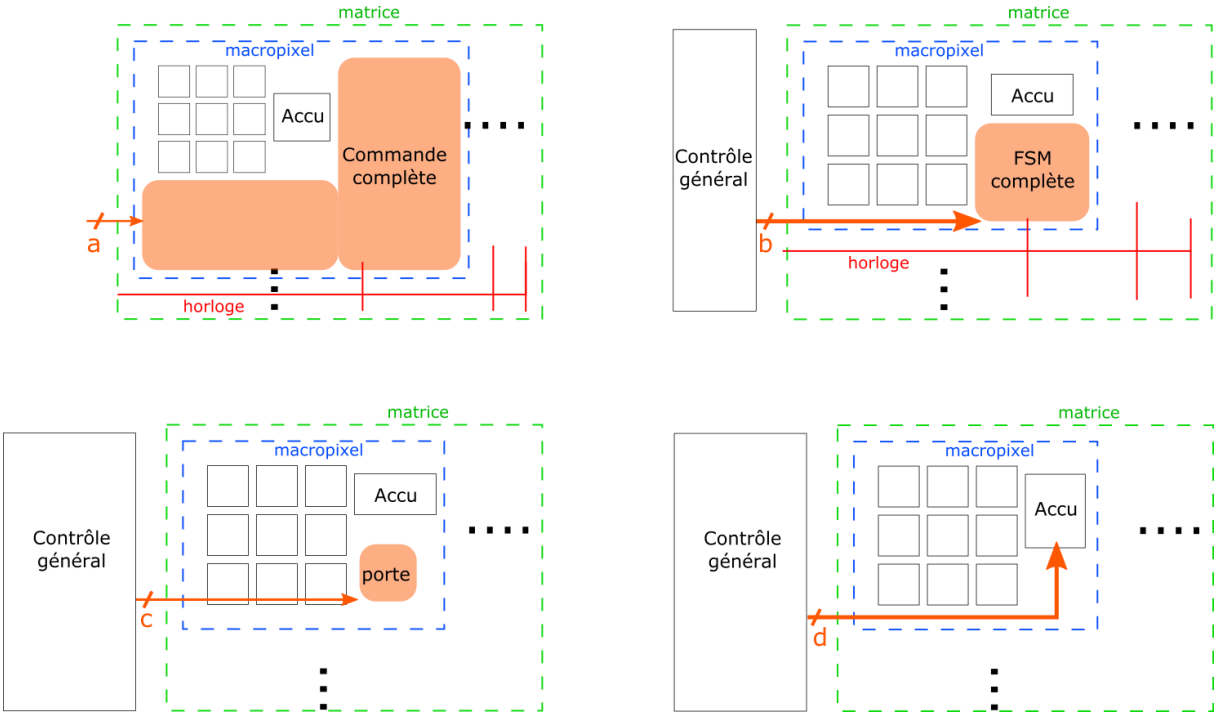


Figure 108 : Différentes possibilités de distribution de l'électronique numérique : tout le circuit dans chaque macropixel, une partie importante (par exemple une machine à états (FSM) de contrôle), seulement quelques portes ou rien (on a b > d > c).

Ensuite certains petits circuits (quelques portes) proches des sorties pourraient être implémentés dans le macropixel. Il s'agit alors de comparer la surface occupée à la surface gagnée en enlevant des pistes. De simples portes logiques permettent en plus de ne pas distribuer une horloge à travers la matrice (voir Figure 108). Par exemple, l'accumulateur a 6 commandes, dont 2 qui sont la somme de deux autres. Deux portes OU placées dans le macropixel pourraient ainsi supprimer deux pistes traversant la matrice. Ces considérations dépendent énormément de la technologie.

Ici, la technologie utilisée est AMS 0.35µm C35B4C3. Dans cette technologie une porte OU a une surface de 13.1x7.3µm² tandis qu'une piste occupe 0.5+0.4µm de large, soit une surface d'environ 50µm² pour un pitch de 50µm. Ainsi la distribution du circuit numérique ne présente pas d'intérêt dans cette technologie. Tout le circuit numérique est donc placé à l'extérieur de la matrice dans le circuit proposé dans ce travail mais ces considérations sont à rediscuter au cas par cas pour chaque technologie.

5.6 Interconnexion entre circuits numériques et analogiques

5.6.1 Commandes

Le circuit numérique a donc des entrées qui le lient à l'extérieur (horloge, remise à zéro, commande de configuration et d'activité), et des sorties qui permettent de commander le circuit analogique (Figure 109).

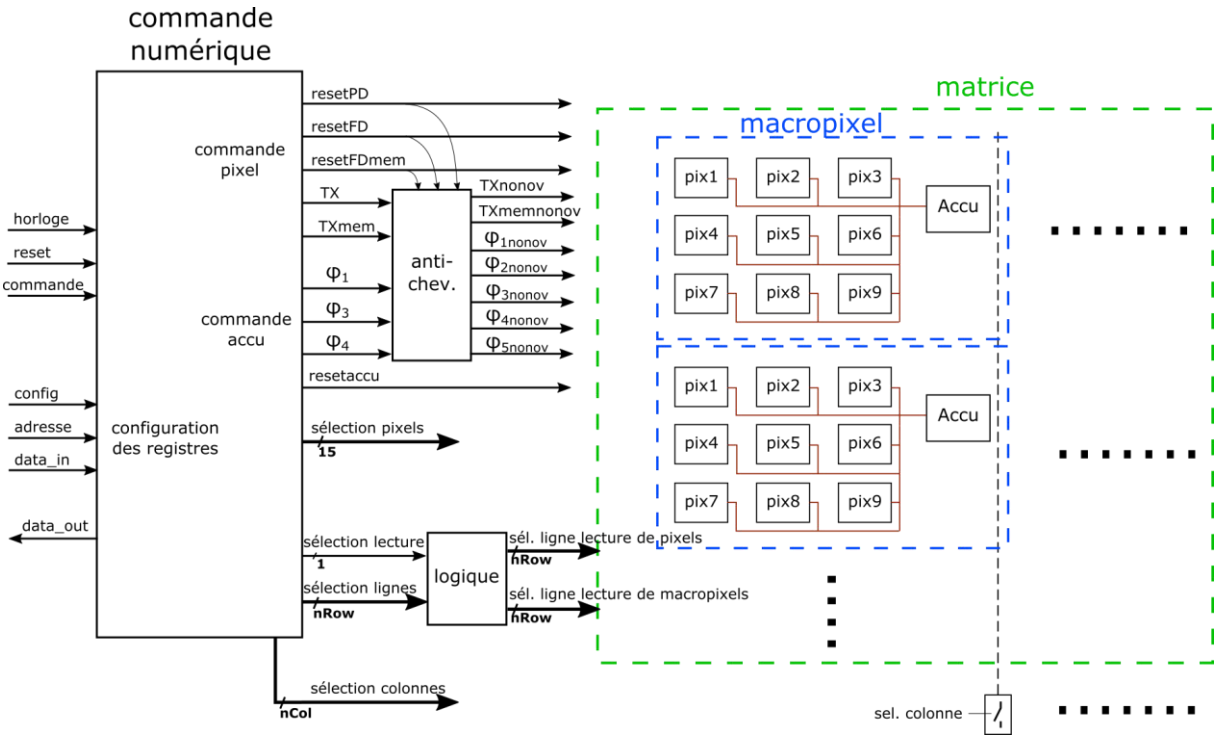


Figure 109 : Schéma bilan du système, avec les connexions de la commande numérique.

On a vu que pour limiter le nombre de broches du circuit la lecture est faite en série. Les sorties de macropixels étant en colonne, il faut sélectionner la colonne en bas de colonne et la ligne par des pistes traversant la matrice. Il doit être possible de lire la valeur de sortie d'un accumulateur ou celle d'un pixel grâce à des interrupteurs dans le macropixel. Pour limiter le nombre de transistors et de pistes parcourant la matrice, on choisit une implémentation en 2 interrupteurs NMOS avec chacun leur commande, l'un pour lire des pixels dans une ligne et l'autre pour lire des sorties d'accumulateurs dans une ligne. Pour une ligne *i* on a donc les signaux :

$$sel\_lecture\_pixel(i) = \text{not}(sel\_lecture) \text{ AND } sel\_ligne(i)$$
$$sel\_lecture\_accu(i) = sel\_lecture \text{ AND } sel\_ligne(i)$$

avec *sel_lecture* un signal issu de la commande numérique qui indique si on lit un accumulateur ou un pixel. La Figure 111 représente graphiquement ce système.

En ce qui concerne la sélection de pixel dans le macropixel, elle est identique pour tous les macropixels en parallèle lorsqu'il s'agit de faire une convolution spatiale. Il y aurait donc 9 pistes de sélection de pixels (autant que de pixels dans un macropixel). En revanche pour une différence temporelle, il faut sélectionner une mémoire puis son pixel associé, dont la place diffère selon le macropixel considéré comme le rappelle la Figure 110. Par exemple, pour une différence temporelle le pixel à mémoire n°2 doit être sélectionné en même temps que le pixel à mémoire n°6, puis leurs mémoires respectives, toujours simultanément ; alors que pour une convolution le n°2 n'est pas sélectionné en même temps que le n°6 qui est dans le macropixel voisin... Il est possible de gérer ceci en ayant des lignes de commandes spécifiques pour les pixels à mémoires, ce qui ferait 9 pistes de commandes pour les pixels (9 pixels par macropixel), 9 autres pour les pixels à mémoires (dans le motif

élémentaire 6x6 il y a 9 mémoires) et 3 pour les mémoires (3 mémoires maximums gérées par un PE), soit un total de 21 pistes. Pour réduire leur surface occupée dans la matrice, il est préférable d'utiliser 2 transistors de sélection par pixel à mémoire, l'un commandé par les pistes de sélections de pixel pour convolution spatiale, et l'autre par une piste de sélection de pixel pour différence temporelle. Ce qui fait 9 pistes de sélection de pixels (9 pixels par macropixel), 3 autres pour les pixels à mémoires et 3 pour les mémoires, soit 15 pistes seulement au total, au prix d'un transistor supplémentaire dans chaque pixel à mémoire. La Figure 111 illustre cette implémentation sur le cas d'un macropixel TR, avec des mémoires sur les pixels 2 et 8 du macropixel. Le pixel 8 a 3 commandes de sélection : la *sel_pix_8* pour être le 8ème dans une convolution spatiale, et *sel_pix_DT_2* et *sel_mem_2* pour être le 2ème pixel à mémoire traité dans une différence temporelle.

L'accumulateur étant plus sensible aux défauts de ses interrupteurs (injection de charges, etc.) on tire 5 pistes dans la matrice (φ_1, φ_3, φ_4, $\varphi_1 + \varphi_3$ et $\varphi_3 + \varphi_4$) plutôt que de doubler des interrupteurs.

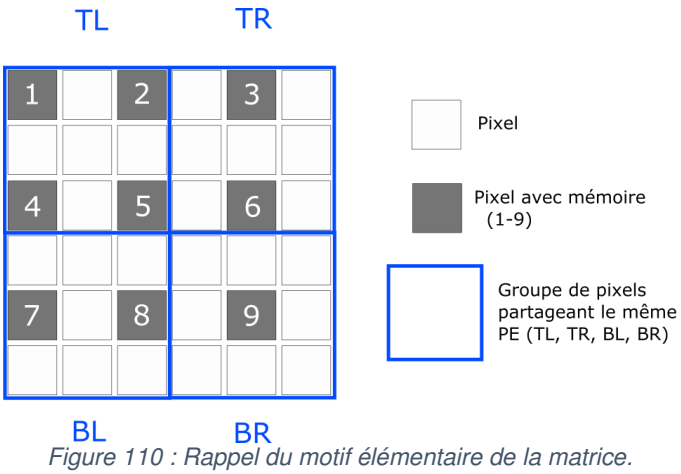


Figure 110 : Rappel du motif élémentaire de la matrice.

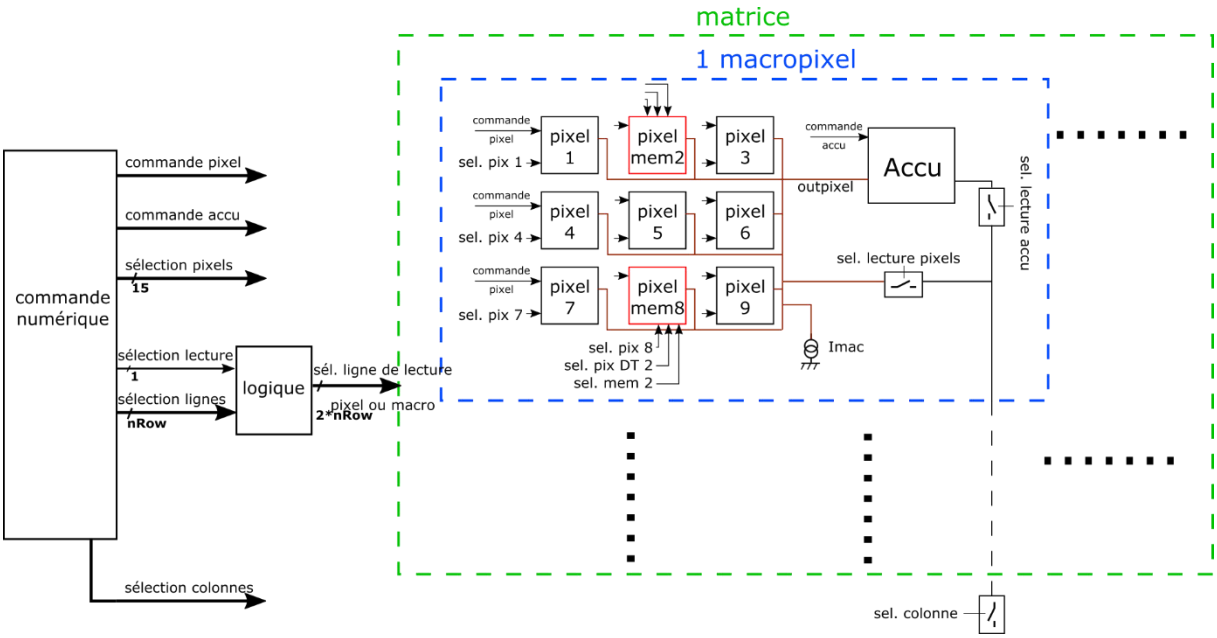


Figure 111 : Schéma du système avec en particulier les sélections de pixels/mémoires et de sorties sur un macropixel à deux mémoires).

5.6.2 Anti-chevauchement

Les signaux numériques issus du bloc de commande sont tous modifiés au front montant d'horloge : l'ouverture de certains peut donc chevaucher la fermeture d'autres. Cela peut entrainer des pertes d'informations dans le circuit analogique : par exemple, dans l'accumulateur, des charges

envoyées vers la tension de référence au lieu d'être accumulées sur la capacité de sortie. Des circuits supplémentaires sont donc nécessaires pour garantir que les signaux de commande des interrupteurs ne se chevauchent pas, c'est-à-dire qu'un interrupteur ne doit se fermer que si ceux d'une autre commande sont complètement fermés (*non-overlapping*). Des circuits simples d'anti-chevauchement sont donc insérés entre le bloc de commande numérique et la matrice.

Pour l'accumulateur, trois signaux sont concernés : φ_1, φ_3 et φ_4. Les deux premiers correspondent au chargement d'une valeur sur la capacité d'entrée et le dernier au transfert de ces charges sur la capacité de sortie. Pour passer à 1, les signaux φ_1 et φ_3 doivent donc attendre chacun que φ_4 soit revenu à zéro, et φ_4 quant à lui doit attendre que φ_1 et φ_3 à la fois soient à zéro. L'anti-chevauchement a été implémenté de façon très simple, avec des lignes de retard et quelques portes logiques comme l'indique la Figure 112. Un signal est retardé et inversé par rapport à son original (grâce à un nombre impair d'inverseurs en série) et son passage à 1 permet de déclencher l'autre signal, bien après le passage à zéro du premier. Pour indiquer qu'un signal est passé par un circuit d'anti-chevauchement, on rajoute le suffixe *nonov* à son nom.

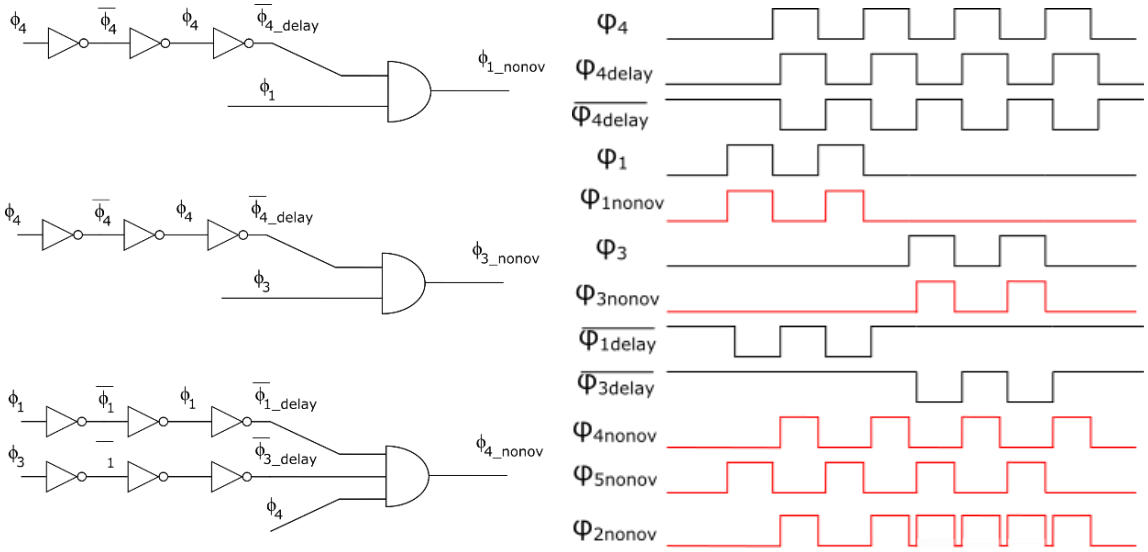


Figure 112 : Schéma des circuits anti-chevauchement et chronogramme pour les commandes de l'accumulateur ($\varphi_2=\varphi_3+\varphi_4$ et $\varphi_5=\varphi_1+\varphi_3$).

Les signaux de commande complémentaires de l'accumulateur, $\varphi_1 + \varphi_3$ et $\varphi_3 + \varphi_4$, sont obtenus à partir de φ_{1nonov}, φ_{3nonov} et φ_{4nonov} par de simples portes logiques. Ils ne se chevauchent pas non plus (voir chronogrammes de la Figure 112).

Pour les signaux de commande du pixel, il ne faut pas que les remises à zéro puissent être activées en même temps qu'un transfert d'information. Il faut donc garantir la fin du signal *resetFD* avant le début du transfert d'information de la PD vers la FD. Le signal TX doit donc monter à 1 seulement après avoir attendu la descente de *resetFD* lorsque *resetPD* n'est pas présent (voir le rappel du chronogramme en Figure 113). Un circuit comme celui donné en Figure 114 le permet. Le même type de circuit est nécessaire également pour la chaîne mémoire.

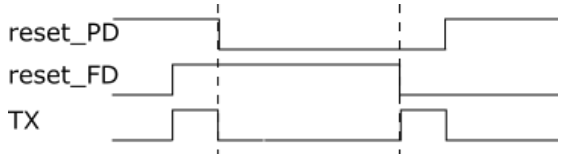


Figure 113 : Rappel du chronogramme de commande d'un pixel.

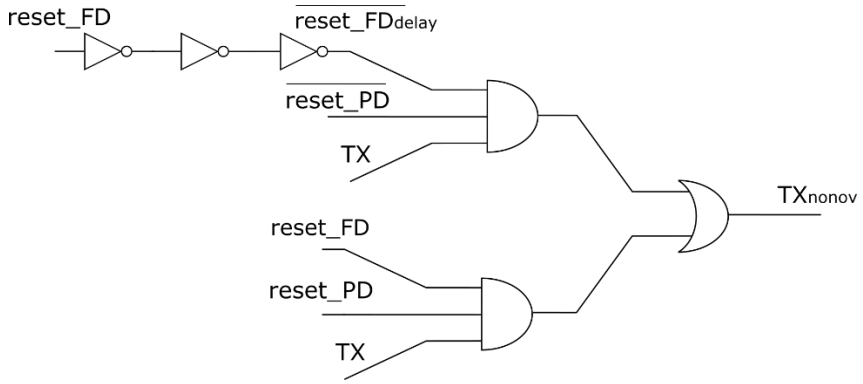


Figure 114 : Schéma du circuit d'anti-chevauchement de TX.

Pour ne pas perdre l'information transférée sur la FD, il faut également attendre la fin du transfert (retour à zéro du signal TX) avant de remettre la PD à sa valeur initiale. On considère que le circuit numérique le fait déjà puisqu'il possède un état d'attente (*acknowledge*) pendant lequel le potentiel de cathode de la PD n'est pas modifié, avant de commencer un nouveau cycle (voir annexe 3).

5.6.3 Connexion inter-macropixels

Pour la simplicité de notre preuve de concept, la possibilité de faire une convolution complète ou une taille de masque autre que 3x3 pixels n'est pas implémentée. Pour cela, comme détaillé au chapitre 3, il suffirait que les PEs puissent s'échanger des valeurs entre eux. Ici les PEs sont indépendants les uns des autres et tous identiques. En revanche les macropixels ne sont pas tous identiques, et peuvent avoir besoin d'interconnexions.

En effet, l'imageur proposé dans ce travail est composé de 4 macropixels. Le macropixel en haut à gauche du motif (TL pour *Top Left*) possède 4 pixels à mémoires alors que le macropixel en bas à droite (BR pour *Bottom Right*) n'en possède qu'une. Pour optimiser les temps de calculs, on a vu au chapitre 3 que l'on fait calculer une différence temporelle sur le 4ème pixel à mémoire de TL par l'accumulateur de BR. Pour cela, on réalise les branchements indiqués en Figure 115 : le pixel *pix9* de TL est lié par un transistor de sélection commandé par *sel_pix_9* à l'accumulateur de son macropixel, TL, tandis que des transistors commandés par *sel_mem_2* et *sel_pix_DT_2* permettent de donner la valeur de ce pixel et de sa mémoire à l'accumulateur de BR pour une différence temporelle.

5.6.4 Temps de montée pour injection de charges

Le PE étant un accumulateur à capacités commutées, l'ouverture et la fermeture des interrupteurs provoque des parasites par de l'injection de charges. Comme on l'a vu précédemment, une application majeure de notre système, à savoir la détection de piétons, tolère beaucoup d'erreurs sur le prétraitement et c'est pourquoi notre PE peut travailler en calcul approximé. En revanche, l'influence de l'injection de charges sur un circuit dessiné (et a fortiori réalisé) est accrue du fait de la présence de capacités parasites. Sans alourdir le circuit en lui-même, l'injection de charges peut être réduite par des modifications des signaux de commandes : le décalage de fermeture et ouverture d'interrupteurs en opposition de phase mais liés au même nœud du circuit et/ou le ralentissement des commutations des signaux. La deuxième peut se faire simplement en réglant les buffers d'entrée de matrice. En effet leur dimensionnement, selon la taille de la matrice choisie (en nombre de pixels), permet d'ajuster le temps de montée des signaux de commande à la valeur voulue. Cet ajustement peut-être fait par analyse paramétrique sur les vues extraites du circuit de la matrice (chapitre 6, section 6.2.1).

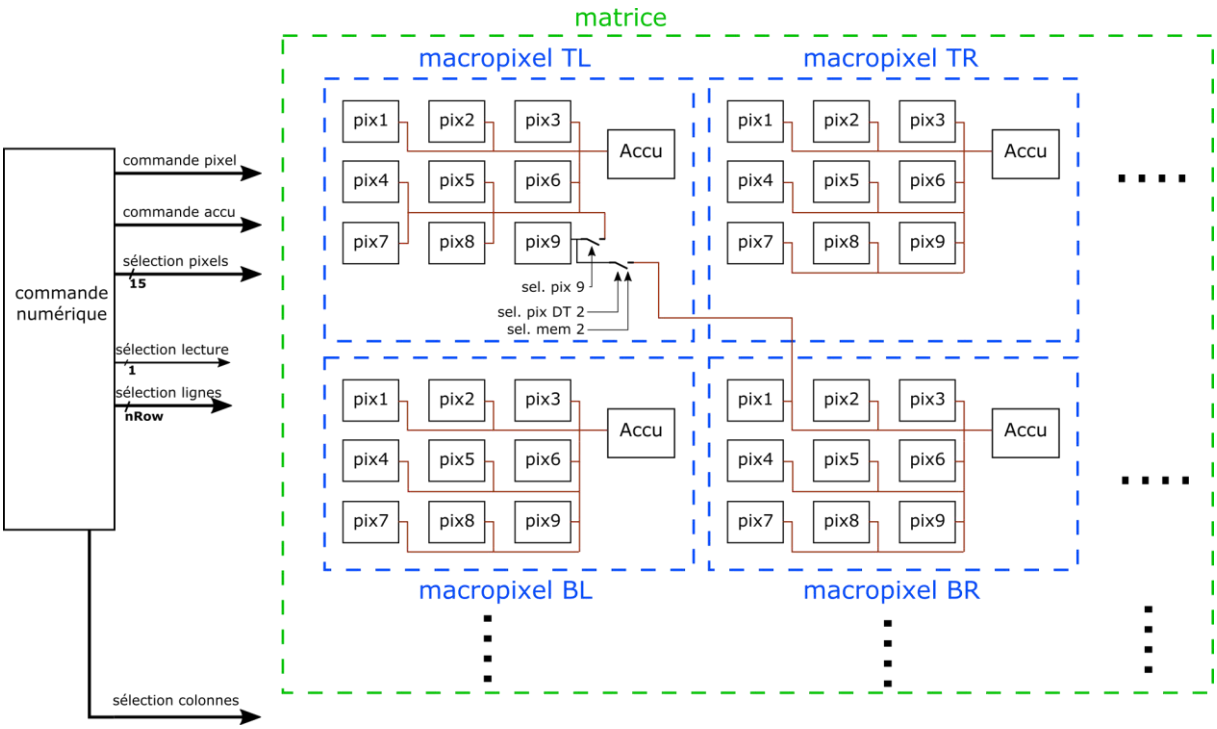


Figure 115 : Schéma du système avec en particulier le lien entre les macropixels TL et BR.

5.7 Dessin : considérations, analyse et réalisation

Afin de valider une preuve de concept et de mesurer l'impact des circuits rajoutés sur la surface photosensible, la matrice a été dessinée. Après avoir exposé les spécifications de placement et routage, les dessins (ou *layout*) des circuits proposés sont donc présentés.

5.7.1 Placement des composants

Tout d'abord pour avoir une image non distordue sur un écran, toutes les photodiodes doivent être régulièrement espacées, que ce soit dans un macropixel ou entre macropixels : il faut donc distribuer l'accumulateur dans le macropixel, sans le mettre d'un bloc à côté des 9 pixels comme il a été représenté schématiquement jusqu'ici.

Ensuite, pour limiter le bruit spatial fixe, c'est-à-dire pour que tous les pixels aient la même réponse, il est préférable qu'ils soient tous identiques. Les chaînes de lecture directes (non mémoire) des photodiodes doivent donc être toutes placées de la même manière par rapport à la photodiode. Par exemple on choisit de placer l'électronique d'un pixel sous sa photodiode. Pour avoir une matrice de photosites régulière on a donc un espace équivalent à l'électronique d'un pixel entre deux photodiodes, latéralement. C'est-à-dire que chaque photodiode possède en-dessous d'elle le reste de son pixel, et un espace disponible à sa droite. Ces 9 espaces disponibles dans un macropixel doivent contenir des circuits à mémoires, l'accumulateur et des bus de signaux. En remarquant que la capacité de sortie de l'accumulateur est égale à la capacité de diffusion d'un pixel, et que le reste de l'accumulateur représente à peu près la surface de l'électronique d'un pixel, on propose le placement de la Figure 116. Un bus de signaux prend 3 espaces, l'accumulateur 2 et donc il reste 4 espaces disponibles pour les mémoires. Le bus est répété entre chaque macropixel pour limiter la distance entre le bus et les transistors qu'il commande (afin d'éviter le risque d'effet antenne sur du métal 3, voir la suite de la section), et légèrement distribué entre les autres lignes et colonnes de pixels vu le nombre important de pistes.

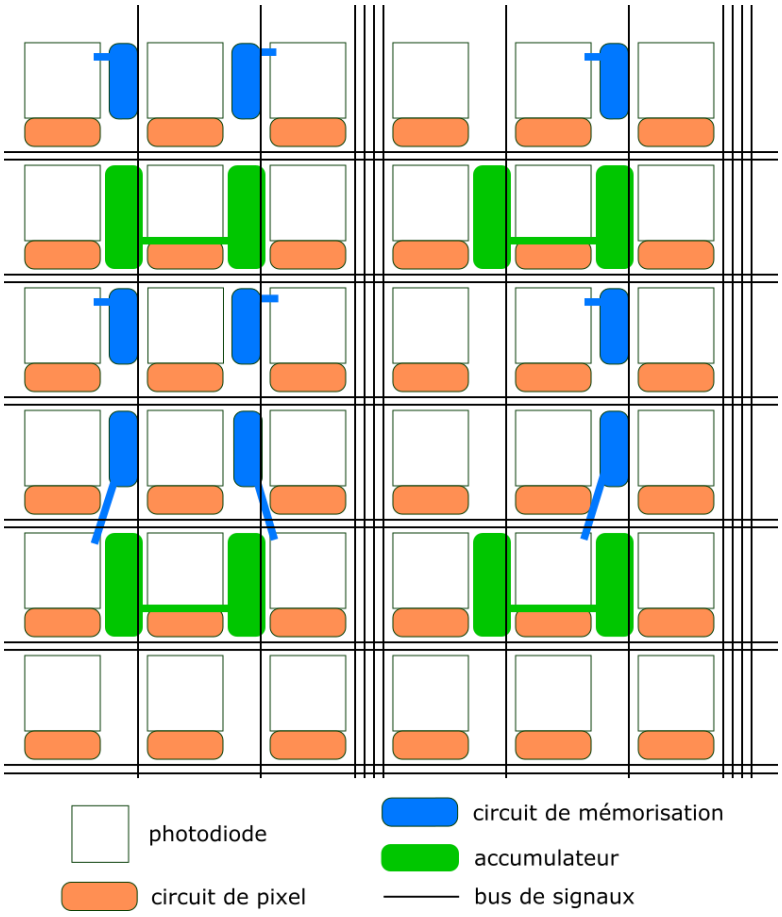


Figure 116 : Placement des différents circuits dans le motif élémentaire de la matrice : 6x6 pixels.

Tous les pixels sont identiques mais selon le macropixel la place de la mémoire par rapport au pixel correspondant varie : on obtient 5 types de pixels à mémoire différents (ils peuvent avoir le même dessin en ce qui concerne la chaîne classique, seule la chaîne qui contient la mémoire diffère). La Figure 116 montre qu'il reste des zones libres, mais on limite le FPN en cherchant un minimum de symétrie : tous les pixels et tous les accumulateurs sont identiques.

De nombreux signaux doivent parcourir la matrice : les signaux de commande des pixels et des accumulateurs, les signaux de sélection de ligne, de pixels et de type de lecture (accumulateur ou pixel), les sorties de macropixels et les alimentations analogiques (V_{DD}, $V_{DD}/4$ et la masse). Parmi ces signaux, les commandes de sélection de lignes de macropixels sont au nombre d'une par ligne tandis que les sorties de macropixels sont au nombre d'une par colonne. Les premières sont donc à router en lignes, les secondes en colonne, et tout le reste des signaux a une direction libre. La place de chaque piste dans le macropixel est choisie pour éviter tout croisement entre un signal numérique qui commute pendant un calcul et les pistes internes de l'accumulateur, en particulier celles qui relient la capacité de sortie au reste du circuit de l'accumulateur, schématisé par les traits verts sur la Figure 116. Cela évite de brouiller les signaux par diaphonie.

Enfin, la technologie utilisée possède 2 niveaux de polysilicium et 4 niveaux de métallisation. Les capacités ne sont disponibles qu'en couches de polysilicium. Le métal 4, le plus haut, est réservé pour faire un masque : les photons ne doivent pouvoir arriver que sur les photodiodes et pas sur les circuits électroniques environnant pour éviter la création de charges qui bruiteraient le signal. Pour limiter les capacités parasites entre pistes métalliques, on restreint chaque métal à une direction : un métal sur deux est vertical et l'autre horizontal. Ici par exemple les métaux 1 et 3 sont utilisés pour les pistes horizontales et le métal 2 pour les pistes verticales. Les pistes de commande doivent parcourir la matrice et sont reliées à des grilles de transistors ; leur longueur est trop grande par rapport au nombre de transistors pour éviter l'effet antenne. Une solution est de faire un petit détour par un métal de plus haut niveau entre la piste considérée et le transistor. Le métal 3 étant le plus haut niveau utilisable, seules les pistes de métal 1 et 2 peuvent être utilisées pour les signaux de commande.

5.7.2 Pixel

La Figure 117 présente le dessin d'un pixel. La photodiode mesure 30x30µm² et est entourée d'un anneau de garde distant de 3µm (contrainte de la technologie). Le circuit du pixel est placé à côté, avec la capacité de diffusion qui occupe la plus grande place. Sa forme est choisie pour que le circuit utilise la longueur de la photodiode et de son anneau, tout en étant le plus étroit possible.

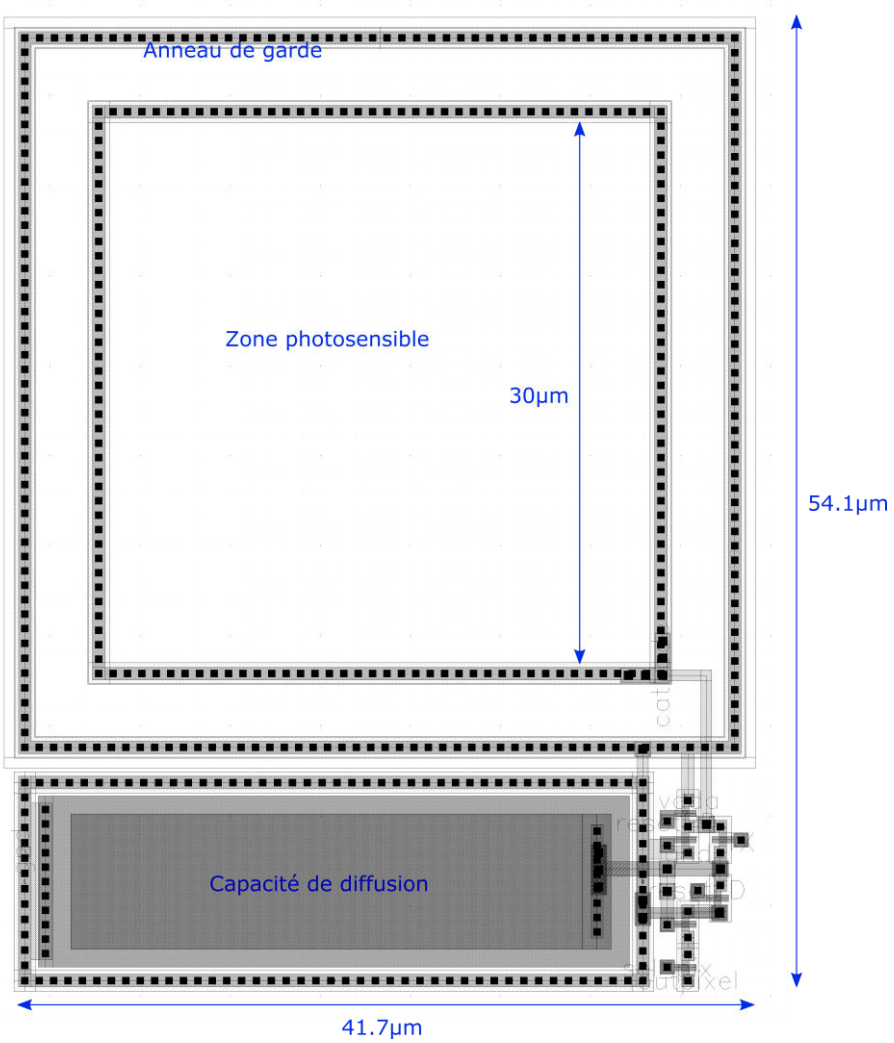


Figure 117 : Dessin d'un pixel.

5.7.3 Pixels avec mémoire

Les pixels à mémoire possèdent exactement les mêmes éléments que les pixels classiques, avec en plus le circuit de pixel qui est dupliqué pour constituer la chaîne à mémoire. Celle-ci est positionnée de différentes façons selon le placement de la Figure 116, les circuits résultant sont présentés en Figure 118.

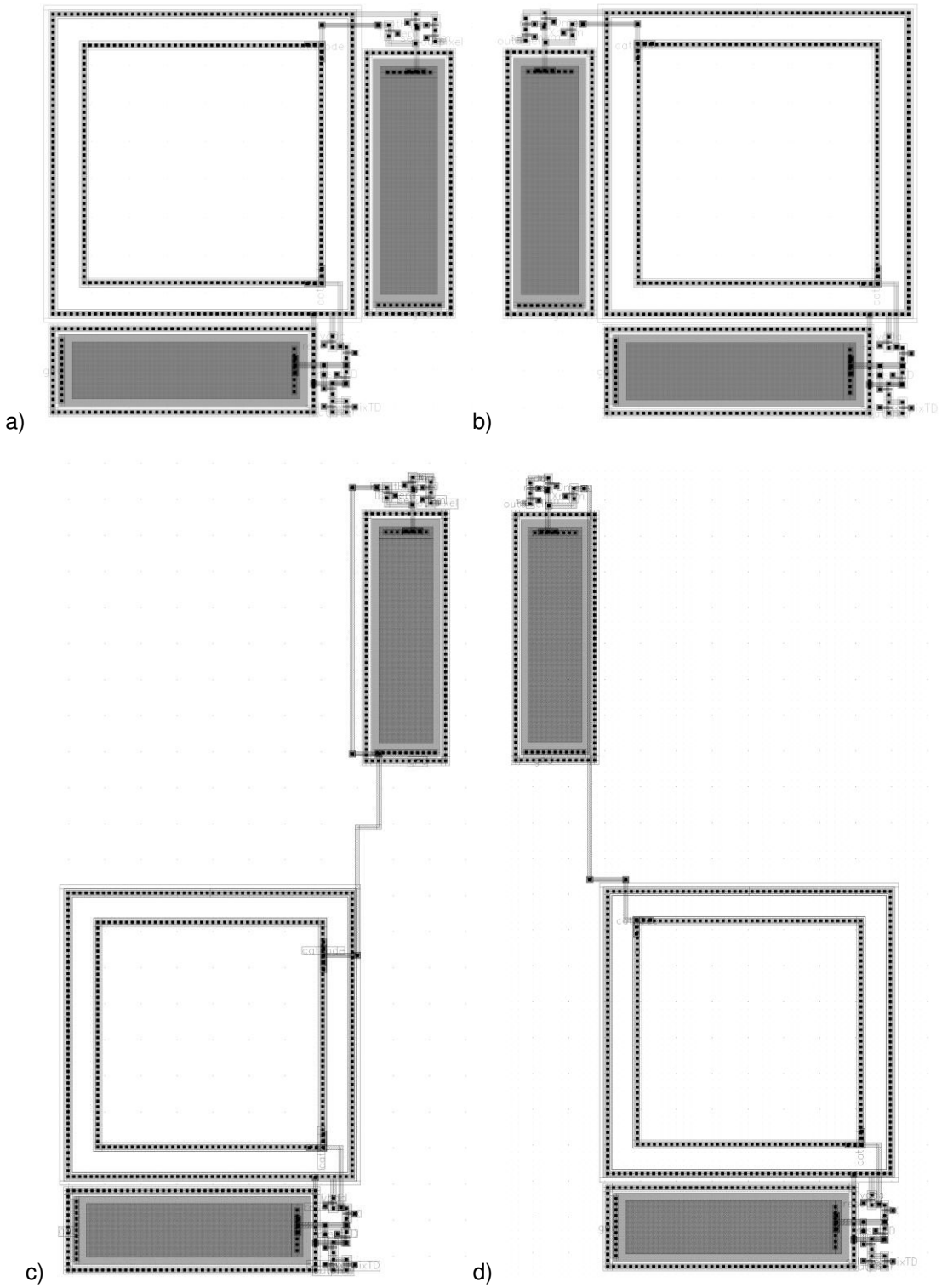


Figure 118 : Pixels à mémoires : (a) avec mémoire à droite pour macropixels TL et TR, (b) avec mémoire à gauche pour TL et TR, (c) avec mémoire en haut à droite pour BL et BR et (d) en haut à gauche pour BL.

5.7.4 Accumulateur

Le schéma de l'accumulateur est rappelé en Figure 119 et son dessin présenté en Figure *120*. La capacité de sortie est à droite du dessin avec un transistor de remise à zéro et tout le reste du circuit est dans la partie gauche. Les électrodes basses des capacités (en polysilicium 1) sont choisies pour les points du schéma les plus éloignés possibles de l'inverseur afin de limiter l'influence des capacités

parasites. Au chapitre 4, il a été montré que le désappariement entre les tailles de C_{in} et de $C_{out} = 10 \times C_{in}$ n'a que peu d'influence. Pour gagner de la place elles sont donc dessinées telles quelles, sans utiliser de méthode de limitation du désappariement comme le *common centroid.*

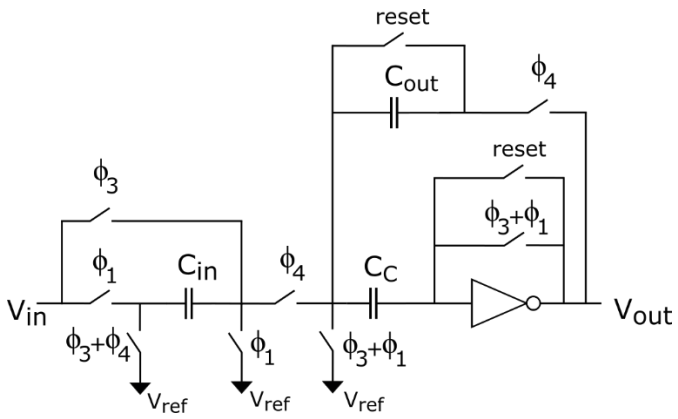


Figure 119 : Rappel du schéma de l'accumulateur avec les commandes.

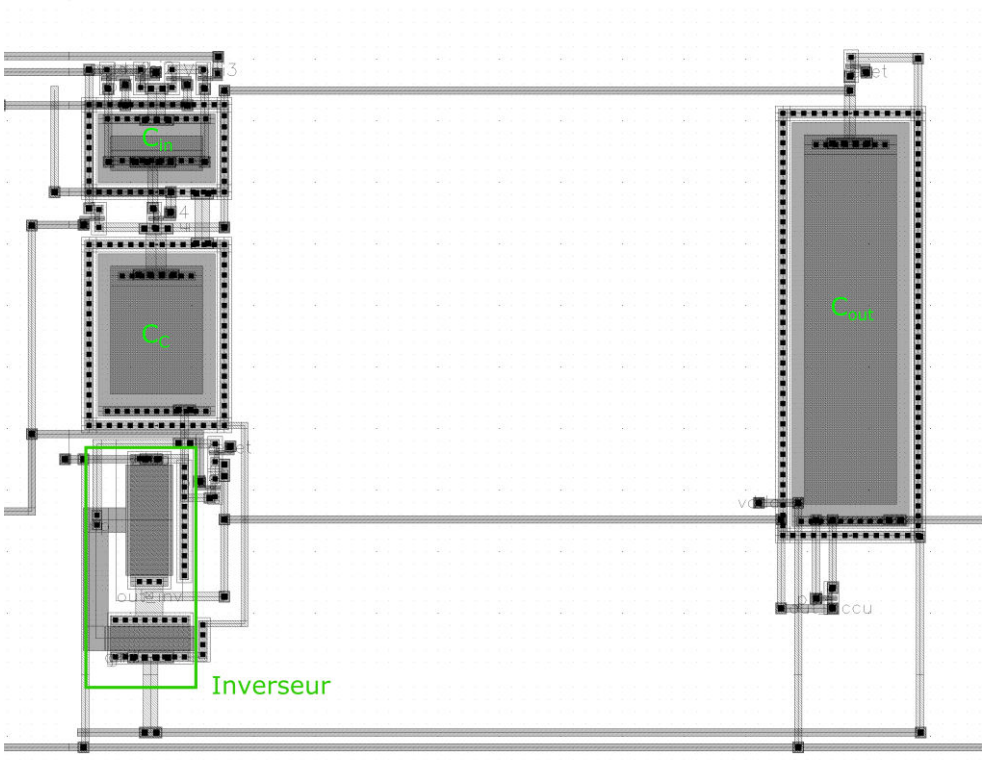


Figure 120 : Dessin de l'accumulateur.

5.7.5 Macropixels

Conformément au placement décidé, les macropixels sont de 4 types, avec chacun des pixels classiques et des accumulateurs identiques, mais qui diffèrent par leurs pixels à mémoire. Ils forment le motif élémentaire de la matrice. La Figure 121 présente le macropixel TL avec ses différents éléments. Des capacités fantômes sont rajoutées dans les espaces libres pour symétriser l'ensemble, présenté en Figure *122*.

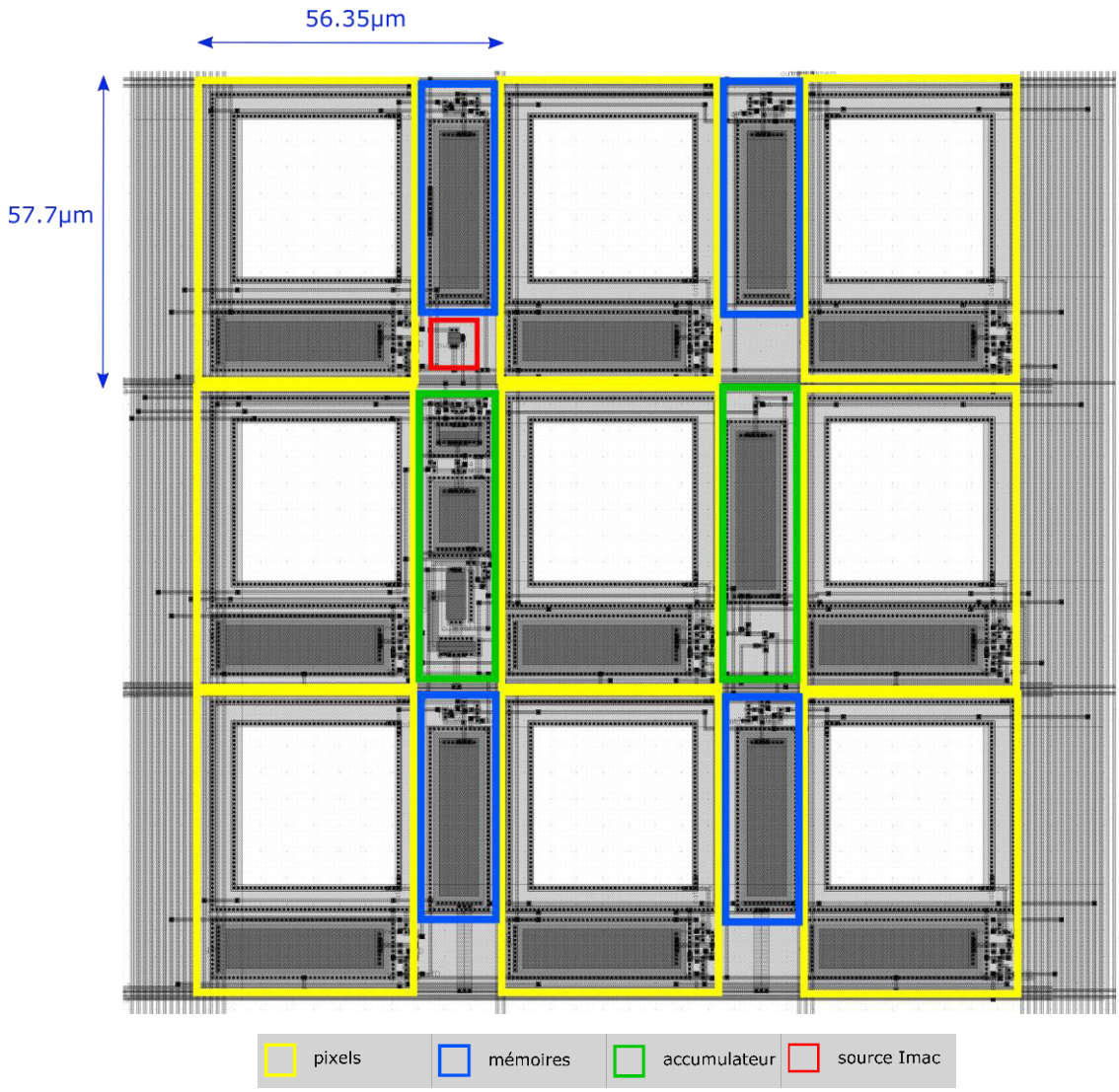


Figure 121 : Dessin d'un macropixel (TL).

5.7.6 Positionnement par rapport à l'état de l'art

Les caractéristiques en termes de surfaces et de fonctionnalités de ce circuit sont résumées dans le Tableau 14. L'indice de versatilité vaut 4.5 puisque le circuit est capable de calculer une convolution spatiale à coefficients programmables (3), de calculer une différence temporelle (1) et de prendre une image sans traitement (0.5). Cela permet de le placer parmi l'état de l'art des imageurs intelligents en utilisant les figures de mérites définies au chapitre 2, comme le montre la Figure 123.

L'objectif de ce travail est d'améliorer le compromis entre versatilité et $FoM_{surface}$ de l'état de l'art. Or on peut voir sur la Figure 123 que le circuit proposé se place un peu moins bien que celui de [77 - Massari 2005]. Ce circuit présente la même versatilité, et un $FoM_{surface}$ légèrement meilleur alors que son architecture est de type 1 PE par pixel. Le principe de l'architecture proposée dans ce travail étant de regrouper les éléments de calculs par macropixels, le circuit qui en découle aurait dû obtenir un meilleur *FoMsurface*. Cette différence avec le résultat obtenu peut s'expliquer notamment par un dessin de macropixel peut-être sous-optimisé. En particulier toutes les recommandations de dessin de la technologie ont été respectées, comme la distance entre l'anneau de garde et la photodiode, les anneaux de garde des capacités, l'absence de métal au-dessus d'un transistor, etc. Le dessin d'un autre circuit de l'équipe de [77 - Massari 2005], présenté dans [121 - Massari 2007] est beaucoup plus dense que le dessin proposé dans ce chapitre. Celui-ci a été fait avec relativement peu d'expérience et de recul en matière de dessin d'imageur. Il semblerait intéressant de le retravailler avec une connaissance

plus approfondie des règles pouvant être outrepasseées pour le densifier, que ce soit au niveau du pixel ou du placement-routage.

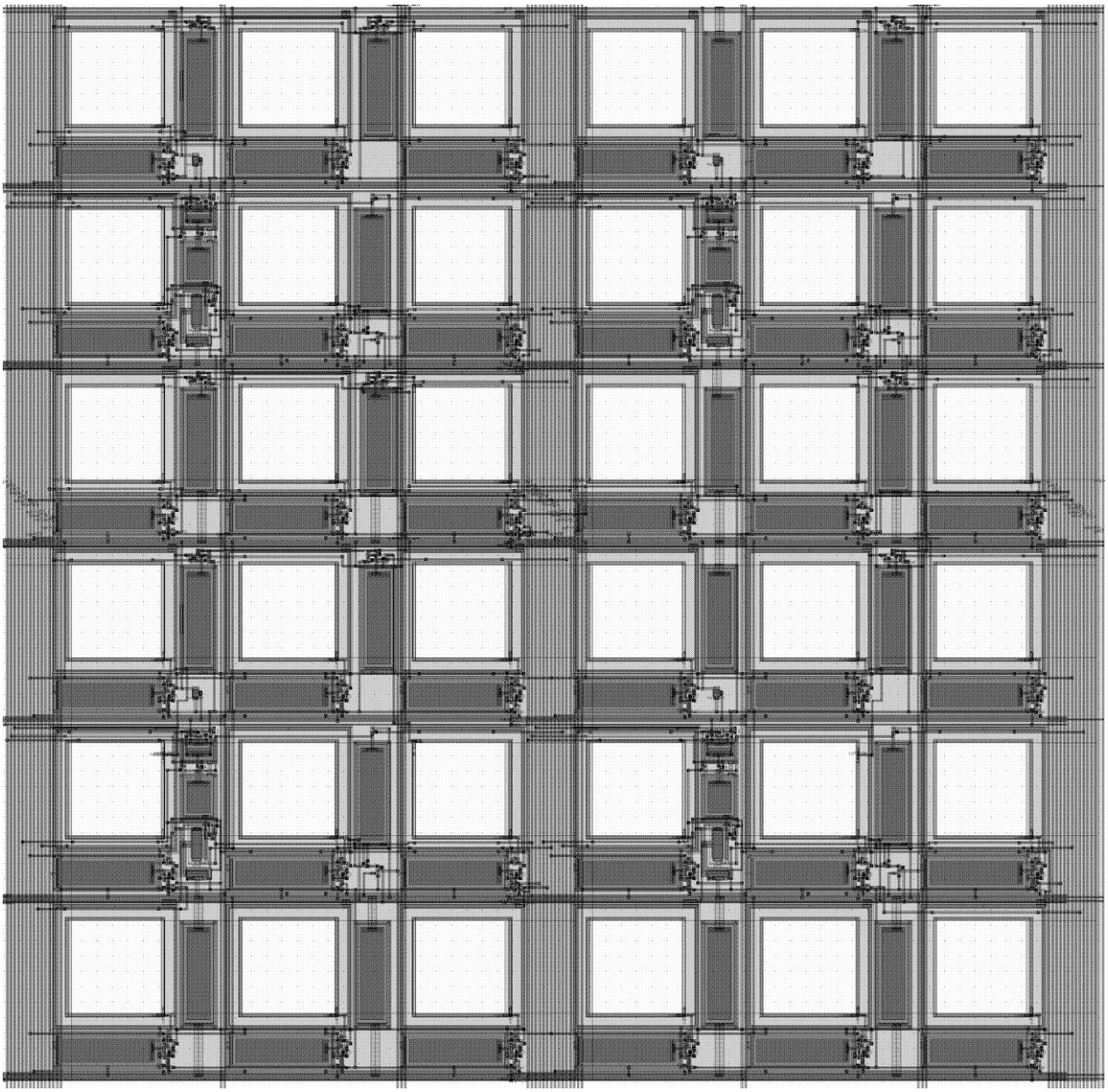


Figure 122 : Dessin du motif de 6x6 pixels, i.e. 4 macropixels différents, formant le motif élémentaire de la matrice.

Technologie	AMS (0.35µm 4M2P) C35B4C3
Taille du pixel (pitch)	56.35x57.7µm²
Surface photosensible	30x30µm²
Facteur de remplissage (FF)	28%
$FoM_{surface}$	18.4
Versatilité	Convolution spatiale, différence temporelle, imagerie
Indice de versatilité	4.5

Tableau 14 : Caractéristiques du circuit proposé.

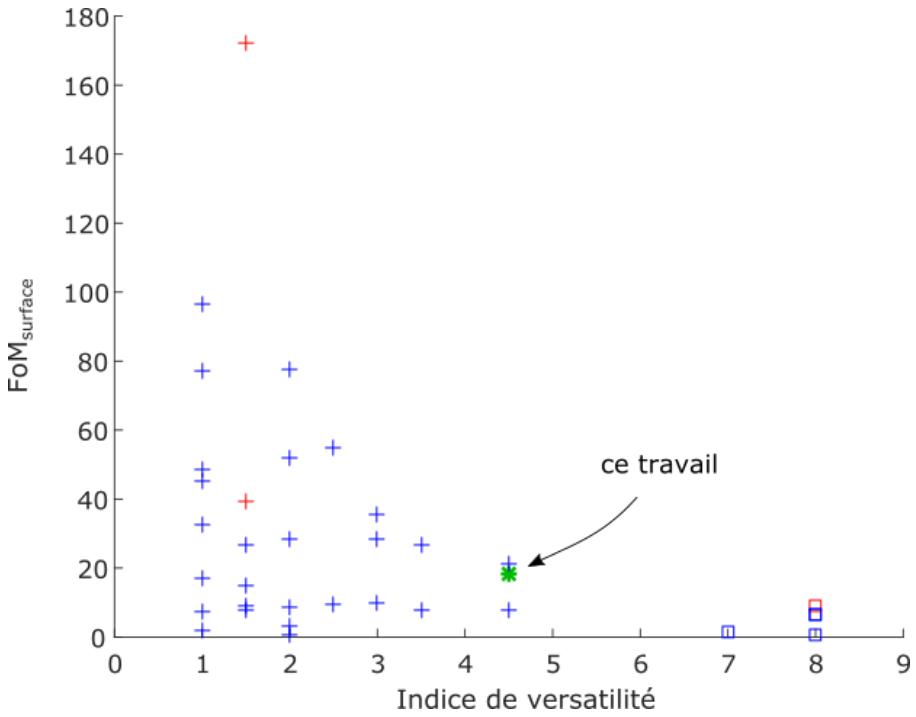


Figure 123 : Classement de l'état de l'art avec (en vert) le circuit proposé.

Le $FoM_{surface}$ défini au chapitre 2 prend en compte la taille du pixel et la technologie, ce qui parait plus réaliste que de s'en tenir au FF seul. Cependant une autre notion importante pourrait être la variation de FF due à l'ajout de calcul dans la matrice. En effet, quand la « technologie » est réduite à son nœud, elle prend en compte les tailles minimales de transistors. Cela impacte la surface occupée par le PE, mais celle-ci dépend aussi de d'autres règles de dessin de la technologie. La variation du FF permet au contraire de tenir compte du FF maximal pour un pixel sans calcul avec le même type de dessin et permettrait de mieux qualifier l'architecture elle-même. Cette donnée n'étant pas fournie dans les articles présentant des systèmes, on ne peut pas s'en servir pour comparer les différents imageurs intelligents de la littérature. Dans le circuit proposé, l'imageur sans PE aurait un facteur de remplissage de 34% seulement (30x30µm² de photodiode, 57.7µm de hauteur de pixel et 46.18µm de largeur de pixel, en comptant un tiers de bus de signaux (41.7µm + 4.48µm)).

Ce chapitre a donc présenté l'implémentation d'une nouvelle architecture d'imageur intelligent décrite dans les chapitres précédents. Ses éléments analogiques (pixels, accumulateur) ont été dimensionnés, son contrôle numérique décrit en VHDL, ses interconnexions expliquées et sa matrice dessinée. Il en résulte une dégradation de facteur de remplissage de 6% seulement mais le circuit ne se place pas en meilleure position que l'état de l'art selon les figures de mérites introduites au chapitre 2. Le chapitre suivant présente les résultats de simulations du circuit en vue extraite afin de valider le circuit sur ses fonctionnalités.

Chapitre 6 : Validation et estimation des performances du système

Le système présenté dans ce travail est constitué d'une matrice de pixels et de circuits de calcul (PE) analogiques, ainsi que d'un circuit de contrôle numérique comme le rappelle la Figure 124. Cette architecture et ses différents éléments ont été conçus dans les chapitres précédents. Ce chapitre présente des résultats de simulation pour valider ce concept.

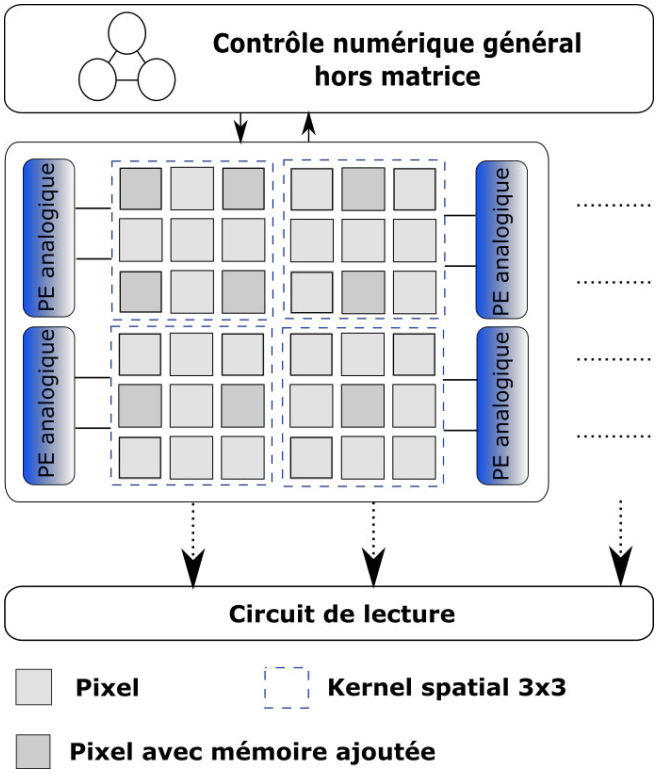


Figure 124 : Rappel de l'architecture du système proposé.

6.1 Environnement de test et validation

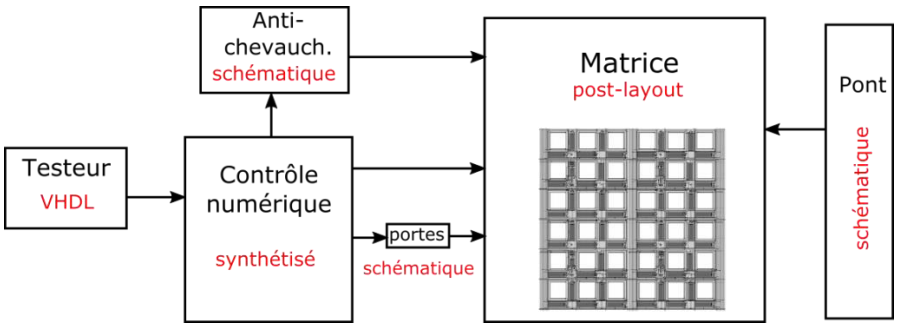


Figure 125 : Banc de test pour les simulations présentées.

Afin de valider le circuit conçu durant ces travaux, un environnement de test a été mis en place. Des simulations de co-vérification ont été réalisées à chaque étape de conception, mais pour la clarté de la discussion seuls les résultats finaux sont présentés ici. Le banc de test est présenté sur la Figure 125 et est composé de plusieurs blocs distincts :

- **le circuit de contrôle numérique :** il gère la commande de l'imageur et du traitement analogique et est configurable par l'extérieur. Il est utilisé dans sa version synthétisée et non placée-routée dans l'optique de gain de temps de simulation, avec la configuration donnée dans le Tableau 15. Il est commandé par un **bloc testeur** codé en VHDL qui fournit l'horloge, la remise à zéro générale et les autres commandes normalement fournies par l'utilisateur, extérieur au système proposé.

Temps	Intégration	Reset photo-diode	Transfert TX	Chargement de C_{in}	Transfert sur C_{out}	Reset de l'accumulateur	Lecture
Durée (ns)	300	40	60	80	80	160	40

Tableau 15 : Configuration utilisée dans les simulations présentées.

- **le circuit analogique** : il est constitué d'une matrice de pixels et de PEs pour assurer les traitements locaux. Les simulations utilisent une matrice de 4 macropixels, c'est-à-dire le motif élémentaire à répliquer. Les circuits ont été dessinés, et sont utilisés avec les parasites résistifs et capacitifs extraits (*post-layout*, ou vue extraite). Seul 1 motif élémentaire est simulé pour limiter le temps de simulation qui comme nous le verrons ultérieurement est un élément prohibitif.

- **le bloc d'antichevauchement** : il assure le décalage temporel des commandes qui ne doivent pas fermer des interrupteurs donnés avant que d'autres ne soient complétement ouverts. Il est simulé sous forme de schématique, avec des chaines de retard de 11 inverseurs dimensionnés à $L_n = L_p =$ 3.5µm et $W_n = W_p = 1$µm. Quelques portes logiques font également interface entre les blocs numériques et analogiques pour l'adressage des sélections de lecture des lignes (une porte NAND et une NOT pour chaque signal de sélection de ligne d'accumulateurs et une NOT et une NOR pour chaque signal de sélection de lecture de lignes de pixels, cf. chapitre 5). Aucun buffer n'est rajouté ici.

- **les alimentations** extérieures : les tensions hautes (V_{DD}) et la masse, sont simulées par des sources de tension idéales. L'alimentation interne, le $V_{DD}/4$, nécessaire à l'accumulateur et à la source de courant du macropixel, est fournie par un pont diviseur de tension simulé en schématique. Ce pont est formé de 4 transistors dimensionnés grands, à W_p =100µm et L_p = 2µm.

Afin de faciliter la validation du comportement du circuit, le protocole de test prévoit une analyse par fonction d'activation du circuit. Ainsi, deux cas d'utilisation sont exploités : la convolution spatiale et la différence temporelle.

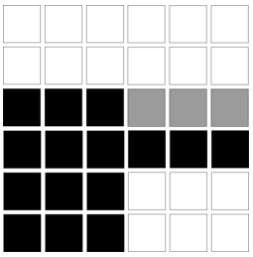
6.2 Convolution spatiale

Le premier traitement testé est la convolution spatiale. Des résultats de simulations transitoires sont présentés pour vérifier le bon déroulement temporel des traitements. Une image complète prétraitée par notre circuit est également proposée pour juger de la qualité d'une détection de contours. Enfin, des simulations de Monte Carlo sont réalisées et leurs résultats sont comparés aux résultats de prétraitement pour une détection de piéton fonctionnelle.

6.2.1 Résultats transitoires

Une simulation mixte dans les conditions décrites ci-dessus en section 6.1 a donc été réalisée sur une image simple présentée en Figure 126, avec comme commande une prise d'image suivie d'une convolution spatiale sur un masque de Sobel horizontal. Les résultats transitoires sont présentés en Figure 127, Figure 129 et Figure 131 pour les signaux de commande des pixels, ceux des accumulateurs et les signaux du circuit analogique respectivement. La Figure 128 reprend les résultats

de la Figure 127 en superposant les courbes des signaux *TXnonov* et *resetFD* afin de montrer l'impact du bloc d'antichevauchement.



-1	-2	-1
0	0	0
1	2	1

Figure 126 : Image (6x6 pixels) et masque avec lesquels les résultats transitoires des figures de cette section ont été obtenus.

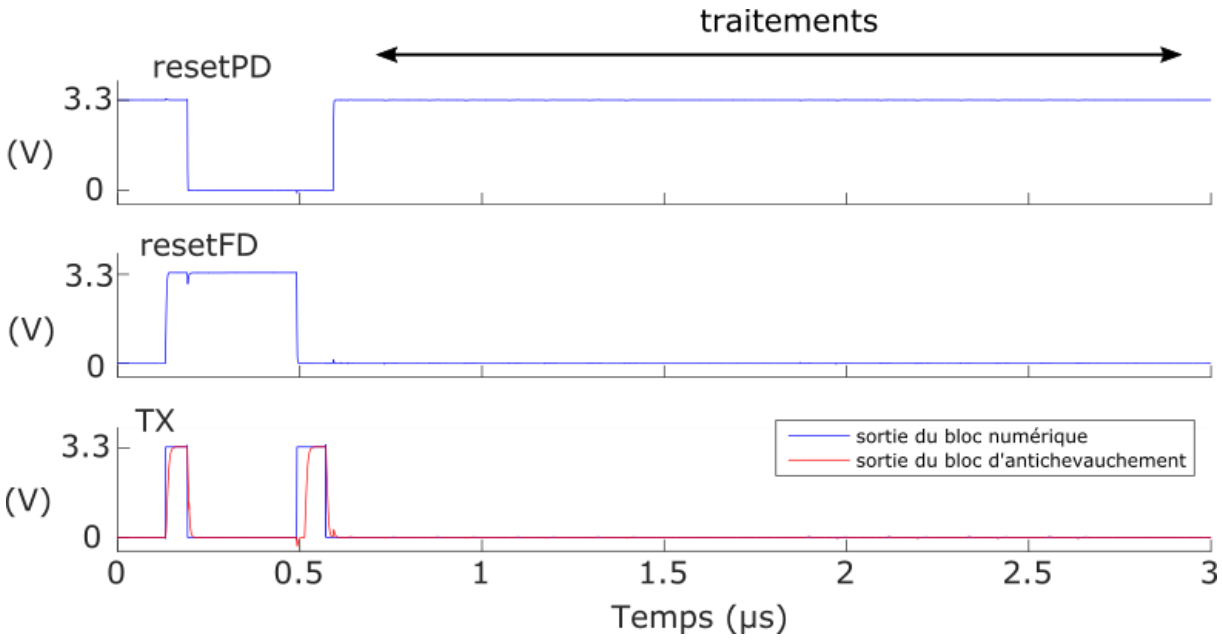


Figure 127 : Résultats de simulation transitoire : resetPD, resetFD, TX et TXnonov (en rouge).

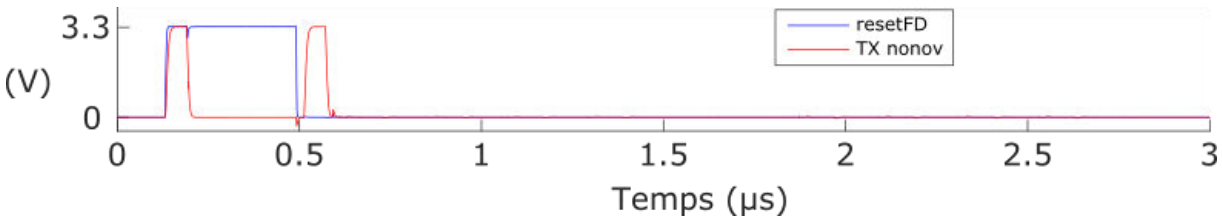


Figure 128 : Résultats de simulation transitoire : resetFD et TXnonov. Ces deux signaux ne se chevauchent pas à la fin du temps d'intégration (i.e. à 0.5µs sur la figure).

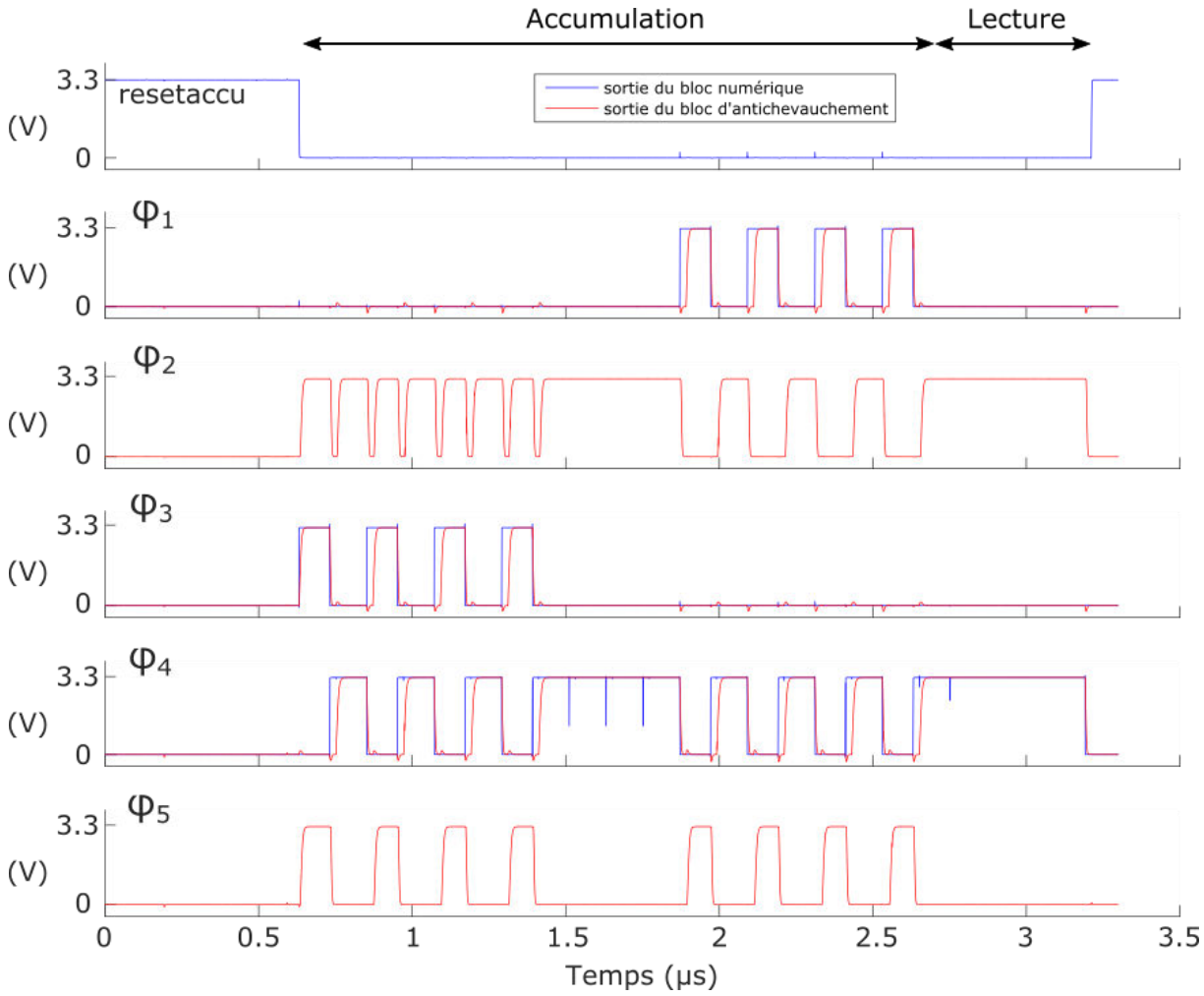


Figure 129 : Résultats de simulation transitoire pour les commandes des accumulateurs. En bleu les signaux en sortie du bloc de contrôle numérique synthétisé, et en rouge les signaux en sortie du bloc d'anti-chevauchement.

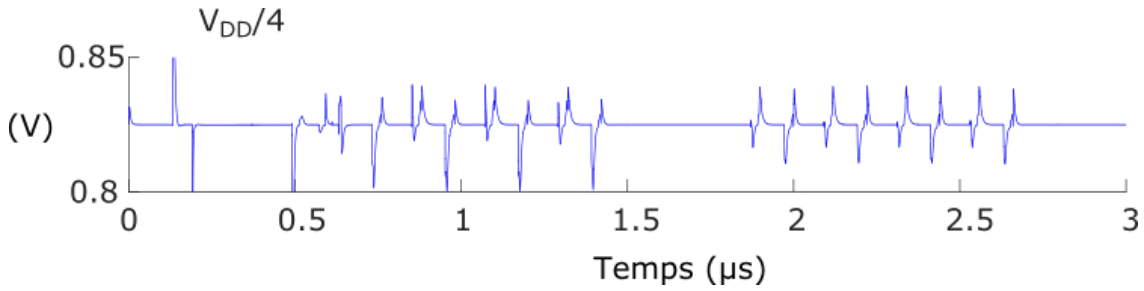


Figure 130 : Résultat de simulation : alimentation $V_{DD}/4$, la tension de référence V_{ref} pendant une prise d'image suivie d'une convolution de type Sobel (cf. figures ci-dessus).

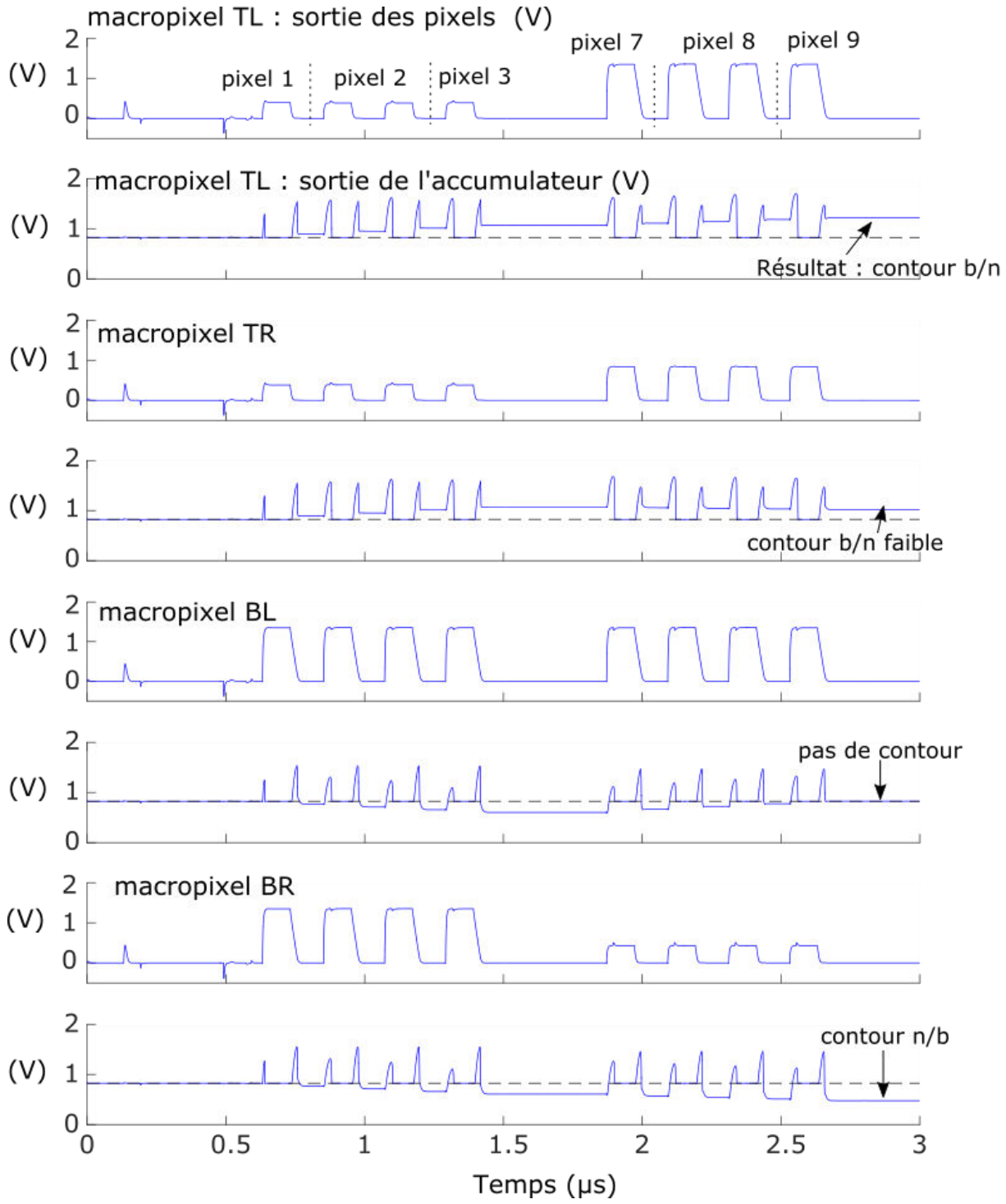


Figure 131 : Résultats de simulation : sorties de pixels (i.e. l'entrée de l'accumulateur) et sorties d'accumulateurs pour chaque macropixel (n pour noir et b pour blanc). Les lignes pointillées représentent la tension de référence V_{ref} *= 0.825V.*

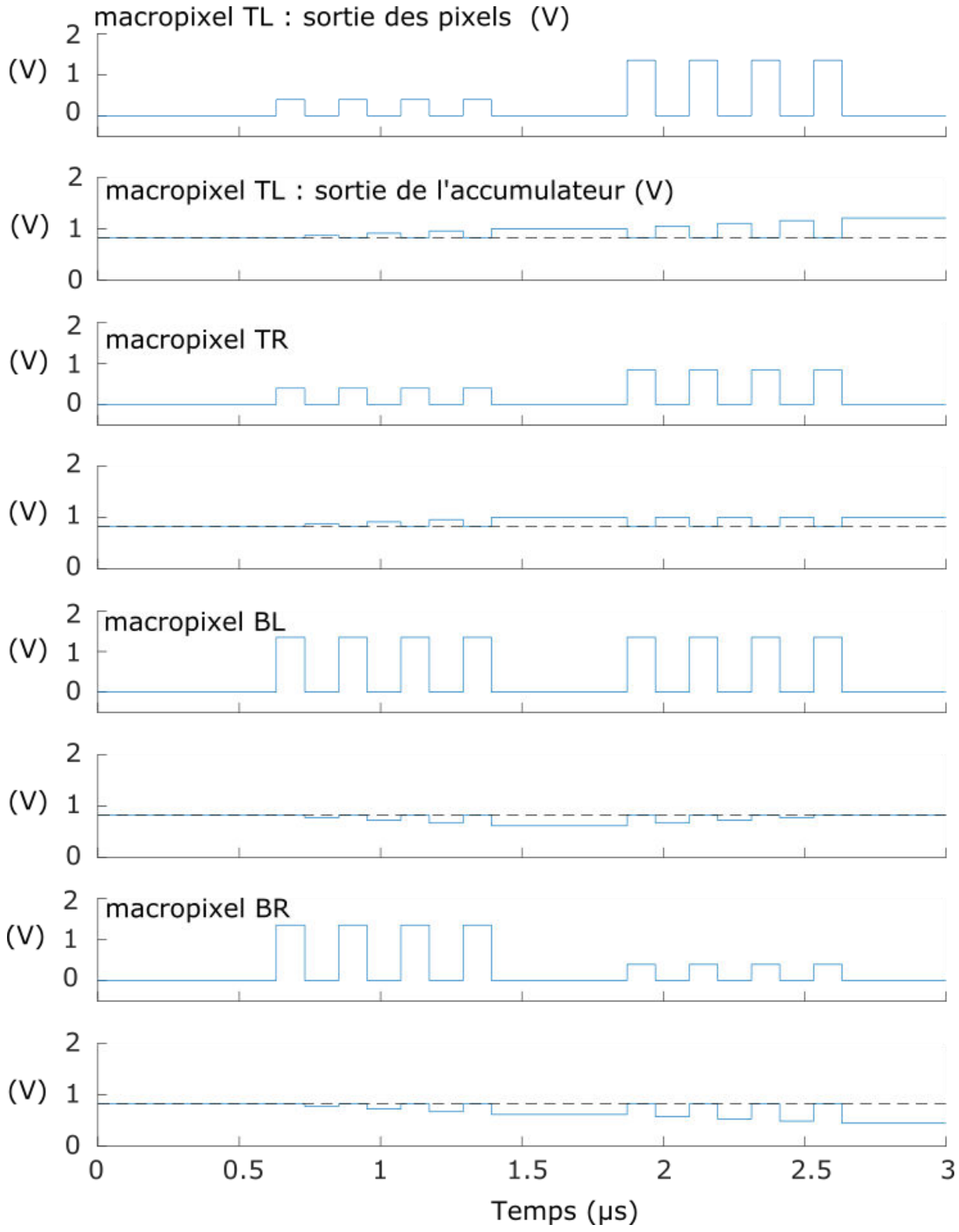


Figure 132 : Résultats transitoires attendus du circuit analogique (simulation fonctionnelle Matlab).

Les circuits de commande des pixels et des accumulateurs correspondent à ce qui est attendu (voir chapitre 5). Les signaux en sortie du bloc d'anti-chevauchement sont nettement séparés (Figure 128). La conception et le dimensionnement proposés du bloc d'antichevauchement sont donc suffisants pour assurer sa fonction. La contrainte temporelle porte sur le temps maximum d'une convolution. Elle aurait pu influencer le choix du temps de non-chevauchement des signaux mais comme on va le voir

dans la suite de cette section, le circuit est plus rapide que nos spécifications. Le temps de non-chevauchement a donc été surdimensionné. Le pont diviseur de tension a lui aussi été surdimensionné (100µm/2µm) pour pouvoir supporter une éventuelle évolution du circuit. La Figure 130 montre que la tension $V_{DD}/4$ a un établissement correct à chaque étape de calcul, ce qui valide la fonction du pont diviseur.

Les résultats du circuit analogique (Figure 131) sont également conformes aux attentes (Figure 132). En effet, sur l'image de la Figure 126, les 3 premiers pixels du macropixel TL doivent avoir une valeur basse (proche de 0.3V, minimum de l'excursion du pixel conçu) tandis que les 3 derniers doivent avoir une valeur très élevée (proche de 1.4V). Ce sont bien ces valeurs que nous retrouvons sur la première courbe de la Figure 131. Ensuite, la tension de sortie de l'accumulateur varie au fur et à mesure de l'accumulation. Toujours pour le macropixel TL, les 3 premiers pixels (blancs, avec des coefficients négatifs) font diminuer progressivement la valeur de sortie de l'accumulateur, tout comme les 3 suivants et derniers (noirs, avec des coefficients positifs). La sortie finale d'un accumulateur indique le type de contours : il suffit de comparer cette valeur à la tension de référence V_{ref} (en pointillés sur la Figure 131). La Figure 133 compare les résultats obtenus sur le circuit aux résultats qu'aurait obtenu un accumulateur idéal avec les mêmes valeurs de pixels. L'accumulateur idéal est défini selon l'équation 4.4 (chapitre 4). Les résultats de la Figure 133 montrent une légère différence entre un accumulateur idéal et l'accumulateur dessiné mais les résultats de ce dernier restent tout de même conformes à nos attentes.

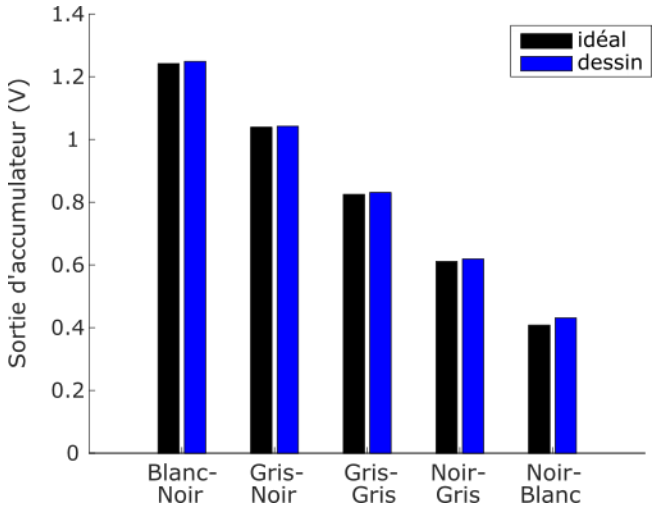


Figure 133 : Comparaison entre les sorties d'un accumulateur idéal et de l'accumulateur en vue extraite (dans le macropixel BL) pour une convolution spatiale de type Sobel sur différents contours.

La Figure 134 compare les réponses de l'accumulateur en vue extraite, pour différents temps de montée des signaux de commande, à la réponse d'un accumulateur idéal, pour 3 cas de contours. On voit qu'un temps de montée de 2ns induit environ 40% d'erreur à cause des injections de charges (cf. chapitre 5 section 5.6.1) contre environ 12% et 10% pour des temps de 10ns et 15ns respectivement. Cette étude a été menée en modifiant artificiellement les temps de montée des signaux issus de la commande numérique synthétisée (qui a un temps de montée de 2ns). Le bloc d'antichevauchement présenté permet d'atteindre, pour le motif de 6x6 pixels simulé, un temps de montée de 14ns (Figure 128), ce qui est suffisant au vu de la Figure 134.

Pendant le calcul de la combinaison linéaire par le masque de Sobel (cf. Figure 129), les 3 premiers pixels sont accumulés selon leurs coefficients respectifs. Avant de passer aux 3 derniers pixels, on peut remarquer une plage de 0.5µs environ pendant laquelle seul le signal φ_4 reste à 3.3V : il s'agit du temps que met le contrôle numérique à passer les coefficients nuls du masque (voir annexe 3). Cela n'influe pas sur le résultat, mais ralentit le calcul complet (0.5µs sur 2.5µs). Une optimisation du code VHDL du contrôle engendrerait une complexité plus grande que le gain en temps, vu nos spécifications. En effet, on voit qu'avec les paramètres choisis (voir Tableau 15), les signaux ont un établissement correct. Le calcul complet d'une combinaison linéaire par un masque de Sobel 3x3, soit 8 accumulations et 3 coefficients nuls, est de 2µs seulement. Ceci est bien meilleur que les spécifications demandées (10ms maximum par convolution, voir chapitre 4). En extrapolant ce résultat, ce circuit pourrait donc réaliser 500k convolutions de type Sobel par seconde. Une opération unitaire

étant un couple chargement-transfert dans l'accumulateur, qui dure ici 0.2µs (Figure 131), le PE fonctionne à 5 MOPS.

On a montré que les commandes étaient correctes pour une convolution spatiale et que par exemple quelques contours bien différents donnent en sortie des accumulateurs des résultats cohérents. Cela va être complété par la simulation du circuit en convolution spatiale sur une image entière.

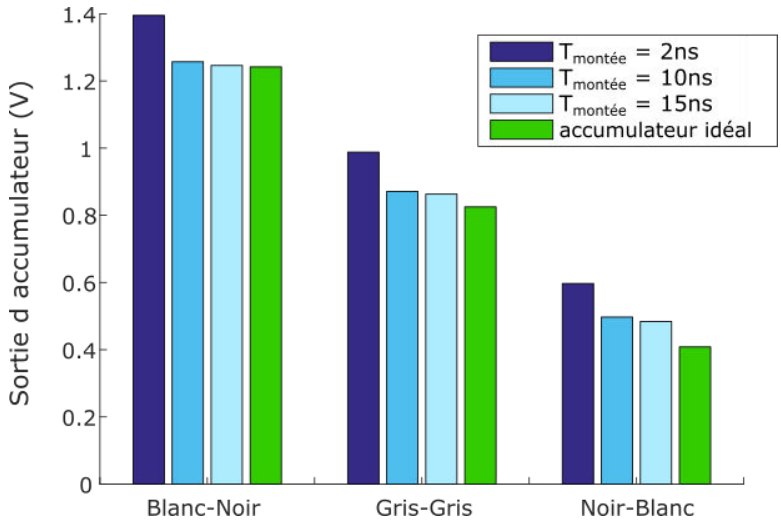


Figure 134 : Comparaison des sorties d'un accumulateur dessiné pour différents temps de montée des signaux de commande, et d'un accumulateur idéal.

6.2.2 Validation sur une image

Pour simuler une exposition à la lumière correspondant à une image donnée, le niveau de gris de chaque pixel est converti en une valeur de photocourant grâce à un script Matlab. Le blanc est transcrit en la valeur maximale de photocourant avant éblouissement de la photodiode pour le temps d'intégration utilisé, à savoir 1000nA. Dans le circuit de test, le photocourant est injecté à la cathode de la photodiode par une source de courant idéale.

Une simulation unitaire correspond toujours à un seul motif de 6x6 pixels. L'image originale est donc découpée en morceaux de cette taille, et 7225 simulations sont nécessaires pour une image de 510x510 pixels comme celle des poivrons (Figure 135). L'image de résultats est ensuite reconstituée par un autre script Matlab.

Le nombre de simulations à faire pour une seule image étant si élevé, des simulations purement analogiques sont préférées à des simulations mixtes. Pour cela, les signaux résultant de la commande numérique et du bloc d'anti-chevauchement sont récupérés d'une simulation mixte puis utilisés comme stimuli dans une simulation analogique. Chaque simulation analogique d'une convolution d'un masque de Sobel sur un motif de 6x6 pixels dure environ 3 minutes, contre 6 heures en simulation mixte (ordinateur 32 cœurs à 2.40GHz).

De telles simulations ont été réalisées pour calculer la convolution sous-échantillonnée de l'image des poivrons par un masque de Sobel horizontal. Le résultat est présenté en Figure 135 avec l'image originale et la simulation fonctionnelle de la CSE telle que présentée au chapitre 3. Ces deux résultats sont très proches, le circuit fonctionne donc comme attendu. Le masque de convolution est programmable, mais le temps de simulation est trop grand pour en tester plusieurs.

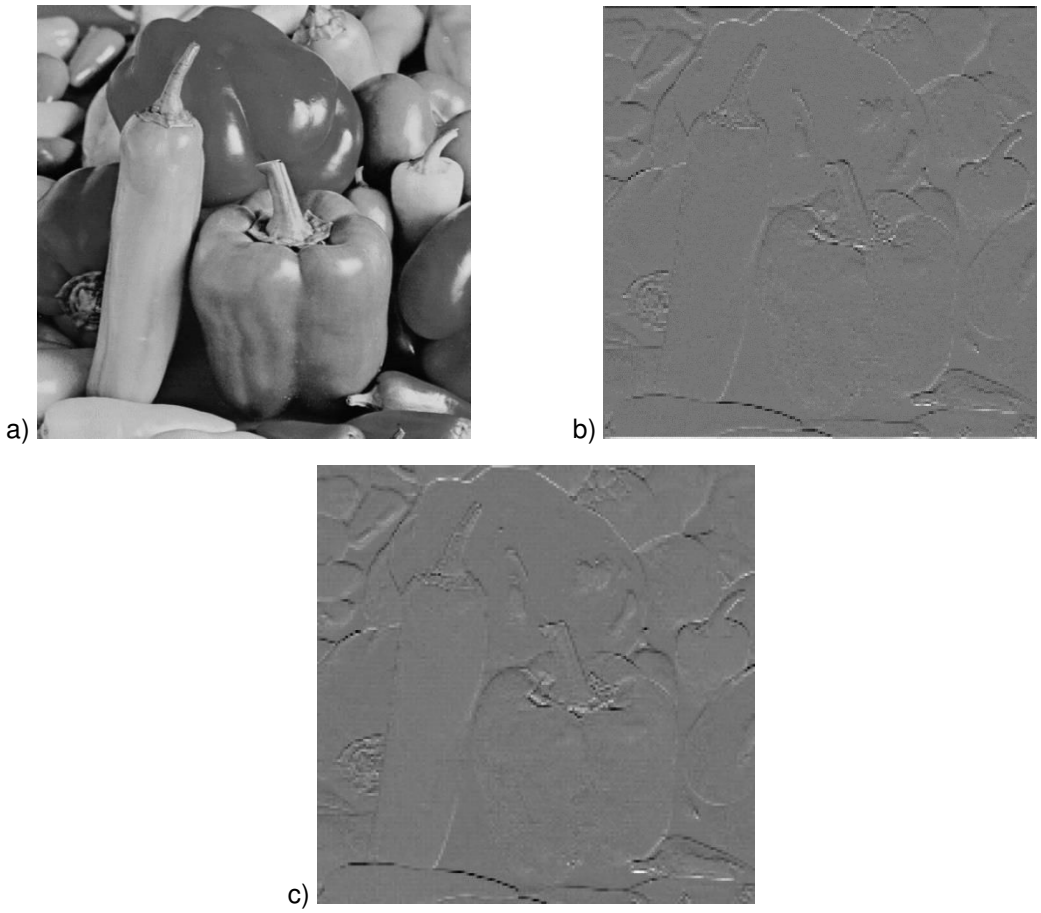


Figure 135 : (a) Image initiale des poivrons (512x512 pixels), (b) résultat en simulation fonctionnelle et (c) résultat en simulation sur le circuit proposé (convolution par le masque de Sobel horizontal) (170x170 pixels).

6.2.3 Validation par détection de piétons

Comme on l'a vu au chapitre 3, une détection de piétons est un critère plus objectif de validation du circuit de prétraitement. Ici notre système peut renvoyer le gradient obtenu par convolution sous-échantillonnée (CSE). Pour détecter des piétons, l'algorithme HOG est simulé fonctionnellement sur de telles images prétraitées et les résultats sont utilisés par un algorithme d'apprentissage automatique. Cet apprentissage automatique est entrainé sur des images prétraitées avant d'être utilisé en outil de décision sur d'autres images prétraitées. L'entrainement demande 600 images positives (avec piéton) et 600 images négatives (sans piéton), et le test au moins une centaine d'autres images de chaque catégorie. Ce nombre d'images à prétraiter (chacune dans au moins deux directions du gradient) avec notre circuit est trop important pour que le temps de simulation en post-layout soit raisonnable.

Cette validation est donc réalisée sur des circuits schématiques, et les résultats unitaires et en simulation de Monte-Carlo sont comparés à ceux des simulations en vue extraite.

6.2.3.1 Détection de piétons sur circuit schématique

Pour réaliser une détection de piétons dans le même cadre que celui du chapitre 3, la même base de données d'images positives et négatives est utilisée. Les 1500 images sont prétraitées par le circuit schématique, pour des convolutions spatiales avec le masque de détection de contours 'simple' horizontal puis le vertical (donc 3000 images convoluées). L'algorithme HOG est ensuite appliqué à ces images prétraitées, et enfin les vecteurs obtenus sont fournis à un apprentissage. Les résultats sont donnés dans le Tableau 16, à côté des résultats obtenus en simulation fonctionnelle et discutés au chapitre 3. Les résultats sur circuit schématique sont comparables à ceux des simulations fonctionnelles en termes de faux négatifs. Les faux positifs sont plus nombreux, mais comme cela a été discuté au chapitre 3, leur impact est moins important. Le circuit est donc adéquat à une détection de piétons.

	Masque simple - classique - fonctionnelle	Masque simple - CSE - fonctionnelle	Masque simple - CSE - circuit schématique
Faux positifs (%)	0	2	7
Faux négatifs (%)	6	8.5	11

Tableau 16 : Résultats de détection de piétons : avec des masques simples, en convolution classique ou CSE, et en simulation fonctionnelle ou sur circuit schématique.

6.2.3.2 Comparaison des résultats du circuit schématique et du circuit en vue extraite

Pour tenter d'extrapoler ce résultat à des simulations en vue extraite plus proches d'un circuit réalisé, le résultat d'une convolution spatiale par un masque de Sobel horizontal sur un circuit en schématique est comparé au résultat de la même simulation sur un circuit en vue extraite. L'image de poivrons donnait de bons résultats mais nous allons comparer ici sur une image typique utilisée parmi d'autre pour la détection de piétons. La Figure 136 montre que les deux résultats sont toujours très semblables. Une détection de piétons pourrait donc fonctionner sur le circuit dessiné.

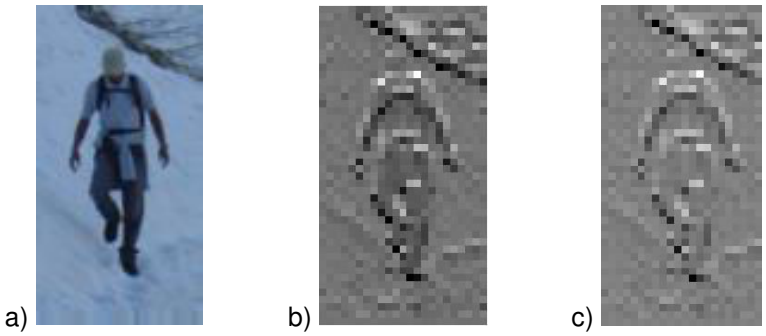


Figure 136 : *Image initiale de piéton (a), et résultats en simulation (Sobel horizontal) sur circuit (b) schématique et (c) en vue extraite.*

6.2.3.3 Etude en simulation de Monte-Carlo

En raison des variations engendrées par la fabrication, les circuits fabriqués donneront des résultats plus ou moins proches de ceux de la Figure 136(c). Pour évaluer ces variations, des simulations de Monte-Carlo ont été réalisées sur 2 types de contours. Les résultats sont présentés sur la Figure 137.

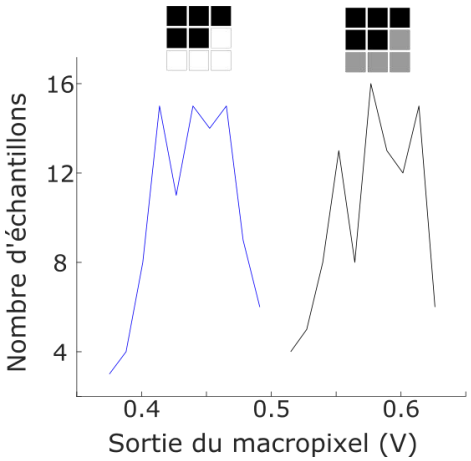


Figure 137 : Résultats de simulations de Monte-Carlo (100 essais) sur vue extraite d'un macropixel (convolution de type Sobel sur deux types de contour présentés au-dessus de chaque courbe).

Pour savoir si une détection de piétons fonctionnerait avec toutes ces variations, le script Matlab utilisé pour les simulations fonctionnelles (chapitre 3) est modifié. Celui-ci prend désormais en compte les pixels et les accumulateurs dimensionnés, et permet de simuler l'influence des erreurs vues sur les simulations de Monte-Carlo. Tout d'abord les images sont converties en valeurs de sortie de pixels : un pixel blanc, codé à 255, correspond à une sortie de pixel de 0.3V tandis qu'un pixel noir à une sortie de 1.4V (voir chapitre 5). Ces images sont ensuite convoluées par un masque en suivant l'équation d'un accumulateur idéal, avec une erreur rajoutée en sortie :

$$V_{out} = \frac{C_{in}}{C_{out}} \left(\sum_{i=1}^{N_{pixels}} s_i * coef_i * \left(V_{in(i)} - V_{ref} \right) \right) + V_{ref} \qquad (6.1)$$

et

$$V_{out_{erreur}} = V_{out} + bruit \qquad (6.2)$$

avec C_{in} et C_{out} les capacités d'entrée et de sortie respectivement,

N_{pixels} le nombre de pixels (9 pour un masque de taille 3x3),

s_i et $coeff_i$ les signes et coefficients en valeur absolue correspondant au pixel i du masque,

V_{ref} la tension de référence (V_{DD}/4, soit 0.825V),

$V_{in(i)}$ la sortie du pixel i et

bruit une erreur aléatoire de distribution uniforme entre -Err et +Err (voir Tableau 17).

Les résultats de détection de piétons ainsi obtenus en simulations fonctionnelles (Matlab) sont donnés dans le Tableau 17. Les résultats de simulation avec erreur sont moyennés sur 15 traitements complets pour limiter les effets des fonctions pseudo-aléatoires du logiciel.

	Classique	CSE	CSE avec erreur		
Err =			2mV	5mV	10mV
Faux positifs (%)	0	2	3	7.3	14.2
Faux négatifs (%)	6	8.5	9.9	13.6	22.2

Tableau 17 : Résultats de simulations fonctionnelles de détection de piéton sans ou avec erreur sur une sortie de prétraitement représentative du circuit idéal.

Les résultats indiquent que 5mV d'erreur en valeur absolue dégradent déjà les performances de l'algorithme de détection de piétons. Or les distributions de Monte-Carlo de notre circuit en vue extraite (Figure 137) s'étalent sur près de 50mV de part et d'autre de la valeur attendue. Une détection de piétons ne fonctionnerait pas sur la majorité des circuits fabriqués.

La convolution spatiale sous-échantillonnée, étudiée ici à travers le traitement des contours, a des résultats adéquats lorsqu'on regarde une image de contours. En revanche, une détection de piéton simple telle que décrite au chapitre 3 et basée sur ce prétraitement ne fonctionnerait a priori pas sur les prototypes, sans modification des algorithmes de plus haut niveau et/ou une amélioration de la précision de notre circuit.

L'étude fonctionnelle du chapitre 3 indiquait qu'une erreur de l'ordre de 10% sur chaque coefficient du masque de CSE ne dégradait pas trop la détection de piétons. Cela nous a conduits à choisir une approche en calcul approximé pour dimensionner le PE. Or on constate ici que l'erreur autorisée sur le résultat dans ce circuit à capacités commutées est très faible. Cette différence peut venir du fait que dans le cadre du chapitre 3 l'erreur était introduite de la même manière sur tous les masques d'une image ; c'est-à-dire que le même masque erroné est utilisé pour toutes les convolutions d'une image. Cela ne correspond pas à la réalité des variations de process entre macropixels. De même les hypothèses de l'étude présentée dans cette section sont à l'autre extrême, avec une décorrélation totale des disparités de process entre macropixels d'un même circuit. Il s'agit d'un pire cas, atteint ou non en fabrication. Le choix effectué de travailler en calcul approximé semble donc erroné. La méthodologie de conception de circuit SC développée au chapitre 4 serait à réappliquer en prenant en compte une erreur autorisée plus faible.

6.3 Différence temporelle

En plus de la convolution spatiale, le circuit proposé peut également calculer des différences temporelles sous-échantillonnées par 2x2 pixels, cette architecture ayant été justifiée au chapitre 3. Cette fonction va être étudiée dans cette section.

6.3.1 Résultats transitoires

Afin d'observer une différence temporelle, le circuit placé dans les conditions décrites dans la section 6.1 de ce chapitre est simulé avec en commande la prise d'une image, puis la prise d'une image à stocker sur les mémoires et enfin une différence temporelle. Celle-ci est effectuée avec un coefficient 2 pour accentuer la plage des résultats :

$$V_{out} = 2 * V_{pix} - 2 * V_{mem}$$

avec V_{out} la sortie de l'accumulateur en fin d'opération,

V_{pix} la valeur de sortie d'un pixel et

V_{mem} la valeur de la mémoire correspondant à ce pixel.

Comme stimuli pour cette différence temporelle, les images de la Figure 138 sont utilisées. Le système proposé calcule une différence temporelle sous-échantillonnée par 2x2 pixels, seuls 9 pixels sont donc représentés avec un niveau de gris pertinent sur cette figure de 6x6 pixels. Les 9 différences de niveaux de gris proposées sont représentatives des excursions possibles : des différences extrêmes noir - blanc ou blanc - noir, mais aussi des changements plus faibles ou nul (pixel du milieu). Les trois différences noir - gris foncé, gris foncé - gris clair, et gris clair - blanc par exemple représentent la même différence.

Pour rappel, une différence temporelle dans notre architecture est réalisée en 3 cycles. Chaque cycle correspond à un calcul de différence par chaque PE et à la lecture des résultats en sortie des accumulateurs. Chaque PE gère 2 ou 3 pixels à mémoire, il y a donc besoin de 3 cycles mais seuls les PE de macropixels TL travaillent pendant le troisième cycle (voir la fin du chapitre 3 et le chapitre 5 pour la commande numérique). Ces cycles s'observent bien sur les résultats transitoires présentés en Figure 140 et Figure 141. On y voit également que les blocs numérique et d'anti-chevauchement donnent les résultats attendus (voir chapitre 5), comme pour la convolution spatiale.

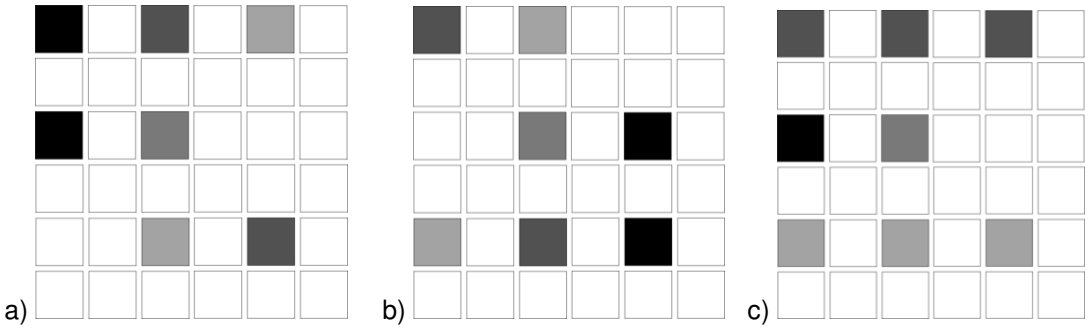


Figure 138 : Images utilisées en stimulus dans la simulation transitoire de différence temporelle entre deux images consécutives : (a) première image et (b) seconde image. C) Résultat attendu de la différence entre les deux images : 3 passages de 'foncé vers clair' identiques, un passage noir-blanc, un gris-gris, un blanc-noir et 3 'clair vers foncé' identiques.

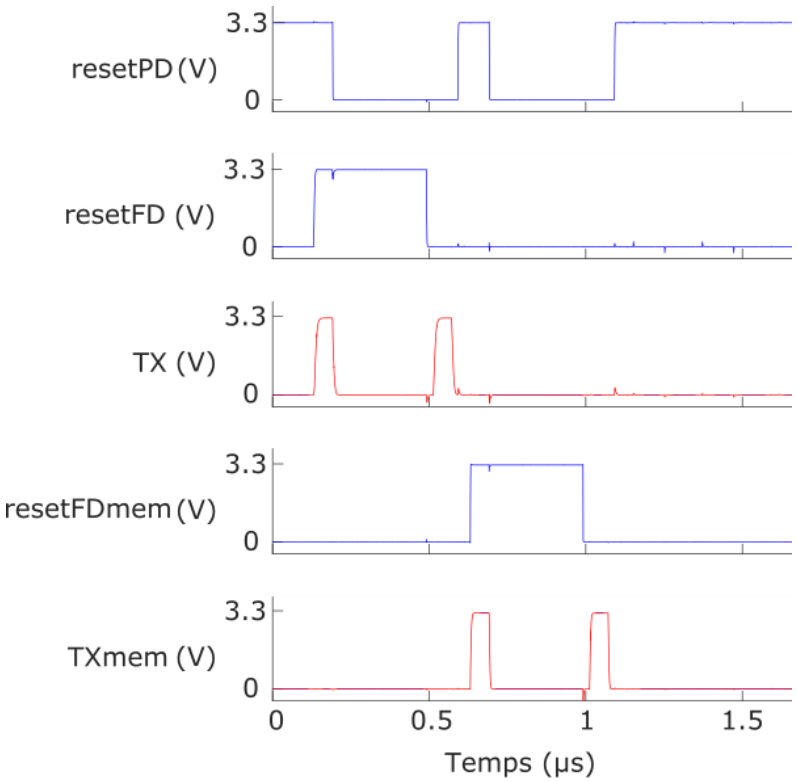


Figure 139 : Résultats en simulation transitoire : signaux de commande des pixels (en bleu : sortie du bloc numérique, et en rouge : sortie du bloc d'antichevauchement).

En ce qui concerne le circuit analogique, la Figure 141 présente les signaux en sortie des pixels et en sortie d'accumulateur pour chaque macropixel. Elle doit être comparée à la Figure 142 qui présente les résultats attendus. Durant chaque cycle, la sortie des pixels prend deux fois la valeur du pixel et deux fois la valeur de la mémoire correspondante. La sortie de l'accumulateur varie en conséquence et le résultat de la différence temporelle peut être lu par l'écart de la sortie à la tension de référence (représentée en pointillés). Sur les différents changements de couleur proposés le circuit fonctionne comme attendu sur chacun de ses macropixels.

Le temps de calcul par le PE est très court : environ 1µs pour 4 opérations, soit 3µs pour une différence temporelle complète. Ceci correspondrait à 330k différences temporelles (à coefficient 2) par seconde. En revanche, le temps de lecture peut être très limitant, d'autant plus qu'il faut faire 3 lectures de chaque macropixel pour une seule image de différence temporelle. En particulier, ici la lecture se fait macropixel par macropixel (et non ligne de macropixels par ligne de macropixels) ce qui allonge le temps total de lecture : il représente ici un tiers du temps total utilisé (1.5µs sur 4.5µs). Mais cela dépend plus de l'extérieur du circuit et du nombre de sorties allouées que de l'architecture.

Le circuit distingue bien les changements de niveaux de gris très particuliers comme ceux de la Figure 138. Ce résultat est complété par une différence temporelle sur deux images réelles consécutives.

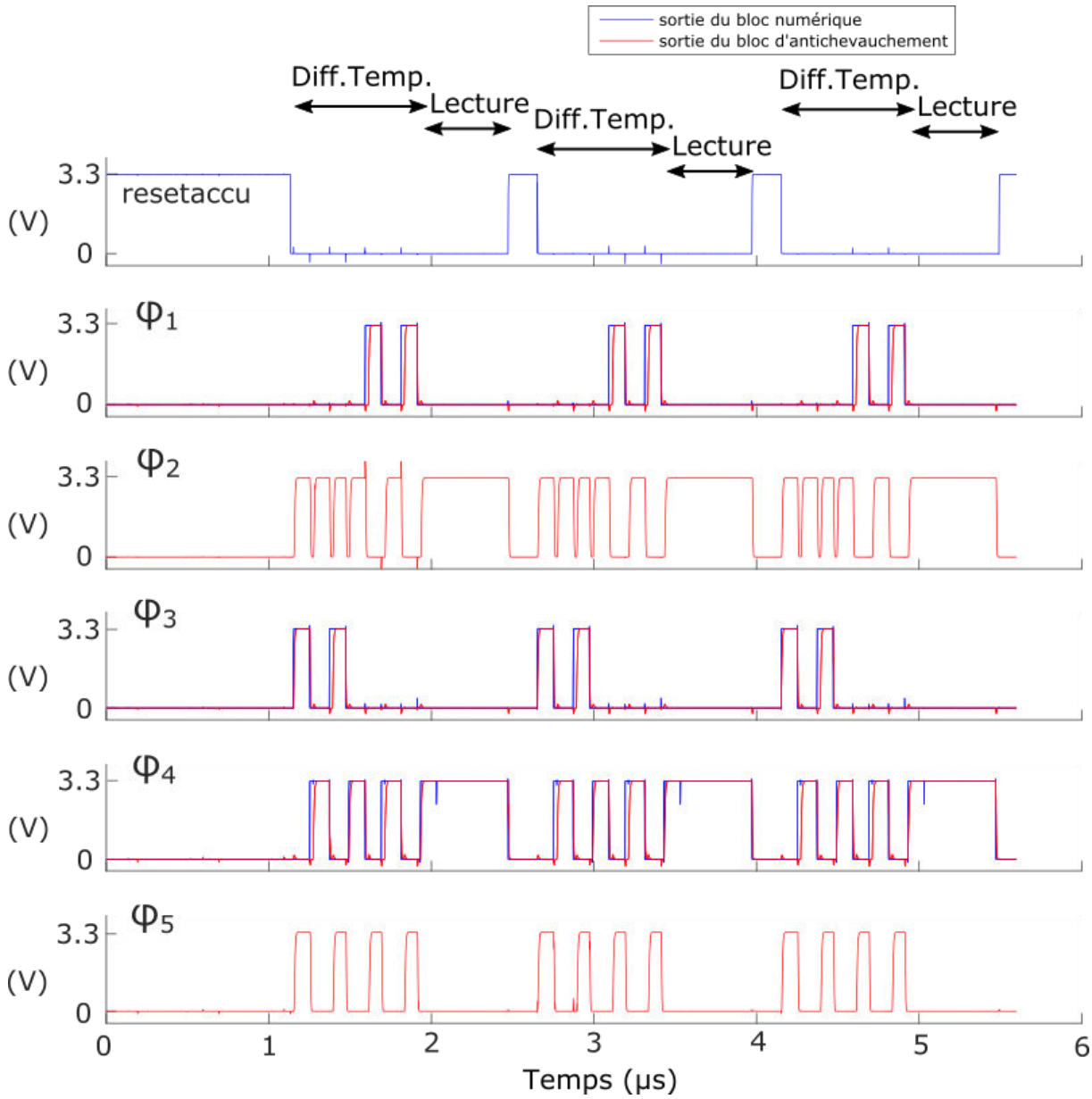


Figure 140 : Résultats en simulation transitoire : commandes d'accumulateurs pour une différence temporelle amplifiée par 2.

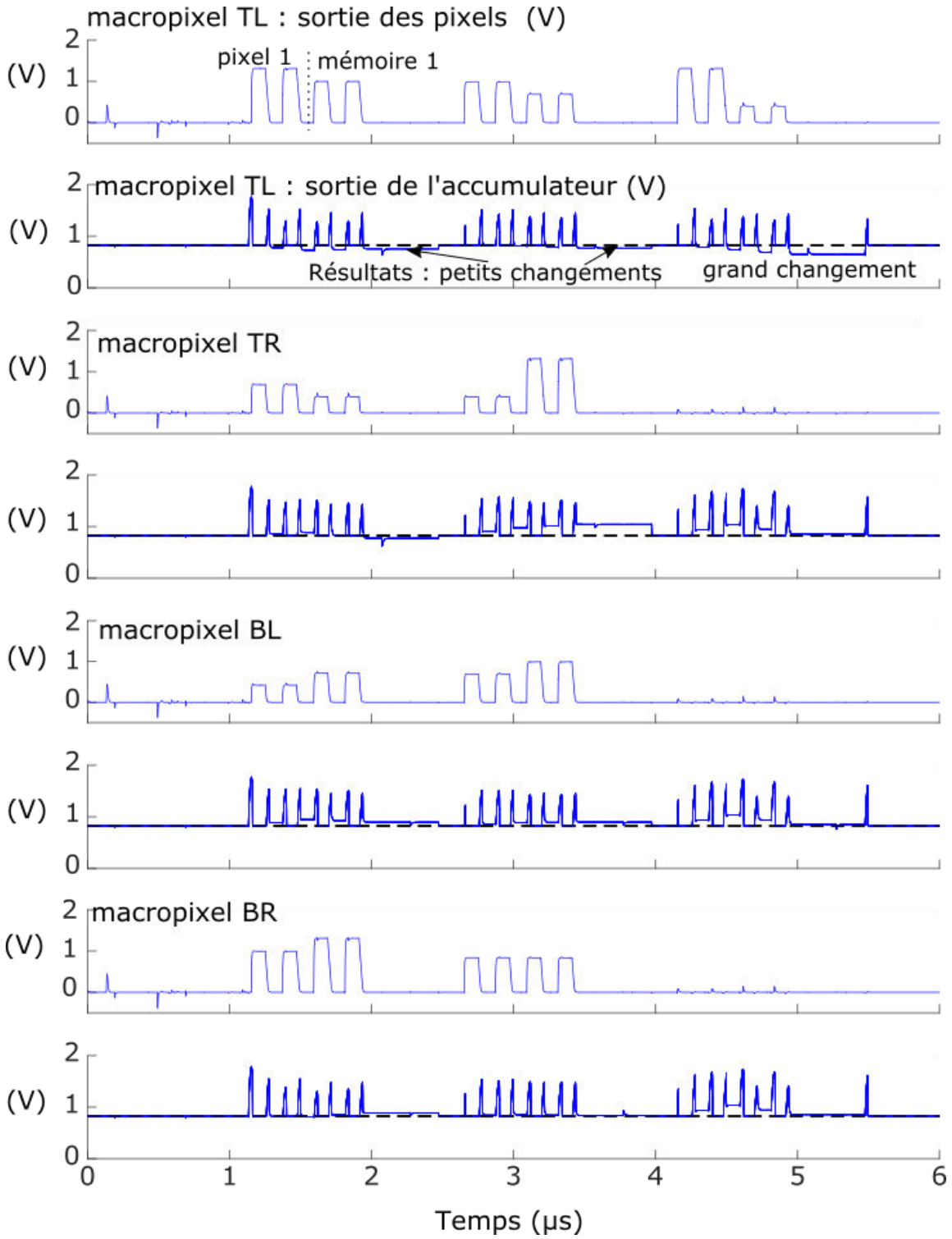


Figure 141 : Résultats en simulation transitoire : sorties de pixels et sorties d'accumulateurs pour une différence temporelle amplifiée par 2.

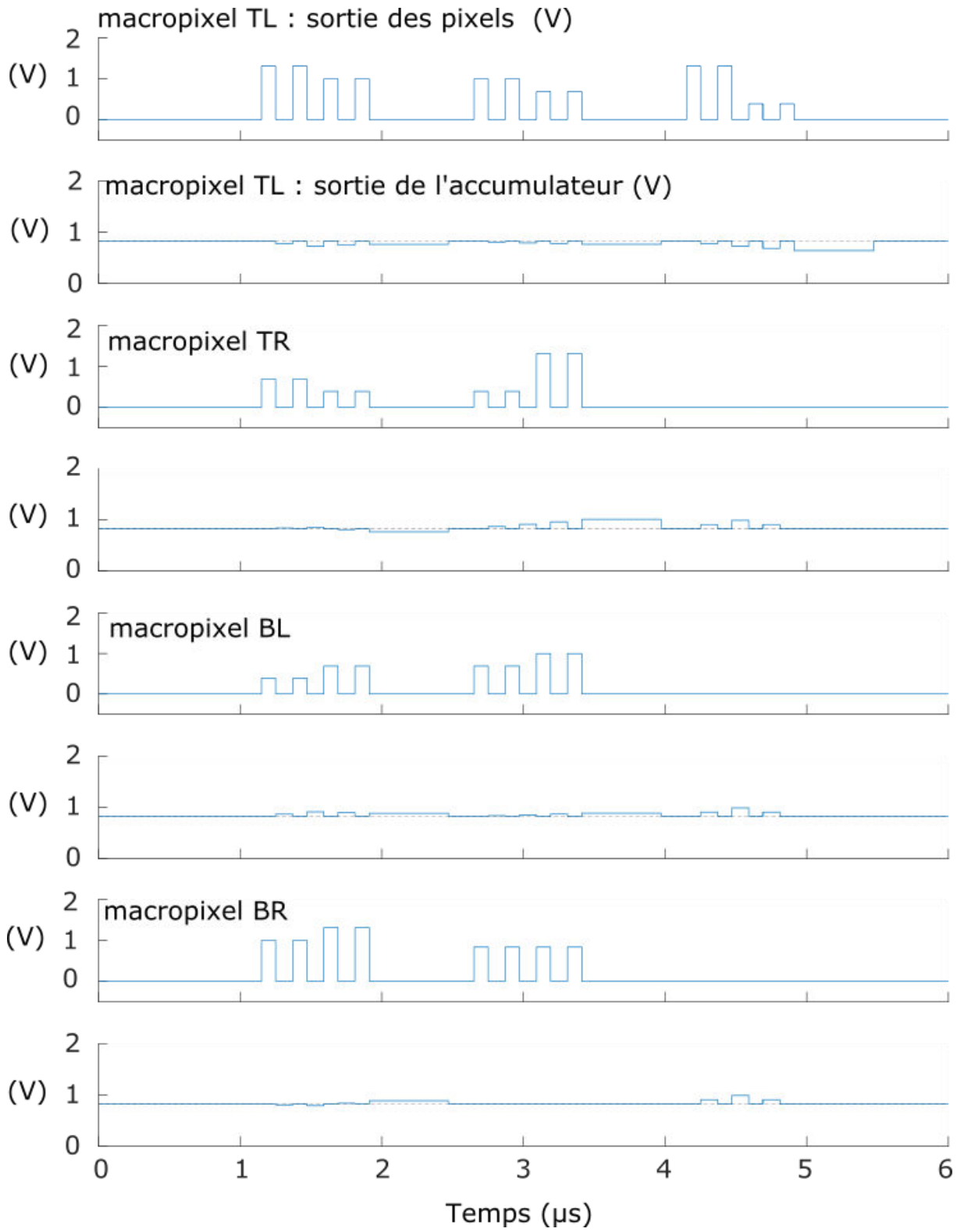


Figure 142 : Résultats transitoires attendus du circuit analogique (simulation fonctionnelle Matlab).

6.3.2 Validation sur deux images consécutives

Deux images consécutives sont extraites d'une vidéo d'un homme marchant et remuant les bras (utilisées également au chapitre 3). La même méthodologie que pour une convolution spatiale sur une image entière est appliquée : chaque image est découpée en blocs de 6x6 pixels. L'image utilisée est présentée en Figure 143, elle est formée de 2120 blocs. La Figure 143 illustre également les résultats en simulation fonctionnelle et sur le circuit proposé.

Dans les deux cas, les mouvements de bras et de jambes sont identifiables de la même manière, ce qui valide la fonctionnalité de notre circuit. En revanche, l'image de résultat de la simulation du circuit présente des défauts tels qu'un quadrillage sur le fond et l'apparition des nuances de gris de l'image originale. Cela peut s'expliquer par les choix de placement des composants dans la matrice qui ont été effectués. La section 5.7.1 du chapitre 5 a détaillé le placement de ces composants (photodiodes, circuits de pixels, mémoires,…) dans le motif de 6x6 pixels. Les pixels sont tous identiques mais la place des mémoires et donc leurs connexions diffèrent. Un photocourant ne donne pas la même valeur en sortie d'un pixel et en sortie de sa mémoire, et cette différence varie selon le pixel à mémoire utilisé dans le motif. La différence de valeur entre le pixel et sa mémoire explique la réapparition des nuances de gris des images originales sur les zones sans mouvement. La variation de cette différence entre les pixels à mémoire du motif élémentaire explique le quadrillage obtenu : celui-ci est la répétition de la réponse du motif élémentaire de 6x6 pixels sur une zone unie et sans mouvement comme le montre la Figure 144. Comme on le voit sur la Figure 143, ces erreurs introduites sont suffisamment faibles pour cela ne soit pas gênant pour identifier des mouvements classiques tels que ceux de l'homme.

Cette remarque sur la différence de réponse d'un pixel selon le placement de sa mémoire conforte le choix de placer tous les circuits directs de pixels de la même façon par rapport à leur photodiode, tel que cela a été présenté au chapitre 5. Sans cela, les résultats de convolution spatiale présentés ci-dessus seraient dégradés.

La différence temporelle n'a pas été simulée en Monte-Carlo au vu des temps de simulation et du fait que pour cette fonction nous n'avons pas de critère pour juger si la distribution des résultats serait trop étalée ou pas.

6.4 Estimation de consommation

Les simulations sur vues extraites permettent d'estimer la consommation de la matrice. Ses sources de consommation principales sont les inverseurs des PEs et les sources de courant I_{mac} polarisant les pixels. L'alimentation du circuit est 3.3V. Un inverseur consomme 21µA de courant en permanence, soit 70µW. Les sources de courants délivrent 5µA lorsqu'un pixel est sélectionné, c'est-à-dire lors d'un chargement de la capacité d'entrée pendant un calcul, soit pendant environ la moitié du temps de calcul. Ceci donne une estimation de consommation moyenne par macropixel de 78µW pendant un calcul. Avec l'estimation de vitesse de 5MOPS présentée dans la section 6.2.1, cela revient à 64GOPS/W.

Les deux circuits ayant le même indice de versatilité dans l'état de l'art présenté au chapitre 2 sont des puces qui ont été effectivement fabriquées. Massari *et al*. mesurent 6mW de consommation pour une matrice de 32x32 pixels qui réalise le même genre de prétraitements que le circuit proposé ici [77 - Massari 2005]. Cela est du même ordre de grandeur que les 78µW estimés pour 3x3 pixels dans notre cas. En revanche, Fernandez-Berni *et al*. mesurent une consommation beaucoup plus faible : 5.6mW sur une puce de 176x144 pixels [78 - Fernández-Berni 2011]. Cela peut s'expliquer par le fait que le circuit de calcul est passif, contrairement à ceux de [77 - Massari 2005] et de ce travail. En effet, ici la plus grosse consommation vient de l'inverseur utilisé en amplificateur qui utilise 70µW en permanence. Nos estimations et ces comparaisons nous mènent donc à conclure qu'un circuit de PE en capacités commutées n'est pas le meilleur choix d'un point de vue consommation.

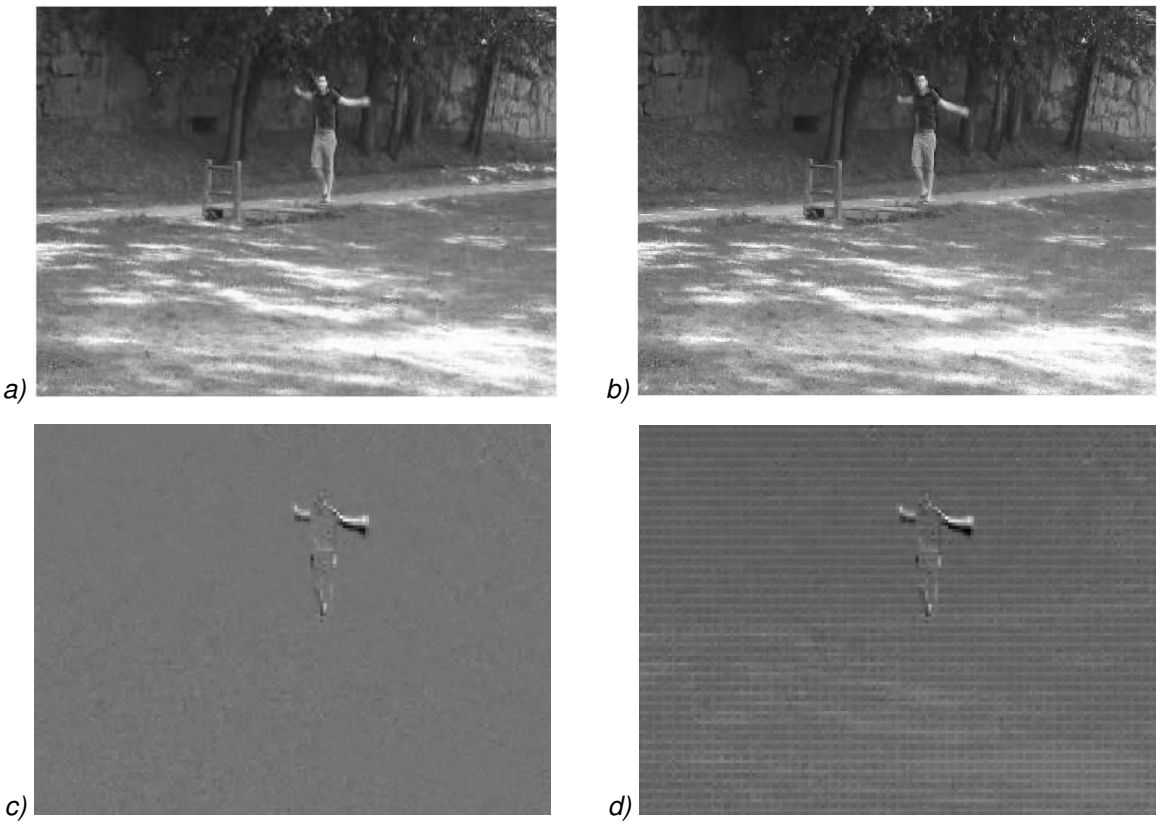


Figure 143 : (a, resp. b) Première (resp. deuxième) image originale (320x240 pixels), (c, resp. d) différence sous-échantillonnée par 2x2 pixels entre les images a et b en simulation fonctionnelle (resp. en simulation du circuit) (160x120 pixels).

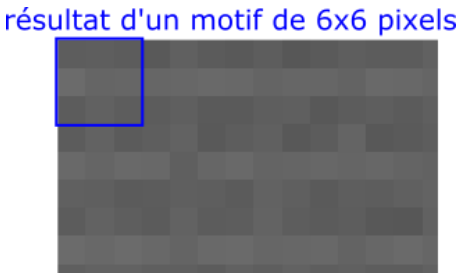


Figure 144 : Détail du coin nord-ouest de la Figure 143(d).

L'architecture haut niveau a bien été choisie pour limiter la consommation du système (pas de transmission ni de conversion analogique numériques des images brutes) mais le circuit du PE n'a pas été conçu en basse consommation. Une amélioration de notre travail peut donc être de concevoir un autre type de circuit de PE capable des mêmes fonctions que le circuit SC proposé ici mais à plus basse consommation.

L'imageur intelligent proposé dans ce travail a donc été testé au niveau dessin pour la matrice analogique et au niveau synthèse logique pour le contrôle numérique. Il a été validé sur des simulations transitoires et sur des images complètes, à la fois en détection de contours de type Sobel (pour représenter la convolution spatiale) et en différence temporelle. Une détection de piétons complète n'a pas pu être réalisée mais les éléments présentés indiquent que les prototypes ne donneraient pas de résultats suffisants sans adéquation plus approfondie de l'algorithme de haut niveau avec l'architecture

du circuit proposé. Le choix initial de concevoir un PE travaillant en calcul approximé est à revoir pour une application telle quelle de détection de piétons, et le circuit à reconcevoir en conséquence, selon la méthodologie du chapitre 4. En termes de consommation, chaque macropixel nécessite en moyenne 70µW pendant un calcul ce qui est élevé comparé à un imageur intelligent utilisant un circuit passif de PE.

Conclusion et perspectives

Le travail présenté dans ce mémoire est fondé sur le constat qu'un imageur intelligent doit faire face à un compromis entre surface photosensible disponible et versatilité des traitements intégrés dans la matrice de façon parallèle. Deux figures de mérites ont été introduites pour classer les architectures proposées dans la littérature en termes d'efficacité spatiale des traitements d'une part et de programmabilité d'autre part. Le $FoM_{surface}$ évalue la surface utilisée par l'architecture en essayant de s'affranchir de l'impact de la technologie et de la taille du pixel choisi. L'indice de versatilité propose une quantification de la versatilité des imageurs intelligents en affectant une valeur à chaque fonction réalisée. Ces deux figures de mérites permettent de montrer la limite des meilleurs compromis actuels des architectures massivement parallèles, c'est-à-dire qui bénéficient d'un calcul rapide, distribué dans la matrice, au détriment de la surface photosensible.

Ce travail s'est donc intéressé à une architecture par macropixels, afin de réduire l'impact sur la surface photosensible d'une architecture de calcul massivement parallèle. Une approche algorithmique, résultant de la mise en adéquation de prétraitements classiques à cette architecture, a été proposée : la convolution spatiale et la différence temporelle sous-échantillonnées. Une validation fonctionnelle a été effectuée. Il a été montré que l'utilisation de la CSE dégrade dans une limite acceptable les performances d'un algorithme de plus haut niveau tel que la détection de piéton. Une nouvelle architecture d'imageur intelligent a donc été proposée, pour effectuer des convolutions sous-échantillonnées par 3x3 pixels et des différences temporelles sous-échantillonnées par 2x2 pixels. Il s'agit d'une matrice dont le motif élémentaire contient 6x6 pixels, regroupés en 4 macropixels possédant chacun un élément de calcul et entre 1 et 4 mémoires. Ces PEs doivent être programmables pour effectuer selon la commande des convolutions spatiales de masque quelconque ou des différences temporelles amplifiées, c'est-à-dire des combinaisons linéaires de pixels ou de leur mémoire. Leur contrainte de conception essentielle est une surface minimisée afin de ne pas dégrader la surface photosensible de la matrice.

Notre étude a conduit au choix d'un calcul séquentiel analogique grâce à un accumulateur de charges, contrôlé par une commande numérique. Une méthodologie de dimensionnement de ce circuit à capacités commutées sous contrainte de surface a été proposée : le dimensionnement des capacités est l'élément critique et dépend essentiellement du nombre d'accumulations souhaitées. Les compromis entre précision du calcul et surface occupée ont été discutés en calculant notamment l'influence du désappariement des capacités. Cette étude a montré que le dessin des capacités n'est pas nécessairement critique, selon la précision souhaitée pour les calculs.

Enfin, ce travail a proposé une implémentation du système. Les éléments de la matrice, c'est-à-dire les pixels avec ou sans mémoire et les accumulateurs ont été codimensionnés puis dessinés. Il en résulte un facteur de remplissage de 28% et un $FoM_{surface}$ de 18.4 pour un indice de versatilité de 4,5, ce qui se positionne dans la meilleure partie de l'état de l'art, mais sans faire mieux pour autant. Le bloc numérique a été réalisé en VHDL et synthétisé, il est capable de commander de façon programmable des captures d'images, des convolutions spatiales de masque quelconque, des différences temporelles amplifiées ou non et des lectures d'images ou de résultats de prétraitement.

Les circuits numériques et analogiques conçus ont été simulés conjointement, le premier synthétisé et le second en vue extraite, interfacés par un circuit d'antichevauchement des commandes de l'accumulateur. Les fonctions du système ont été validées sur des résultats transitoires et des images complètes, à la fois pour la différence temporelle et la convolution spatiale dans le cas du masque Sobel horizontal (il s'agit du seul masque testé à cause des temps de simulation trop importants). Un résumé des caractéristiques du circuit est présenté dans le Tableau 18.

Technologie	AMS C35 B4C3 : 4 métaux, 2 poly
Pas de l'imageur (pitch)	56.35x57.7µm²
Facteur de remplissage	28%
FoMsurface	18.4
Type de PE	Capacités commutées
Taille du PE	(12x57µm²)x2
Consommation du PE	70µW
Fonctions	Convolution spatiale programmable, différence temporelle et prise d'image brute
Indice de versatilité	4.5

Tableau 18 : Caractéristiques du circuit proposé, simulé en vue extraite.

Si les images obtenues sont correctes visuellement, en revanche des simulations de Monte-Carlo sur vue extraite suggèrent que l'algorithme de détection de piéton ne fonctionnerait pas tel quel avec la plupart des prototypes. En effet, l'étude fonctionnelle nous a conduits à choisir un dimensionnement de PE pour du calcul approximé alors qu'une étude plus approfondie des erreurs introduites dans l'architecture à capacités commutées choisie indique un besoin de précision plus grande sur les résultats de calcul. Une première piste d'amélioration serait donc de réutiliser la méthodologie de conception analogique développée, avec des contraintes plus grandes en précision : redimensionner les capacités d'entrée ou de compensation, ne pas négliger les techniques de limitation des injections de charges comme les interrupteurs fantômes,... Cela réduirait le $FoM_{surface}$ obtenu. Pour éviter cela, l'adéquation entre l'algorithme de détection de piétons et le circuit proposé pourrait aussi être plus explorée : une adaptation des paramètres du HOG plus poussée que ce qui est proposé ici, une autre méthode d'apprentissage automatique, un entrainement particulier, etc. De plus, une calibration du circuit en post-traitement des résultats pourrait également améliorer sa précision (en limitant l'influence des variations de process).

Par ailleurs l'objectif ici était de proposer une preuve de concept. En premier lieu, des éléments du système mériteraient une optimisation comme la conception du pixel et en particulier sa photodiode, le dessin général de l'analogique pour obtenir un meilleur $FoM_{surface}$, mais aussi les temps caractéristiques des commandes (période des commandes de l'accumulateur, temps d'antichevauchement,…).

Des améliorations de l'architecture peuvent aussi être étudiées. Par exemple, dans le circuit à capacités commutées à base d'inverseur, la variation du point de fonctionnement de l'inverseur est inconnue et variable d'un inverseur à l'autre, et induit une erreur sur le résultat. Dans le circuit étudié, celle-ci est compensée par l'ajout d'une capacité. Or le point de fonctionnement est variable d'un inverseur à l'autre mais constant pour un inverseur donné. La correction du point de fonctionnement par calibration pourrait donc être étudiée. Cela permettrait d'enlever du macropixel la capacité de compensation, qui occupe dans le dessin proposé ici une place non négligeable.

Ensuite, les mémoires introduites dans la matrice, au nombre d'une pour 4 pixels dans cette architecture, sont réalisées par l'ajout d'un circuit de pixel supplémentaire : quelques transistors (transfert, sélection, drain commun) et une capacité de diffusion. Certains pixels ont donc deux chaines de transfert et stockage de charges issues de la photodiode. Cette architecture de pixel est proche de celle des pixels à capacité de débordement : la capacité mémoire pourrait être utilisée en stockage des charges issues d'une photodiode éblouie. Cela permettrait d'augmenter la dynamique de l'imageur. Le circuit présenterait donc une fonction imagerie HDR en plus de ses fonctions actuelles. L'image à grande dynamique serait sous-échantillonnée par 2x2 pixels vu la répartition des capacités mémoires. Une première version de commande pour cette fonction a été intégrée au bloc numérique, mais pas encore testée sur le circuit analogique.

Par ailleurs si la convolution sous-échantillonnée ne tolère pas une grande erreur, la convolution complète semble plus adaptée à du calcul approximé comme le suggèrent des résultats de simulation fonctionnelle à compléter. Pour faire une convolution classique avec des ressources rassemblées par macropixels, les PEs doivent pouvoir s'échanger des résultats entre eux. Ainsi, l'architecture en macropixels et le circuit de PE dimensionné dans ce travail pourraient être conservés, sous réserve de modifier le contrôle numérique et les interconnexions pour avoir des PEs communicants entre eux, capables de réaliser des convolutions classiques.

Le motif élémentaire de la matrice proposée est asymétrique pour réaliser une convolution spatiale sous-échantillonnée par 3x3 pixels et une différence temporelle par 2x2 pixels. Le circuit proposé est fonctionnel sur ces deux aspects, avec une erreur sur la convolution spatiale indépendante de cette asymétrie. En revanche les images de différence temporelle font apparaître des artefacts dus au placement asymétrique des composants dans les macropixels. Ils semblent négligeables, mais un critère plus objectif de qualité de la différence temporelle (équivalent à la détection de piétons en convolution spatiale par exemple) serait nécessaire pour l'assurer.

En termes de consommation d'énergie, le circuit de PE proposé n'est pas très efficace puisqu'il possède un inverseur qui consomme un courant statique. Une piste d'amélioration possible est de suspendre son alimentation lorsqu'il ne travaille pas, c'est-à-dire au repos du système ou encore pendant les prises d'images. Pour réduire la consommation pendant les traitements, il conviendrait d'étudier une autre architecture de PE qu'un circuit à capacités commutées actif.

Enfin, l'implémentation proposée dans ce travail a été réalisée en technologie standard, i.e. non optique et planaire. Cependant, les technologies de type BSI ou 3D deviennent de plus en plus courantes. Elles permettent de placer plus de calculs dans la matrice, sans détériorer la surface photosensible : le facteur de remplissage est proche de 100%. L'architecture par macropixel n'est pas pour autant inutile dans ces technologies : regrouper les éléments de calculs sur la surface de plusieurs pixels permet toujours théoriquement d'introduire plus de versatilité pour une même taille de pixels. Il pourrait donc être intéressant d'implémenter l'architecture de système proposée ici dans une technologie 3D ou BSI. Ce changement de technologie, ou une évolution vers un nœud technologique plus petit, pourrait aussi modifier l'architecture proposée. En effet, le contrôle numérique a été choisi comme extérieur à la matrice dans cette implémentation au vu de la discussion proposée. Mais plus de place pour le PE et des transistors plus petits pourraient conduire à une distribution de portes logiques dans la matrice pour diminuer le nombre de pistes de commandes.

Pour terminer, ces travaux ont contribué à explorer les architectures d'imageurs intelligents par macropixels en proposant une validation fonctionnelle avec critère de qualité de traitement, une nouvelle architecture asymétrique, une méthodologie de conception de circuits à capacités commutés sous contrainte de surface et un système dimensionné et dessiné. Les performances finales, en termes de précision ou de surface occupée sont en deça de celles espérées, mais l'étude des améliorations envisagées devraient permettre d'améliorer le compromis entre versatilité et surface photosensible de l'état de l'art.

Au-delà de cette architecture, l'intégration en général de traitements dans le capteur semble prometteur. Dans le cas de la vision, les technologies planaires seront bientôt obsolètes mais les technologies 3D devraient permettre de résoudre le compromis avec la surface photosensible. Des projets portant sur l'intégration de plus de traitements, ou de traitements de plus haut niveau, par répartition spatiale des calculs dans un imageur en 3D seraient à développer. Les projets à venir pourraient également se pencher, dans un contexte plus large, sur la comparaison d'implémentation en calcul approximé en numérique (ADC de faible résolution, calcul approximé) et en analogique, en termes de surface, consommation, facilité d'implémentation et versatilité.

Bibliographie

[1] Bovik, A., *The Essential Guide To Image Processing*. 2009.

[2] Jähne, B., *Digital Image Processing*. 2005.

[3] Sponton, H. and Cardelino, J., "A Review of Classic Edge Detectors," vol. 5, pp. 90–123, 2015.

[4] Roberts, L.G., "Machine perception of three-dimensional solids," *Phd, Massachusetts Inst. Technol.*, 1963.

[5] Harris, C. and Stephens, M., "A Combined Corner and Edge Detector," *Procedings Alvey Vis. Conf. 1988*, pp. 147–151, 1988.

[6] Hough, P.V.C., "Machine analysis of bubble chamber pictures," *2nd Int. Conf. High-Energy Accel. Instrum.*, vol. 73, pp. 554–558, 1959.

[7] Duda, R.O. and Hart, P.E., "Use of the Hough transform to detect lines and cures in pictures," *Commun. Assoc. Comput. Mach.*, vol. 15, no. 1, pp. 11–15, 1972.

[8] Nomura, Y., Yamamoto, S., Yoshihara, K., and Hashimoto, T., "A feasibility study of accurate 3D measurement of ships using dense stereo vision system," *2016 Techno-Ocean*, pp. 562–565, 2016.

[9] Kolar, A. and Duranton, M., "Integrated three dimensional vision sensor," *US Pat. 9532030*, 2016.

[10] Dalal, N. and Triggs, B., "Histograms of Oriented Gradients for Human Detection," 2005.

[11] Viola, P. and Jones, M.J., "Rapid object detection using a boosted cascade of simple features," *Comput. Vis. Pattern Recognit.*, vol. 1, p. I--511--I--518, 2001.

[12] Feruglio, S., Lu, G.-N., Garda, P., and Vasilescu, G., "A Review of the CMOS Buried Double Junction (BDJ) Photodetector and its Applications," *Sensors*, vol. 8, no. 10, pp. 6566–6594, 2008.

[13] Belbachir, A.N., *Smart Cameras*. 2010.

[14] Weckler, G.P., "Operation of p-n Junction Photodetectors in a Photon Flux Integrating Mode," *IEEE J. Solid-State Circuits*, vol. 2, no. 3, pp. 65–73, 1967.

[15] Navarro, D., "Architecture et Conception de Rétines Silicium CMOS : Application à la mesure du flot optique," *Phd*, 2003.

[16] Kleinman, D.A., "The Forward Characteristic of the PIN Diode," *Bell Syst. Tech. J.*, vol. 35, no. 3, pp. 685–706, 1956.

[17] Caramona Fernandes, C.M. and Torres Pereira, J.M., "Bandwidth modeling and optimization of PIN photodiodes," *EUROCON 2011 - Int. Conf. Comput. as a Tool - Jt. with Conftele 2011*, pp. 2–5, 2011.

[18] Seitz, P. and Theuwissen, A.J.P., *Single-Photon Imaging*. 2011.

[19] Cavadore, C., "Design and characterization of CMOS-APS," *These*, vol. 1, no. 1, pp. 1–239.

[20] Ohta, J., *Smart CMOS Image Sensors and Applications*. 2007.

[21] Knickerbocker, J.U., Andry, P.S., Dang, B., Horton, R. r., Patel, C.S., Polastre, R.J., Sakuma, K., Sprogis, E.S., Tsang, C.K., Webb, B.C., and Wright, S.L., "3D silicon integration," *2008 Electron. Components Technol. Conf.*, pp. 538–543, 2008.

[22] Akahane, N., Sugawa, S., Adachi, S., Mori, K., Ishiuchi, T., and Mizobuchi, K., "A Sensitivity and Linearity Improvement of a 100dB Dynamic Range CMOS Image Sensor using a Lateral Overflow Integration Capacitor," *IEEE J. Solid-State Circuits*, vol. 44, no. 10, pp. 352–354, 2006.

[23] Mann, S. and Picard, R.W., "On being 'undigital' with digital cameras : extending dynamic range by combining differently exposed pictures," *Proc. IS T Annu. Meet. 48th.*, 1996.

[24] Peizerat, A., Guezzi, F., Dupret, A., Jalby, R., Bruno de Sa, L., Benetti, M., Guicquero, W., and Blanchard, "A 120dB DR and 5µm pixel pitch imager based on local integration time adaptation," pp. 5–8, 2015.

[25] Zhao, X., Bermak, A., and Boussaid, F., "A CMOS digital pixel sensor with photo-patterned micropolarizer array for real-time focal-plane polarization imaging," *2008 IEEE-BIOCAS Biomed. Circuits Syst. Conf. BIOCAS 2008*, no. 1, pp. 145–148, 2008.

[26] Bayer, B.E., "Color Imaging Array," *U. S. Pat. 3971 065*, p. 10, 1976.

[27] Fossum, E., "CMOS Image Sensors : Electronic Camera-On-A-Chip," vol. 44, no. 10, pp. 1689–1698, 1997.

[28] Noble, P.J.W., "Self-scanned silicon image detector arrays," *IEEE Trans. Electron Devices*, vol. 15, no. 4, pp. 202–209, 1968.

[29] Mori, M., Katsuno, M., Kasuga, S., Murata, T., and Yamaguchi, T., "1/4-inch 2-mpixel MOS image sensor with 1.75 transistors/pixel," *IEEE J. Solid-State Circuits*, vol. 39, no. 12, pp. 2426–2430, 2004.

[30] Takayanagi, I., Mo, Y., Ando, H., Kawamura, K., Yoshimura, N., Kimura, K., Otaka, T., Matsuo, S., Suzuki, T., Brady, F., and Nakamura, J., "A 600x600 Pixel, 500 fps CMOS image sensor with a 4.4 um pinned photodiode 5-transistor global shutter pixel," *Image Sens. Work.*, pp. 2–5, 2007.

[31] Gogniat, G., Milojevic, D., Morawiec, A., and Erdogan, A., *Algorithm-Architecture Matching for Signal and Image Processing*. 2011.

[32] Kavadias, S., Dierickx, B., Scheffer, D., Alaerts, A., Uwaerts, D., and Bogaerts, J., "A logarithmic response CMOS image sensor with on-chip calibration," *IEEE J. Solid-State Circuits*, vol. 35, no. 8, pp. 1146–1152, 2000.

[33] Sugawa, S., Akahane, N., Adachi, S., Ishiuchi, T., Mori, K., and Mizobuchi, K., "A 100dB Dynamic Range CMOS Image Sensor using a Lateral Overflow Integration Capacitor," *Image (Rochester, N.Y.)*, vol. 44, no. 10, pp. 352–354, 2005.

[34] Marsh, B., Das, D., Sedgwick, I., Turchetta, R., Bayer, M., Correa, J., Gottlicher, P., Lange, S., Marras, A., Shevyakov, I., Smoljanin, S., Viti, M., Wunderer, C.B., Xia, Q., Zimmer, M., Cautero, G., Giuressi, D., Menk, R., Stebel, L., Yousef, H., Marchal, J., Pedersen, U., Rees, N., Tartoni, N., and Graafsma, H., "PERCIVAL: The design and characterisation of a CMOS image sensor for direct detection of low-energy X-rays," *2014 IEEE Nucl. Sci. Symp. Med. Imaging Conf. NSS/MIC 2014*, pp. 3–6, 2014.

[35] Villa, F., Lussana, R., Bronzi, D., Tisa, S., Tosi, A., Zappa, F., Dalla Mora, A., Contini, D., Durini, D., Weyers, S., and Brockherde, W., "CMOS Imager With 1024 SPADs and TDCs for Single-Photon Timing and 3-D Time-of-Flight," *IEEE J. Sel. Top. Quantum Electron.*, vol. 20, no. 6, 2014.

[36] Wu, X. and Bermak, A., "A low power digital pixel sensor with a dynamically biased ADC," *2008 Int. SoC Des. Conf. ISOCC 2008*, vol. 1, pp. 105–108, 2008.

[37] Chen, D.G., Matolin, D., Bermak, A., and Posch, C., "Pulse Modulation Imaging - Review and Performance Analysis," *Biomed. Circuits Syst. IEEE Trans.*, vol. 5, no. 1, pp. 64–82, 2011.

[38] Zhang, M. and Bermak, A., "A compact digital pixel sensor architecture using predictive coding scheme," *2008 IEEE Sensors*, pp. 961–964, 2008.

[39] Ng, K.T., Chen, S., Boussaid, F., and Bermak, A., "Compact Gray-Code Counter/Memory

Circuits for Spiking Pixels," *4th IEEE Int. Symp. Electron. Des. Test Appl. (delta 2008)*, pp. 506–511, 2008.

[40] Lichtsteiner, P., Posch, C., and Delbruck, T., "A 128x128 120dB 15μs latency asynchronous temporal contrast vision sensor," *IEEE J. Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, 2008.

[41] Uhring, W., Millet, L., Misischi, B., Rarbi, F., Guellec, F., Dzahini, D., Maciu, O., Kammerer, J.-B., and Sicard, G., "A Scalable Architecture for Multi Millions Frames per Second CMOS Sensor With Digital Storage," *IEEE Int. New Circuits Syst. Conf.*, pp. 252–255, 2018.

[42] Zarandy, A., *Focal-plane Sensor-processor Chips*. 2011.

[43] Bourrasset, C., Maggiani, L., Sérot, J., Berry, F., and Pagano, P., "Distributed FPGA-based Smart Camera Architecture for Computer Vision Applications," *Int. Conf. Distrib. Smart Cameras*, no. Nios Ii, 2013.

[44] Zhang, W., Fu, Q., and Wu, N., "A programmable vision chip based on multiple levels of parallel processors," *IEEE J. Solid-State Circuits*, vol. 46, no. 9, pp. 2132–2147, 2011.

[45] Yang, J., Yang, Y., Chen, Z., Liu, L., Liu, J., and Wu, N., "A Heterogeneous Parallel Processor for High-Speed Vision Chip," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8215, no. c, pp. 1–1, 2016.

[46] Lindgren, L., Melander, J., Johansson, R., and Möller, B., "A Multiresolution 100-GOPS 4-Gpixels/s Programmable Smart Vision Sensor for Multisense Imaging," *IEEE J. Solid-State Circuits*, vol. 40, no. 6, pp. 1350–1359, 2005.

[47] Wu, N., "Neuromorphic vision chips," *Sci. CHINA Inf. Sci.*, vol. 33, no. 5, pp. 38–46, 2018.

[48] Posch, C., Serrano-Gotarredona, T., Linares-Barranco, B., and Delbruck, T., "Retinomorphic event-based vision sensors: Bioinspired cameras with spiking output," *Proc. IEEE*, vol. 102, no. 10, pp. 1470–1484, 2014.

[49] Thorpe, S.J., Brilhault, A., and Perez-Carrasco, J.A., "Suggestions for a biologically inspired spiking retina using order-based coding," *ISCAS 2010 - 2010 IEEE Int. Symp. Circuits Syst. Nano-Bio Circuit Fabr. Syst.*, pp. 265–268, 2010.

[50] Carmona-Galán, R., Zarándy, Á., Rekeczky, C., Földesy, P., Rodríguez-Pérez, A., Domínguez-Matas, C., Fernández-Berni, J., Liñán-Cembrano, G., Pérez-Verdú, B., Kárász, Z., Suárez-Cambre, M., Brea-Sánchez, V., Roska, T., and Rodríguez-Vázquez, Á., "A hierarchical vision processing architecture oriented to 3D integration of smart camera chips," *J. Syst. Archit.*, vol. 59, no. 10 PART A, pp. 908–919, 2013.

[51] Gruev, V., Spiegel, J. Van Der, Philipp, R.M., and Etienne-Cummings, R., "Image sensor with general spatial processing in a 3D integrated circuit technology," *2006 IEEE Int. Symp. Circuits Syst.*, pp. 4963–4966, 2006.

[52] Suarez, M., Brea, V., Fernandez-Berni, J., Carmona-Galan, R., Linan, G., Cabello, D., and Rodriguez-Vazquez, Á., "CMOS-3D Smart Imager Architectures for Feature Detection," *IEEE J. Emerg. Sel. Top. Circuits Syst.*, vol. 2, no. 4, pp. 723–736, 2012.

[53] Gruev, V. and Etienne-Cummings, R., "Implementation of steerable spatiotemporal image filters on the focal plane," *IEEE Trans. Circuits Syst. II Analog Digit. Signal Process.*, vol. 49, no. 4, pp. 233–244, 2002.

[54] Mehta, S. and Etienne-Cummings, R., "Normal Flow Measurement Visual Motion Sensor," *IEEE ISCAS 2006*, no. 1, pp. 959–962, 2006.

[55] Döge, J., Hoppe, C., Reichel, P., and Peter, N., "A 1 Megapixel HDR Image Sensor SoC with Highly Parallel Mixed-Signal Processing," *Image Sens. Work.*, 2015.

[56] Soell, C., Shi, L., Roeber, J., Reichenbach, M., Weigel, R., and Hagelauer, A., "Low-Power Analog Smart Camera Sensor for Edge Detection," *Image Process. (ICIP), 2016 IEEE Int. Conf.*, 2016.

[57] Takahashi, N. and Shibata, T., "A row-parallel cyclic-line-access edge detection CMOS image sensor employing global thresholding operation," *ISCAS 2010 - 2010 IEEE Int. Symp. Circuits Syst. Nano-Bio Circuit Fabr. Syst.*, pp. 625–628, 2010.

[58] Massari, N., De Nicola, M., Cottini, N., and Gottardi, M., "A 64x64 pixels 30µW vision sensor with binary data compression," *Sensors, 2010 IEEE*, pp. 118–122, 2010.

[59] Gottardi, M. and Lecca, M., "A 35 µW 64 x 64 Pixels Vision Sensor Embedding Local Binary Pattern Code Computation," *Circuits Syst.*, pp. 0–3, 2018.

[60] Ikeda, H., Tsuji, K., Asai, T., Yonezu, H., and Shin, J.K., "A novel retina chip with simple wiring for edge extraction," *IEEE Photonics Technol. Lett.*, vol. 10, no. 2, pp. 261–263, 1998.

[61] Costas-Santos, J., Serrano-Gotarredona, T., Serrano-Gotarredona, R., and Linares-Barranco, B., "A spatial contrast retina with on-chip calibration for neuromorphic spike-based AER vision systems," *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 54, no. 7, pp. 1444–1458, 2007.

[62] Lenero-Bardallo, J.A., Häfliger, P., Carmona-Galán, R., and Rodríguez-Vázquez, Á., "A Bio-Inspired Vision Sensor With Dual Operation and Readout Modes," *IEEE Sens. J.*, vol. 16, no. 2, pp. 317–330, 2016.

[63] Dubois, J., Ginhac, D., Paindavoine, M., and Heyrman, B., "A 10 000 fps CMOS sensor with massively parallel image processing," *IEEE J. Solid-State Circuits*, vol. 43, no. 3, pp. 706–717, 2008.

[64] Zhu, H. and Asada, K., "A superparallel image filtering digital-pixel-sensor employing a compressive multiplication technique," *IEEE Int. Conf. Electron. Circuits Syst.*, pp. 363–366, 2014.

[65] Katic, N., Schmid, A., and Leblebici, Y., "A retina-inspired robust on-focal-plane multi-band edge-detection scheme for CMOS image sensors," *Midwest Symp. Circuits Syst.*, pp. 683–686, 2014.

[66] Shi, B.E., "A low-power orientation-selective vision sensor," *IEEE Trans. Circuits Syst. II Analog Digit. Signal Process.*, vol. 47, no. 5, pp. 435–440, 2000.

[67] Barbaro, M., Burgi, P.Y., Mortara, A., Nussbaum, P., and Heitger, F., "A 100 x 100 pixel silicon retina for gradient extraction with steering filter capabilities and temporal output coding," *IEEE J. Solid-State Circuits*, vol. 37, no. 2, pp. 160–172, 2002.

[68] Nishimura, M. and Van Der Spiegel, J., "A CMOS Image Processing Sensor for the Detection of Image Features," *Analog Integr. Circuits Signal Process.*, vol. 45, no. 3, pp. 263–279, 2005.

[69] Yin, C. and Hsieh, C.-C., "A 1V 14kfps smart CMOS imager with tracking and edge-detection modes for biomedical monitoring," *2013 Int. Symp. onVLSI Des. Autom. Test*, pp. 1–4, 2013.

[70] Yin, C., Chiu, C., and Hsieh, C., "A 0.5V, 14.28kframes/s, 96.7dB Smart Image Sensor With Array-Level Image Signal Processing for IoT Applications," *IEEE Trans. Electron Devices*, vol. 63, no. 3, pp. 1–7, 2016.

[71] Simoni, A., Torelli, G., Maloberti, F., Sartori, A., Plevridis, S.E., and Birbas, A.N., "A Single-Chip Optical Sensor with Analog Memory for Motion Detection," *IEEE J. Solid-State Circuits*, vol. 30, no. 7, pp. 800–806, 1995.

[72] Chi, Y.M., Mallik, U., Clapp, M.A., Choi, E., Cauwenberghs, G., and Etienne-Cummings, R., "CMOS camera with in-pixel temporal change detection and ADC," *IEEE J. Solid-State Circuits*, vol. 42, no. 10, pp. 2187–2196, 2007.

[73] Huang, J., Guo, M., and Chen, S., "A Dynamic Vision Sensor with Direct Logarithmic Output and Full-frame Picture-On-Demand," *IEEE Int. Symp. Circuits Syst.*, 2017.

[74] Cottini, N., Gottardi, M., Massari, N., Passerone, R., and Smilansky, Z., "A 33µW 64×64 pixel vision sensor embedding robust dynamic background subtraction for event detection and scene interpretation," *IEEE J. Solid-State Circuits*, vol. 48, no. 3, pp. 850–863, 2013.

[75] Olumodeji, O.A., Bramanti, A.P., and Gottardi, M., "Memristor-Based Pixel for Event-Detection

Vision Sensor," *IEEE SENSORS*, pp. 1–4, 2015.

[76] Kim, D. and Culurciello, E., "Tri-mode smart vision sensor with 11-transistors/pixel for wireless sensor networks," *IEEE Sens. J.*, vol. 13, no. 6, pp. 2102–2108, 2013.

[77] Massari, N., Gottardi, M., Gonzo, L., Stoppa, D., and Simoni, A., "A CMOS image sensor with programmable pixel-level analog processing," *IEEE Trans. Neural Networks*, vol. 16, no. 6, pp. 1673–1684, 2005.

[78] Fernández-Berni, J., Carmona-Galán, R., and Carranza-Gonzalez, L., "FLIP-Q: A QCIF resolution focal-plane array for low-power image processing," *IEEE J. Solid-State Circuits*, vol. 46, no. 3, pp. 669–680, 2011.

[79] Fernandez-Berni, J., Acasandrei, L., Carmona-Galan, R., Barriga-Barros, A., and Rodriquez-Vazquez, A., "Power-efficient focal-plane image representation for extraction of enriched Viola-Jones features," *ISCAS 2012 - 2012 IEEE Int. Symp. Circuits Syst.*, no. 1, pp. 3122–3125, 2012.

[80] Carmona-Galan, R., Fernandez-Berni, J., and Rodriguez-Vazquez, Á., "Automatic DR and Spatial Sampling Rate Adaptation for Secure and Privacy-Aware ROI Tracking Based on Focal-Plane Image Processing," *Image Sens. Work.*, 2015.

[81] Fernández-Berni, J., Carmona-Galán, R., and Rodríguez-Vázquez, Á., "Methodology and Implementation of Early Vision Tasks for the Viola-Jones Processing Framework," *Work. Archit. Smart Cameras*, 2013.

[82] Poikonen, J., Laiho, M., and Paasio, A., "Locally adaptive image sensing with the 64x64 cell MIPA4k mixed-mode image processor array," *IEEE Int. Symp. Circuits Syst.*, pp. 93–96, 2009.

[83] Carey, S.J., Lopich, A., Barr, D.R.W., Wang, B., and Dudek, P., "A 100,000 fps Vision Sensor with Embedded 535GOPS / W 256x256 SIMD Processor Array," *VLSI Circuits (VLSIC), 2013 Symp.*, pp. 182–183, 2013.

[84] Lopich, A. and Dudek, P., "ASPA: Focal plane digital processor array with asynchronous processing capabilities," *Proc. - IEEE Int. Symp. Circuits Syst.*, pp. 1592–1595, 2008.

[85] Rodríguez-Vázquez, A., Liñán-Cembrano, G., Carranza, L., Roca-Moreno, E., Carmona-Galán, R., Jiménez-Garrido, F., Domínguez-Castro, R., and Meana, S.E., "ACE16k: The third generation of mixed-signal SIMD-CNN ACE chips toward VSoCs," *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 51, no. 5, pp. 851–863, 2004.

[86] Clapp, M.A. and Etienne-Cummings, R., "A dual pixel-type array for imaging and motion centroid localization," *IEEE Sens. J.*, vol. 2, no. 6, pp. 529–548, 2002.

[87] Massari, N. and Gottardi, M., "A 100 dB dynamic-range CMOS vision sensor with programmable image processing and global feature extraction," *IEEE J. Solid-State Circuits*, vol. 42, no. 3, pp. 647–657, 2007.

[88] Tabet, M. and Hornsey, R., "Cmos image sensor camera with focal plane edge detection," *Can. Conf. Electr. Comput. Eng.*, pp. 1129–1133, 2001.

[89] Mizuno, S., Fujita, K., Yamamoto, H., Mukozaka, N., and Toyoda, H., "A 256 × 256 compact CMOS image sensor with on-chip motion detection function," *IEEE J. Solid-State Circuits*, vol. 38, no. 6, pp. 1072–1075, 2003.

[90] Hong, C.S. and Hornsey, R., "On-Chip Binary Image Processing with CMOS Image Sensors," *Proc. SPIE 4669, Sensors Camera Syst. Sci. Ind. Digit. Photogr. Appl. III*, 2002.

[91] Basset, C.J.-M., "CMOS Imaging Technology with Embedded Early Image Processing," *These, Calif. Inst. Technol.*, 2007.

[92] Dupret, A., Klein, J.O., and Nshare, A., "A DSP-like analogue processing unit for smart image sensors," *Int. J. Circuit Theory Appl.*, vol. 30, no. 6, pp. 595–609, 2002.

[93] Angotzi, G.N. and Barbaro, M., "A CMOS imager for embedded systems with integrated, real-time, motion-detection capabilities and digital output," *IEEE SENSORS*, pp. 274–277, 2008.

[94] Elouardi, A., Bouaziz, S., Dupret, A., Lacassagne, L., Klein, J.O., and Reynaud, "A smart architecture for low-level image computing," *J. Comput. Sci.*, vol. 5, no. 3, pp. 1–19, 2008.

[95] Cho, J., Park, S., Choi, J., and Yoon, E., "Area-efficient and low-power implementation of vision chips using multi-level mixed-mode processing," *Int. Work. Cell. Nanoscale Networks their Appl.*, pp. 1–2, 2014.

[96] Zhao, B., Zhang, X., and Chen, S., "A 64 × 64 CMOS Image Sensor With On-Chip Moving Object Detection and Localization," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 4, pp. 581–588, 2012.

[97] Lee, C., Chao, W., Lee, S., Hone, J., Molnar, A., and Hong, S., "A Low Power Edge Detection Image Sensor Based on Parallel Digital Pulse Computation," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 1, no. 1, pp. 1–1, 2015.

[98] Benetti, M., Gottardi, M., and Smilansky, Z., "A 80µW 30fps 104x104 all-nMOS pixels CMOS imager with 7-bit PWM ADC for robust detection of relative intensity change," *Eur. Solid-State Circuits Conf.*, no. 1, pp. 303–306, 2013.

[99] Gottardi, M., Massari, N., and Jawed, S.A., "A 100µW 128x64 Pixels Contrast-Based Asynchronous Binary Vision Sensor for Sensor Networks Applications," *IEEE J. Solid-State Circuits*, vol. 44, no. 5, pp. 1582–1592, 2009.

[100] Cottini, N., Gasparini, L., De Nicola, M., Massari, N., and Gottardi, M., "A CMOS ultra-low power vision sensor with image compression and embedded event-driven energy-management," *IEEE J. Emerg. Sel. Top. Circuits Syst.*, vol. 1, no. 3, pp. 299–307, 2011.

[101] Sicard, G., Abbas, H., Chefi, A., and Amhaz, H., "Simple intra-pixel interaction for smart CMOS image sensors," *Int. Work. Cell. Nanoscale Networks their Appl.*, pp. 4–5, 2014.

[102] Liu, X., Zhang, M., and Van Der Spiegel, J., "A low-power multifunctional cmos sensor node for an electronic facade," *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 61, no. 9, pp. 2550–2559, 2014.

[103] Choi, J., Park, S., Cho, J., and Yoon, E., "A 3.4-µW object-adaptive cmos image sensor with embedded feature extraction algorithm for motion-triggered object-of-interest imaging," *IEEE J. Solid-State Circuits*, vol. 49, no. 1, pp. 289–300, 2014.

[104] Suarez, M., Brea, V.M., Fernandez-Berni, J., Carmona-Galan, R., Cabello, D., and Rodriguez-Vazquez, A., "Low-Power CMOS Vision Sensor for Gaussian Pyramid Extraction," *IEEE J. Solid-State Circuits*, vol. 52, no. 2, pp. 483–495, 2017.

[105] Martel, J.N.P., Chau, M., Cook, M., and Dudek, P., "Pixel interlacing to trade off the resolution of a cellular processor array against more registers," *2015 Eur. Conf. Circuit Theory Des. ECCTD 2015*, pp. 1–4, 2015.

[106] Schmitz, J.A., Gharzai, M.K., Balkır, S., Hoffman, M.W., White, D.J., and Schemm, N., "A 1000 frames / s Vision Chip Using Scalable Pixel-Neighborhood-Level Parallel Processing," *IEEE J. Solid-State Circuits*, vol. 52, no. 2, pp. 556–568, 2017.

[107] Brunetti, A., Buongiorno, D., Trotta, G.F., and Bevilacqua, V., "Computer vision and deep learning techniques for pedestrian detection and tracking: A survey," *Neurocomputing*, vol. 300, pp. 17–33, 2018.

[108] Maggiani, L., Bourrasset, C., Petracca, M., Berry, F., Pagano, P., and Salvadori, C., "HOG-Dot: A Parallel Kernel-Based Gradient Extraction for Embedded Image Processing," *IEEE Signal Process. Lett.*, vol. 22, no. 11, pp. 2132–2136, 2015.

[109] Dalal, N., "INRIA Person Dataset," *http://pascal.inrialpes.fr/data/human/*.

[110] Sadeghi, N., Gaudet, V., and Schlegel, C., "Analog DFT processors for OFDM receivers: Circuit mismatch and system performance analysis," *IEEE Trans. Circuits Syst. I Regul. Pap.*, vol. 56, no. 9, pp. 2123–2131, 2009.

[111] Chaoui, H., "CMOS analogue adder," *Electron. Lett.*, vol. 31, no. 3, pp. 180–181, 1995.

[112] Gilbert, B., "A Precise Four-Quadrant Multiplier with Subnanosecond Response," *IEEE J. Solid-State Circuits*, vol. 3, no. 4, pp. 365–373, 1968.

[113] Dudek, P. and Hicks, P.J., "A CMOS general-purpose sampled-data analog processing element," *IEEE Trans. Circuits Syst. II Analog Digit. Signal Process.*, vol. 47, no. 5, pp. 467–473, 2000.

[114] Lam, K.K.K. and Copeland, M.A., "Noise-cancelling switched- capacitor (sc) filtering technique," *Electron. Lett.*, vol. 19, no. 20, pp. 19–20, 1983.

[115] McCreary, J.L. and Gray, P.R., "All-MOS Charge Redistribution Analog-to-Digital Conversion Techniques—I," *IEEE J. Solid-State Circuits*, vol. SC-10, no. 6, 1975.

[116] Nagaraj, K., Vlach, J., Viswanathan, T.R., and Singhal, K., "SWITCHED-CAPACITOR INTEGRATOR WITH REDUCED SENSITIVITY TO AMPLIFIER GAIN," *Electron. Lett.*, pp. 1103–1105, 1986.

[117] Nauta, B., *Analog Cmos Filters for Very High Frequencies*. 1993.

[118] Razavi, B., "The Switched-Capacitor Integrator [a Circuit for All Seasons]," *IEEE Solid-State Circuits Mag.*, vol. 9, no. 1, pp. 9–11, 2017.

[119] Chae, Y. and Han, G., "Low Voltage, Low Power, Inverter-Based Switched-Capacitor Delta-Sigma Modulator," *IEEE J. Solid-State Circuits*, vol. 44, no. 2, pp. 458–472, 2009.

[120] Razavi, B., "Introduction to Switched-Capacitor Circuits," *Des. Analog C. Integr. Circuits*, vol. 1, 2000.

[121] Massari, N., Syed, A.J., and Gottardi, M., "A High Dynamic Range Time-Based Edge Detection Algorithm for a Vision Sensor," *ICECS*, pp. 1063–1066, 2007.

Annexe 1

Cette annexe présente une étude complémentaire d'adéquation entre algorithme de différence temporelle et architecture par macropixels.

1 Différence « flou-net »

Une autre solution au problème de différence temporelle en ressources limitées serait de moyenner la réponse de plusieurs pixels voisins sur une seule mémoire, puis de faire la différence entre chaque pixel de la seconde image et la moyenne mémorisée correspondante (voir). On nomme ce principe « flou-net ». Les résultats sont présentés en . En comparant à la différence temporelle complète (chapitre 3), on voit que des artefacts apparaissent : du mouvement est détecté là où il n'y en avait pas, comme dans l'herbe. Cela s'explique par le fait que l'on a parfois moyenné des groupes de pixels qui présentaient un contraste fort entre eux, le pixel suivant apparait donc différent de la moyenne mémorisée alors qu'il était identique à son homologue précédent. Aucun seuil simple, commun à toute l'image, ne permet de retrouver tout le mouvement sans aucun bruit. La présente les résultats pour un moyennage sur des carrés de 2x2 pixels. Les résultats pour un moyennage sur 3x3 pixels sont bien pires, et sur des rectangles de 2x1 ou 1x2 guère mieux.

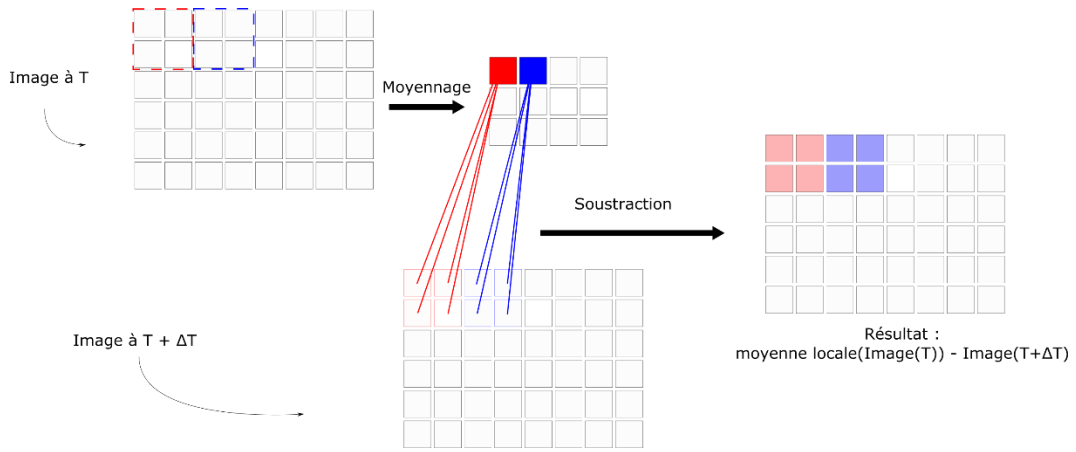


Figure 145 : Principe de la différence temporelle flou - net : une première image est moyennée, puis la seconde est prise en plein résolution et on en soustrait chaque pixel à la moyenne mémorisée correspondante.

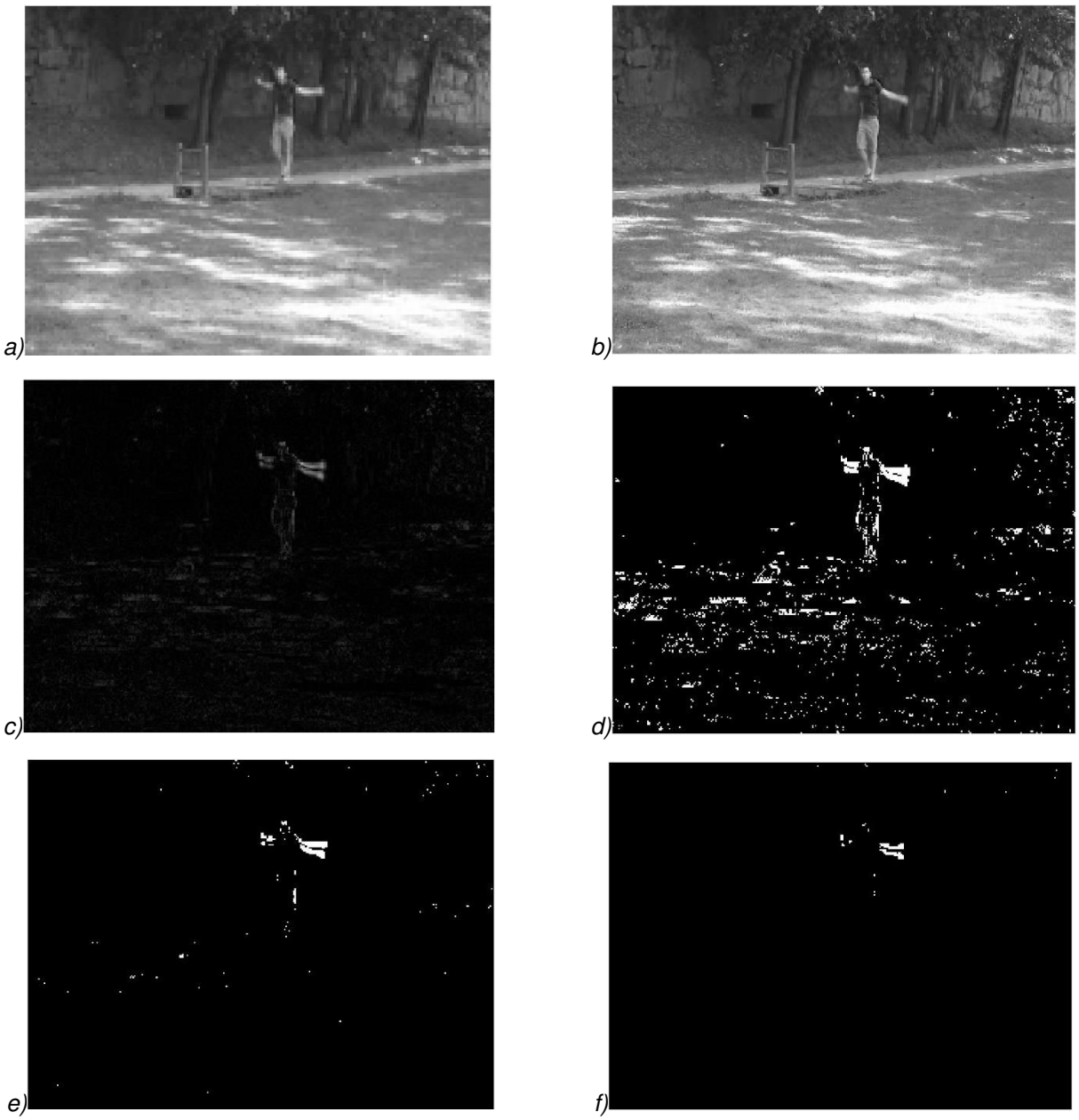


Figure 146 : (a) Première image moyennée sur 2x2 pixels (donc 120x160 pixels) (b) deuxième image (240x320 pixels) (c) différence a-b (d) a-b seuillée à 0.1 comme 1-d) (e) a-b seuillée à 0.2 (f) a-b seuillée à 0.3 (g) a-b seuillée à 0.4. Les images résultats ont autant de pixels que la vidéo initiale : 240x320.

2 Différence « flou-net » corrigée par morphologie

Pour éliminer le bruit causé par le moyennage, on peut penser à des méthodes simples sur images binaires comme la morphologie [7 - Bovik 2009]. Elle pourrait être appliquée hors de la matrice, l'idée ici est de voir si on peut récupérer l'information non bruitée. On a essayé une morphologie fermée puis ouverte et l'inverse sur différentes images flou-net avec différents seuils (comme en). La présente les résultats : on ne retrouve pas du tout les mouvements de la différence temporelle idéale, il y a trop de perte d'information.

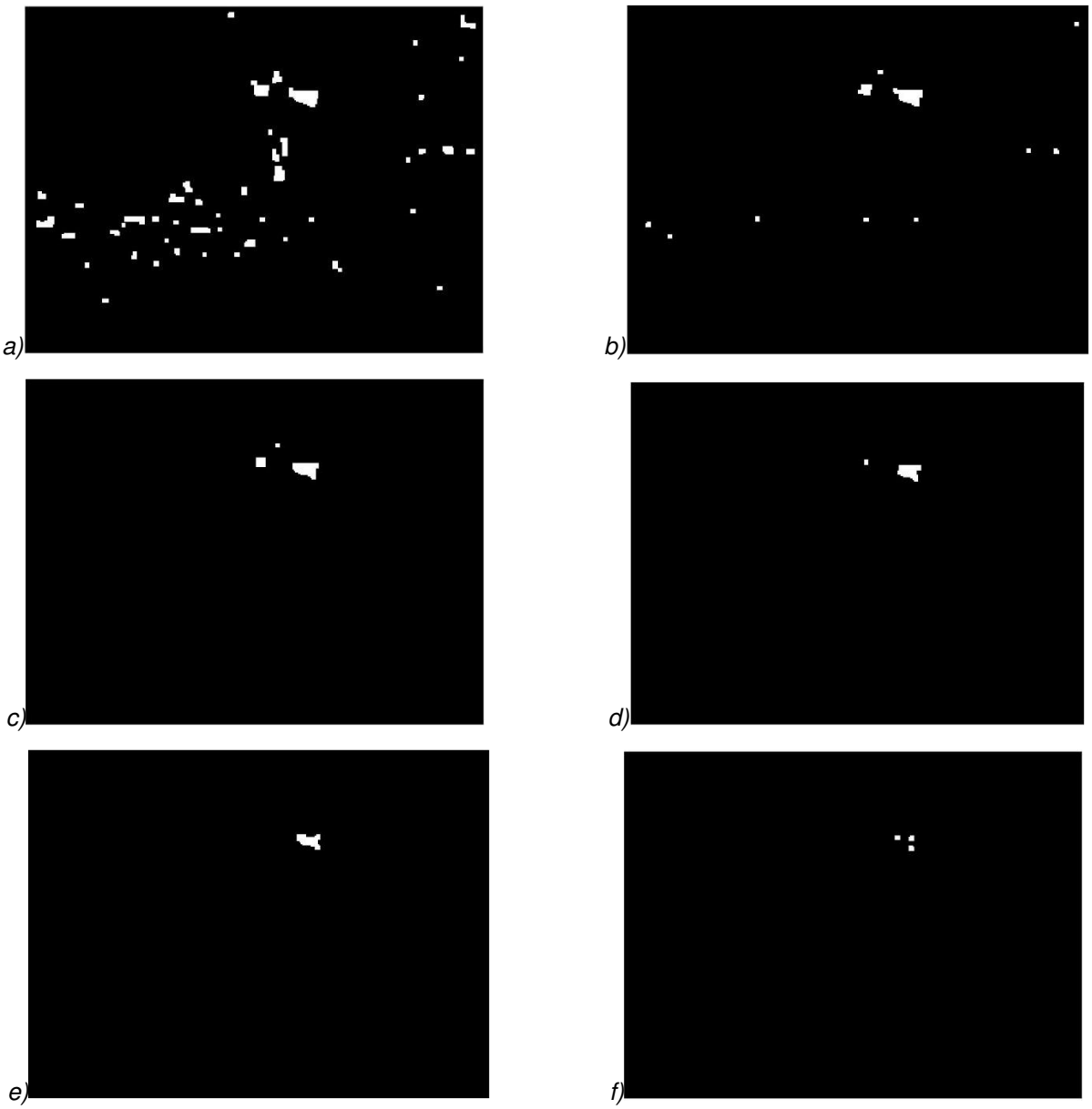


Figure 147 : Dans le même contexte que la , on fait suivre le seuillage par une morphologie : (a) seuil 0.1 et fermée-ouverte (b) seuil 0.1 et ouverte-fermée (c) seuil 0.2 et fermée-ouverte (d) seuil 0.2 et ouverte-fermée (e) seuil 0.3 et fermée-ouverte (f) seuil 0.3 et ouverte-fermée.

3 Différence « flou-net » corrigée par soustraction de contours

Une autre solution pour éliminer ce bruit dû à la différence flou-net serait de soustraire les contours de l'image puisque ce sont eux qui provoquent ce bruit. On peut utiliser les contours de la première image ou de la deuxième, puisque ceux que l'on veut enlever n'ont pas bougé. Ces contours peuvent être disponibles localement grâce au filtrage spatial. La présente plusieurs cas : des différences flou-net suivies de soustractions de contours complets ou en CSE, seuillées ou non. Lorsque le contours est obtenu en CSE, la valeur d'un point de l'image des contours est donnée à tous ses pixels voisins pour recréer une image de contours en pleine résolution.

Les résultats sont très loin de la différence temporelle idéale. Des traitements plus compliqués pourraient certainement effacer ce bruit sans perdre l'information, mais le but de notre système n'est pas de fournir un résultat qui doit être longuement corrigé par la suite.

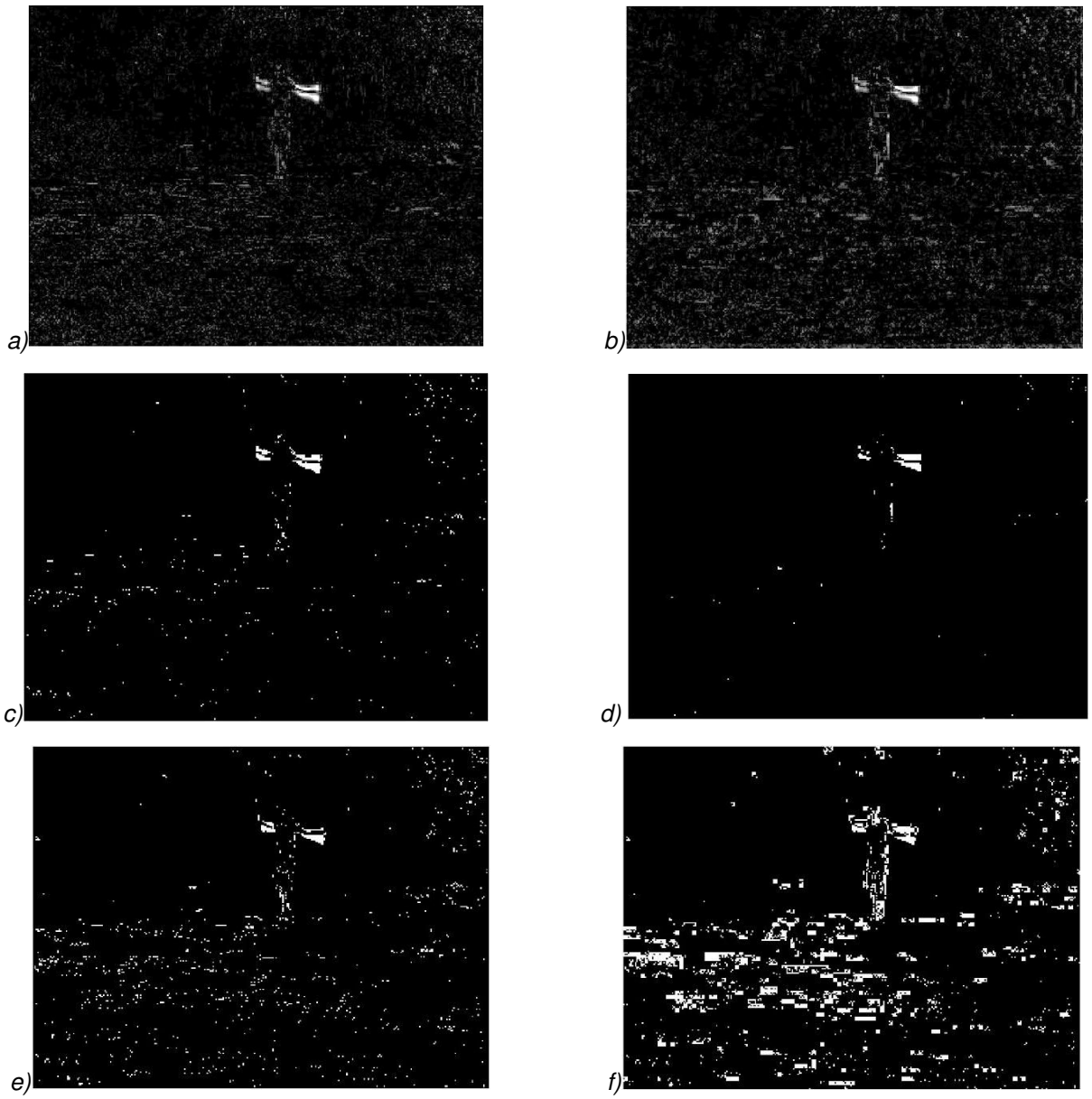


Figure 148 : Différences (a) flou2x2 - net - contours obtenus par Sobel (b) flou2x2 - net - contours obtenus par Sobel en CSE (c) flou 2x2 - net - contours obtenus par Sobel, seuillée à 0.08 (d) flou 2x2 - net - contours obtenus par Sobel CSE, seuillée à 0. 15 (e) (flou 2x2 - net) seuillée à 0.1 - contours obtenus par Sobel seuillés à 0.1 (f) idem e avec contours obtenus par Sobel CSE. Toutes ces images ont bien 240x320 pixels.

Annexe 2

Cette annexe détaille le calcul pour l'obtention de l'équation 4.7 du chapitre 4.

Notons *A* le gain réel de l'inverseur (A<0, typiquement entre -500 et -100 à son maximum) et modifions les équations de fonctionnement du circuit à capacités commutées. On note V_{ccLeft} le potentiel à gauche de C_C et V_{OFF} le point de repos de l'inverseur ; le reste des notations est identiques aux équations précédentes.

Pendant la première phase φ_1 :

$$Q_{in1} = C_{in}\left(-V_{in1} + V_{ref}\right) \qquad \text{et} \qquad Q_{out1} = 0$$

Pendant la phase $\varphi 4/\varphi 2$:

$$V_{CcLeft1bis} = V_{ref} + \frac{V_{out1bis} - V_{OFF}}{A}$$

$$Q_{in1bis} = C_{in}\left(V_{CcLeft1bis} - V_{ref}\right) = + C_{in}\,\frac{V_{out1bis} - V_{OFF}}{A}$$

$$Q_{out1bis} = Q_{out1} - \left(Q_{in1} - Q_{in1bis}\right)$$

Donc $C_{out}\left(V_{out1bis} - V_{CcLeft1bis}\right) = Q_{out1bis} = C_{in}\left(V_{in1} - V_{ref}\right) + C_{in}\,\frac{V_{out1bis} - V_{offset}}{A}$

$$C_{out}\left(V_{out1bis} - V_{ref} - \frac{V_{out1bis} - V_{OFF}}{A}\right) = C_{in}\left(V_{in1} - V_{ref}\right) + C_{in}\,\frac{V_{out1bis} - V_{OFF}}{A}$$

$$V_{out1bis}\left(1 - \frac{1}{A} - \frac{C_{in}}{AC_{out}}\right) = \frac{C_{in}}{C_{out}}\left(V_{in1} - V_{ref} - \frac{V_{OFF}}{A}\right) + V_{ref} - \frac{V_{OFF}}{A}$$

Pendant la deuxième phase φ_1 :

$$Q_{in2} = C_{in}\left(-V_{in2} + V_{ref}\right) \qquad \text{et} \qquad Q_{out2} = Q_{out1bis}$$

Pendant la phase $\varphi 4/\varphi 2$ suivante :

$$V_{CcLeft2bis} = V_{ref} + \frac{V_{out2bis} - V_{OFF}}{A}$$

$$Q_{in2bis} = C_{in}\left(\left(V_{ref} + \frac{V_{out2bis} - V_{OFF}}{A}\right) - V_{ref}\right) = + C_{in}\,\frac{V_{out2bis} - V_{OFF}}{A}$$

$$Q_{out2bis} = Q_{out2} - \left(Q_{in2} - Q_{in2bis}\right)$$

$$C_{out}\left(V_{out2bis} - \left(\cancel{V_{ref}} + \frac{V_{out2bis} - V_{OFF}}{A}\right)\right) = C_{out}\left(V_{out1bis} - \left(\cancel{V_{ref}} + \frac{V_{out1bis} - V_{OFF}}{A}\right)\right) + C_{in}\left(V_{in2} - V_{ref}\right) + C_{in}\,\frac{V_{out2bis} - V_{OFF}}{A}$$

Donc

$$V_{out2bis}\left(1 - \frac{1}{A} - \frac{C_{in}}{AC_{out}}\right) = V_{out1bis}\left(1 - \frac{1}{A}\right) + \frac{C_{in}}{C_{out}}\left(V_{in2} - V_{ref} - \frac{V_{OFF}}{A}\right)$$

D'où pour N>1 :

$$V_{out1bis}\left(1-\frac{1}{A}-\frac{C_{in}}{AC_{out}}\right)=\frac{C_{in}}{C_{out}}\left(V_{in1}-V_{ref}-\frac{V_{OFF}}{A}\right)+V_{ref}-\frac{V_{OFF}}{A}$$

$$V_{out(N)bis}\left(1-\frac{1}{A}-\frac{C_{in}}{AC_{out}}\right)=V_{out(N-1)bis}\left(1-\frac{1}{A}\right)+\frac{C_{in}}{C_{out}}\left(V_{in(N)}-V_{ref}-\frac{V_{OFF}}{A}\right)$$

Ou encore :

$$V_{out(N)bis}=\frac{V_{out(N-1)bis}\left(1-\frac{1}{A}\right)+\frac{C_{in}}{C_{out}}\left(V_{in(N)}-V_{ref}-\frac{V_{OFF}}{A}\right)}{\left(1-\frac{1}{A}-\frac{C_{in}}{AC_{out}}\right)}$$

Et par itération :

$$V_{out(N)bis}=\frac{\frac{V_{out(N-2)bis}\left(1-\frac{1}{A}\right)+\frac{C_{in}}{C_{out}}\left(V_{in(N-1)}-V_{ref}-\frac{V_{OFF}}{A}\right)}{\left(1-\frac{1}{A}-\frac{C_{in}}{AC_{out}}\right)}\left(1-\frac{1}{A}\right)+\frac{C_{in}}{C_{out}}\left(V_{in(N)}-V_{ref}-\frac{V_{OFF}}{A}\right)}{\left(1-\frac{1}{A}-\frac{C_{in}}{AC_{out}}\right)}$$

Pour N = 2 on a :

$$V_{out(2)bis}=\frac{\frac{C_{in}}{C_{out}}\left(V_{in(1)}-V_{ref}-\frac{V_{OFF}}{A}\right)}{\left(1-\frac{1}{A}-\frac{C_{in}}{AC_{out}}\right)^{2}}\left(1-\frac{1}{A}\right)+\frac{\frac{C_{in}}{C_{out}}\left(V_{in(2)}-V_{ref}-\frac{V_{OFF}}{A}\right)}{\left(1-\frac{1}{A}-\frac{C_{in}}{AC_{out}}\right)}+V_{ref}$$

Soit en générique :

$$V_{out(N)bis}=\frac{C_{in}}{C_{out}}\sum_{k=1}^{N}\frac{\left(V_{in(k)}-V_{ref}-\frac{V_{OFF}}{A}\right)}{\left(1-\frac{1}{A}-\frac{C_{in}}{AC_{out}}\right)^{N-k+1}}\left(1-\frac{1}{A}\right)^{N-k}+V_{ref}$$

Et avec le signe s_K de l'accumulation:

$$V_{out(N)bis}=\frac{C_{in}}{C_{out}}\sum_{k=1}^{N}s_k\frac{\left(V_{in(k)}-V_{ref}-\frac{V_{OFF}}{A}\right)}{\left(1-\frac{1}{A}-\frac{C_{in}}{AC_{out}}\right)^{N-k+1}}\left(1-\frac{1}{A}\right)^{N-k}+V_{ref} \qquad (4.7)$$

Annexe 3

Cette annexe décrit le contrôle numérique conçu à partir des spécifications.

1 Spécifications

Fonctions :

Configuration : écriture / lecture des registres
Prise d'image
Prise d'image à mémoriser
Prise d'image en HDR
Convolution avec le masque en mémoire + lecture des macropixels
Différence temporelle amplifiée avec le coefficient en mémoire + lecture des macropixels
Lecture des pixels en pleine résolution
Lecture des mémoires

Entrées du circuit numérique:

clock[1] : l'horloge,
reset[1] : la remise à zéro générale,
adress[4] : l'adresse du registre concerné pour une configuration,
rw[1] : lecture (1) ou écriture(0) du registre adressé pendant une configuration,
datain[1] : entrée série des données à charger dans les registres,
command[4] : choix de la fonction à réaliser,
vddD[1] et gndD[1] : les alimentations.

Sorties du bloc numérique à destination de l'analogique :

reset_PD, resetFD, resetFDmem, TX, TXmem : commandes du pixel,
sel_pix, sel_mem, sel_pixTD : sélection de pixel, de mémoire ou de pixel pour différence temporelle, dans un macropixel,
sel_row, sel_col : sélection de ligne ou colonne de macropixels,
phi1, phi3, phi4, reset_accu : commandes de l'accumulateur,
sel_read : sélection du type de lecture (pixels ou macropixels).

Sorties du bloc numérique à destination de l'utilisateur :

dataout [1] : sortie série des données lues dans un registre pendant une configuration,
ack_data, ack_image, ack_conv, ack_read[4] : signalement de fin de tâche,
flag_pixel, flag_row [2] : signalement de changement de pixel ou ligne, en lecture.

Entrée *command* :

0001 prise d'image
0010 Prise d'image à mémoriser
0011 Convolution avec le masque en mémoire + lecture macros
0100 Différence temporelle + lecture macros
0101 Lecture full résolution
0110 Prise d'image HDR
0111 Lecture des mémoires
1000 Film de convolutions
1001 Film de différences temporelles
1010 Film HDR

On veut un système de commandes qui soit pratique pour le prototypage. Les fonctions qu'il doit réaliser sont donc des fonctions élémentaires (prise d'image, convolution, etc.). Ces fonctions sont à combiner pour un cas réel : par exemple enchainer prise d'image et lecture en pleine résolution. Les fonctions « film » enchaînent automatiquement ces fonctions élémentaires pour trois applications particulières.

Le circuit doit être programmable : des registres contiennent les coefficients du masque de convolution spatiale, le coefficient de la différence temporelle amplifiée et les temps (d'intégration, de reset de la photodiode, etc.). Ces registres sont lus et/ou écrits pendant une configuration grâce aux entrées *adress*, *rw,* et *datain*. Lorsque le système est au repos, l'entrée *adress* est prioritaire : si elle est différente de 0000, une configuration doit commencer. Si l'adresse est nulle, alors la fonction déclenchée est définie par l'entrée *command*. Toute action élémentaire entamée est menée à terme même si les entrées changent.

2 Conception

Le système numérique proposé pour répondre à ces spécifications a en premier niveau une machine à états (fsm pour *finite state machine*) *generalcontrol*, et en second niveau 4 autres fsm : *datacontrol*, *imagecontrol*, *computationcontrol* et *readoutcontrol*.

La machine à états *generalcontrol* présentée en gère l'ensemble du système, ses états déclenchent les autres machines à états et ils attendent un *acknowledge* de la part de la fsm déclenchée pour revenir en état de repos *idle* : voir tableau ci-dessous.

La fonction de prise d'image HDR est intégrée à la structure générale, mais encore en étude.

Etat	Signal supplémentaire	Fsm déclenchée	Signal attendu
data	-	datacontrol (go_data)	ack_data
imaging	-	imagecontrol (go_im)	ack_im
imagingMem	mem_image		
imagingHDR	hdr_image		
conv	-	computationcontrol (go_conv)	ack_conv
td	tempdiff		
readingFull		readoutcontrol (go_read)	ack_read
readingMem			

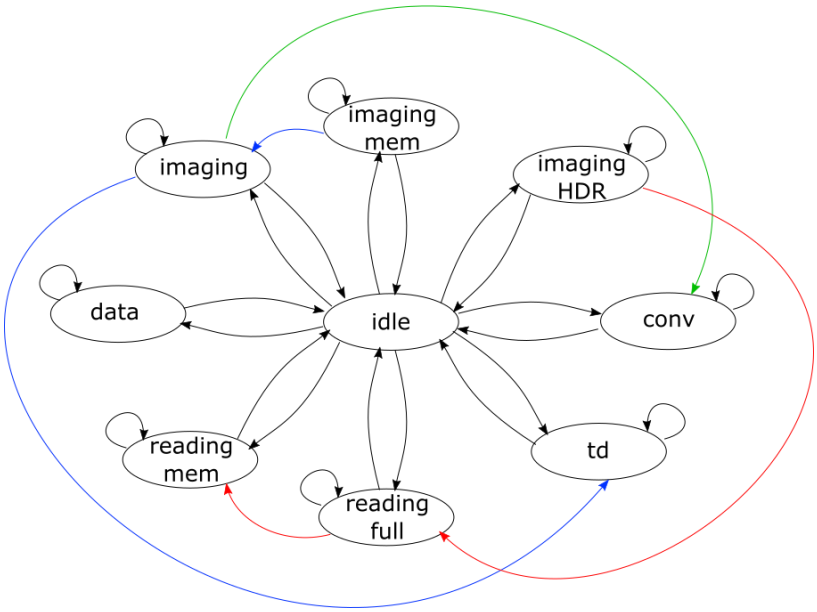


Figure 149 : Machine à états principale generalcontrol (vert : lien utilisés par la commande film de convolution, bleu : film de différence temporelle, rouge : film HDR).

2.1 Configuration

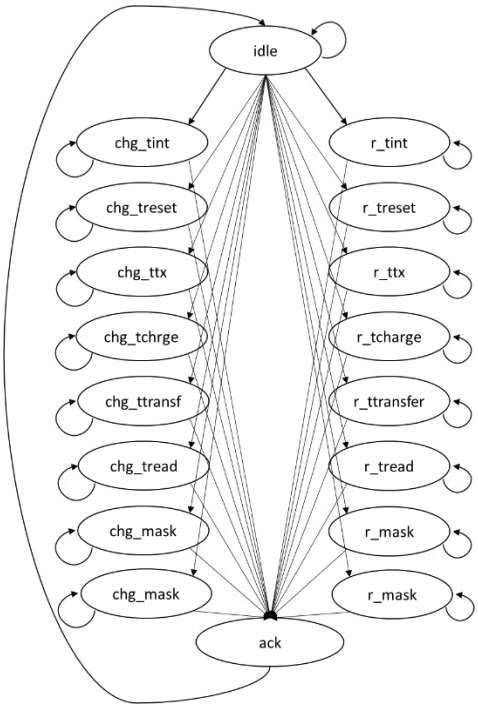


Figure 150 : Machine à états datacontrol.

La configuration doit permettre d'enregistrer le masque, le coefficient de différence temporelle et les temps d'intégration *tint*, de reset du pixel *treset*, de transfert sur la FD *ttx*, de transfert sur l'accumulateur *tcharge*, de transfert vers la sortie de l'accumulateur *ttransfer*, de reset de l'accumulateur *tresetaccu* et de lecture d'une sortie *tread*. Le module *datacontrol* est donc une machine à états (fsm) qui gère la charge ou la lecture de registres selon les entrées *datain, rw* et *adress*. Elle est déclenchée directement par l'extérieur par une adresse non nulle si *generalcontrol* était dans l'état *idle*. Elle bloque

alors *generalcontrol* dans un état *data* jusqu'à l'émission d'un signal *ack_data* à la fin de chaque charge ou lecture (configuration de chaque donnée (masque ou temps) individuellement) comme l'illustre la .

Les données à charger sont disponibles en série, pour limiter le nombre d'entrées/sorties, sur l'entrée *datain*. L'entrée *adress* permet d'identifier le registre, et l'entrée *rw* indique si l'on écrit dans les registres (0) ou si on les lit (1). En cas de lecture, la sortie *dataout* présente en série le masque ou le temps demandé par l'adresse. Une valeur de l'entrée *adress* identifie le registre concerné :

adress ="0001" : tint ;
adress ="0010" : treset ;
adress ="0011" : ttx ;
adress ="0100" : tcharge ;
adress ="0101" : ttransfer ;
adress ="0110" : tresetaccu ;
adress ="0111" : tread ;
adress ="1000" : mask ;
adress ="1001" : coeftd.

Les temps sont tous codés sur *nbittint* (10) bits. Une valeur 1 correspond à une période d'horloge, 20ns (50MHz) dans les simulations. Ils sont chargés dans des registres à décalages. Un compteur *cpt_times* permet d'attendre l'arrivée des 10bits.

Le masque est composé de 9 coefficients de *nbitcoef* (5) bits chacun (1 bit de signe et *nbitcoef*-1 bits de valeur). On reçoit le premier coefficient en premier, en commençant par son bit le plus fort. Par exemple un masque de Sobel = [-1 -2 -1 0 0 0 1 2 1] correspond à 1 puis 0 puis 001 10010 10001 00000 etc. Pour la charge du masque dans un tableau de 9 coefficients de 3 bits chacun, on utilise un compteur de bits et un compteur de coefficients.

Le coefficient de la différence temporelle (*coeftd*) est codé sur *nbitcoef* bits également. Il est chargé dans un simple registre à décalage. Pour limiter le nombre de compteurs, on réutilise *cpt_times* en l'arrêtant à *nbitcoef*. Comme ce compteur décrémente, on suppose que *nbittint* > *nbitcoef*.

Pour le prototype, afin d'éviter les problèmes éventuels de configuration, on fixe des valeurs à tous ces registres au moment du reset général :

tint = 15
treset = 2
ttx = 3
tcharge = 4
ttransfer = 4
tresetaccu = 8
tread = 2
coeff_td = 2
mask = sobel horizontal

La fsm *datacontrol* est entièrement programmable du point de vue des nombres de bits, sauf pour le nombre de coefficients du masque, fixé à 9.

2.2 Imagerie

Pour prendre une image, la machine à états principale (*generalcontrol*) déclenche via le signal *go_image* une autre machine à états (*imagecontrol*) qui gère la prise d'image (reset du pixel, intégration, tx) jusqu'à rendre la main avec le signal *ack_image*. Cette fsm gère la prise de l'image aussi bien pour une image stockée en mémoires (*reset_pd, txmem, resetFDmem*) ou en chaîne directe (*reset_pd, tx,*

resetFD), selon que la machine à état principale soit dans l'état *imagingTD* ou *imaging*, c'est-à-dire qu'elle envoie le signal *mem_image* ou non. Elle utilise donc les valeurs des registres *tint, treset* et *ttx*.

Comme l'illustre la , deux chemins dans la machine à états ont été définis pour gérer une mémorisation ou une image classique puisque les commandes du pixel sont distinctes à chaque étape (à cause du rajout du reset sur la FD du pixel). Après l'état *acknowledge*, on utilise un état supplémentaire *idlebis* pour laisser le temps à *generalcontrol* de reconnaître le signal de fin de tâche et d'enlever son signal *go_image*. Pour gérer une prise d'image HDR, il suffit de rajouter une troisième branche avec les états possédant les commandes voulues pour exploiter la capacité de mémoire en capacité de débordement. Celle-ci est en cours d'étude.

Chaque état de reset, d'intégration ou de tx active non seulement les signaux correspondants de commande du pixel, mais aussi un compteur dédié (via des signaux *go_cpt_tint*, *go_cpt_treset*,…). Il décompte le temps correspondant à l'état, depuis la valeur du registre (*treset, tint* ou *ttx*).

On peut remarquer qu'on ne reset la FD que pendant les premiers états d'une nouvelle prise d'image (reset du pixel et intégration). Cela permet bien de garder une valeur de pixels pour différents traitements (plusieurs convolutions successives par exemple, ce qui est en particulier utile pour un calcul de gradient complet, dans 2 directions).

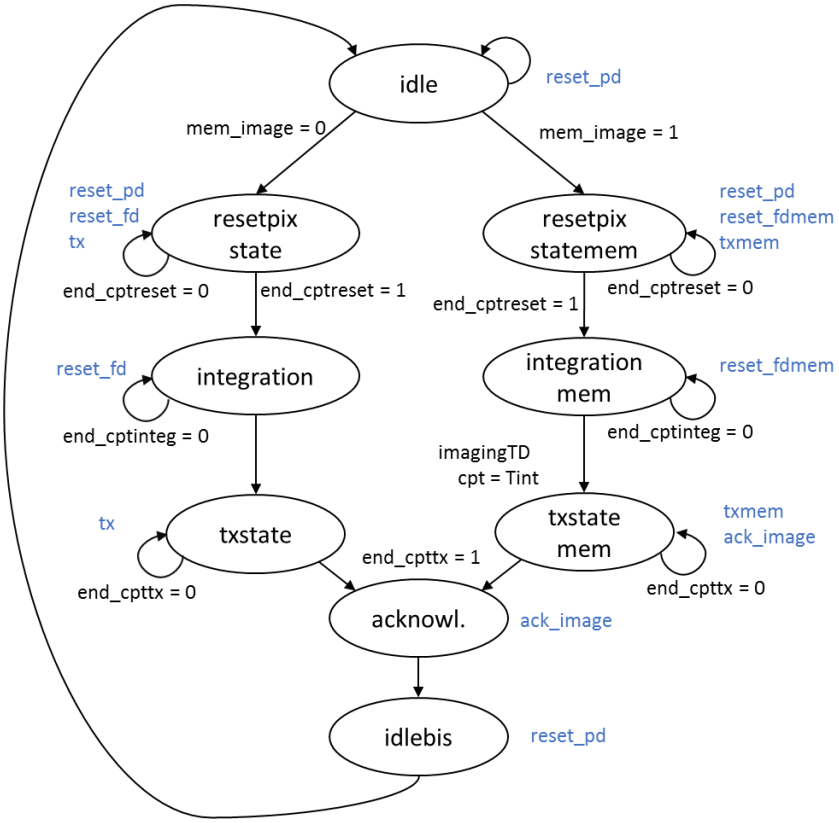


Figure 151 : Machine à états imagecontrol avec en bleu les signaux qu'elle commande et en noir les signaux à recevoir pour transiter). Les signaux go_cpt_XXX ne sont pas représentés pour la clarté, mais permettent l'avancement du compteur leur correspondant, et correspondant au type d'état (reset, integ ou tx). De plus quel que soit l'état, si go_image retombe à zéro, imagecontrol revient en idle.

2.3 Calcul

Les prétraitements d'image sont gérés par la fsm *computationcontrol*, déclenchée par *go_conv*. Le calcul est réalisé à partir de 3 états principaux : la charge de la valeur du pixel sur la capacité d'entrée de façon positive (*chargeplus*), la charge négative (*chargeminus*), et le transfert des charges vers la sortie pour l'accumulation (*transfer*) (voir).

Pour une convolution, on utilise le masque : la valeur absolue du coefficient d'un masque donne le nombre de charges/transferts nécessaires pour chaque pixel du groupe de 3x3 pixels. Après la

convolution, on lit la matrice de macropixels en appelant la fsm *readoutcontrol* (état *reading* de *computationcontrol*).

Pour une différence temporelle, c'est *coeftd* qui indique le nombre de *chargeplus* et de chargeminus à faire pour chaque pixel. On lit ensuite la matrice de macropixels (état *reading*). On fait cette séquence calcul/lecture 3 fois, i.e. jusqu'à ce que tous les pixels avec mémoire aient subi la différence temporelle.

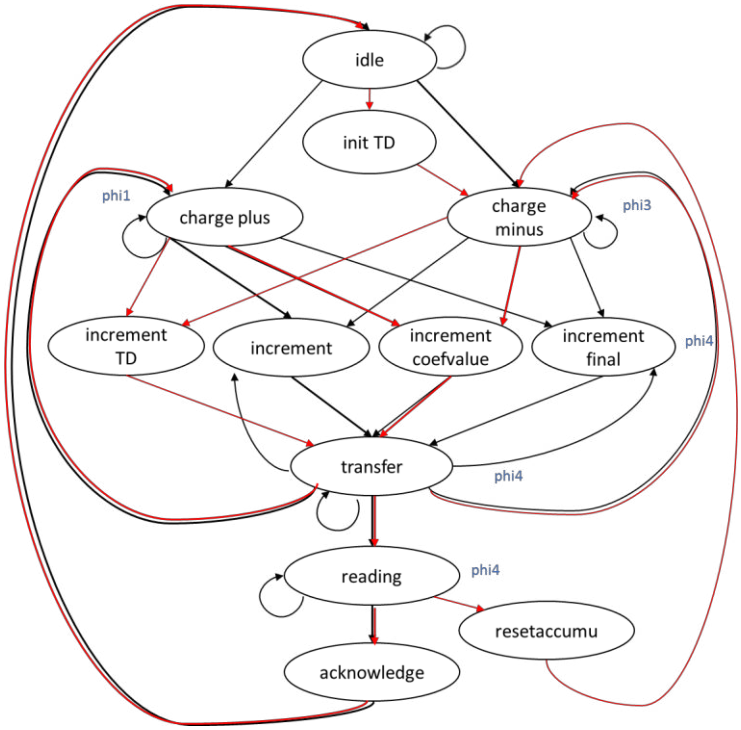


Figure 152 : Machine à états computationcontrol (avec en noir les chemins de calculs de convolution et en rouge les chemins de différence temporelle) (seuls les signaux de commandes de l'accumulateur sont représentés (en bleu) pour la clarté du schéma).

Dans le détail, de l'état de repos, on va vers une charge si la fsm générale déclenche *go_conv*.

- Convolution : après une charge, on incrémente les compteurs : soit le compteur *cpt_conv* de coefficients du masque (*increment*), soit le compteur de la valeur du coefficient du masque (*incrementcoefvalue*). L'état *incrementfinal* permet de ne pas incrémenter le compteur de coefficients avant la phase de transfert. Le choix entre *chargeminus* et *chargeplus* se fait selon le signe du coefficient en cours lors d'une convolution.

- Différence temporelle : après initialisation des compteurs (par défaut bons pour la convolution) avec la bonne valeur (*initTD*), on fait autant de chargeminus/transfert que *coeftd*, puis de même avec chargeplus/transfert, on lit la matrice, on reset l'accumulateur et on recommence (3 fois en tout). Pour cela, on incrémente un compteur jusqu'à *coeftd* grâce à l'état *incrementcoefvalue*. L'état *incrementTD* sert alors à incrémenter le compteur *cpt_conv* lorsque l'on a atteint *coeftd* sans remettre à zéro le compteur de valeur de coefficients, qui atteindra plus tard 2x*coeftd*. Le choix entre *chargeminus* et *chargeplus* se fait donc selon la parité de *cpt_conv*, tout comme le choix entre la sélection du pixel ou de la mémoire.

Deux compteurs permettent de décompter les temps de charge et de transfert issus des registres *tcharge* et *ttransfer*.

Les pixels possédant une mémoire sont différents selon le macropixel (Top/Bottom + Left/Right), la commande de sélection de pixel pour une différence temporelle se fait donc sur trois autres signaux. La sélection des mémoires se fait également sur trois autres signaux. On a donc un bus de 15 signaux: 9 de sélection de pixel dans un macropixel pour une convolution (les *sel_pix*) et 2x3 de

sélection de pixel avec leur mémoire associée pour un calcul de différence temporelle (les *sel_pixTD* et *sel_mem*).

L'état *reading* permet de conserver φ_4 (et donc une sortie valide de l'accumulateur) pendant la lecture, et de déclencher celle-ci par le signal *go_read* envoyé à la fsm *readoutcontrol*. Une fois la lecture terminée, le signal *ack_read* envoyé par la fsm *readoutcontrol* permet de déclencher *ack_conv* et de remettre la fsm *computationcontrol* en état *idle*.

Il convient de remarquer que dans le design actuel, cette machine à états prend un peu de temps sur les coefficients nuls dans une convolution spatiale. En effet, pendant l'état *transfer*, on teste la valeur du coefficient suivant et s'il est nul on retourne directement en état *increment* et donc ensuite en *transfer.* Un coefficient nul prend donc le temps d'un état et d'un transfert supplémentaires. La machine à états pourrait être optimisée, mais au vu du temps global d'une convolution très court (2µs pour un masque de type Sobel, soit 8 accumulations et 3 coefficients nuls), on se contente de cette version pour faire une preuve de concept sans alourdir le contrôle numérique.

2.4 Lecture

Tous les types de lectures sont gérés par la fsm *readoutcontrol*. Elle peut être déclenchée directement par *generalcontrol* pour une lecture d'image de résolution complète ou une lecture des mémoires, ou déclenchée par *computationcontrol* pour une lecture des sorties de macropixels. Cette dernière revient à une lecture classique puisqu'il y a un bus de sortie (« bus colonne ») par colonne de macropixels. La fsm *computationcontrol* lance le *go_read*, ce qui déclenche la fsm *readoutcontrol*. Celle-ci sélectionne une ligne de macropixels et toutes les colonnes une à une avant de passer à la ligne suivante (). Toutes les valeurs de macropixels seront donc disponibles en série en sortie de matrice. Le temps de lecture de chaque sortie de macropixel est défini par le registre *tread*, un compteur part donc de cette valeur pour gérer le temps à rester dans l'état *read* (voir).

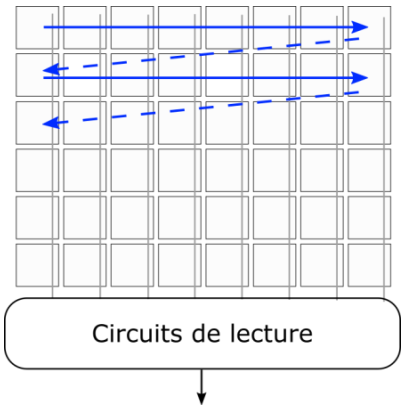


Figure 153 : Lecture classique : ligne par ligne, et dans chaque ligne, colonne par colonne.

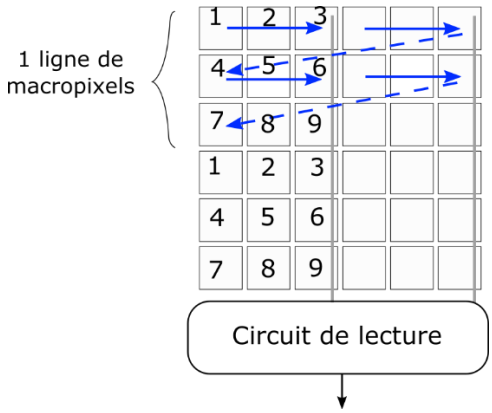


Figure 154 : Lecture de pixels avec des bus colonnes par macropixels (les numéros ne désignent que le nom des pixels).

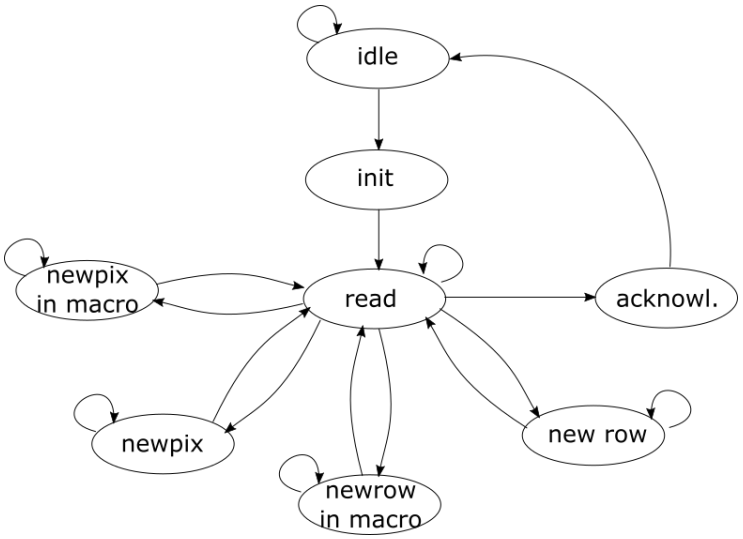


Figure 155 : Machine à états readoutcontrol. Les états newrowinmacrorow et newpixinmacropix ne sont utilisés que pour les lectures en résolution complète.

Pour lire la matrice complète en pleine résolution, c'est-à-dire les valeurs de sortie des pixels, il est possible de ne faire qu'une accumulation à la fois dans un PE. Cela donne en sortie une valeur correspondante au pixel sélectionné (masque de type [1 0 0 ; 0 0 0 ; 0 0 0]) :

$$V_{out_{accu}} = \frac{C_{in}}{C_{out}}\left(V_{pix1} - V_{ref}\right) + V_{ref}.$$

On peut ainsi appliquer un tel masque à toute la matrice, lire en sortie de macropixels les valeurs des pixels 1 dans chaque macropixel, charger un autre masque (avec une autre place pour le coefficient 1) et répéter cela 9 fois (9 pixels par macropixels). La sortie série obtenue est ensuite à recomposer dans l'ordre des pixels.

Cette technique n'ajoute aucun circuit dans la matrice mais force à charger 9 masques et à recomposer l'ordre des valeurs de la sortie. Elle peut être intéressante pour une évolution du système. Ici, pour plus de facilité d'utilisation, on rajoute plutôt un interrupteur dans la matrice pour pouvoir sélectionner, en lecture, les sorties de pixels à la place des sorties de macropixels. Cela permet également de pouvoir lire la sortie des pixels sans passage par l'accumulateur, ce qui est pratique pour tester les différentes parties du prototype.

Une lecture est donc faite ici en suivant le schéma de la . Pour chaque ligne de macropixels, on sélectionne une à une les colonnes de macropixels. Pour chaque colonne de macropixels, on sélectionne les pixels 1 puis 2 puis 3. Puis à nouveau pour chaque colonne de macropixels, on sélectionne les pixels 4 puis 5 puis 6 ; et une dernière fois pour chaque colonne de macropixels, on sélectionne les pixels 7 puis 8 puis 9. On peut alors passer à la ligne de macropixels suivante. Le chronogramme correspondant est présenté dans le chapitre 5. La machine à état de lecture classique est utilisée, mais quand on est sur un macropixel on fait trois lectures, et on fait trois fois chaque ligne. Dans la fsm, les états *new pix in macro* et *newrow in macro row* servent respectivement à changer de pixel dans un macropixel (passer du n°2 au n°3 par exemple) et de ligne dans un macropixel (passer d'une ligne de pixels 1-2-3 à une ligne de pixels 4-5-6 par exemple).

Pour une éventuelle imagerie HDR ou simplement pour tester le prototype, il est possible de lire les valeurs des mémoires. Pour cela, l'équivalent d'une lecture en pleine résolution est effectué, avec une sélection des mémoires au moment adéquat à la place des sélections de pixels. Cela prend plus de temps que nécessaire pour ne lire que les mémoires, mais n'alourdit pas la machine à états.

Il y a donc un bus de sélection de lignes de macropixels (*sel_row*), un bus de sélection de colonne de macropixels (*sel_col*) et un bus de sélection de pixels dans un macropixel pour une lecture directe. Ce bus *sel_pix* est associé aux 9 premières composantes du bus de sélection de pixel pour un calcul dans l'accumulateur par un OU logique dans *generalcontrol*. La sélection de colonne se fait en bas de chaque colonne, à l'extérieur de la matrice, tandis que la sélection de ligne de macropixels se

fait au niveau de chaque macropixel. Pour chaque ligne de macropixel, on associe le *sel_row* à *sel_read*, un signal qui détermine si l'on lit une sortie d'accumulateur ou de pixel.

Enfin, pour signaler à l'utilisateur le passage, pendant une lecture, à un nouveau pixel ou à une nouvelle ligne, des signaux *flag_pix* et *flag_row* sont utilisés.

Liste des publications liées à ce travail (décembre 2018)

Conférences :

J. Le Hir, A. Kolar and F. Vinci dos Santos, "*Distributed mixed-signal architecture for programmable smart image sensors*", IEEE Int. New Circuits and Systems Conf. (NEWCAS), 2017. DOI : 10.1109/NEWCAS.2017.8010178

Best student paper *award* (*1st place*)

J. Le Hir, F. Vinci dos Santos and A. Kolar, "*An accuracy-area tradeoff for mixed-signal computation in programmable smart image sensors*", IEEE Int. New Circuits and Systems Conf. (NEWCAS), 2018.

Journal :

J. Le Hir, A. Kolar and F. Vinci dos Santos, "*Distributed mixed-signal architecture for programmable smart image sensors*", Analog Integrated Circuits and Signal Processing (2018) vol 97 pp 493-501. DOI : 10.1007/s10470-018-1342-y

Liste des abbréviations utilisées

AAA : adéquation algorithme-architecture ;

ADC : *analog-to-digital converter*, convertisseur analogique numérique ;

AER : *address event representation*, représentation par adresse d'événement ;

APS : *active pixel sensor*, pixel actif ;

BL : *bottom left*, macropixel en bas à gauche du motif de 6x6 pixels ;

BR : *bottom right*, macropixel en bas à droite du motif de 6x6 pixels ;

BSI : *back-side illuminated*, circuit illuminé par l'arrière ;

CDS : *correlated double sampling,* double échantillonnage corrélé ;

CMOS : *complementary metal-oxide-semiconductor ;*

CSE : convolution sous-échantillonnée ;

FD : *floating diffusion*, diffusion flottante ;

FPGA : *field-programmable gate array,* matrice de portes reprogrammables ;

FPN : *fixed pattern noise*, bruit spatial fixe ;

FSM : *finite state machine*, machine à états ;

GPU : *graphics processing unit ;*

HDR : *high dynamic range*, grande dynamique ;

HOG : *histogram of oriented gradients*, histogramme de gradients orientés ;

PD : photodiode ;

PE : *processing element*, élément de calcul ;

PFM : *pulse frequency modulation*, modulation de fréquence d'impulsions ;

PPS : *passive pixel sensor*, pixel passif ;

PWM : *pulse width modulation*, modulation par largeur d'impulsion ;

SC : *switched-capacitor*, circuit à capacités commutées ;

SF : *source follower*, drain commun ;

SVM : *support vector machine*, machine à support de vecteurs ;

SPAD : *single photon avalanche diode*, diode à avalanche détectant des photons uniques ;

TL : *top left*, macropixel en haut à gauche du motif de 6x6 pixels ;

TR : *top right*, macropixel en haut à doite du motif de 6x6 pixels ;

TX : *transfer gate*, porte de transfert dans le pixel.

Liste des signaux de commande utilisés

Commandes du pixel

resetPD : remise à V_{DD} de la photodiode ;

resetFD : remise à V_{DD} du noeud FD ;

TX : transfert de charges de la photodiode vers le noeud FD.

Le suffixe "mem" représente les mêmes signaux, pour la chaîne qui contient la mémoire.

Le suffixe "nonov" indique que le signal est passé par un circuit d'antichevauchement.

Commandes de l'accumulateur

resetaccu : remise à zéro de l'accumulateur entre deux traitements ;

$\varphi_{1,...5}$: commandes des interrupteurs de l'accumulateur.

Le suffixe "nonov" indique que le signal est passé par un circuit d'antichevauchement.

Commandes de sélection :

Sel_lecture : sélectionne le type de lecture : des sorties de pixels ou des sorties de macropixels.

Sel_pix : sélection de pixels parmi les 9 du macropixel (1 bit / pixel : 9), pour convolution spatiale.

SelpixTD : sélection de pixels possédant une mémoire (1 bit / mémoire : 3) dans les macropixels, pour une différence temporelle. 3 mémoires maximums sont gérées par un macropixel.

Sel_mem : sélection de mémoires dans les macropixels (1 bit / mémoire : 3).

Sel_row : sélection de lignes de macropixels.

Sel_col : sélection de colonne de macropixels.

université PARIS-SACLAY

ÉCOLE DOCTORALE
Physique et ingénierie : électrons, photons, sciences du vivant (EOBE)

Titre : Conception mixte d'un capteur d'images intelligent intégré à traitements locaux massivement parallèles

Mots clés : microélectronique, conception mixte analogique/numérique, imageur intelligent, traitement d'image

Résumé : Les capteurs intelligents permettent aux systèmes embarqués d'analyser leur environnement sans transmission de données brutes, consommatrice d'énergie. Ce mémoire présente donc un travail sur un imageur intégrant du traitement d'image. Deux figures de mérite sont introduites pour classer l'état de l'art des imageurs intelligents en fonction de leur versatilité et de leur préservation de la surface photosensible. Cela met en évidence un compromis que ce travail essaie d'améliorer en explorant une approche par macropixels. En effet, en regroupant les éléments de calculs (PEs) pour plusieurs pixels, les traitements sont à la fois massivement parallèles et potentiellement plus versatiles à surface photosensible donnée. Une adaptation du filtrage spatial et du filtrage temporel en adéquation avec une architecture par macropixels est proposée (sous-échantillonnage par 3x3 pixels et par 2x2 pixels respectivement), et validée fonctionnellement. Une architecture d'imageur en macropixels asymétriques est donc présentée. Le PE conçu est un circuit analogique à capacités commutées, programmable par un contrôle numérique extérieur à la matrice. Son dimensionnement est discuté pour des compromis entre surface et précision des calculs, avant d'être implémenté en calcul approximé pour notre cas. La matrice proposée a été simulée en vue extraite et présente des images de résultats de détection de contours ou de différence temporelle corrects, avec un facteur de remplissage de 28%.

Title : Mixed co-design for an integrated smart image sensor with massively parallel local image processing

Keywords : microelectronics, mixed signal, smart image sensor, vision system, image processing

Abstract : Smart sensors allow embedded systems for analysing their environment without any transmission of raw data, which consumes a lot of power. This thesis presents an image sensor integrating image processing tasks. Two figures of merit are introduced in order to classify the state of the art of smart imagers regarding their versatility and their preservation of photosensitive area. This shows a trade-off that this work aims at improving by using a macropixel approach. By merging processing elements (PEs) between several pixels, processing tasks are both massively parallel and potentially more versatile at given photosensitive area. An adaptation of spatial and temporal filtering, matching such an architecture is proposed (downsampling by 3x3 and 2x2 pixels respectively for each processing task) and functionnally validated. An architecture of asymmetric macropixels is thus presented. The designed PE is an analog switched capacitor circuit that is controlled by out-of-matrix digital electronics. The sizing of the PE is discussed over the trade-off between accuracy and area, and implemented in an approximate computing approach in our study. The proposed matrix of pixels and PEs is simulated in post-layout extracted views and shows good results on computed images of edge detection or temporal difference, with a 28% fill factor.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France