RenPar'19 / SympA'13 / CFSE'7
Toulouse, France, du 09 au 11 septembre 2009

Unité flexible de calcul de disparité pour applications d'aide à la conduite

Mehdi Darouich[1], Stéphane Guyetant[1], Dominique Lavenier[2]

[1] CEA LIST
PC 94, F-91191 Gif-Sur-Yvette
{mehdi.darouich,stephane.guyetant}@cea.fr
[2] ENS Cachan Bretagne / IRISA
Campus de Beaulieu ; 35042 Rennes
dominique.lavenier@irisa.fr

Résumé
Cet article présente une unité flexible de calcul de scores d'appariement pour la génération de cartes de disparité dans des applications de stéréovision, et plus particulièrement dans le cadre d'applications d'aide avancée à la conduite. Cette unité est capable de générer des scores selon plusieurs méthodes et sur des fenêtres de voisinages de tailles et de formes différentes, tout en respectant les contraintes du temps-réel et de l'embarqué. Ainsi, notre solution matérielle calcule les scores d'appariement sur des voisinages 15x15 sur une paire d'images VGA, à une fréquence de 40 images par secondes, avec 64 niveaux de disparité et fonctionne à 200 MHz. Nous avons effectué une synthèse ASIC en technologie 65nm low power de l'unité flexible de calcul de disparité et estimé sa taille à environ 3 mm^2, vérifiant ainsi son caractère embarqué.

Mots-clés : architecture reconfigurable, stéréovision, disparité, flot de données

---

1. Introduction

Les applications d'aide avancée à la conduite (ou Advanced Driver Assistance Systems, ADAS) visent à apporter plus de sécurité et de confort aux utilisateurs de véhicules. Un nombre croissant de ces applications utilise une ou plusieurs caméras comme capteurs. L'utilisation de deux caméras, ou stéréovision, permet notamment d'extraire l'information de profondeur de la scène, ce qui peut profiter à un grand nombre d'applications comme la détection d'obstacles, la détection de piétons, la reconnaissance de panneaux signalétiques ou le suivi de voie de circulation. Plusieurs types d'informations peuvent ainsi être extraits de la carte de profondeur et être utilisés à plus haut niveau dans l'application. Par exemple, l'extraction préalable du plan de la route à partir de la carte de profondeur permet la détection des obstacles potentiels [8]. Cette information est aussi utilisée pour limiter le nombre de tests pour l'étape de classification d'une application de détection de piétons [11] [2]. Dans [10], l'information de profondeur est utilisée pour la détection et le suivi de la voie de circulation du véhicule.

Ces applications sont soumises à plusieurs contraintes. Du fait du contexte applicatif, les ADAS doivent répondre à de fortes contraintes de sûreté d'exécution et de temps-réel : les capteurs utilisés ont des résolutions et des fréquences image élevées pour garantir la précision et la réactivité du système. Le caractère embarqué implique, de son côté, des contraintes de consommation et d'encombrement (efficacité silicium). De plus, ce domaine applicatif couvrant un large spectre algorithmique, il faut que les architectures matérielles associées soient flexibles : plusieurs applications, nécessitant l'extraction d'informations différentes peuvent ainsi être portées sur le même circuit intégré, réduisant ainsi les coûts de fabrication.

L'unité de calcul que nous présentons dans cet article permet la génération de la carte de disparité qui est une étape clé de la stéréovision. En effet cette étape est très calculatoire et influe sur les résultats des autres blocs en aval de la chaîne de stéréovision. Afin de répondre aux contraintes précédemment citées

et de garantir un niveau de flexibilité suffisant, la majorité du calcul flot de données - le calcul des scores d'appariement - est effectuée au sein d'une matrice de calcul flexible reconfigurable statiquement.
Nous présentons dans la première partie de cet article la chaîne de calcul de disparité, les différents algorithmes associés et les solutions matérielles existantes. L'unité flexible de calcul de disparité est exposée en détail dans la seconde partie. Les perspectives d'intégration sont présentées en conclusion.

2. Algorithmes de disparité

La stéréovision nécessite l'utilisation de deux caméras pour extraire la profondeur d'une scène. Pour un point donné de la scène, la profondeur est liée au décalage entre les projections de ce point sur les deux images, ou disparité, comme le montre la figure 1. La profondeur P est ensuite extraite à partir des caractéristiques géométriques de la paire stéréoscopique, la base B et la focale f, et de la disparité D grâce à la formule $P = B \cdot f/D$. Pour une image de référence donnée, l'ensemble des disparités - appelé carte de disparité - est déterminé par appariement de chacun de ses pixels avec les pixels de la seconde image. Le calcul de la carte de disparité est le plus souvent précédé d'une étape de redressement de la paire stéréoscopique qui consiste à générer, par interpolation des pixels des images d'entrée, les images qui auraient été prises par deux capteurs d'axes optiques parallèles. Ce redressement apporte une simplification dans la recherche du pixel correspondant qui s'effectue sur la même ligne que le pixel de référence. La zone de recherche est bornée par une disparité maximale qui correspond à la distance minimale de détection et qui dépend de la géométrie de la paire stéréoscopique et de la résolution des images.

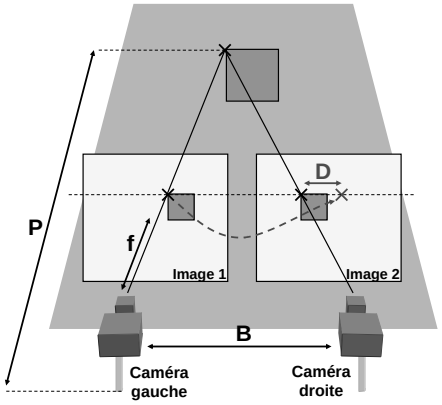


FIG. 1 – La disparité d'un point de l'image de référence correspond au décalage de ce point dans la seconde image. Elle est inversement proportionelle à la profondeur P.

La génération d'une carte de disparité se divise en deux étapes : le calcul de scores d'appariement et le choix de la paire de pixels donnant la bonne disparité. Le calcul des scores d'appariement s'effectue pour chaque paire de pixels analysée en prenant en compte leur voisinage. Ces scores sont représentatifs du degré de similarité entre les deux pixels analysés. Lors de la seconde étape, les scores d'appariement sont utilisés pour extraire la meilleure disparité pour chaque pixel de l'image de référence. Cette mise en correspondance peut s'effectuer selon des méthodes locales, semi-globales (sur les lignes par exemple) ou globales (sur toute l'image). Les méthodes locales sont les plus communément utilisées pour des applications embarquées. La plus simple consiste à choisir la disparité donnant le meilleur score, noté WTA (Winner Take All). Nous nous intéressons plus particulièrement dans cet article à la phase de génération des scores d'appariement, étape très calculatoire, mais régulière et hautement parallélisable.

La qualité des cartes générées peut être jugée selon deux critères principaux : le taux d'erreurs et la densité. La densité représente le taux de pixels dont la disparité est estimée valide. Le taux d'erreurs

représente le ratio de pixels de disparité erronée parmi les pixels de disparité estimée valide. Un compromis est à faire entre ces deux critères, le cas idéal étant une carte de disparité avec une forte densité et un taux d'erreur très faible. La valeur de ces deux critères dépendent de l'application et de l'utilisation de la carte de disparité en aval de la chaîne de traitement. Ainsi, certaines applications nécessitent une carte de densité la plus dense possible, quitte à ce que le taux d'erreur soit plus élevé [1].
La robustesse aux conditions applicatives est un autre critère de qualité. La variation des paramètres d'illumination entre les deux caméras de la paire stéréoscopique est une source d'erreur dans le calcul de disparité. Une des qualités d'un algorithme de disparité est l'insensibilité à ces variations qui peuvent être assimilées à une transformation affine des niveaux de gris X, tel que définie dans la formule $X' = G \cdot X + B$. Cette qualité dépend notamment de la méthode utilisée pour le calcul de score. Nous préciserons, par la suite, les méthodes qui sont insensibles à la variation du gain G et celles qui sont insensibles à la variation du biais B.

2.1. Calcul des scores d'appariement

Le calcul du score d'appariement d'une paire de pixel s'effectue selon plusieurs métriques de base qui fournissent des résultats différents selon la nature des paires d'images traitées. Elles sont de deux types : mesure de similarité (le score est d'autant plus grand que les fenêtres sont similaires) et de dissimilarité. Ces scores peuvent être calculés sur des fenêtres de voisinage de taille et de forme différentes. Ces paramètres influent à la fois sur la quantité de ressources nécessaires et sur la qualité du résultat. Le tableau 1 présente les principales métriques utilisées pour le calcul des scores d'appariement, et précise leur type et leur robustesse à la variation des paramètres d'illumination des deux caméras.

2.1.1. Métriques pour le calcul de scores

Les métriques les plus communément utilisées se basent sur l'intensité des pixels. La somme des écarts absolus, notée SAD (Sum of Absolute Differences) est la plus simple et de ce fait l'une des plus utilisée [13]. La somme des écarts au carré, notée SSD (Sum of Squared Differences), utilise un carré à la place de la valeur absolue ce qui a pour effet d'accroître les écarts mais aussi d'augmenter la dynamique des scores et la complexité des calculs. Ces deux métriques sont des métriques de dissimilarité et sont sensibles aux différences de gain et de biais entre les deux images. La mesure de corrélation croisée normalisée [12], notée NCC (Normalized Cross-Correlation), a pour avantage d'être insensible aux différences de gain. C'est une mesure de similarité. Elle implique cependant une complexité matérielle importante du fait de l'utilisation de la multiplication, de la racine carrée et de la division, et reste sensible aux différences de biais. Cette sensibilité peut être éliminée en utilisant des métriques centrées sur la moyenne des voisinages. Ainsi les mesures ZSAD (Z pour Zero-mean) et ZSSD sont insensibles aux différences de biais et la mesure ZNCC est insensible aux différences de gain et de biais. Le calcul de la moyenne sur le voisinage implique cependant un surcoût calculatoire non négligeable. Ainsi pour une ZSAD, environ 35% d'opérations supplémentaires sont nécessaires par rapport à la SAD.
Les métriques non-paramétriques ont pour particularité de ne pas se baser sur l'intensité des pixels pour évaluer le degré de similarité. De ce fait, elles sont insensibles aux différences de gain et de biais. La mesure basée sur le recensement [16] (Census) s'effectue en deux étapes. La première consiste à créer un vecteur de recensement dont les composantes sont le résultat de la comparaison entre chaque pixel du voisinage et le pixel central. La seconde étape consiste à générer le score d'appariement en calculant la distance de Hamming entre les vecteurs recensement de chaque fenêtre de voisinage. La mesure de rang [16] (Rank) consiste à filtrer l'image avec un filtre de Rank avant de générer le score à l'aide d'une métrique basée sur l'intensité, la SAD généralement. Ce filtre consiste, pour chaque pixel de l'image, à dénombrer les pixels du voisinage dont l'intensité est plus élevée que celle du pixel central. Ensuite, on utilise généralement la SAD pour calculer le score.
Aucune des métriques précédemment présentées ne se détache du lot comme étant meilleure que les autres pour le calcul de la carte de disparité. D'un point de vue calculatoire, certaines métriques se révèlent rédhibitoires si l'on veut faire du calcul embarqué et temps-réel. Ainsi nous éliminons les métriques (Z)SSD et (Z)NCC car elles utilisent des opérateurs trop complexes. Nous proposons la mise en place d'une unité de calcul flexible permettant de calculer des scores en utilisant les métriques de (Z)SAD, Census et Rank.

Mesure	C(x,y,d)	Type	G/B
SAD	$\sum_{(u,v)\in N(x,y)} \lvert I_1(u,v) - I_2(u+d,v)\rvert$	D	-
SSD	$\sum_{(u,v)\in N(x,y)} (I_1(u,v) - I_2(u+d,v))^2$	D	-
NCC	$\frac{\sum_{(u,v)\in N(x,y)} (I_1(u,v)\cdot I_2(u+d,v))}{\sqrt{\sum_{(u,v)\in N(x,y)} (I_1(u,v)^2)}\cdot\sqrt{\sum_{(u,v)\in N(x,y)} (I_2(u+d,v)^2)}}$	S	G
ZSAD	$\sum_{(u,v)\in N(x,y)} \lvert(I_1(u,v) - \bar{I_1}) - (I_2(u+d,v) - \bar{I_2}))\rvert$	D	B
ZSSD	$\sum_{(u,v)\in N(x,y)} ((I_1(u,v) - \bar{I_1}) - (I_2(u+d,v) - \bar{I_2}))^2$	D	B
ZNCC	$\frac{\sum_{(u,v)\in N(x,y)} ((I_1(u,v)-\bar{I_1})\cdot(I_2(u+d,v)-\bar{I_2}))}{\sqrt{\sum_{(u,v)\in N(x,y)} (I_1(u,v)-\bar{I_1})^2}\cdot\sqrt{\sum_{(u,v)\in N(x,y)} (I_2(u+d,v)-\bar{I_2})^2}}$	S	GB
Census	$\sum_{(u,v)\in N(x,y)} ((I_1(u,v) > Ic_1)XOR(I_2(u+d,v) > Ic_2))$	D	GB

TAB. 1 – Caractéristiques des métriques pour le calcul du coût C(x,y,d) entre le pixel de référence (x,y) et le pixel (x,y-d). Les mesures sont soit de similarité (S), soit de dissimilarité (D). La colonne G/B indique si la mesure est insensible aux différences de gain (G) et/ou de biais (B)

2.1.2. Taille et forme des fenêtres

Le voisinage pris en compte dans le calcul des scores d'appariement est généralement rectangulaire et de taille fixe. La taille de voisinage influe sur la quantité de données à traiter et donc sur les ressources matérielles nécessaires au calcul de score et la dynamique sur laquelle il est codé. Une fenêtre de grande taille (15x15 par exemple) convient pour traiter des zones peu texturées. Cependant, comme une grande zone est couverte, la précision sur les contours est faible. Une fenêtre de petite taille (3x3 par exemple) donne de meilleurs résultats au niveau des contours mais est sensible au bruit.

Afin de pallier à ce problème, une pondération peut être appliquée aux pixels du voisinage en fonction de leur intensité et de leur distance au centre de la fenêtre [15]. Ainsi, dans un grand voisinage, les pixels proches du pixel central au niveau de la distance et de l'intensité ont un poids plus important car ils ont de fortes chances d'appartenir au même objet. Cette technique est très calculatoire car il faut notamment recalculer les poids liés à l'intensité pour chaque fenêtre de voisinage. Cependant, le principe de poids géométrique peut être conservé en utilisant, par exemple, des poids en puissance de 2 permettant d'utiliser un décalage plutôt qu'une multiplication ou une division.

L'utilisation de fenêtres multiples pour le calcul d'un score d'appariement a été envisagée pour pallier au problème de taille des voisinages. Une méthode adaptée au calcul temps-réel combinant cinq fenêtres a été proposée dans [5]. Pour chaque paire de pixel, on utilise quatre fenêtres supplémentaires dont les centres sont les extrémités de la fenêtre centrale. Les deux meilleurs scores sont extraits et ajoutés au score de la fenêtre centrale. On obtient ainsi une fenêtre résultante dont la forme s'adapte localement à la scène.

Lorsque la métrique utilisée est non-paramétrique (Census ou Rank par exemple), l'augmentation de taille de la fenêtre de voisinage n'entraîne pas la déformation des contours des objets. Dans ce cas, les voisinages de grandes tailles sont à privilégier, bien que la dynamique du score dépende du nombre de pixels dans le voisinage, nombre qui augmente rapidement.

3. Architectures matérielles existantes

Il existe de nombreuses architectures qui adressent le problème des ADAS en général. IMAPCAR de NEC[1] est un processeur SIMD, constitué de 128 éléments de calcul VLIW cadencés à 100 MHz, embarquant 300 Ko de mémoire pour une consommation de moins de 2W. Une application de stéréovision à base de programmation dynamique a été portée sur une version antérieure à ce processeur, IMAP-VISION [7]. Le calcul de la carte de disparité est effectué en moins de 100 ms sur une région d'intêret de 128x128 pixels, pour 64 niveaux de disparité. L'architecture EyeQ2 de Mobileye[2], est composée de deux processeurs RISC à virgule flottante MIPS34K associés à cinq blocs de calcul dédiés et trois processeurs vectoriels. Elle embarque 512Ko de mémoire et consomme 3W à une cadence de 332 MHz, en

[1] www.necel.com/applications/en/automotive/imapcar

[2] www.mobileye.com

technologie 90nm. Les différents blocs de calcul étant parallèles, EyeQ2 est adaptée pour le traitement de plusieurs applications en simultané. Ces applications vont de la détection de changement de voie à la reconnaissance de panneau de signalisation en passant par la détection de collision avant. La stéréovision est utilisable puisqu'un des blocs de calcul est dédié à la disparité et peut traiter 2 pixels par cycles.
D'autres architectures adressent plus précisément le problème de la génération de la carte de disparité. La mesure généralement utilisée pour comparer les performances de ces architectures est le pixel de disparité par seconde, noté PDS (Pixel Disparities per Second). Cela correspond au nombre de comparaisons entre pixels effectué par seconde. Ainsi, le processeur Deepsea [14], qui utilise la métrique Census pour calculer la carte de disparité, offre une performance de 2,6 GPDS. En effet, ce système est capable de traiter des images 512x480 à une fréquence de 200 fps pour une consommation inférieure à 1W. La plage de disparité balayée est de 52 pixels mais la taille du voisinage utilisé n'est pas donnée. Dans [9], le calcul des scores d'appariement s'effectue aussi à l'aide de la métrique Census. L'utilisation d'un opérateur dédié pour le calcul de la distance de Hamming permet d'obtenir une fréquence image de 130 fps avec des images VGA (640x480), un niveau de disparité maximum de 64 pixels et une taille de voisinage allant de 3x3 à 7x7, ce qui correspond aussi à 2,6 GPDS. Cette architecture utilise environ 11100 éléments logiques et 174 Kbits de mémoire sur un composant FPGA (Altera Stratix 1S40). L'architecture VoC, décrite dans [6], se base sur une matrice reconfigurable capable de calculer à chaque cycle un score d'appariement à base de SAD sur un voisinage 7x7 ou neuf scores SAD 5x5. La zone de recherche du correspondant n'est pas limitée à une ligne horizontale et est définie par l'utilisateur. Elle a été synthétisée à 158 MHz sur un FPGA Virtex XC6000 et occupe 32459 slices, soit 80% du FPGA. Ses performances s'élèvent à 1,4 GPDS crête. Plus récemment, un processeur de stéréovision utilisant la SAD a été proposé [4]. Il opère à 120 MHz et peut traiter des images QVGA (320x240) à une fréquence de 140 fps pour une taille de fenêtres de voisinage de 11x11 et une disparité maximale de 64 pixels, ce qui donne une performance de 0,7 GPDS.
La plupart des architectures présentées ci-dessus offrent de bonnes performances mais avec une flexibilité limitée voire inexistante, soit au niveau de la métrique utilisée, soit par rapport au voisinage pris en compte. Or, les conditions applicatives et les besoins concernant la carte de disparité diffèrent d'une application à l'autre, et cette flexibilité est importante pour permettre le portage d'un nombre important de ces applications au sein de la même architecture.

4. Présentation de notre solution matérielle

L'unité de calcul de score que nous proposons a pour but de répondre au besoin de flexibilité tant au niveau des métriques utilisées pour le calcul des scores qu'au niveau de la taille et de la forme des fenêtres de voisinage, tout en respectant les contraintes liées au domaine des ADAS. Ainsi l'architecture doit être capable de traiter des paires d'images VGA monochromes 8 bits avec 64 niveaux de disparité pour une fréquence de 40 fps (frame per second). Cette fréquence correspond à une contrainte temps-réel forte d'application pré-crash [3]. Dans un soucis de limitation de la consommation en puissance, la fréquence du système est fixée à 200 MHz. L'unité de calcul de score doit donc être capable de générer quatre scores d'appariement par cycle. La taille de fenêtre maximale choisie est de 15x15, compromis entre un voisinage couvrant une surface importante et la dynamique des scores ainsi limitée à 16 bits. Le calcul de quatre scores par cycle sur une fenêtre de voisinage de taille 15x15 implique 900 paires de pixels à traiter par cycle.
La figure 2 présente la structure générale de la solution matérielle proposée. Le calcul des scores d'appariement s'effectue en parallèle au sein de quatre matrices de calcul flexibles. Chaque matrice contient 225 éléments de calcul (EC) et est associée à un arbre de réduction flexible (ARF). Les matrices sont alimentées en pixels par deux buffers de lignes de 10 Ko chacun. Chaque matrice génère au moins un score par cycle.
Nous présentons dans la section 4.1 les buffers de lignes et l'utilisation de la localité des données pour pallier au problème de la bande passante. Nous détaillons ensuite la structure d'une matrice de calcul ainsi que l'élément de calcul dans la section 4.2 . Enfin, nous présenterons dans la section 4.3 le modèle d'arbre de réduction reconfigurable proposé pour atteindre la flexibilité souhaitée pour les fenêtres de voisinage.

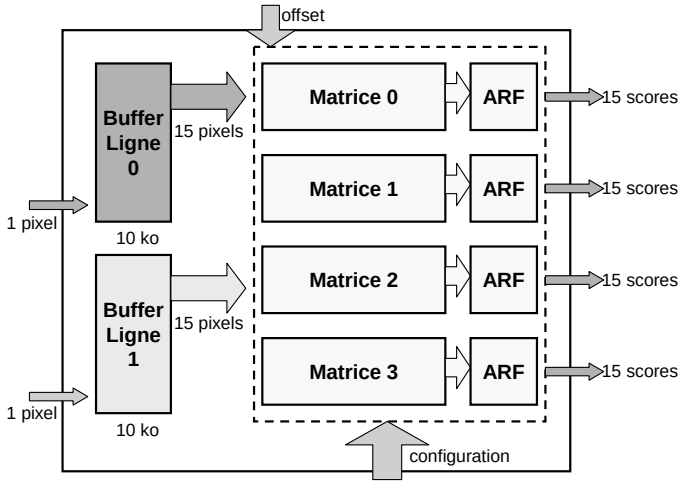


FIG. 2 – Structure générale de l'unité de calcul de scores d'appariement. Deux buffers de lignes alimentent les quatre matrices de calcul flexibles. Chaque matrice est associée à un arbre de réduction flexible et génère de 1 à 15 scores d'appariement.

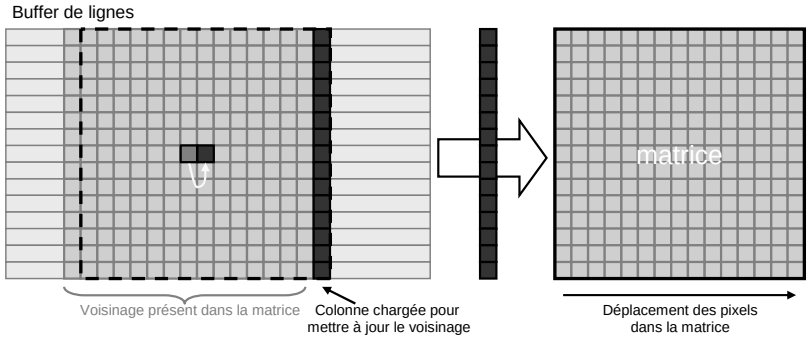


FIG. 3 – Les deux voisinages en cours de traitement sont stockés dans la matrice. Le passage au voisinage suivant s'effectue en mettant à jour le voisinage stocké. Pour effectuer cette mise à jour, la matrice lit une colonne de 15 pixels dans le buffer de lignes.

4.1. Buffers de lignes

Le contexte embarqué des architectures pour ADAS implique une contrainte de surface forte. Ainsi, la quantité de mémoires embarquées est limitée. Sachant qu'une image VGA nécessite 307 Ko, il n'est pas envisageable de stocker la paire d'images d'entrée en mémoire locale. Il est cependant nécessaire de stocker les lignes d'image contenant les voisinages utilisés pour le calcul de score, soit 15 lignes pour chaque image. Ainsi, deux mémoires locales de 10 Ko ont été mises en place. Appelées buffers de lignes sur la figure 2, elles peuvent contenir 16 lignes de 640 pixels chacune. Il est possible d'effectuer un accès mémoire par cycle sur chaque ligne. En considérant que la mise à jour du buffer nécessite une écriture par cycle, la bande passante en sortie de chaque buffer de lignes est de 15 octets par cycle.
La bande passante entre la mémoire locale et les matrices de calcul est un point critique. En effet, il n'est pas envisageable de lire à chaque cycle l'ensemble des pixels des deux voisinages pour le calcul

de chaque score d'appariement, soit 225 paires de pixels pour un voisinage 15x15. Comme on peut le remarquer sur la figure 3, les voisinages de deux pixels consécutifs diffèrent d'une seule colonne de pixels. Nous avons tiré avantage de cette propriété de localité des données en stockant au sein de la matrice de calcul les pixels de la paire d'images. Ainsi, le voisinage utilisé pour le calcul d'un score est mis à jour par le chargement d'une colonne de pixel (soit 15 pixels) pour obtenir le voisinage du pixel suivant.

4.2. Matrice de calcul

La plupart des fonctions de génération de scores d'appariement sont de la forme de l'équation (1), avec C le score généré, P_R un pixel appartenant à N_R - voisinage du pixel de référence, et P_A un pixel appartenant à N_A - voisinage du pixel analysé. Afin de générer un score par cycle, le calcul $f(P_R, P_A)$ doit être effectué sur toutes les paires de pixels en un cycle.

$$C = \sum_{N_R, N_A} f(P_R, P_A) \tag{1}$$

Nous proposons l'utilisation d'une matrice contenant autant d'éléments de calcul qu'il y a de paires de pixels à traiter, soit 225 dans notre cas. Chaque élément prend en charge le calcul de $f(P_R, P_A)$ à chaque cycle. Le traitement des 900 paires de pixels à chaque cycle nécessaire à la génération des quatre scores par cycle implique la mise en place de quatre de ces matrices en parallèle.
Comme le montre la figure 4, à chaque EC sont associés deux registres pixels, r0 et r1, contenant chacun un pixel des deux voisinages en cours de traitement dans la matrice. Ces registres communiquent avec les registres pixel des deux ECs de gauche et des deux ECs de droite permettant un déplacement horizontal bilatéral des pixels au sein de la matrice, avec un pas de 1 ou 2. Ce choix est expliqué plus loin dans la partie 4.2.2. Chacune des deux entrées de l'unité fonctionnelle de l'EC peut se connecter soit à un des deux registres pixel, soit au registre colonne (Rcol), soit à la sortie d'un des ECs voisins. Les différents cas sont expliqués en partie 4.2.1.

4.2.1. Elément de calcul

Notre but est de proposer un élément de calcul capable d'effectuer le calcul élémentaire de la SAD et du CENSUS, en un cycle, tout en proposant un niveau de flexibilité élevé pour d'éventuelles améliorations, notamment la possibilité d'effectuer des pondérations. La possibilité d'intégrer des métriques plus complexes (SSD, NCC) n'a pas été retenue pour des raisons de limitation en surface, les opérateurs de carré et de multiplication étant trop coûteux.
Les principales opérations prises en charge par l'élément de calcul que nous proposons sont résumées dans la figure 5. Il permet notamment d'effectuer les opérations de soustraction, de soustraction absolue et de comparaison. Un décalage sur le résultat de ces opérations peut être effectué par la suite. Une des particularités de cet élément est qu'il est capable de mémoriser une entrée pour l'utiliser comme opérande dans une des opérations citées ci-dessus. Il est aussi possible de mémoriser le résultat binaire de la comparaison entre les deux entrées et de l'utiliser ensuite lors d'une opération logique avec le résultat d'une autre comparaison. Les deux entrées de l'EC sont sur 8 bits, signés ou non. Le résultat de l'opération est stocké dans un registre de sortie et est codé sur 8 bits signés ou non.
Les opérations disponibles nous donnent ainsi accès au calcul de la SAD, du CENSUS et à l'ajout/retrait d'un offset. Cet offset est calculé en dehors de l'unité de calcul et stocké dans le registre colonne (R col dans la figure 4). Il peut correspondre notamment à la moyenne utilisée dans le calcul de ZSAD. Ainsi, en associant deux ECs adjacents, il est possible d'effectuer une ZSAD. Le premier effectue le centrage du pixel par rapport à la moyenne du voisinage. Le second EC récupère en entrée le résultat du premier EC et calcule la différence absolue. Cependant, comme le premier EC ne peut effectuer qu'un centrage par cycle, le pixel de référence centré doit être calculé en premier et mémorisé dans le second EC. Ensuite, le premier EC effectue le centrage des pixels à analyser à chaque cycle et le second EC calcule la différence absolue en utilisant le pixel de référence centré stocké dans le registre.
Le décalage en sortie de l'EC s'effectue sur 1 bit à droite et 2 bits à gauche. Il est donc possible d'attribuer un poids de 1/2, 1, 2 et 4 au résultat de l'opération principale de l'EC, et ainsi obtenir une pondération sur les fenêtres de voisinage. Les bits de configuration de l'opérateur de décalage sont contenues dans le registre de configuration de l'EC.

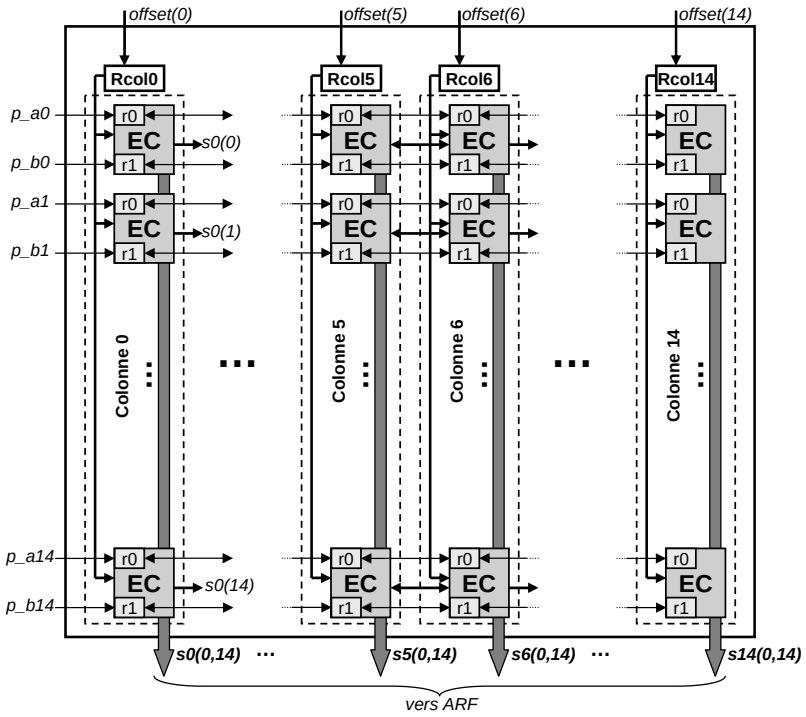
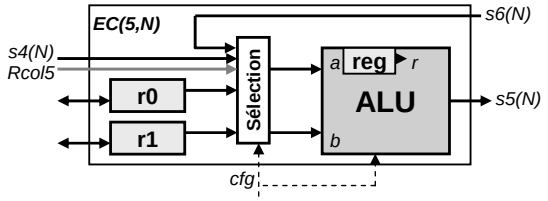


FIG. 4 – La matrice de calcul est composée de 15 colonnes de 15 éléments de calcul chacune. Les pixels des deux voisinages traités sont contenus dans les registres pixels r0 et r1 des ECs. Chaque colonne possède un registre Rcol pouvant être connecté aux entrée de ses ECs. Les sorties de tous les ECs sont connectées aux entrées de l'arbre de réduction flexible (ARF).



Opérations principales de l'ALU:

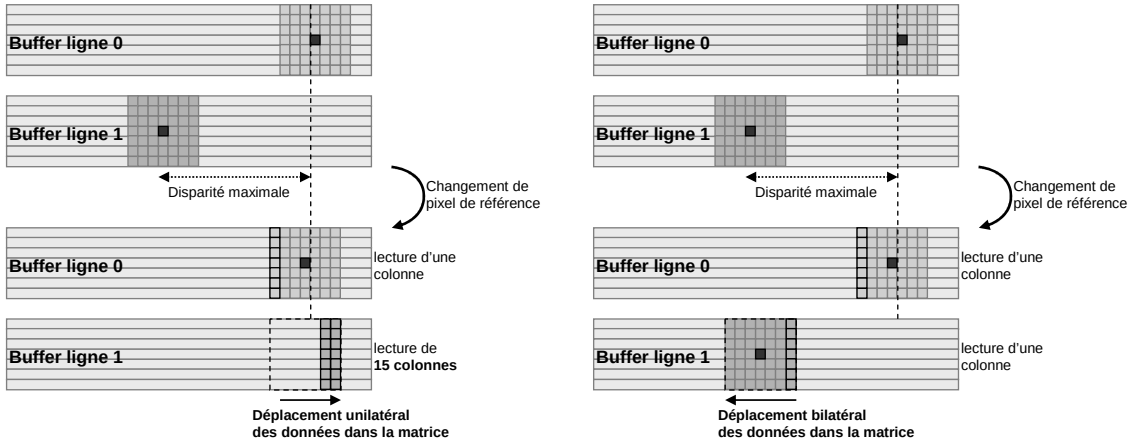
- mémorisation d'entrée	**r = a**
- soustraction:	**s = a – b** ou **s = r – b**
- soustraction pondérée:	**s = (a – b) << p** ou **s = (r – b) << p**
p = cte	**s = (a – b) >> p** ou **s = (r – b) << p**
- différence absolue	**s = |a – b|** ou **s = |r – b|**
- différence absolue pondérée	''
- comparaison	**s = (a > b)**
- comparaison puis mémorisation	**r' = (a > b)**
- comparaison puis opération logique avec valeur mémorisée	**s = ((a > b) XOR r')**

FIG. 5 – Structure de l'élément de calcul et détails des principales opérations effectuées par l'ALU. Chacune des deux entrées de l'élément de calcul peut être connectée à un des deux registres pixels, au registre de la colonne Rcol, ou à la sortie d'un des deux éléments de calcul voisins. Le coefficient de pondération p est une constante contenue dans la configuration cfg

4.2.2. Déplacement de données au sein de la matrice

Comme nous l'avons vu précédemment, les pixels des voisinages en cours de traitement sont stockés au sein de la matrice de traitement. Comme l'illustre la figure 6(a), pour la génération des scores relatifs

à un pixel de référence donné, le voisinage sur l'autre image glisse à mesure que l'on se déplace d'un pixel testé à l'autre. Lorsque le score relatif au dernier pixel à tester est généré, le voisinage du pixel de référence est mis à jour par le chargement d'une colonne de pixel.



(a) Cas du parcours unilatéral de la matrice

(b) Cas du parcours bilatéral de la matrice

FIG. 6 – Parcours de la matrice de calcul. Dans le cas où le déplacement des pixels est unilatéral, à chaque changement de pixel de référence, la lecture des 15 colonnes du premier voisinage à analyser est nécessaire. Cette pénalité de 15 cycles est supprimée si les pixels de voisinage peuvent se déplacer dans la matrice dans les deux sens.

Dans le cas d'une matrice où le déplacement s'effectuerait dans un seul sens (de droite à gauche dans le cas de la figure 6(a)), au chargement d'un nouveau pixel de référence, le premier pixel à tester n'étant pas adjacent au dernier pixel testé, il est nécessaire de charger l'ensemble de son voisinage. Si l'on ne considère qu'une seule matrice, la pénalité de chargement est de 15 cycles, correspondant au temps pour traverser la matrice de calcul. De plus, cette pénalité implique la nécessité de générer cinq scores d'appariement par cycle au lieu de quatre par cycle pour pouvoir répondre aux contraintes du temps-réel. Dans ce cas, cinq matrices en parallèle sont nécessaires, soit 25% d'augmentation en surface. Dans le cas de 4 matrices en parallèle la pénalité est plus importante.
En permettant le parcours de la matrice de calcul dans les deux sens, la pénalité de chargement est annulée. En effet, si l'on se reporte à la figure 6(b), lors du changement de pixel de référence, il est possible d'effectuer le balayage dans le sens inverse et d'utiliser le voisinage du dernier pixel analysé pour mettre à jour le voisinage du nouveau pixel à tester. Les résultats de synthèse ASIC que nous avons effectué ont montré que cette solution nécessite à peu près 11% de surface en plus par rapport à la solution à quatre matrice à déplacement unilatéral, ce qui est un bon compromis comparé aux 25% d'augmentation de surface entraînée par l'ajout d'une matrice supplémentaire.

4.3. Arbre de réduction flexible

Le nombre de données à sommer à chaque cycle au sein de la matrice atteint un maximum de 225 dans le cas d'une fenêtre 15x15. Cette sommation peut être réalisée à l'aide d'un arbre de réduction simple. Cependant, nous avons fait le choix de rendre l'arbre de réduction reconfigurable afin de pouvoir définir le voisinage sur lequel est calculé le score. Ainsi, seuls sont pris en compte les éléments de calcul traitant les pixels appartenant au voisinage qui nous intéresse. Quelques aménagements de l'arbre reconfigurable donne la possibilité à la matrice de calcul de générer plusieurs scores. Enfin, il nous a paru

nécessaire de permettre le recouvrement entre plusieurs fenêtres de voisinage au sein de la même matrice pour donner accès à la technique de fenêtres multiples. En effet, cette technique fournit de bon résultats pour le calcul de la carte de disparité.
L'arbre de réduction flexible que nous proposons est composé d'additionneurs à trois entrées. Pour chacun de ces additionneurs, 3 bits de configuration spécifient les entrées prises en compte dans le calcul. L'arbre de réduction se divise en trois niveaux, ainsi que l'illustre la figure 7.
Le premier niveau est constitué de 15 sous-arbres. Chaque sous-arbre effectue la réduction sur une colonne en particulier. Ses entrées sont directement connectées aux sorties des ECs de la colonne et il fournit 3 sorties parmi les 7 possibles au deuxième niveau. Le second niveau est constitué de trois réseaux en couches. Il possède 45 entrées et 15 sorties. Le dernier niveau permet de fusionner les sous-scores générés par le niveau 2 pour obtenir de 1 à 15 scores.

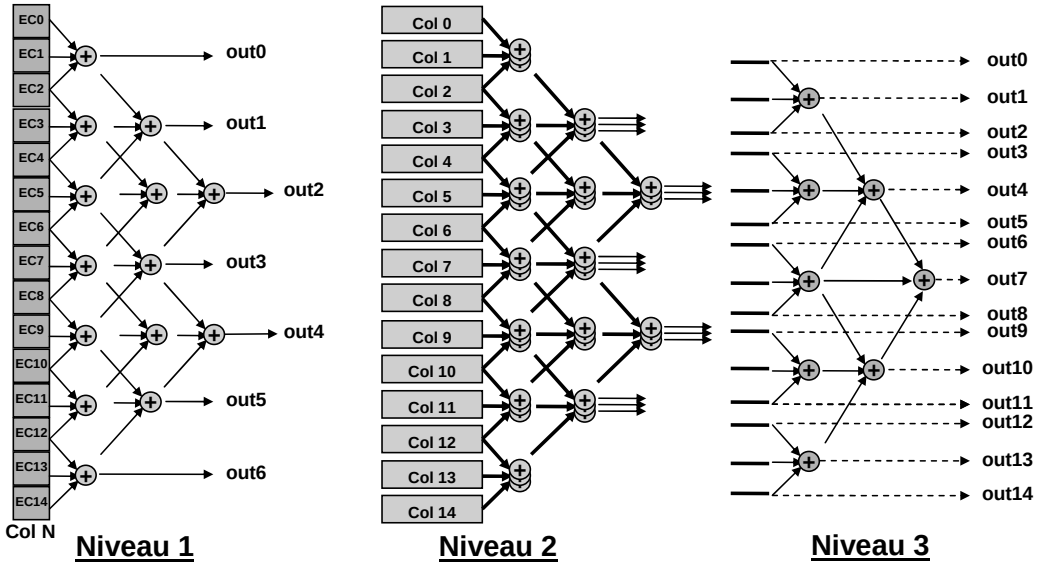


FIG. 7 – Structure de l'arbre de réduction flexible. Il se divise en trois niveaux. Le niveau 1 est constitué de 15 sous-arbres,chacun connecté à une colonne de la matrice de calcul. L'arbre de réduction génère jusqu'à 15 scores

La taille des fenêtres de voisinage générées par l'ARF varie entre 2x2 et 15x15 dans le cas d'une fenêtre simple et entre 3x3 et 9x9 en fenêtres multiples. Il est possible d'obtenir un recouvrement de trois pixels horizontalement et verticalement entre trois fenêtres maximum. Cependant, il ne peut y avoir plus de trois fenêtres multiples sur la même colonne. Chaque ARF utilise 260 additionneurs 3 voies. La configuration de la totalité des additionneurs de l'arbre nécessite 852 bits. Le choix de la fenêtre de voisinage étant généralement fixe pour une application donnée, la configuration de l'ARF est statique - fixée à l'initialisation - et ne pose donc pas de problème de pénalité de configuration pendant l'exécution.
Nous avons conçu une version dégradée de cet arbre utilisant 188 additionneurs 3 voies, soit 28% d'additionneurs en moins. Cet arbre de réduction dégradé possède les mêmes caractéristiques mis à part qu'il ne permet le recouvrement que de deux fenêtres multiples. Il nécessite 636 bits de configuration. Cette version pourra être privilégiée dans le cas où le recouvrement entre les fenêtres de voisinage n'est pas primordial.

4.4. Résultats

L'unité de calcul de score flexible est capable de générer quatre scores d'appariement par cycle à une fréquence de 200 MHz sur des voisinages 15x15. Ceci correspond au calcul de la disparité d'une paire d'images VGA (640x480) à une cadence de 40 fps et pour 64 niveaux de disparité, soit environ 0,8 GPDS. Nous avons effectué la synthèse ASIC d'une version de l'unité de calcul de score ne possèdant pas la capacité de décalage en sortie des éléments de calcul. Cette synthèse a été effectuée en technologie 65nm Low-Power, à 200 MHz. L'unité de calcul de score flexible occupe une surface d'environ 3 mm^2.

En dehors de l'aspect flexible, notre solution possède un potentiel adaptatif. Ainsi, selon les besoins, elle peut être dimentionnée à la synthèse. Pour une puissance de calcul plus élevée, l'ajout d'une ou plusieurs matrices peut être envisagé. Le calcul de scores d'appariement sur des voisinages plus importants peut être réalisé en augmentant la taille des matrices. Enfin, l'utilisation de buffers de lignes de plus grande capacité permet de traiter des paires d'images de résolution plus importante.

De même, l'utilisation d'un nombre moins élevé de matrices ou de matrices plus petites est envisageable dans le cas d'une solution basse consommation.

5. Conclusion et perspectives

La génération de la carte de disparité est une étape importante dans les applications d'aide avancée à la conduite basées sur la stéréovision. Vu le grand nombre de solutions algorithmiques proposées pour obtenir des cartes de disparité de caractéristiques variables, la flexibilité des solutions matérielles est une nécessité.

L'unité de calcul flexible présentée dans cet article effectue le calcul des scores d'appariement, étape très calculatoire dans l'estimation de la disparité. Composée de quatre matrices de calcul associées chacune à un arbre de réduction, l'unité de calcul génère au moins quatre scores d'appariement par cycle, selon les métriques SAD, ZSAD ou CENSUS. Le modèle d'arbre de réduction flexible utilisé permet de générer à chaque cycle un score sur une fenêtre de voisinage allant de 2x2 à 15x15 et jusqu'à 15 scores sur des fenêtres multiples de taille 3x3 à 9x9, avec possibilité de recouvrements. Deux buffers de ligne, de 10 Ko chacun, fournissent deux colonnes de 15 pixels par cycle aux matrices de calcul.

Notre architecture est capable de générer quatre scores par cycle à une fréquence de 200 MHz, soit une performance de 0,8 GPDS. Elle peut ainsi calculer les scores d'appariement sur un voisinage 15x15 pour l'estimation de la disparité d'une paire d'images VGA (640x480) à une cadence de 40 fps et pour 64 niveaux de disparité maximum. L'intégration prochaine d'un opérateur de décalage permettra la mise en place de la technique de pondération géométrique.

L'intégration de notre unité de calcul au sein d'un SoC ouvre la voie au portage de divers applications de stéréovision et plus particulièrement d'applications d'aide avancée à la conduite. Ainsi, une perspective intéressante serait d'intégrer cette unité dans un coprocesseur de calcul de disparité capable de gérer plusieurs méthodes de mise en correspondance, comme des méthodes locales avancées et des méthodes semi-globales. L'aspect flexible peut même être mis à profit pour des calculs de type filtrage. Ainsi, cette unité de calcul peut très bien prendre en charge le filtrage de Rank, voir une extraction de contour par filtre de Sobel.

Bibliographie

1. Nikolay Chumerin and Marc M. Van Hulle. Ground plane estimation based on dense stereo disparity. *The Fifth International Conference on Neural Networks and Artificial Intelligence, Minsk, Belarus*, pages 209–213, May 27-30 2008.
2. D. M. Gavrila, J. Giebel, and S. Munder. Vision-based pedestrian detection : The PROTECTOR system. In *In IEEE Intelligent Vehicles Symposium*, pages 13–18, 2004.
3. A. Greiner, F. Petrot, M. Carrier, M. Benabdenbi, R. Chotin-Avot, and R. Labayrade. Mapping an obstacles detection, stereo vision-based, software application on a multi-processor system-on-chip. *Intelligent Vehicles Symposium, 2006 IEEE*, 1 :370–376, 2006.
4. Sang-Kyo Han, SeongHoon Woo, Mun-Ho Jeong, and Bum-Jae You. Improved-quality real-time stereo vision processor. pages 287–292, Jan. 2009.

5. Heiko Hirschmüller, Peter R. Innocent, and Jon Garibaldi. Real-time correlation-based stereo vision with reduced border errors. *Int. J. Comput. Vision*, 47(1-3) :229–246, 2002.
6. R.P. Jacobi, R.B. Cardoso, and G.A. Borges. Voc : a reconfigurable matrix for stereo vision processing. pages 6 pp.–, April 2006.
7. Gerold Kraft and Pieter Jonker. Real-time stereo with dense output by a simd-computed dynamic programming algorithm. In *PDPTA '02 : Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*, pages 1031–1036. CSREA Press, 2002.
8. R. Labayrade, D. Aubert, and J.-P. Tarel. Real time obstacle detection in stereovision on non flat road geometry through "v-disparity" representation. *Intelligent Vehicle Symposium, 2002. IEEE*, 2 :646–651, 2002.
9. A. Naoulou, J.-L. Boizard, J.Y. Fourniols, and M. Devy. An alternative to sequential architectures to improve the processing time of passive stereovision algorithms. In *Field Programmable Logic and Applications, 2006. FPL '06. Int. Conf. on*, pages 1–4, 2006.
10. S. Nedevschi, R. Schmidt, T. Graf, R. Danescu, D. Frentiu, T. Marita, F. Oniga, and C. Pocol. 3d lane detection system based on stereovision. *Intelligent Transportation Systems, 2004. Proceedings. The 7th Int. IEEE Conf. on*, pages 161–166, Oct. 2004.
11. A. Sappa, D. Geronimo, F. Dornaika, and A. Lopez. Stereo vision camera pose estimation for on-board applications. In Rustam Stolkin, editor, *Scene Reconstruction, Pose Estimation and Tracking*, chapter 3, pages 39–50. I-Tech Education and Publishing, 2007.
12. Luigi Di Stefano and Stefano Mattoccia. Fast template matching using bounded partial correlation. *Mach. Vision Appl.*, 13(4) :213–221, 2003.
13. W. Van der Mark and D.M. Gavrila. Real-time dense stereo for intelligent vehicles. *Intelligent Transportation Systems, IEEE Transactions on*, 7(1) :38–50, 2006.
14. John Iselin Woodfill, Gaile Gordon, and Ron Buck. Tyzx deepsea high speed stereo vision system. In *CVPRW '04 : Proc. of the 2004 Conf. on Computer Vision and Pattern Recognition Workshop (CVPRW'04) Volume 3*, page 41, Washington, DC, USA, 2004. IEEE Computer Society.
15. Kuk-Jin Yoon and In So Kweon. Adaptive support-weight approach for correspondence search. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(4) :650–656, 2006.
16. Ramin Zabih and John Woodfill. Non-parametric local transforms for computing visual correspondence. In *ECCV '94 : Proc. of the Third European Conf.-Volume II on Computer Vision*, pages 151–158, London, UK, 1994. Springer-Verlag.