N° d'ordre : 3606

THÈSE

présentée par

Madaín PÉREZ PATRICIO

pour obtenir le grade de Docteur
de l'Université des Sciences et Technologies de Lille

Discipline : Automatique et Informatique Industrielle

Stéréovision dense par traitement adaptatif temps réel : Algorithmes et implantation

Soutenue le 25 février 2005

Jack-Gérard POSTAIRE	Président	Professeur à l'USTL
Patrick GORIA	Rapporteur	Professeur à l'IUT Le Creusot
Edwige PISSALOUX	Rapporteur	Professeur à l'Université de Rouen
Abdelaziz BENSRHAIR	Examinateur	Professeur à l'INSA de Rouen
Sebastien AMBELLOUIS	Examinateur	Chargé de recherche à l'INRETS
Miguel ARIAS ESTRADA	Examinateur	Professeur à l'INAOE
François CABESTAING	Co-Directeur de thèse	MCF-HDR à l'USTL
Olivier COLOT	Co-Directeur de thèse	Professeur à l'USTL

Thèse préparée au **L**aboratoire d'**A**utomatique, **G**énie **I**nformatique et **S**ignal

LAGIS UMR CNRS 8146

Remerciements

Le travail présenté dans cette thèse a été réalisé au Laboratoire d'Automatique, Génie Informatique et Signal (LAGIS) de l'Université des Sciences et Technologies de Lille.

Je tiens à exprimer toute ma gratitude à Celui qui nous a équipé d'un système stéréoscopique de série si puissant.

Je tiens à remercier Monsieur Jack-Gérard Postaire, professeur à l'USTL, pour m'avoir accueilli au sein de son équipe, pour avoir accepté la présidence du jury et pour son soutien depuis mon arrivée au laboratoire.

Je tiens également à remercier Monsieur François Cabestaing, maître de conférences HDR à l'USTL, pour avoir co-dirigé cette thèse. C'est grâce à ses conseils très pertinents que ce travail a été mené à bien.

Je tiens à exprimer toute ma gratitude à Monsieur Olivier Colot, professeur à l'USTL, pour avoir co-dirigé mes travaux de recherche. Cette thèse n'aurait pu aboutir sans sa direction.

Ma reconnaissance et mes vifs remerciements vont à Madame Edwige Pissaloux pour avoir accepté d'être l'un de rapporteurs de ce mémoire.

J'exprime ici ma gratitude à Monsieur Patrick Gorria pour avoir également accepté de juger mon travail.

Je suis très reconnaissant à Monsieur Abdelaziz Bensrhair, d'avoir bien voulu siéger à cette commission d'examen.

Monsieur Miguel Arias Estrada a accepté de juger mon travail. Je lui suis très reconnaissant de faire partie des membres du jury.

Je souhaite remercier vivement Monsieur Sebastien Ambellouis pour avoir accepté de juger mon mémoire et faire partie du jury de cette thèse.

Je ne peux pas oublier de remercier Dany, Judith, Daysi, et mes parents pour avoir

accepté de se lancer avec moi dans cette aventure.

Table des matières

Table des figures

Introduction

De nombreuses applications récentes, principalement dans le domaine de la robotique mobile, nécessitent de disposer d'une représentation tridimensionnelle du monde réel. Le système reconstitue habituellement une carte plus ou moins détaillée de son environnement en fusionnant les informations issues des différents capteurs dont il est doté. Les caméras vidéo, associées à un système de vision artificielle, font partie des capteurs qui peuvent être utilisés à cette fin.

Dans certains cas, la perception de l'environnement permet à un robot mobile de suivre une trajectoire prédéterminée [Faugeras *et coll.* 1993] où de détecter et poursuivre une cible [Matsumoto *et coll.* 1997]. Dans ces situations précises, la perception de l'environnement est employée afin de détecter les obstacles présents sur le chemin emprunté par le robot, de telle sorte qu'il puisse modifier ponctuellement la trajectoire à suivre afin de les éviter. La tâche de planification globale de la trajectoire reste cependant à la charge d'un opérateur humain.

Dans d'autres situations, l'intervention d'un opérateur humain est exclue, ce qui complique encore la tâche de navigation autonome. Par exemple, pour les robots destinés à l'exploration de la surface des autres planètes du système solaire, le temps de communication important — de l'ordre de 20 minutes entre la Terre et Mars — interdit toute intervention des opérateurs dans la tâche de navigation. Ainsi, il faut doter les robots mobiles de systèmes capables de percevoir l'environnement de façon extrêmement précise et fiable [Urmson et Dias 2002].

Qu'il s'agisse ou non de faire intervenir un opérateur humain, les informations fournies par les capteurs doivent être prises en compte en un temps minimum, afin de garantir une actualisation rapide de la représentation interne de l'environnement. C'est pourquoi, en parallèle avec le développement de nouveaux capteurs, les chercheurs travaillent également

de façon intensive sur les techniques d'accélération des traitements. Une solution consiste à employer des architectures de traitement adaptées à la structure des données fournies par le capteur, plutôt qu'une structure standard de type ordinateur séquentiel.

I.1 Assistance à la conduite automobile

On a assisté durant ces dernières années à l'apparition de systèmes de transport automatisés, tel le véhicule automatique léger (VAL) fonctionnant à Lille et Toulouse. Ces véhicules guidés naviguent en aveugle dans un environnement protégé, aucun obstacle n'étant sensé se trouver sur leur trajectoire. Une nouvelle génération de systèmes confient le guidage à des automatismes exploitant les images fournies par des caméras installées à l'avant du véhicule. Par exemple, dans le véhicule CIVIS[1] le système de guidage utilise un système de vision artificielle.

Dans le domaine de l'automobile, les constructeurs de véhicules ont également travaillé au développement de systèmes de perception de l'environnement. L'objectif n'est pas obligatoirement le pilotage automatique du véhicule, ce qui nécessitera encore quelques années de recherche et développement, mais uniquement l'assistance au conducteur. Dans ce cas, le système pallie les défauts du conducteur — le manque de vigilance, la mauvaise appréciation des distances, etc. — en l'avertissant d'un danger potentiel. Dans ces applications, la perception de l'environnement du véhicule constitue également un élément clé dont dépend la fiabilité globale du système.

Différentes technologies de capteurs peuvent être employées pour reconstituer une carte tridimensionnelle représentative d'une scène. Le radar émet une onde électromagnétique vers l'environnement et détermine la présence d'objets en analysant le temps de retour des ondes réfléchies par ces derniers. Le sonar et le lidar fonctionnent selon le même principe, émettant un signal sonore pour le premier, et un faisceau laser pour le second. Dans les systèmes exploitant la lumière structurée, un projecteur éclaire la scène avec un ou plusieurs motifs [Devernay *et coll.* 2002, Zhang *et coll.* 2002]. Une caméra enregistre les déformations de ces motifs causées par les objets afin de reconstruire la scène par triangulation. Tous ces capteurs sont qualifiés d'actifs, car ils émettent un signal, puis

[1] http ://www.matra-transport.fr/pages/produits/Civis.htm

analysent les signaux réfléchis par les objets présents dans la scène.

Les capteurs qualifiés de passifs n'exploitent que la lumière provenant d'une source extérieure qui a été réfléchie par les objets présents dans la scène. Les caméras constituent le premier élément de la chaîne dont l'objectif est de mimer le système visuel des animaux ou de l'être humain : le système de vision artificielle. Certains systèmes utilisent une seule caméra, fournissant une image de la scène perçue depuis un unique point de vue. Dans ce cas, le contenu de l'image est analysé afin de reconnaître, jusqu'à un certain niveau de précision, les objets présents dans la scène.

D'autres systèmes de vision, qui nous intéressent plus particulièrement dans ce travail, utilisent deux caméras pour observer la scène. Ils utilisent donc une technique similaire à celle employée dans les systèmes biologiques, la plupart des animaux dotés d'un système de vision ayant au moins deux yeux. Le fait de disposer de deux images de la scène au même instant permet, sous certaines conditions, de reconstituer l'information tridimensionnelle qui a été perdue au cours du processus de formation des images.

I.2 La stéréovision

À partir des images fournies par au moins deux caméras, il devient possible de reconstruire la troisième dimension qui a été perdue dans les images. Ce principe est utilisé dans les systèmes stéréoscopiques lesquels utilisent plusieurs caméras. Un système stéréoscopique classique n'utilise que deux caméras, comme l'indique la figure I.1. L'utilisation d'un nombre plus élevé de caméras permet habituellement de valider les résultats obtenus [Mulligan *et coll.* 2002]. Nous nous limitons à la description des systèmes n'incluant que deux caméras.

Les images acquises par le système stéréoscopique sont appelées respectivement image gauche et image droite. Elles sont obtenues au même instant à partir de deux points de vue légèrement différents. De ce fait, les projections dans les deux images d'un même point de la scène n'ont pas les mêmes coordonnées. La projection dans l'image droite est décalée par rapport à la projection dans l'image gauche. Le décalage entre les coordonnées des projections est appelé disparité. La disparité dépend directement de la distance du point au système de caméras. Plus le point est proche du système de caméras, plus la disparité

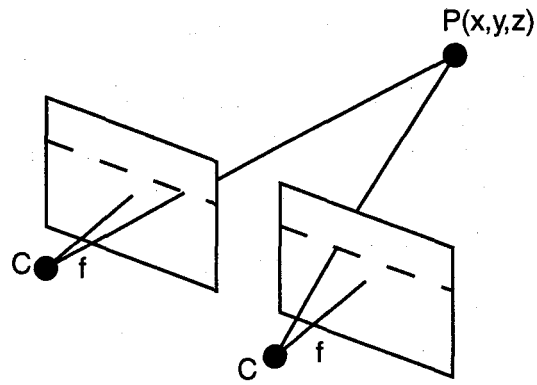


FIG. I.1 : Un système stéréoscopique classique.

est grande et inversement. À partir de la disparité, une simple technique de triangulation permet de calculer la distance entre le point de la scène et les caméras.

Pour déterminer la disparité, il faut repérer dans les images les deux pixels, qualifiés d'homologues, qui correspondent à un point unique dans la scène. Cette recherche des points homologues est appelée appariement. Plusieurs facteurs interviennent dans le fait que l'appariement est la tâche la plus complexe dans un système de vision stéréoscopique. En raison du nombre élevé d'opérations nécessaires pour apparier les points, le temps de calcul correspondant peut devenir prohibitif. Cependant, dans les applications visées par le système que nous présentons dans cette thèse, un traitement temps-réel doit être réalisé.

I.3 Accélération des traitements

Quand un système doit fonctionner à la cadence vidéo et traiter des images à pleine résolution, le nombre d'opérations élémentaires devant être réalisées chaque seconde est impressionnant. Une architecture de calcul classique, comme un ordinateur de type PC, traite les données de façon séquentielle. Il faut donc utiliser des boucles imbriquées, balayant successivement les images, puis les lignes, puis chaque pixel, afin d'implanter un algorithme de vision artificielle. Malgré l'augmentation constante des cadences auxquelles les données sont traitées, les architectures de calcul classiques ne sont pas capables d'at-

teindre le débit nécessaire dans les applications temps-réel qui travaillent avec des images à pleine résolution.

Afin de traiter les images sur une architecture classique, en respectant la cadence imposée par les caméras, il faut réaliser des compromis ou employer des astuces de programmation. On peut diminuer la cadence en ne traitant pas certaines images, ou diminuer le nombre de données en abaissant la résolution des images. Par exemple, le système stéréoscopique décrit dans [Mulligan *et coll.* 2002] utilise des images de taille 320×240 et ne fournit que 1.75 cartes de disparité par seconde. On peut également utiliser les instructions spécialisées d'extension multimédia (MMX) disponibles dans les processeurs récents [Martínez *et coll.* 2003, Stefano *et coll.* 2002, Stefano et Mattoccia 2000].

Cette dernière solution exploite le fait que des traitements identiques sont opérés sur des données différentes et peuvent donc être réalisés au même instant par plusieurs unités de traitement. La plupart de algorithmes de stéréovision remplissent cette condition : ils réalisent les mêmes types d'opérations sur des données différentes. On peut alors décomposer les techniques de calcul, en faisant apparaître ce qu'on qualifie de parallélisme, c'est-à-dire le fait que plusieurs calculs peuvent être réalisés simultanément par des unités de traitement indépendantes.

Dans [Kanade *et coll.* 1995, Faugeras *et coll.* 1993, Konolige 1997, Choudhary *et coll.* 1990], des architectures utilisant des processeurs numériques de signaux (Digital Signal Processor, ou DSP) ont été présentées. Les DSPs ont des instructions dédiées au traitement des signaux qui exploitent le parallélisme des algorithmes. Le traitement global reste séquentiel, mais certaines instructions spécifiques traitent plusieurs données simultanément. Pour obtenir des performances temps-réel, on peut également utiliser plusieurs processeurs DSP, chacun traitant une partie de l'image. L'inconvénient de ce type d'architecture est qu'elle n'est pas reconfigurable [Williamson 1998].

Dans [Yang *et coll.* 2003, Yang et Pollefeys 2003, Woetzel et Koch 2004] les auteurs ont utilisé les instructions spécialisées des processeurs intégrés dans les cartes graphiques actuelles. Ces cartes traitent les données en parallèle au moyen d'une architecture pipeline. Le nombre d'instructions disponibles est très restreint, mais parfaitement optimisé car la plupart des instructions sont exécutées en un seul top d'horloge. En revanche, la précision arithmétique disponible dans les unités de traitement intégrées sur ces cartes est rarement

suffisante. Ainsi, plusieurs caméras sont nécessaires pour lever les ambiguïtés engendrées par des calculs en faible précision.

Plusieurs travaux ont été présentés également afin de transformer les caméras en un dispositif plus spécialisé, appelé rétine artificielle [Barbaro et Raffo 2001, Mahowald 1994, Maher *et coll.* 1989]. Le capteur et les dispositifs de traitement forment ainsi un ensemble capable de fournir des informations de haut niveau qui permettent de réduire le nombre d'opérations ultérieures à réaliser et ainsi d'accélerer les traitements. Hélas, la résolution de ces dispositifs reste encore limitée et la plupart des algorithmes de stéréovision standard ne sont pas adaptés pour être implantés dans ce type d'architecture [Moini 1999].

I.4 Architectures spécialisées pour la stéréovision

Pour pallier les inconvénients des architectures non spécialisées, ou tout au moins non adaptées au traitement en temps-réel des images, certaines équipes ont développé des structures de calcul dédiées. Les processeurs ACADIA [Wal *et coll.* 2000] ou PAPRICA [Broggi *et coll.* 1997] utilisent des circuits logiques développés sur mesure (ASICs) qui optimisent l'usage des ressources matérielles et permettent l'implantation de traitements stéréoscopiques en temps-réel. Cependant, le temps de développement de ces dispositifs est long et le coût de développement est élevé [Darabiha *et coll.* 2003].

Plus récemment, les circuits logiques spécifiques utilisés dans ces architectures ont été remplacés par des circuits programmables à haute densité, comme les FPGAs (Field Programmable Gate Array). Ces dispositifs, qui sont par nature reconfigurables, disposent d'un grand nombre de portes logiques, ce qui permet l'implantation d'un système complet dans un seul composant. Ils sont bien adaptés au traitement des images à la cadence vidéo, car ils travaillent à des vitesses comparables à celles des circuits dédiés, tout en apportant la flexibilité d'un système programmable. Comme dans les ASICs, chaque module est développé et optimisé pour réaliser une opération spécifique [Herzen 1997].

Des architectures intégrant plusieurs FPGAs sur une même carte ont été décrites dans la littérature [Woodfill et Herzen 1997, Darabiha *et coll.* 2003]. Dans cet exemple, des méthodes de synchronisation complexes entre composants ont été introduites, puisque les unités de traitement sont réparties dans plusieurs circuits. Des architectures mixtes,

utilisant simultanément des FPGAs et des DSPs, ont également été décrites dans la littérature [Batlle *et coll.* 2002, Ohm *et coll.* 1998]. Leur efficacité est également limitée par les problèmes liés à la synchronisation des différentes unités de traitement.

L'utilisation d'un seul composant, disposant d'un nombre suffisant de portes logiques élémentaires permet de pallier ces inconvénients. Au sein du LAGIS, un Système Temps-Réel d'Extraction et d'Analyse du Mouvement (STREAM) a été développé durant les dix dernières années [Cabestaing *et coll.* 1991; 1999]. Il est composé d'une partie dédiée à la numérisation des signaux vidéo et d'un circuit logique FPGA assurant le traitement des images.

I.5 Plan du mémoire

Les résultats présentés dans ce mémoire découlent d'un travail de recherche qui visait plusieurs objectifs. Il s'agissait tout d'abord d'implanter sur le processeur STREAM les algorithmes standard de stéréovision qui avaient déjà été proposés dans la littérature. D'autre part, il fallait vérifier que ces techniques de stéréovision à la cadence vidéo pouvaient être intégrées dans les systèmes de détection d'obstacles développés dans notre laboratoire durant ces dernières années. En parallèle avec ces aspects relativement technologiques, nous avions comme objectif de proposer une technique originale de mise en correspondance stéréoscopique. Ces trois aspects sont présentés dans ce mémoire, en suivant un plan qui ne respecte pas l'ordre chronologique dans lequel ils ont été abordés.

Dans le chapitre 1, nous présentons les concepts de base de la stéréovision et les techniques qui permettent de reconstruire l'information tridimensionnelle à partir de deux images d'une même scène. Dans ce chapitre, nous décrivons également les méthodes utilisées pour mettre en correspondance des primitives extraites des images. Étant donné qu'aucune de ces techniques n'est parfaite, nous en présentons systématiquement les avantages et inconvénients.

Les algorithmes de stéréovision exploitant une mesure de corrélation entre les deux images du couple stéréoscopique semblent les mieux adaptées à une implantation sur une architecture spécifique de calcul. C'est la raison pour laquelle nous nous sommes particulièrement intéressés à ces méthodes, qui font l'objet du chapitre 2. Certains algorithmes

utilisant plusieurs fenêtres et susceptibles d'être implantés dans une architecture spécialisée de calcul, sont notamment analysés en détail dans ce chapitre.

L'étude approfondie de ces algorithmes nous a permis de proposer un nouvel algorithme qui est présenté dans le chapitre 3. Nous l'avons appelé SBAN, ce qui signifie Similarity-Based Adaptive Neighborhood (voisinage adaptatif basé sur la similarité de pixels). Afin de calculer la corrélation entre deux régions, l'algorithme n'utilise que les pixels du voisinage qui sont similaires au pixel dont on cherche le correspondant. Plusieurs critères de similarité entre pixels sont proposés et analysés dans ce chapitre. Cet algorithme peut également être implanté dans une architecture spécialisée afin de traiter les images en temps-réel. Une étude comparative du comportement des différents algorithmes retenus est présentée dans ce chapitre.

Dans le chapitre 4, nous décrivons l'architecture du processeur STREAM. Ce processeur utilise un composant FPGA reconfigurable disposant d'un grand nombre de ressources matérielles. Nous montrons comment les algorithmes présentés dans le chapitre précédent sont décomposés, suivant une hiérarchie qui repose uniquement sur quelques modules élémentaires, afin d'être implantés dans le processeur STREAM. Ces modules sont spécifiés au moyen d'un langage de description de composants de haut niveau. Nous terminons ce chapitre en comparant les ressources matérielles nécessaires à l'implantation des algorithmes présentés dans les chapitres 2 et 3.

Enfin, le dernier chapitre présente deux applications, visant à améliorer la sécurité dans les transports terrestres, dans lesquelles notre algorithme peut être avantageusement utilisé. Ces deux applications ayant fait l'objet de travaux antérieurs, notamment de plusieurs thèses de Doctorat, nous ne présenterons que les aspects directement liés à la stéréovision en temps-réel. La première application, développée dans le cadre d'un projet PREDIT 2 nommé STATUE, vise à détecter les obstacles présents à l'avant d'un véhicule guidé automatisé évoluant en site protégé. La deuxième application est très similaire, puisqu'il s'agit cette fois de détecter les obstacles présents sur la route à l'avant d'un véhicule automobile. Ces travaux ont été menés dans le cadre d'un projet financé par le Conseil Régional Nord-Pas de Calais, intitulé RaViOLi (Radar et Vision Orientables, Lidar)

Afin de conclure, dans la dernière partie de ce mémoire, nous résumons tous les aspects

abordés durant cette thèse. Nous présentons également plusieurs voies suivant lesquelles ces recherches peuvent, ou vont, être poursuivies.

Chapitre 1

La stéréovision

L'être humain dispose d'un système de vision puissant et complexe. Les images perçues par les yeux sont transmises au cerveau dans lequel elles sont traitées afin d'en extraire les informations pertinentes. L'une des informations que le cerveau peut reconstituer à partir des images est la distance à laquelle se trouvent les objets qui entourent l'individu. Celle-ci est calculée sur la base des différences existant entre les deux images de la scène, puisqu'elles sont obtenues par des yeux qui sont séparés d'une certaine distance.

Chez l'être humain, le temps de réponse du système visuel est court mais les distances sont évaluées de façon relativement imprécise. Cela nous permet néanmoins de nous repérer très efficacement dans l'espace, de nous déplacer rapidement dans un environnement comportant de nombreux obstacles, ou encore de conduire un véhicule à grande vitesse.

Les performances des systèmes de vision artificielle sont encore loin d'égaler celles des systèmes visuels naturels. Malgré tout, dans certaines applications comme la navigation autonome de robots mobiles, l'infographie ou la réalité virtuelle, la construction d'une représentation tridimensionnelle détaillée et précise de la scène est nécessaire. Des contraintes temps-réel sont également imposées par ces applications, ce qui implique le traitement d'une grande quantité d'informations à une cadence élevée.

De nombreux systèmes stéréoscopiques ont été développés par les chercheurs en vision artificielle, qui ont proposé des techniques de traitement d'images permettant de reconstruire l'information 3D de la scène observée. Comme dans le système visuel animal, deux capteurs d'images, cette fois des caméras, regardent la même scène depuis des positions légèrement différentes [Edelman et Weinshall 1989]. Les deux images sont cependant ac-

quises au même instant. La figure 1.1 présente la configuration classique d'un système stéréoscopique :

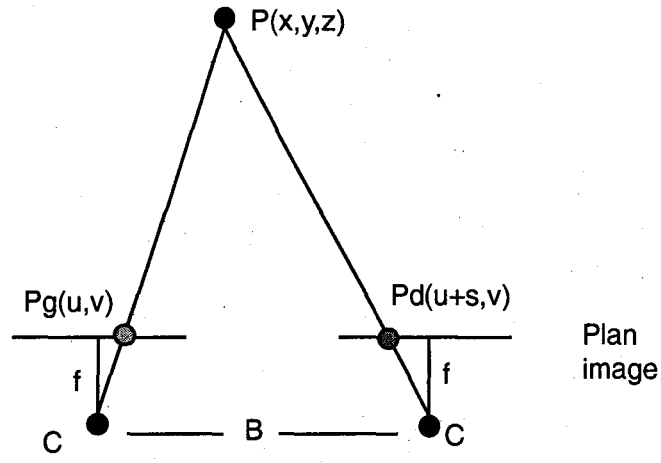


FIG. 1.1 : Un système stéréoscopique.

Le modèle sténopé est presque toujours utilisé pour décrire les caméras. Ce modèle fait intervenir un centre optique C, encore appelé centre de projection, et un plan image. La distance f séparant le centre de projection du plan image est appelée distance focale. La distance B séparant les centres optiques est appelée entraxe.

1.1 Calibrage d'un banc stéréo

Le calibrage d'un système stéréoscopique consiste à calculer les paramètres nécessaires pour définir la transformation des coordonnées d'un point dans l'espace 3D vers les coordonnées 2D de leurs projections dans les images [Heikkila et Silven 1997].

Les paramètres intrinsèques expriment les caractéristiques propres à chaque caméra du banc stéréo, indépendamment de la géométrie de l'association. Ils définissent la transformation des coordonnées d'un point dans l'espace vers le plan image d'une caméra. L'équation (1.1) définit la matrice des paramètres intrinsèques en coordonnées homo-

gènes :

$$M_{int} = \begin{vmatrix} \alpha_u & 0 & U_0 & 0 \\ 0 & \alpha_v & V_0 & 0 \\ 0 & 0 & 1 & 0 \end{vmatrix} . \quad (1.1)$$

dans laquelle α_u et α_v sont les dimensions des pixels sur le capteur, qui sont fournies par le constructeur de la caméra, et (U_0, V_0) sont les coordonnées du pixel situé au centre de l'image.

Les paramètres extrinsèques expriment la position et l'orientation des caméras dans l'espace une fois qu'elles ont été associées afin de constituer le banc stéréo. Ces paramètres sont également décrits par une matrice, présentée dans l'équation (1.2). Cette matrice correspond à la combinaison d'une matrice de rotation (R) et d'une matrice de translation (T).

$$M_{ext} = \begin{vmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{vmatrix} . \quad (1.2)$$

Plusieurs méthodes de calibrage ont été décrites dans la littérature [Zhang 2000, Tsai 1987, Heikkila et Silven 1997]. Elles utilisent un objet où une mire de géométrie connue [Devernay 1997], ou mettent en correspondance plusieurs points dans une séquence d'images [Faugeras *et coll.* 1992]. Pour le calibrage de notre système stéréoscopique, nous avons utilisé des outils standard disponibles sur internet [1].

1.2 Rectification des images

La figure 1.2 décrit la géométrie épipolaire d'un système stéréoscopique. L'épipole d'une caméra est la projection dans son plan image du centre optique de la deuxième caméra. La ligne épipolaire est le lieu des pixels d'une image pouvant correspondre à un pixel unique de l'autre image [Issa 2004]. Le pixel $P1$ est appelé conjugué ou homologue du pixel $P2$. La contrainte épipolaire impose que le conjugué $P2$ d'un pixel $P1$ appartienne à la ligne épipolaire correspondante dans la deuxième image.

[1] http ://www.vision.caltech.edu/bouguetj/calib doc/index.html

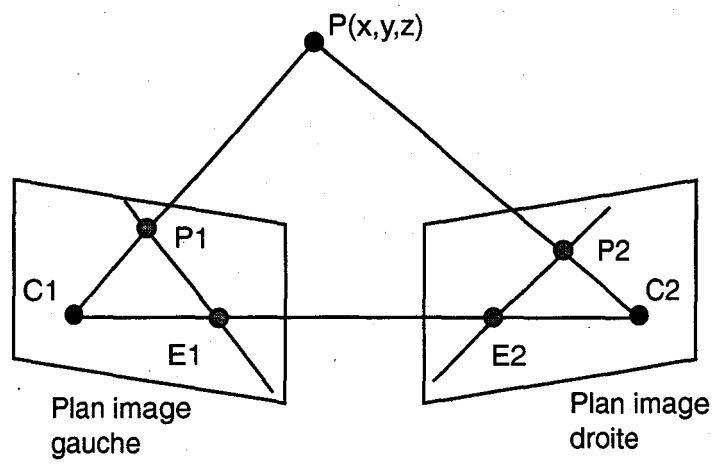


FIG. 1.2 : La géométrie épipolaire.

Les techniques de rectification visent à faire correspondre les droites épipolaires aux lignes des images. Pour ce faire, on re-projette les images dans un plan parallèle à la ligne reliant les centres optiques, de telle sorte que les lignes épipolaires soient parallèles et horizontales [Faugeras *et coll.* 1993]. Plusieurs méthodes de rectification des images ont été décrites dans la littérature [Fusiello *et coll.* 2000b, Hartley 1999].

Après rectification des images, la recherche du conjugué d'un pixel est simplifiée, puisque le conjugué ne doit être recherché que dans la ligne correspondante de l'autre image [Mellor *et coll.* 1996]. Cette simplification est d'autant plus appréciable lorsque la technique est implantée sur un processeur câblé, puisque les calculs peuvent alors être réalisés sur les lignes correspondant au balayage vidéo.

1.3 Mise en correspondance

Dans la suite de ce mémoire, nous supposons que les images ont été rectifiées. Dans ces conditions, un point $P(x, y, z)$ de l'espace 3D se projette dans les plans image gauche et droit sur les pixels de coordonnées $P_g(u, v)$ et $P_d(u + d, v)$ respectivement. On peut constater que les coordonnées horizontales des projections ne sont pas identiques, mais décalées d'une valeur d. Ce décalage est appelé *disparité*. La disparité dépend de la distance du point au système de caméras ; plus les points sont proches du système stéréoscopique, plus la disparité est grande et inversement.

Le problème de la mise en correspondance consiste donc à créer des couples de primitives — une pour chaque image — qui correspondent aux mêmes éléments dans la scène. Les primitives à apparier peuvent être des pixels, des contours ou des régions [Horaud et Monga 1995].

Plusieurs facteurs peuvent influer sur la complexité de la mise en correspondance des primitives. Les occultations, par exemple, sont des zones de l'image existant dans une image mais pas dans l'autre, soit parce que les zones sont cachées par un objet, soit parce qu'elles sont dans le champ de vue d'une caméra et hors du champ de vue de l'autre. Dans la plupart des algorithmes on ne tient pas compte des occultations dans la procédure de mise en correspondance. Une étape de post-traitement permet la détection des occultations, souvent basée sur le principe de la vérification directe-inverse [Fusiello *et coll.* 2000b], lequel sera décrit dans un chapitre postérieur. D'autres algorithmes intègrent la détection des occultations au moment de la mise en correspondance [Intille et Bobick 1994, Geiger *et coll.* 1995]. Les auteurs considèrent que les occultations apportent des informations complémentaires qui simplifient la procédure de mise en correspondance.

Une autre source d'erreur est la distorsion perspective. Une région inclinée n'a pas la même taille et la même forme dans les images à cause des différents points de vue. Les méthodes proposées dans [Devernay 1997, Borga et Knutsson 1998] tiennent compte de cette distorsion.

En connaissant le modèle de projection de chaque caméra et la relation spatiale entre les deux caméras on peut calculer par triangulation les coordonnées 3D d'un point à partir de ses projections dans les images [Horaud et Monga 1995]. La distance z d'un point au système de caméras peut être calculée à l'aide de l'équation (1.3) :

$$z = \frac{f \times B}{d} \tag{1.3}$$

où f est la distance focale, B l'entraxe et d la disparité.

Pour réduire la complexité des algorithmes de mise en correspondance, donc réduire le temps de calcul, plusieurs contraintes peuvent être utilisées :

- La contrainte Lambertienne : l'intensité de la projection dans chaque image d'un point 3D doit être indépendante du point de vue [Devernay 1997]. Cette contrainte n'est plus valable si les deux caméras ont des caractéristiques différentes ou si elles sont réglées différemment car les différences optiques créent des différences entre les

niveaux de gris des pixels correspondant à un même point de la scène [Lee *et coll.* 1999]. Plusieurs mesures permettent de réduire les effets des différences de réglage [Martin et Crowley 1995, Zabih et Woodfill 1994].

- La contrainte fronto-parallèle : la plupart des algorithmes de mise en correspondance supposent que les surfaces sont parallèles aux plans image des deux caméras. Cette contrainte n'est plus valable en présence de surfaces inclinées.
- La contrainte d'ordre : Si le pixel $P2$ est situé à droite du pixel $P1$ dans l'image gauche, son conjugué $P2'$ se trouve à droite du conjugué $P1'$ dans l'image droite. Les objets transparents violent cette contrainte [Marr et Poggio 1979, Intille et Bobick 1994]. Ainsi, la plupart des algorithmes de mise en correspondance supposent qu'il n'y a pas d'objet transparent dans la scène observée.
- La contrainte de continuité : comme les surfaces sont localement continues, deux points proches l'un de l'autre sont situés à des distances similaires des deux caméras. Cette contrainte n'est plus valable sur les bords des objets [Zitnick et Kanade 1999].
- La contrainte d'unicité : un pixel dans une image ne peut s'apparier qu'avec un seul pixel dans la deuxième image [Stefano *et coll.* 2002]. Quelques méthodes permettent d'apparier plusieurs pixels d'une image avec un seul pixel dans la deuxième image [Quénot 1996].
- La contrainte de disparité maximale : elle est imposée pour la configuration du système stéréoscopique et le cahier des charges. Cette contrainte définit la distance minimale qui peut être déterminée et, par conséquent, le décalage maximal à faire intervenir dans les calculs.

1.4 Classification des méthodes d'appariement

Les méthodes de stéréovision peuvent être classées en fonction des primitives qu'elles utilisent dans la procédure de mise en correspondance. Selon ce critère, on distingue deux catégories : les méthodes surfaciques et les méthodes à base de primitives.

1.4.1 Méthodes surfaciques

Les méthodes surfaciques exploitent la contrainte de continuité. Ces méthodes consistent à fixer d'abord un pixel d'intérêt dans une image, dite de référence, puis, à l'aide d'une mesure de corrélation de type distance euclidienne ou produit scalaire, à chercher un pixel dans l'autre image dont le voisinage est semblable à celui du pixel de référence [Devernay 1997]. Étant donné que tous les pixels visibles de l'image de référence sont appariés, une carte de disparité dense est obtenue. Un tel résultat est nécessaire dans les applications comme la synthèse d'images, dans laquelle on doit connaître la valeur de disparité dans toutes les régions de l'image, même dans les zones d'occultations et dans les zones homogènes [Scharstein et Szeliski 2002].

Comme le nombre de primitives à traiter et le nombre d'opérations par pixel sont élevés, le temps de calcul devient très important. Des techniques multi-résolution permettent de réduire le temps de calcul [Koschan *et coll.* 1996, Yuan et Subbarao 1998]. Des images à différentes résolutions sont créées. Les disparités calculées dans les images à plus basse résolution sont utilisées pour réduire l'espace de recherche dans les images à plus haute résolution. Cela diminue considérablement le temps de calcul. Un inconvénient de ces approches est que si deux pixels sont appariés de façon erronée dans les images de basse résolution, alors l'erreur sera propagée dans les appariements à plus haute résolution [Franke et Joos 2000].

Les méthodes surfaciques réalisent des calculs dits réguliers, c'est-à-dire une succession d'opérations identiques quel que soit le voisinage traité dans l'image. Elles ne tiennent compte que de l'information locale. En conséquence, les traitements peuvent être décomposés et implantés en parallèle dans le cas d'architectures de calcul spécialisées.

Les méthodes surfaciques sont sensibles aux images contenant des textures répétitives ou des zones avec un niveau de gris homogène. Dans les images qui contiennent des textures répétitives, la fonction de similarité présente plusieurs minimums locaux et il devient difficile de prendre une décision correcte. Dans les images contenant des zones avec un niveau de gris homogène, la fonction de similarité ne présente pas un pic significatif bien défini.

Ces méthodes peuvent être utilisées dans les scènes naturelles, dans lesquelles les zones homogènes et les textures répétitives ne sont pas nombreuses, et contenant des objets qui

possèdent des textures différentes.

1.4.2 Méthodes à base de primitives de haut niveau

La motivation des méthodes exploitant des primitives de haut niveau est de réduire le temps de calcul [Taraglio et Zanela 2000] et d'obtenir des correspondances fiables, même en présence d'une certaine quantité de bruit dans les images. Des primitives considérées de haut niveau, comme les points d'intérêt, les contours, les lignes ou encore les coins sont utilisées [Lhuillier 1998].

Étant donné que les primitives de haut niveau extraites des images sont moins nombreuses que les pixels, le temps de calcul est considérablement réduit [Franke et Joos 2000]. Cependant, les performances de ces méthodes dépendent du nombre de primitives présentes dans les images et elles fournissent des données 3D éparses [Devernay 1997]. La disparité dans les zones de l'image ne contenant pas de primitives peut être calculée par interpolation [Ohm *et coll.* 1998]. Le temps de calcul dépend du nombre de primitives présentes dans les images. Le problème du choix des primitives à utiliser reste difficile à résoudre, car chaque type de primitive de haut niveau ne se comporte correctement que dans certaines situations [Ishikawa 1999].

L'utilisation de telles primitives est principalement justifiée dans les images qui correspondent à des scènes d'intérieur, dans lesquelles on trouve de grandes zones homogènes et où les contours indiquent le passage d'un objet à un autre.

1.5 Techniques de mise en correspondance

Les techniques utilisées pour la mise en correspondance sont très variées. Dans la partie suivante, nous allons présenter les travaux publiés dans la littérature, en les regroupant uniquement en fonction de la technique employée pour l'appariement, sans tenir compte du type de primitive qu'elle exploite.

1.5.1 Mise en correspondance par algorithmes génétiques

Les algorithmes génétiques sont des techniques d'optimisation globale d'un problème fondées sur les mécanismes de la sélection naturelle et de la génétique [Gong et Yang 2002].

Les solutions d'un problème sont représentées sous forme de chromosomes, en utilisant un codage spécifique. Une fonction d'évaluation permet à une population de chromosomes d'évoluer vers la solution du problème. La fonction d'évaluation est définie de telle sorte que ses minimums correspondent à l'une des solutions du problème. L'espace des solutions est exploré grâce aux opérations de sélection et de reproduction. A partir d'une population initiale, l'algorithme évolue durant un certain nombre d'itérations et finalement le chromosome qui correspond à la valeur optimale de la fonction d'évaluation est sélectionné.

Dans [Saito et Mori 1995, Gong et Yang 2002], plusieurs cartes de disparité sont déterminées par une méthode de type corrélation, soit en utilisant plusieurs fenêtres de tailles différentes, soit plusieurs images à différents niveaux de résolution et une seule fenêtre de taille fixe. La fonction d'évaluation tient compte des contraintes de similarité et de continuité. Les algorithmes génétiques sont utilisés pour choisir la disparité qui optimise la fonction d'évaluation.

Dans [Issa *et coll.* 2003], une technique d'optimisation globale qui utilise les contours comme primitives à apparier a été proposée. Les auteurs utilisent une caméra linéaire, ce qui diminue fortement la quantité de données à traiter. Un nouveau codage, appelé codage entier, permet de mieux explorer l'espace des solutions et d'améliorer le temps de convergence. La fonction d'évaluation est construite à partir des contraintes d'unicité, d'ordre et de continuité.

Dans [Goulermas et Liatsis 2001], une technique pour apparier des segments de lignes a été proposée. La similarité entre les primitives est calculée en utilisant des opérateurs de la logique floue. Les algorithmes génétiques sont employés pour évaluer une fonction qui tient compte des similarités entre les contours dans une ligne et les contours dans les lignes voisines.

Pour ces méthodes, le temps de calcul dépend de la taille de la population initiale, donc du nombre de d'appariements testés à chaque itération. Avec une faible population, la convergence de l'algorithme peut être prématurée et le résultat ne correspond pas à la solution optimale. A l'inverse, une grande population augmente le temps nécessaire à la convergence.

1.5.2 Mise en correspondance par des méthodes fréquentielles

Dans les méthodes fréquentielles, les images sont considérées comme deux signaux continus décalés dans le temps. Une opération de convolution est appliquée à chaque image en utilisant des filtres de Gabor ou des filtres de quadrature et la disparité est calculée en fonction de la différence de phase entre les sorties des filtres.

L'avantage de ces méthodes est que la réponse obtenue est continue, ce qui permet d'obtenir des précisions de type subpixel directement après la convolution. Ces méthodes peuvent être implantées sur des architectures spécialisées pour atteindre des performances temps-réel [Porr *et coll.* 1998, Darabiha *et coll.* 2003]. Un algorithme proposé dans [Borga et Knutsson 1998] peut traiter des images à différentes échelles et tenir compte des objets inclinés. Une étude comparative [Cozzi *et coll.* 1997] montre que ces techniques sont robustes aux différences de gain dans les images, mais très sensibles au bruit.

Un des inconvénients de ces méthodes est que la réponse en fréquence reste limitée à une petite bande de fréquences du fait de la caractéristique périodique des oscillateurs. Ainsi, pour travailler avec de grandes valeurs de disparité on doit faire un traitement à différentes résolutions [Crespi *et coll.* 1998] ou utiliser les dérivées des images plutôt que les images elles mêmes [Ouali *et coll.* 1999]. Un autre inconvénient est que la réponse de ces méthodes est précise uniquement sur les bords des objets [van der Mark et Groen 2001].

1.5.3 Mise en correspondance par réseaux de neurones

Le problème de la mise en correspondance peut être envisagé comme un problème d'optimisation d'une fonction d'énergie [Boykov *et coll.* 1999]. Un réseau de neurones qui travaille en mode d'optimisation combinatoire est bien adapté pour trouver la solution de ce type de problème. On doit identifier la fonction de coût représentative du problème de mise en correspondance, qui est ensuite utilisée comme fonction d'énergie du réseau. Après initialisation, le réseau évolue vers son état stable, lequel correspond au minimum de la fonction de coût.

Dans [Dooze 2001] un réseau de neurones de Hopfield a été utilisé pour apparier des pixels de contours. Ce réseau minimise une fonction de coût basée sur les contraintes de signe du gradient, d'unicité, d'ordre et de continuité. Dans [Huang et Liu 1997], le réseau

de neurones est utilisé pour mettre en correspondance des pixels de contours. Ces derniers sont classés en 16 classes en fonction de leur connectivité locale avec leurs huit voisins. La fonction de coût tient compte du signe et de la direction du gradient.

Dans [Taraglio et Zanela 2000] des cellules de réseaux de neurones ont été utilisées. Ces dispositifs analogiques sont capables de traiter les signaux en temps-réel. Les contraintes de similarité et de continuité sont utilisées pour définir la fonction de coût et mettre en correspondance des régions.

Les principaux inconvénients des méthodes de minimisation d'une fonction d'énergie sont le temps de calcul élevé et le fait qu'elles convergent souvent vers le premier minimum local rencontré, lorsque que la fonction de coût présente plusieurs minimums [Boykov *et coll.* 2001].

1.5.4 Mise en correspondance par programmation dynamique

Dans les méthodes fondées sur la programmation dynamique, le problème de mise en correspondance est ramené à la détermination d'un parcours optimal reliant deux points dans l'espace des solutions. On cherche alors à trouver la meilleure trajectoire parmi toutes les trajectoires possibles en passant pour les différents noeuds d'un graphe [Geiger *et coll.* 1995, Gonzalez *et coll.* 1999]. Un coût est assigné à chaque noeud, par exemple, avec une méthode de type corrélation. Le coût global d'une trajectoire est la somme des coûts individuels des noeuds qui appartiennent à cette trajectoire [Brown *et coll.* 2003]. La meilleure trajectoire est celle qui minimise une fonction de coût globale.

Dans [Quénot 1996], les deux images sont coupées en bandes parallèles et chaque paire de bandes est appariée individuellement. Chaque bande est considérée comme une séquence de pixels et un coût local est calculé entre deux colonnes de deux bandes particulières. Les occultations sont autorisées en assignant un groupe de pixels d'une image à un seul pixel de la deuxième image. Une carte dense est obtenue par interpolation et lissage.

L'algorithme proposé dans [Lecoat *et coll.* 1997] initialise chaque élément de la matrice de programmation dynamique avec une mesure de similarité entre pixels issus des deux images. Ensuite, plusieurs trajectoires sont initialisées. Chaque trajectoire évolue indépendamment des autres, son évolution étant contrôlée par les données de la matrice.

Un élément de la matrice ne peut être utilisé que par une seule trajectoire. Le coût global associé à une trajectoire est la somme des coûts des éléments utilisés. Cet algorithme a été implanté dans une architecture dédiée qui utilise un circuit logique programmable [Pissaloux *et coll.* 2000; 1999].

Dans [Intille et Bobick 1994], les auteurs utilisent une structure de données qui contient toutes les mesures de similarité entre les pixels pour chaque valeur de disparité. La similarité des pixels est calculée sur neuf fenêtres de corrélation et seule celle qui minimise l'indice de corrélation est retenue. Un coût est assigné aux pixels considérés comme des occultations. La fonction de coût à minimiser tient compte des occultations et de la similarité des pixels.

Dans [Geiger *et coll.* 1995], les occultations sont utilisées pour réduire la complexité de la mise en correspondance. Les auteurs supposent qu'une discontinuité dans une image correspond à une occultation dans l'autre image [Ishikawa et Geiger 1998]. Cette considération permet la réduction du nombre des disparités à considérer et simplifie les calculs.

Un inconvénient de ces méthodes est qu'une erreur locale peut être propagée sur toute la ligne et corrompre d'autres appariements [Brown *et coll.* 2003]. Un autre inconvénient est le temps de calcul nécessaire pour mettre toutes les primitives en correspondance, qui est toujours très élevé hormis quand le traitement est réalisé par une structure dédiée.

1.5.5 Mise en correspondance par corrélation

Dans les méthodes de mise en correspondance par corrélation, une région rectangulaire autour du pixel à apparier est utilisée. Cette région est appelée fenêtre de corrélation. La fenêtre de corrélation est comparée à une fenêtre similaire dans l'image droite tout au long de la ligne droite épipolaire. Pour chaque déplacement de la fenêtre dans l'image droite, un indice de corrélation est calculé. Le déplacement qui minimise l'indice de corrélation est retenu comme disparité.

Étant donné que le nombre d'opérations est toujours le même pour chaque pixel, le temps de calcul peut être déterminé en fonction de la taille de la fenêtre de corrélation, du nombre de déplacements de la fenêtre et de la taille de l'image. Dans les autres techniques décrites précédemment, le nombre d'opérations nécessaires pour apparier deux primitives est variable et le temps de calcul ne peut pas être établi avec précision à l'avance.

Dans les architectures classiques, le temps de calcul pour les techniques basées sur la corrélation est grand. En contrepartie, étant donné que les indices de corrélation peuvent être calculés en parallèle, ces méthodes présentent l'avantage d'être implantables dans une architecture de calcul spécialisée pour atteindre des performances temps-réel.

1.6 Conclusion

Dans ces chapitre, nous avons décrit les algorithmes de mise en correspondance proposés dans la littérature. Certains fournissent d'excellents résultats mais leur complexité algorithmique est très élevée. Le temps de calcul étant directement lié à la complexité algorithmique, il n'est en général pas possible d'utiliser ces techniques dans les applications qui nécessitent le traitement des images à la cadence vidéo.

D'autres techniques reposent sur des calculs simples et réguliers effectués sur tous les pixels. Ces calculs peuvent être réalisés en parallèle à l'aide d'architectures spécialisées de calcul. Le traitement d'images à la cadence vidéo devient possible grâce à l'utilisation d'algorithmes adaptés et d'architectures spécialisées.

Dans cette thèse, nous nous sommes intéressés aux techniques qui peuvent être implantées sur la carte STREAM, qui est un processeur spécialisé de traitement de séquences d'images développé dans notre laboratoire. Grâce à leur simplicité algorithmique, les algorithmes de type corrélation ont été largement utilisés pour être implantés dans une architecture spécialisée de calcul. Ces techniques font l'objet d'un étude approfondie dans le chapitre suivant.

Chapitre 2

Mise en correspondance par corrélation

Nous avons listé dans le chapitre précédent les avantages et les inconvénients de chaque technique de mise en correspondance. Pour traiter les images en utilisant des architectures spécialisées de calcul, les algorithmes de mise en correspondance par corrélation sont bon candidats car ils font de calcul réguliers et ne tiennent compte que de l'information locale. Dans ce chapitre nous allons détailler ces techniques.

Dans les méthodes de mise en correspondance par corrélation, on définit tout d'abord une région rectangulaire attachée au pixel (x, y) à mettre en correspondance, qui a été sélectionné dans l'une des deux images, utilisée comme référence. Dans la figure 2.1, les images gauche et droite d'une paire stéréoscopique sont présentées, l'image gauche étant utilisée comme image de référence.

La région rectangulaire utilisée est appelée fenêtre de corrélation et le pixel à mettre en correspondance est habituellement placé au centre de cette fenêtre. La contrainte de continuité est utilisée implicitement, car on considère que tous les pixels qui appartiennent à la fenêtre de corrélation ont la même valeur de disparité.

Une fenêtre de corrélation de même dimension est déplacée dans l'autre image sur la droite épipolaire correspondante, c'est-à-dire sur une ligne horizontale puisque les images ont été rectifiées. Pour chaque position $x+s$ de la fenêtre dans l'image droite, un indice de corrélation $C_g(x, y, s)$ entre les deux fenêtres est calculé, l'indice l signifiant que l'image de gauche est prise comme référence. La figure 2.2 présente la fonction de corrélation obtenue après avoir calculé tous les indices de corrélation pour chaque déplacement de la fenêtre de corrélation le long de la droite épipolaire.

(a) Image gauche (b) Image droite

FIG. 2.1 : Couple stéréoscopique.

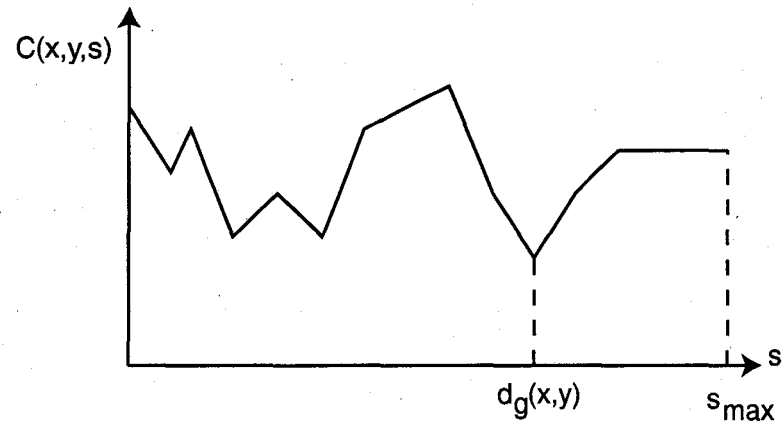


FIG. 2.2 : La fonction de corrélation.

Le minimum de la fonction de corrélation indique la meilleure correspondance entre les fenêtres. Le déplacement qui minimise la fonction de corrélation est retenu comme la valeur de disparité, notée d_g puisque l'image gauche est utilisée comme référence :

$$d_g(x,y) = arg_{min}(C_g(x,y,s)) \; . \tag{2.1}$$

Entre chaque calcul de corrélation, la fenêtre est décalée d'exactement un pixel. Cette technique fournit donc une disparité estimée au pixel près. Pour certaines applications, une précision plus élevée est nécessaire. Une valeur de disparité avec une précision subpixellique peut être déterminée en modélisant par une courbe continue, souvent une parabole, la fonction de corrélation au voisinage de son minimum [Stefano et Mattoccia 2002, Sun

2002]. Ensuite, le minimum de la fonction continue définit une nouvelle valeur estimée pour la disparité. Par exemple, la disparité estimée lorsque l'interpolation est réalisée par une parabole est donnée par l'équation :

$$d'_g = \frac{C_g(x,y,d_g+1) - C_g(x,y,d_g-1)}{2 * (C_g(x,y,d_g+1) + C_g(x,y,d_g-1) - C_g(x,y,d_g))} , \tag{2.2}$$

dans laquelle $C_g(x,y,d_g)$ désigne la valeur minimale de l'indice de corrélation et $C_g(x,y,d_g-1)$ et $C_g(x,y,d_g+1)$ sont les indices de corrélation calculés à gauche et à droite du décalage retenu comme la disparité.

2.1 Mesures de corrélation

L'indice de corrélation peut être calculé en utilisant différentes mesures [Faugeras *et coll.* 1993, Martin et Crowley 1995]. Par exemple, les équations (2.3) et (2.4) définissent respectivement les indices calculés par la somme des différences absolues et la somme des différences au carré entre les niveaux de gris des pixels des deux fenêtres :

$$C_g(x,y,s) = \sum_{i=-w_x}^{i=w_x} \sum_{j=-w_y}^{j=w_y} |I_g(x+i,y+j) - I_d(x+i+s,y+j)| , \tag{2.3}$$

$$C_g(x,y,s) = \sum_{i=-w_x}^{i=w_x} \sum_{j=-w_y}^{j=w_y} |I_g(x+i,y+j) - I_d(x+i+s,y+j)|^2 . \tag{2.4}$$

Dans ces équations, $I_g(x+i,y+j)$ et $I_d(x+i+s,y+j)$ sont les niveaux de gris des pixels $P_g(x+i,y+j)$ et $P_d(x+i+s,y+j)$ dans les images gauche et droite respectivement. Le décalage de la fenêtre de corrélation dans l'image droite est désigné par s, qui prend des valeurs comprises entre 0 et s_{max}. s_{max} est le déplacement maximal de la fenêtre de corrélation, qui détermine la distance minimale qui peut être estimée par la méthode. La fenêtre de corrélation est un rectangle de dimensions $(2w_x+1) \times (2w_y+1)$.

Ces deux mesures sont les plus utilisées car elles sont très simples à calculer et à implanter dans les architectures spécialisées de calcul [Scharstein et Szeliski 2002]. Par contre, elles sont très sensibles aux différences d'illumination entre les deux images [Konolige 1997]. Les variations de gain entre les caméras créent également des différences de niveau de gris entre les pixels homologues, ce qui entraîne de mauvais appariements. Certaines mesures minimisent les erreurs induites par la différence de gain entre les caméras,

comme la corrélation normalisée [Brown *et coll.* 2003] :

$$C(x,y,s) = \frac{\sum_{i=-w_x}^{i=w_x}\sum_{j=-w_y}^{j=w_y}(I_g(x+i,y+j)-\bar{I}_g)\times(I_d(x+i+s,y+j)-\bar{I}_d)}{\sqrt{\sum_{i=-w_x}^{i=w_x}\sum_{j=-w_y}^{j=w_y}(I_g(x+i,y+j)-\bar{I}_g)^2\times(I_d(x+i+s,y+j)-\bar{I}_d)^2}}, \tag{2.5}$$

où $\bar{I}_g$ et $\bar{I}_d$ sont les valeurs moyennes calculées respectivement dans les fenêtres gauche et droite.

2.1.1 Mesures non paramétriques

Quelques mesures sont insensibles aux différences de gain des caméras, telles les mesures basées sur le gradient [Scharstein 1994] et les mesures non paramétriques [Zabih et Woodfill 1994, Schreer *et coll.* 2001]. Les mesures non paramétriques utilisent la relation entre les données plutôt que les données elles-mêmes.

Une mesure non paramétrique souvent utilisée est la transformée Census. Elle indique la relation existant entre chaque pixel et son voisinage à l'aide d'un vecteur à composantes binaires comme le montre la figure 2.3. Pour calculer cette transformée, le niveau de gris du pixel central est comparé avec les niveaux de gris de ses voisins. Une composante du vecteur vaut 0 quand le niveau de gris d'un pixel est inférieur au niveau de gris du pixel central et 1 dans le cas contraire.

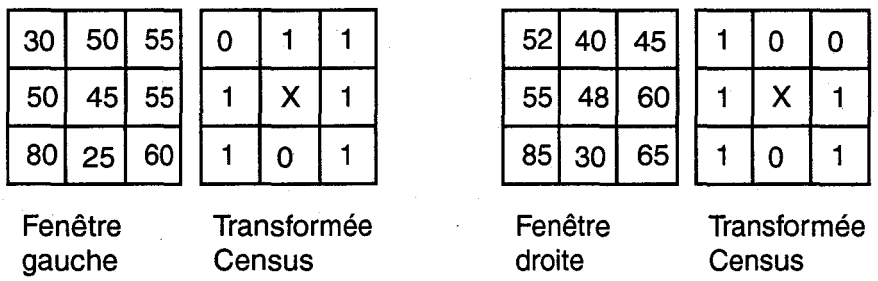


FIG. 2.3 : La transformée Census.

La transformée Census est créée à partir des valeurs binaires contenues dans la fenêtre, lesquelles son lues ligne par ligne de gauche à droite. Dans cet exemple, la transformée

Census de la fenêtre de l'image gauche vaut 01111101 tandis que la transformée Census de la fenêtre de l'image droite vaut 10011101. La similarité entre les deux vecteurs est mesurée en déterminant le nombre de composantes différentes dans les deux vecteurs, c'est-à-dire la distance de Hamming qui les sépare [Woodfill et Herzen 1997]. Dans notre exemple, la distance de Hamming entre les deux vecteurs vaut 3. Nous allons décrire cet algorithme de façon plus précise dans le chapitre suivant de ce mémoire.

Étant donné que dans la procédure de transformation de la luminance d'un pixel en une valeur binaire il y a une perte importante d'information, une modification de la transformée Census a été proposée dans [Lan et Mohr 1995]. Dans cette nouvelle transformée, chaque composante du vecteur n'est plus une valeur binaire, mais une valeur fractionnaire déterminée en analysant l'histogramme des niveaux de gris dans la fenêtre. Les auteurs décrivent une amélioration des performances, surtout en présence d'occultations.

2.1.2 Utilisation de la couleur

Le niveau de gris est l'unique information caractéristique d'un pixel dans une image monochrome. En revanche, dans une image couleur chaque pixel est caractérisé par plusieurs composantes, en général trois : le rouge, le vert et le bleu. La couleur du pixel fournit ainsi des informations additionnelles qui permettent d'améliorer la procédure de mise en correspondance.

Dans [Koschan *et coll.* 1996, Yuan et Subbarao 1998], une texture particulière est projetée sur la scène afin de la structurer, puis l'indice de corrélation est calculé à l'aide des trois composantes couleur :

$$\begin{aligned} C(x,y,s) \;=\; & \sum_{i=-w_x}^{i=w_x} \sum_{j=-w_y}^{j=w_y} ((R_g(x+i,y+j) - R_d(x+i+s,y+j)^2 + \\ & (V_g(x+i,y+j) - V_d(x+i+s,y+j))^2 + \\ & (B_g(x+i,y+j) - B_d(x+i+s,y+j))^2)), \end{aligned} \tag{2.6}$$

où $R_g(x+i,y+j)$, $R_d(x+i+s,y+j)$, $V_g(x+i,y+j)$, $V_d(x+i+s,y+j)$, $B_g(x+i,y+j)$ et $B_d(x+i+s,y+j)$ sont les composantes rouge, verte et bleue des pixels dans les images gauche et droite respectivement. Des améliorations par rapport aux techniques n'utilisant que le niveau de gris ont été rapportées par les auteurs, surtout dans les zones

homogènes. Ces améliorations sont dues principalement à la projection d'une texture sur la scène, mais pas nécessairement à l'utilisation de la couleur.

D'autres espaces de représentation des couleurs peuvent également être utilisés. Dans [Devernay 1997], une correction gamma de l'espace RVB a été utilisée, tandis que dans [Matsumoto *et coll.* 1997], l'espace YCbCr a été utilisé.

Malheureusement, les résultats présentés dans la littérature montrent que l'utilisation de la couleur n'apporte pas d'amélioration sensible par rapport à la mise en correspondance utilisant des images en niveau de gris. La cause principale est la forte corrélation qui existe entre les canaux de chrominance et de luminance, qui correspond au niveau de gris [Devernay 1997]. L'autre inconvénient est que le nombre de calculs à réaliser est plus important que dans le cas des images en niveau de gris, ce qui entraîne une augmentation du temps de traitement [Yuan et Subbarao 1998].

2.2 Choix de la taille de la fenêtre de corrélation

Les méthodes de mise en correspondance par corrélation utilisent une fenêtre de corrélation rectangulaire de taille fixe. Cela simplifie l'implantation logicielle et matérielle de l'algorithme. Elles supposent que tous les pixels qui appartiennent à la fenêtre de corrélation ont la même valeur de disparité, contrainte de continuité qui n'est pas vérifiée par tous les voisinages.

À proximité des discontinuités, qui correspondent aux bords des objets dans la scène donc aux contours dans l'image, tous les pixels de la fenêtre de corrélation ne correspondent pas obligatoirement au même objet. Dans ce cas, on a intérêt à réduire la taille de la fenêtre de corrélation pour respecter la contrainte de continuité. En revanche, dans les zones homogènes de l'image ou celles qui contiennent des textures répétitives, une petite fenêtre ne contient pas suffisamment d'information pour faire apparaître des maximums de corrélation.

C'est la raison pour laquelle l'utilisation d'une fenêtre de taille fixe, ou d'une fenêtre unique, ne donne pas de bons résultats dans toutes les zones de l'image. Ainsi, l'un des problèmes à résoudre avec les méthodes de mise en correspondance par corrélation est de déterminer la taille et la forme optimale de la fenêtre de corrélation.

2.2.1 Méthodes à fenêtres adaptables

L'utilisation d'une fenêtre adaptative semble être une bonne option proposée par plusieurs auteurs. Dans [Kanade et Okutomi 1991], la fenêtre est adaptée pour chaque pixel de l'image par rapport à la variation locale du niveau de gris ou de la disparité elle-même. Pour réduire la complexité du calcul, la forme de la fenêtre reste rectangulaire. Cependant, le résultat dépend d'une valeur de disparité calculée initialement et le temps de traitement est prohibitif. Dans [Scherer *et coll.* 1998], une modification a été proposée qui est fondée sur l'utilisation de la contrainte de similarité. Ainsi, les changements de disparité dépendent des variations locales du niveau de gris. En tenant compte de cette contrainte, le temps de calcul diminue sensiblement.

Dans [Lotti et Giraudon 1994], les contours sont d'abord détectés dans les images avec un filtre Canny-Deriche. La taille de la fenêtre est déterminée par rapport aux contours verticaux et horizontaux détectés dans l'image de référence. La même procédure est appliquée dans la deuxième image. Comme les deux fenêtres peuvent être de tailles différentes à cause des occultations et de la différence de perspective, la fenêtre la plus petite est utilisée pour calculer l'indice de corrélation. Cependant, la forme de la fenêtre est toujours rectangulaire.

Dans [Veksler 2002], une méthode qui utilise des fenêtres non rectangulaires a été proposée. La taille et la forme de la fenêtre de corrélation à utiliser sont sélectionnées en utilisant une technique de minimisation d'énergie basée sur la recherche d'une coupe minimale dans un graphe.

Certains travaux proposent de segmenter d'abord les images et d'utiliser les régions issues de la segmentation en tant que fenêtres de corrélation. On considère alors que tous les pixels qui appartiennent à une région ont la même disparité. Dans [Tao *et coll.* 2001], les images couleur sont segmentées et la même disparité est assignée à tous les pixels qui appartiennent à un segment donné. Dans [Moravec *et coll.* 1998], des opérateurs d'analyse de connexité sont utilisés pour trouver les zones homogènes dans l'image. La taille et la forme de la fenêtre sont déterminées selon la zone connexe déterminée dans l'image segmentée.

Ces travaux montrent l'importance de l'utilisation d'une fenêtre adaptative pour améliorer la mise en correspondance. La taille et la forme de la fenêtre sont parfois obtenues

par des méthodes itératives, ce qui alourdit les calculs et interdit un traitement en temps-réel des images. D'autre part, ces algorithmes ne peuvent pas être directement implantés sur une architecture spécialisée de calcul.

2.2.2 Utilisation de fenêtres multiples

Déterminer la taille et la forme idéales de la fenêtre de corrélation peut nécessiter beaucoup de temps de calcul. L'utilisation d'un nombre réduit de petites fenêtres, réparties à différentes positions dans le voisinage du pixel traité, permet d'aborder ce problème sous un angle différent.

L'algorithme SMW proposé dans [Fusiello *et coll.* 2000a], utilise neuf fenêtres de corrélation. Le pixel à mettre en correspondance est placé à différentes positions dans la fenêtre de corrélation, définissant ainsi plusieurs configurations pour l'analyse comme l'indique la figure 2.4.

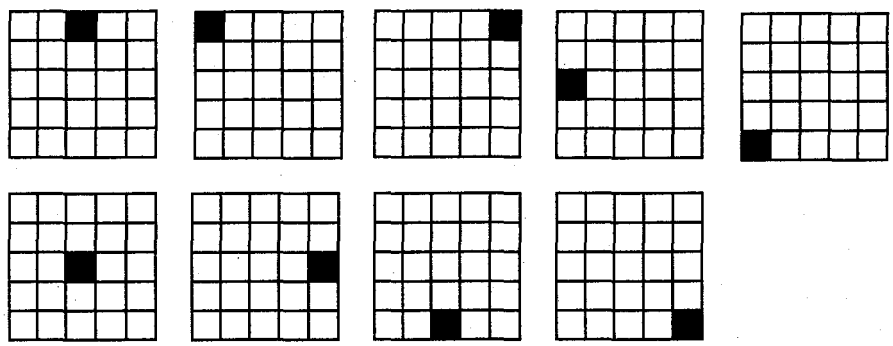


FIG. 2.4 : Les fenêtres de corrélation utilisées dans l'algorithme SMW.

Un indice de corrélation est calculé sur chacune des neuf fenêtres, puis celle qui minimise l'indice de corrélation est retenue. Les pixels qui appartiennent à la fenêtre retenue sont sensés avoir une disparité constante. Une mesure de confiance a également été proposée par les auteurs, qui fait intervenir la variance des valeurs de disparité calculées sur les neuf fenêtres de corrélation.

Dans [Hirschmuller 2001], la mise en correspondance est réalisée avec une fenêtre centrale entourée de plusieurs autres fenêtres, qualifiées de fenêtres supports. Trois configura-

tions avec un nombre différent de fenêtres supports ont été proposées. Ces configurations sont présentées dans la figure 2.5.

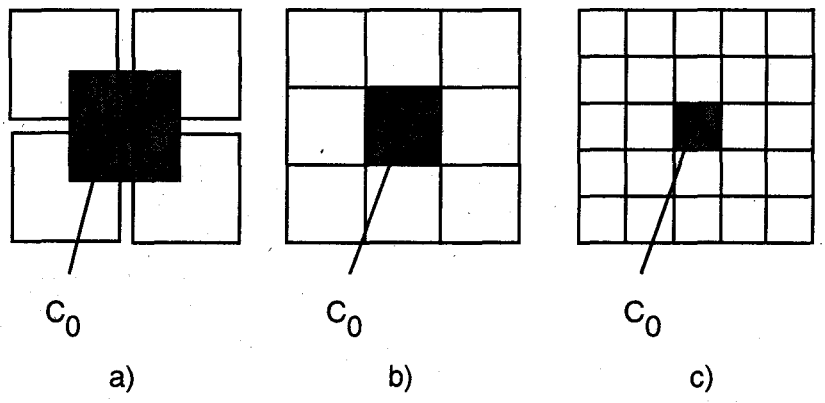


FIG. 2.5 : Configurations avec plusieurs fenêtres dans l'algorithme de Hirschmüller .

Des indices de corrélation sont calculés séparément pour différentes combinaisons des fenêtres supports, puis la combinaison qui minimise l'indice de corrélation global est retenue. Dans cet algorithme, la fenêtre centrale est toujours utilisée. Par exemple, pour la première configuration qui comporte quatre fenêtres supports, les deux plus faibles valeurs C_{i1} et C_{i2} de l'indice de corrélation sont retenues et ajoutées à l'indice de corrélation calculé en utilisant la fenêtre centrale C_0 :

$$C = C_0 + C_{i1} + C_{i2}. \tag{2.7}$$

Dans les autres configurations, qui utilisent un nombre plus élevé de fenêtres supports, quatre et huit valeurs sont retenues respectivement pour être additionnées à la valeur calculée pour le centre.

Ces algorithmes peuvent être implantés dans une architecture spécialisée, comme nous le verrons par la suite. Par contre, le nombre restreint de fenêtres supports utilisées n'est pas suffisant pour traiter correctement certains voisinages complexes.

2.3 Techniques de validation des appariements

Le minimum de la fonction de corrélation n'indique pas toujours la disparité correcte. Les différences de gain dans les caméras, par exemple, créent des différences des niveaux de gris qui peuvent induire des erreurs dans le calcul de la disparité. La présence de zones homogènes ou de textures répétitives dans l'image peuvent également entraîner de mauvais appariements. Cela montre qu'une mesure de confiance doit être établie afin d'indiquer la pertinence de chaque appariement.

2.3.1 Degré de pertinence

En présence de zones homogènes dans les images, la fonction de corrélation ne présente pas de minimum marqué. D'autre part, en présence de textures répétitives, elle présente plusieurs minimums locaux avec des valeurs très proches les unes des autres. Dans ces deux situations, il devient difficile de déterminer la valeur correcte de la disparité.

Soit $C(x, y, s_1)$ l'indice de corrélation calculé avec le déplacement s_1 et $C(x, y, s_2)$ l'indice de corrélation calculé avec le déplacement s_2. On étudie plus particulièrement le cas où s_1 et s_2 définissent des minimums locaux de la fonction de corrélation, comme le décrit la figure 2.6. On suppose que $C(x, y, s_1)$ est inférieur à $C(x, y, s_2)$, donc que la valeur de disparité sélectionnée est $d(x, y) = s_1$.

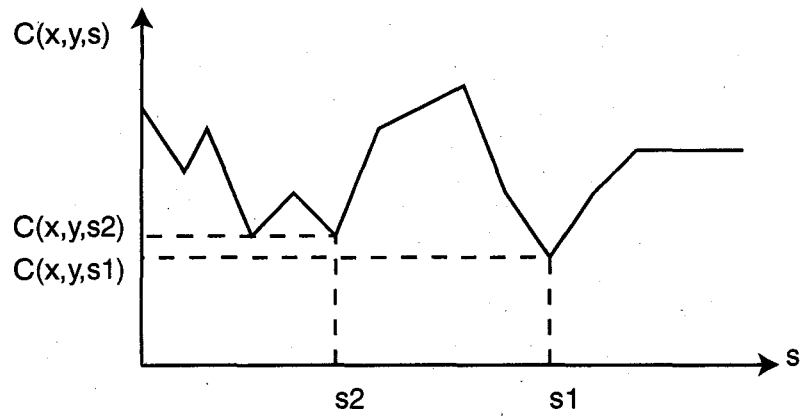


FIG. 2.6 : La fonction de corrélation en présence de plusieurs minimums.

Le degré de pertinence de l'appariement, donc de la disparité estimée, peut être calculé

à l'aide de l'équation :

$$I_c(x,y) = \frac{C(x,y,s2) - C(x,y,s1)}{C(x,y,s2)} . \tag{2.8}$$

Une faible valeur de $I_c(x,y)$ indique que l'appariement n'est pas très fiable, puisqu'une autre valeur du décalage fait également apparaître un minimum significatif dans la fonction de corrélation. Il est possible de considérer que l'appariement est erroné quand la valeur de $I_c(x,y)$ est inférieure à un seuil fixé *a priori* de façon empirique [Hirschmuller 2001].

Une autre critère de pertinence de l'appariement peut être défini en analysant la forme du pic, ou de la vallée, dans la courbe représentative de la fonction de corrélation [Faugeras *et coll.* 1993]. Dans les zones homogènes, le maximum ou le minimum est moins marqué, donc la courbe présente un pic ou une vallée large. En revanche, dans les zones texturées, le pic ou la vallée est plus étroit. Ainsi, on accorde une meilleure confiance à un extremum défini par un pic ou une vallée étroit qu'à un extremum défini de façon plus imprécise sur une courbe aplatie.

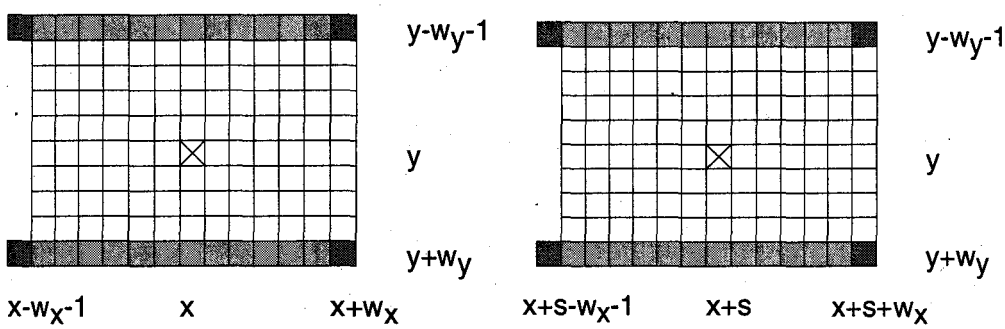
2.3.2 Validation directe-inverse

Dans la phase de mise en correspondance, l'une des images est utilisée comme image de référence, l'image gauche par exemple. Chaque pixel de cette image de référence est apparié avec un pixel de l'autre image afin de définir la disparité qui lui est associée. La validation directe-inverse [Faugeras *et coll.* 1993] consiste à répéter deux fois cette procédure, en choisissant d'abord une image, puis l'autre comme référence. Deux cartes de disparité sont alors obtenues, une pour chaque image de référence.

Soient P_g et P_d deux points homologues, respectivement de l'image gauche et de l'image droite. L'appariement est validé uniquement si la disparité calculée pour P_g en utilisant l'image gauche comme référence, est égale à la disparité calculée pour P_d en utilisant l'image droite comme référence. Cette condition est qualifiée de vérification directe-inverse. La vérification directe-inverse permet l'élimination des faux appariements et la détection des occultations. En effet, si la condition directe-inverse n'est pas validée, on considère qu'il y a une occultation et une disparité minimale est affectée aux pixels [Fusiello *et coll.* 2000a].

2.4 Calcul efficace des indices de corrélation

Afin de diminuer de façon notable le temps de calcul, principalement pour les grandes fenêtres de corrélation, les indices de corrélation peuvent être calculés en utilisant une technique récursive [Faugeras *et coll.* 1993]. Dans les expressions des coefficients $C_g(x, y, s)$ et $C_g(x, y-1, s)$, définissant les coefficients de corrélation pour deux pixels situés à la même position horizontale sur deux lignes successives, de nombreux termes sont communs. Sur la figure 2.7, nous avons représenté les pixels qui sont utilisés pour calculer ces deux valeurs. Les pixels marqués en blanc interviennent dans les deux expressions. Par contre, $C_g(x, y-1, s)$ intègre les pixels grisés situés en haut de la fenêtre, alors que $C_g(x, y, s)$ intègre les pixels grisés situés en bas de cette dernière.



(a) Fenêtre gauche (b) Fenêtre droite

FIG. 2.7 : Simplification du calcul de l'indice de corrélation.

Ainsi, l'indice de corrélation $C_g(x, y, s)$ peut être obtenu à partir de l'indice de corrélation du pixel $C_g(x, y-1, s)$ de la ligne précédente, en utilisant l'équation de récurrence :

$$C_g(x, y, s) = C_g(x, y-1, s) + U_g(x, y, s) \,, \tag{2.9}$$

où $U_g(x, y, s)$ est un coefficient d'actualisation qui inclut uniquement les termes ajoutés pour les pixels de la ligne $y + w_y$ et retirés pour les pixels de la ligne $y - 1 - w_y$.

Le coefficient d'actualisation $U_g(x, y, s)$ peut également être calculé à partir du coefficient d'actualisation $U_g(x-1, y, s)$ du pixel situé à sa gauche sur la même ligne. En effet, dans ce coefficient, on ajoute ou retranche uniquement les termes calculés pour les pixels situés aux coins de la fenêtre, qui sont marqués en gris plus foncé sur la figure 2.7. Ainsi,

le coefficient $U_g(x, y, s)$ peut être obtenu en utilisant une deuxième équation récurrente :

$$\begin{aligned} U_g(x,y,s) &= U_g(x-1,y,s) \\ &+|I_g(x+w_x, y+w_y) - I_d(x+s+w_x, y+w_y)| \\ &-|I_g(x-1-w_x, y+w_y) - I_d(x-1+s-w_x, y+w_y)| \\ &-|I_g(x+w_x, y-1-w_y) - I_d(x-1+s+w_x, y-1-w_y)| \\ &+|I_g(x-1-w_x, y-1-w_y) - I_d(x-1+s-w_x, y-1-w_y| \ . \end{aligned} \tag{2.10}$$

Avec cette technique, l'indice de corrélation peut être déterminé pour le pixel (x, y) et le décalage s en utilisant uniquement 5 opérations élémentaires, plutôt que $(2w_x + 1) \times (2w_y + 1)$ dans le calcul direct. De cette façon, le temps de calcul dépend uniquement de la taille des images et non plus de la taille de la fenêtre de corrélation.

Les indices de corrélation permettant la vérification directe-inverse peuvent également être obtenus de manière efficace. On peut vérifier aisément que les pixels qui interviennent dans le calcul de l'indice de corrélation $C_d(x, y, s)$, quand l'image droite sert de référence, sont les mêmes que ceux utilisés pour calculer l'indice de corrélation $C_g(x-s, y, s)$, quand l'image gauche sert de référence. Plus précisément :

$$C_d(x, y, s) = C_g(x - s, y, s) \ . \tag{2.11}$$

Cette considération permet de diviser le temps de calcul par deux dans la procédure de vérification directe-inverse.

On peut constater également que les données utilisées pour calculer chaque indice de corrélation sont complètement indépendantes les unes des autres. Ainsi, ils peuvent être calculés en parallèle à l'aide d'une architecture de calcul spécialisée. Cette caractéristique ne peut pas être exploitée dans les architectures de calcul séquentielles classiques car chaque indice de corrélation doit être évalué l'un après l'autre en utilisant des imbrications.

2.5 Conclusion

Dans ce chapitre, nous avons présenté de façon détaillée les algorithmes de stéréovision dense par corrélation. Ils présentent plusieurs avantages :

- Ils utilisent des critères simples pour apparier les primitives ;

- Les indices de corrélation peuvent être calculés efficacement et en parallèle ;
- La modification de l'algorithme pour augmenter ou diminuer le nombre de décalages a prendre en compte est simple.

Ainsi, ces algorithmes sont les mieux adaptés à une implantation sur une architecture de calcul spécialisée.

Cependant, la performance de l'algorithme dépend de la forme et la taille de la fenêtre de corrélation utilisée. Il est difficile de trouver la forme et la taille de la fenêtre qui donne à la fois un bon résultat dans les zones homogènes et au voisinage des discontinuités.

Les méthodes utilisant des fenêtres adaptatives déjà décrites dans la littérature sont très gourmandes en temps de calcul. Étant donné que les traitements changent en fonction des données, leur structure algorithmique n'est pas régulière. Elles restent donc difficiles à implanter dans les architectures spécialisées de calcul.

Les algorithmes qui utilisent plusieurs fenêtres peuvent être implantés dans une architecture dédiée. Cependant, le nombre limité de fenêtres utilisées ne suffit pas pour résoudre tous les cas difficiles apparaissant dans des images réelles. Ainsi, reste à trouver une méthode adaptative qui utilise une grande fenêtre dans les zones homogènes, mais uniquement les pixels qui ont la même disparité au voisinage des discontinuités. De plus, il faut que cette méthode soit implantable sur une architecture spécialisée.

Dans le chapitre suivant, nous décrivons un nouvel algorithme de mise en correspondance par corrélation. Il élimine certains pixels de la fenêtre de corrélation, ce qui est équivalent à modifier la taille et la forme de cette fenêtre. D'autre part, la structure régulière des calculs permet d'envisager son implantation sur une architecture spécialisée de traitement d'images.

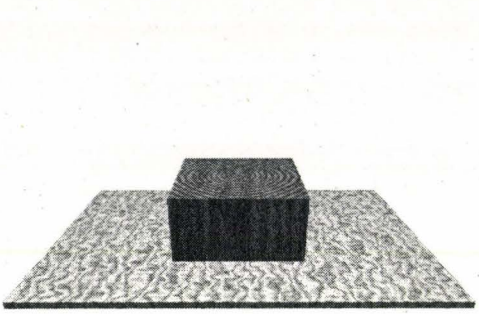
Chapitre 3

Mise en correspondance par voisinage adaptatif basé sur la similarité de pixels

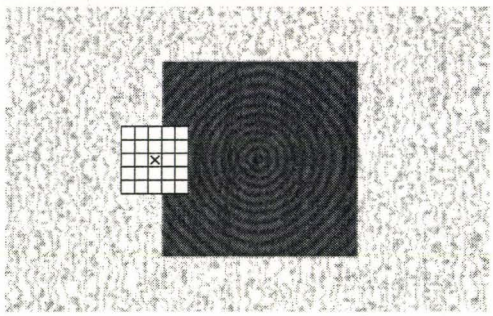
Les algorithmes de mise en correspondance par corrélation supposent que tous les pixels appartenant à la fenêtre de corrélation sont les projections dans l'image de différents points d'un même objet. Ils supposent également que tous ces pixels ont la même valeur de disparité. Ces conditions ne sont plus vérifiées au voisinage des discontinuités, ce qui entraîne des erreurs d'appariement pour les pixels situés dans ces zones de l'image.

Pour illustrer cet inconvénient, nous allons utiliser la figure 3.1.(a), laquelle présente deux objets situés à des distances différentes de la caméra. La figure 3.1.(b) met en évidence un pixel placé à proximité d'une discontinuité et la fenêtre de corrélation correspondante. Une méthode classique, qui utilise tous les pixels de la fenêtre de corrélation (cf. figure 3.1.(c)), risque de conduire à un mauvais appariement. En effet, la fenêtre de corrélation contient des pixels qui sont les projections de points appartenant aux deux objets, donc situés à des distances différentes. En revanche, un algorithme adaptatif n'utilisera que les pixels qui sont les projections d'un seul des deux objets, auxquels on peut donc affecter une disparité unique, comme le montre la figure 3.1.(d).

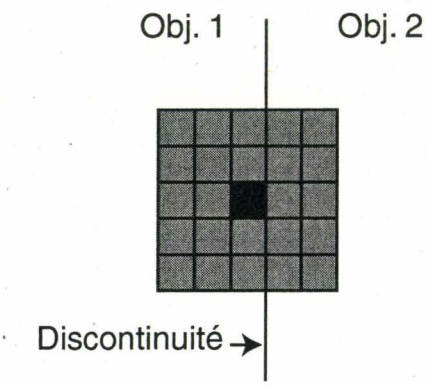
En nous appuyant sur cette observation, nous proposons une nouvelle méthode appelée SBAN (Similarity-Based Adaptive Neighborhood : voisinage adaptatif basé sur la similarité de pixels). Dans cette méthode, une fenêtre de corrélation rectangulaire est initialement définie [Perez *et coll.* 2004b]. En revanche, seulement quelques pixels sélectionnés dans cette fenêtre interviennent dans le calcul de l'indice de corrélation.



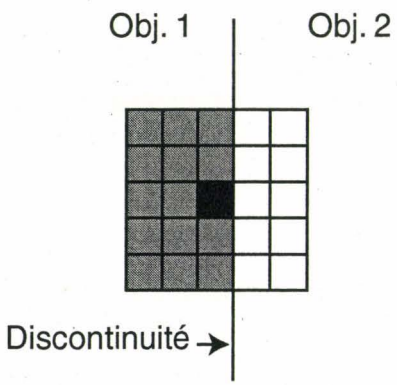
a) Image de synthèse



b) Image gauche



c) Fenêtre fixe



d) Fenêtre adaptative

FIG. 3.1 : Fenêtre fixe ou fenêtre adaptative ?.

Toutes les mesures standard de corrélation peuvent être modifiées selon ce principe. Par exemple, la somme des différences absolues (SDA) devient :

$$C_g(x,y,d) = \sum_{i=-w_x}^{i=w_x} \sum_{j=-w_y}^{j=w_y} |I_g(x+i,y+j) - I_d(x+i+s,y+j)| \cdot \beta_g(x,y,i,j) \ , \qquad (3.1)$$

où $\beta_g(x,y,i,j)$ est fixé à la valeur 1 si les pixels P(x+i,y+j) et P(x+i+s,y+j), avec des niveaux de gris $I_g(x+i,y+j)$ et $I_d(x+i+s,y+j)$ respectivement, sont retenus, et à la valeur 0 si les pixels sont supprimés de la fenêtre initiale.

Le coefficient $\beta_g(x,y,i,j)$ peut être déterminé en fonction de la seule image de référence ou en fonction des deux images. Dans les équations utilisées dans la suite de ce mémoire, pour des raisons de simplicité d'écriture, nous considérons que l'indice affecté au coefficient β décrit le fait qu'il est utilisé dans le calcul de la corrélation faisant intervenir l'image gauche comme référence, et non pour indiquer qu'il est déterminé en fonction du seul contenu de l'image gauche.

Seuls les pixels qui correspondent au même objet doivent être utilisés pour calculer les indices de corrélation. Ainsi, seuls les pixels similaires au pixel central seront retenus, alors que les pixels non similaires seront éliminés de la fenêtre de corrélation. Plusieurs critères de similarité peuvent être utilisés. Par exemple, on peut considérer que deux pixels sont similaires si leurs niveaux de gris sont égaux, ou relativement proches. D'autres attributs liés à la texture ou à la couleur, peuvent naturellement être utilisés.

Supprimer des pixels de la fenêtre initiale est équivalent à changer sa taille et sa forme. Ainsi, le voisinage utilisé pour calculer l'indice de corrélation pour un pixel devient un voisinage qui s'adapte aux formes détectées localement dans l'image, plutôt qu'une simple fenêtre rectangulaire.

La procédure de sélection des pixels est indépendante de la procédure de mise en correspondance et elles peuvent être réalisées en parallèle. Ainsi, cette technique peut être implantée dans une architecture spécialisée de calcul.

Comme dans toutes les méthodes de mise en correspondance par corrélation, le déplacement qui minimise l'indice de corrélation est retenu comme étant la disparité recherchée.

3.1 Critères de similarité

De nombreux critères de similarité peuvent être utilisés pour déterminer quels sont les pixels qui doivent intervenir dans le calcul de l'indice de corrélation. Nous avons testé le comportement de plusieurs de ces critères.

3.1.1 Similarité du niveau de gris des pixels

Souvent, des pixels voisins dont les niveaux de gris sont proches correspondent à un même objet dans la scène et, par conséquent, ont la même disparité [Zhang et Kambhamettu 2002, Cohen *et coll.* 1989, Tao *et coll.* 2001, Koch 1993, Scherer *et coll.* 1998]. Selon ce critère, les pixels qui doivent être retenus dans le calcul de l'indice de corrélation sont ceux qui ont un niveau de gris proche de celui du pixel à mettre en correspondance, qui est situé au centre de la fenêtre rectangulaire initiale. Plus précisément, on calcule le

paramètre $\beta_g(x, y, i, j)$ par l'équation :

$$\beta_g(x, y, i, j) = \begin{cases} 1 & \text{si } |I_g(x+i, y+j) - I_g(x, y)| < T_g(x, y) \\ 0 & \text{sinon} \end{cases},$$

où $T_g(x, y)$ est un seuil utilisé pour indiquer l'écart maximal autorisé entre le niveau de gris d'un pixel appartenant à la fenêtre de corrélation $I_g(x+i, y+j)$ et celui du pixel central $I_g(x, y)$. Le seuil peut être déterminé en fonction des variations du niveau de gris dans la fenêtre de corrélation de l'image de référence, par exemple en utilisant la moyenne des différences absolues (MDA) :

$$T_g(x, y) = \frac{\sum_{i=-w_x}^{i=w_x} \sum_{j=-w_y}^{j=w_y} |I_g(x+i, y+j) - I_g(x, y)|}{(2w_x + 1) \times (2w_y + 1)} . \tag{3.2}$$

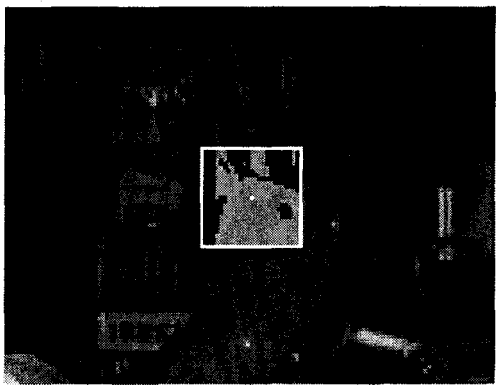
Les figures 3.2.(b), 3.2.(c), 3.2.(d) et 3.2.(e) présentent plusieurs voisinages extraits de l'image 3.2.(a) en utilisant une fenêtre initiale de taille $w_x = w_y = 13$. Le carré indique la position de la fenêtre de corrélation, centrée sur le pixel à mettre en correspondance, lequel est indiqué en blanc. Les pixels grisés sont ceux qui ont été retenus pour faire partie du calcul de l'indice de corrélation, tandis que les pixels noircis sont ceux qui ont été supprimés.

Les fenêtres 1, 2 et 3 sont placées soit dans une zone de l'image assez texturée, soit à proximité d'une discontinuité franche. On peut souligner deux types de problèmes rencontrés avec cette méthode de sélection en examinant ces configurations :

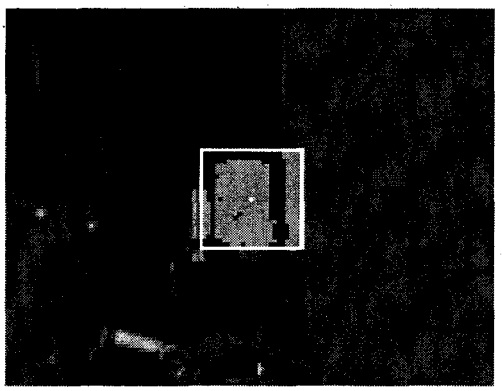
- des pixels avec des niveaux de gris similaires mais qui appartiennent à des objets différents ont été retenus. Ces pixels peuvent introduire des erreurs dans le calcul des indices de corrélation. Une analyse de connexité peut s'avérer utile dans cette situation ;
- des pixels qui appartiennent au même objet mais qui ont des niveaux de gris différents ont été supprimés. Cependant, un nombre important de pixels ont été retenus, ce qui permet d'avoir assez d'information dans le calcul des indices de corrélation.

La fenêtre 4 est placée dans une zone homogène, dans laquelle la plupart des pixels ont été retenus. Cette technique de sélection convient bien dans ces zones de l'image, dans lesquelles on doit utiliser un grand nombre des pixels pour lever les ambiguïtés dans la détermination de la disparité.

a) Image gauche

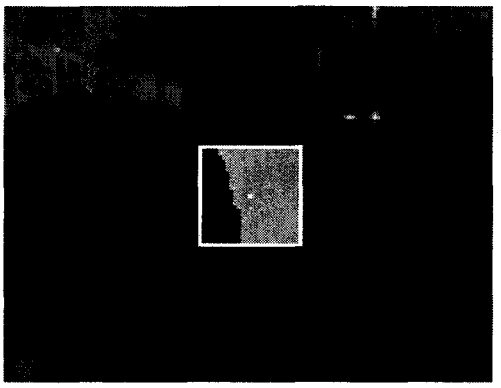


(b) Fenêtre 1



(c) Fenêtre 2

(d) Fenêtre 3



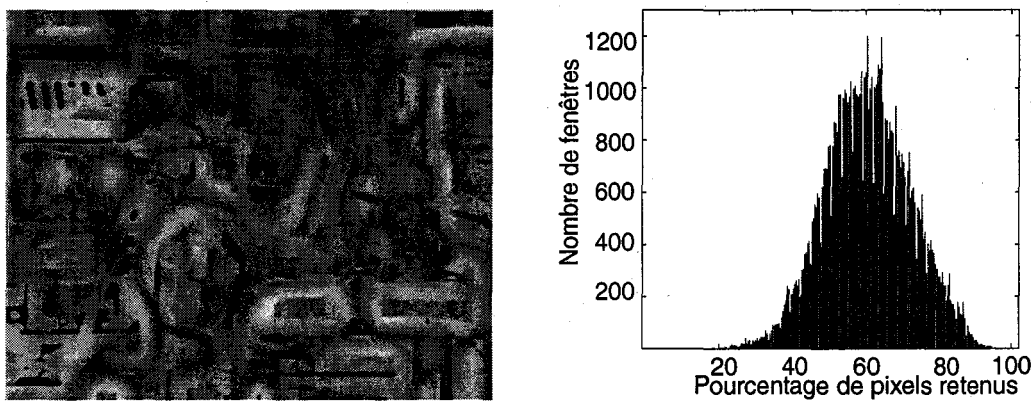
(e) Fenêtre 4

FIG. 3.2 : Les pixels retenus avec le critère de similarité basé sur le niveau de gris.

Les exemples précédents montrent que le voisinage utilisé n'a pas une forme définie et qu'il peut s'adapter aux éléments visuels présents localement dans l'image.

La figure 3.3.(a) présente une image indiquant le nombre des pixels retenus dans le calcul de corrélation pour chaque pixel de l'image 3.2.(a). Dans cette image, le niveau de gris code le nombre de pixels utilisés : clair pour un grand nombre de pixels et foncé pour

un petit nombre de pixels. L'image a été normalisée pour améliorer la lisibilité.



(a) Le nombre des pixels utilisés (b) Distribution du nombre de pixels utilisés

FIG. 3.3 : Le nombre de pixels utilisés et son histogramme

Les variations du niveau de gris dans l'image indiquent que le nombre de pixels retenus n'est pas le même pour chaque pixel à mettre en correspondance, c'est-à-dire que la taille de la fenêtre change pour chaque pixel. On peut constater dans cette image que le nombre de pixels utilisés diminue quand on se rapproche d'un contour..

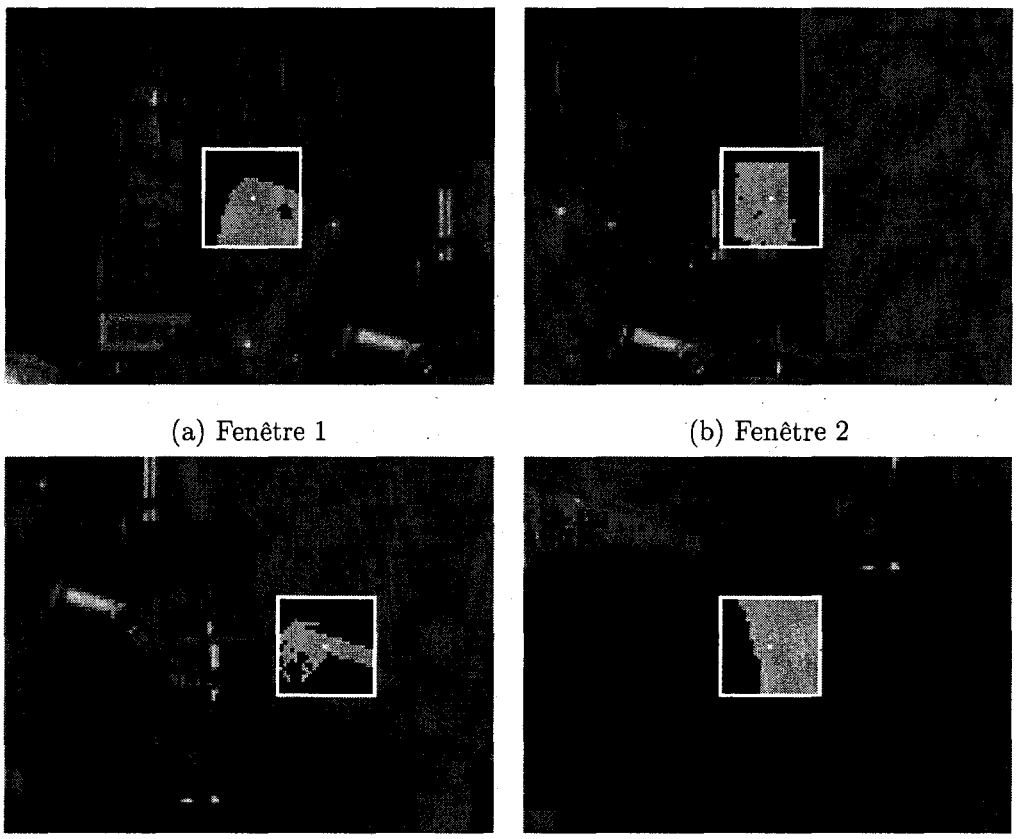
Afin d'évaluer le comportement global de cette méthode de sélection, nous avons analysé la distribution du pourcentage de pixels retenus dans les voisinages associés à tous les pixels de l'image 3.3.(a). La figure 3.3.(b) présente l'histogramme correspondant. L'axe des abscisses correspond au pourcentage de pixels retenus dans un voisinage tandis que l'axe des ordonnées présente le nombre de fenêtres dans l'image 3.3.(a) qui utilisent ce pourcentage de pixels. On constate qu'environ 40 % à 80 % des pixels de la fenêtre initiale sont retenus dans la plupart des cas. Ainsi, la mise en correspondance obtenue avec cette méthode est robuste car la majorité des voisinages adaptés contiennent suffisamment d'informations pour l'appariement.

Étant donné que les pixels sont toujours supprimés de la fenêtre de corrélation, et non ajoutés, il vaut mieux utiliser une fenêtre initiale de grande dimension. Cela permet de disposer d'un support suffisant dans les zones uniformes, sans nuire à la qualité du résultat dans les autres situations, comme nous le verrons en section 3.2.

3.1.2 Analyse de la connexité

On a vérifié dans la section précédente qu'il existe parfois dans une même fenêtre de corrélation des pixels dont les niveaux de gris sont proches mais qui ne correspondent pas au même objet. Ces pixels ne devraient pas intervenir dans le calcul des indices de corrélation puisqu'ils peuvent entraîner de mauvais appariements. Certains de ces pixels peuvent être supprimés au moyen d'une analyse de connexité. Ainsi, deux pixels sont considérés comme similaires quand leurs niveaux de gris sont proches et qu'ils sont connexes.

N'importe quel critère de connexité peut être utilisé, par exemple la connexité de quatre voisins. La figure 3.4 indique les pixels retenus, pour les mêmes fenêtres que celles de la figure 3.2, en utilisant le critère de similarité de niveau de gris des pixels et l'analyse de connexité.



(a) Fenêtre 1 (b) Fenêtre 2

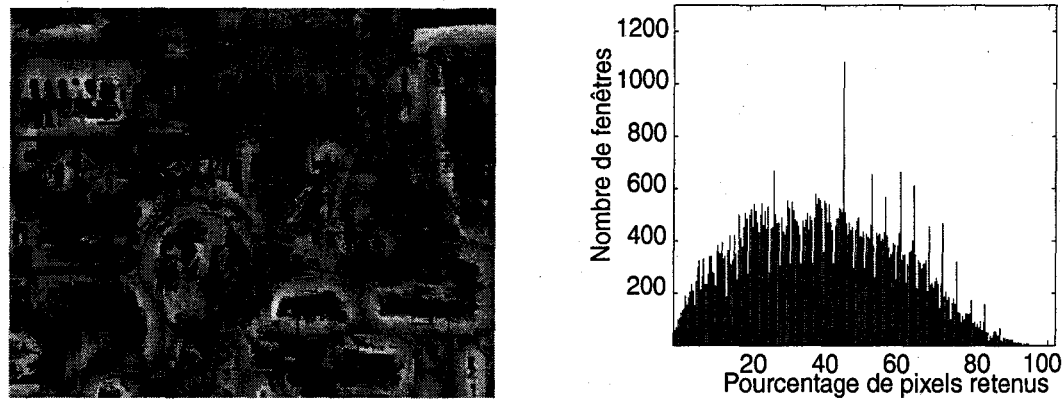
(c) Fenêtre 3 (d) Fenêtre 4

FIG. 3.4 : Les pixels retenus avec l'analyse de connexité.

Dans les fenêtres 1 et 2, seuls les pixels qui appartiennent au même objet que celui

du pixel à mettre en correspondance ont été retenus. On peut également constater que les voisinages utilisés sont mieux adaptés aux formes présentes localement dans l'image. Dans la fenêtre 3, il reste encore des pixels qui appartiennent à des objets différents car la méthode de sélection utilisée ne permet pas de séparer parfaitement des pixels correspondant à des objets différents mais dont les niveaux de gris sont très similaires. La plupart des pixels ont été retenus dans la fenêtre 4, laquelle est placée dans une zone homogène.

La figure 3.5.(a) indique le nombre des pixels retenus pour faire l'appariement de chaque pixel de l'image 3.2.(a) et la figure 3.5.(b) présente l'histogramme de l'image 3.5.(a).



(a) Le nombre des pixels utilisés (b) Distribution du pourcentage de pixels utilisés

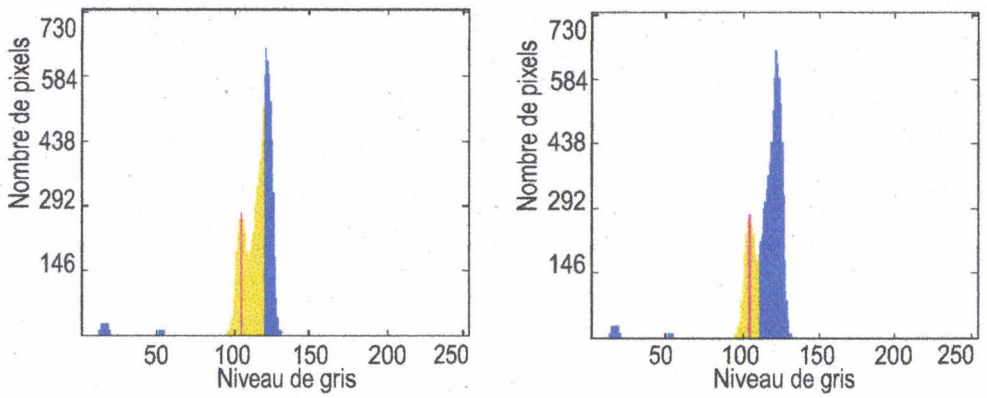
FIG. 3.5 : Le nombre de pixels retenus avec analyse de connexité.

Le nombre des pixels retenus est élevé dans les zones homogènes de l'image tandis qu'au voisinage des discontinuités, le nombre de pixels retenus est beaucoup plus faible. Globalement, le nombre de pixels retenus est plus faible que ceux qui sont retenus sans utiliser le critère de connexité. L'histogramme montre que pour certains pixels, la majorité des voisins a été éliminée dans le calcul de l'indice de corrélation. Dans ce cas, le voisinage support n'est plus suffisant, ce qui peut entraîner des problèmes d'appariement.

3.1.3 Sélection des pixels par seuillage

La figure 3.6 présente l'histogramme des niveaux de gris dans une fenêtre extraite de l'image à proximité d'une discontinuité. Le niveau de gris du pixel central est indiqué en

rouge et les niveaux de gris des pixels retenus sont indiqués en jaune.



(a) MDA (b) Seuil obtenu par seuillage

FIG. 3.6 : Distribution des niveaux de gris dans la fenêtre.

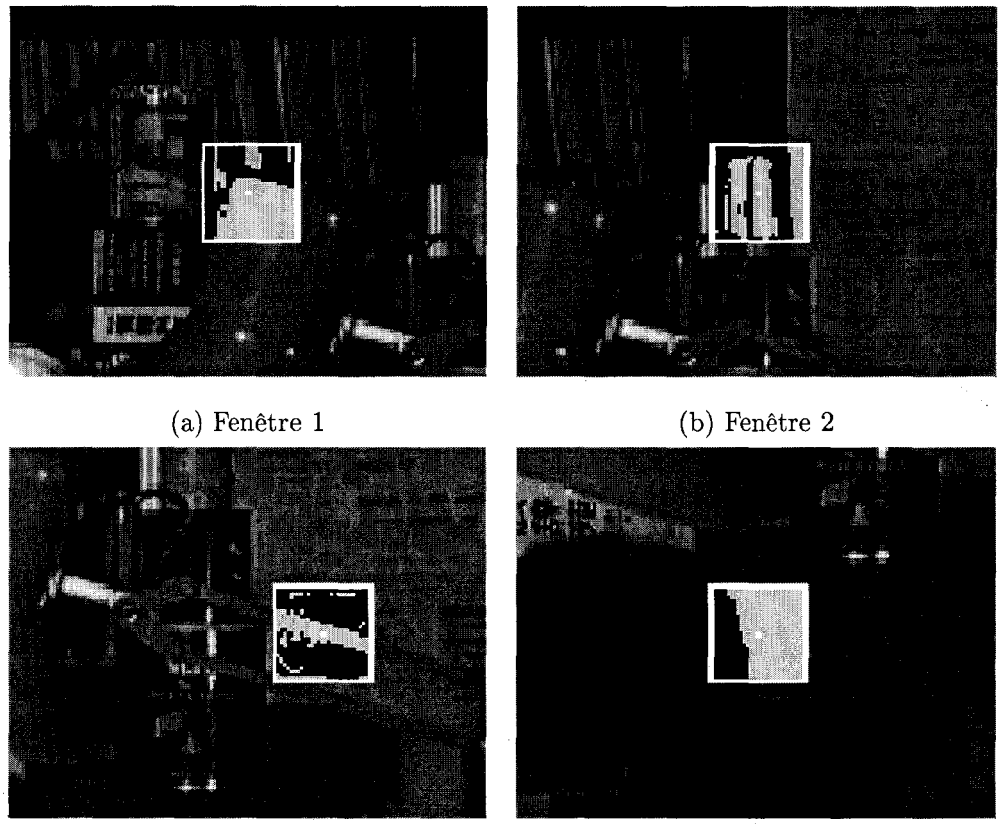
Si on suppose que chaque mode correspond à un objet différent, les pixels retenus avec la MDA correspondent aux projections de différents objets (figure 3.6.(a)). Cela peut entraîner des problèmes d'appariement. Ce problème peut être aggravé quand on augmente la taille de la fenêtre de corrélation.

Le seuil $T_g(x, y)$ peut être déterminé en détectant le mode de l'histogramme auquel appartient le pixel à mettre en correspondance. Plusieurs techniques de segmentation utilisant l'histogramme des niveaux de gris ont été décrites dans la littérature. En ce qui nous concerne, nous avons tout d'abord lissé l'histogramme afin de réduire l'effet du bruit, puis nous avons détecté le mode qui correspond au pixel à apparier par analyse du gradient. La figure 3.6.(b) présente les niveaux de gris des pixels retenus, qui correspondent normalement à un seul objet de la scène.

La figure 3.7 montre les pixels retenus dans les quatre fenêtres utilisées précédemment (cf. figure 3.2) en utilisant la méthode de seuillage de l'histogramme.

Ainsi, les pixels retenus sont presque les mêmes que ceux retenus avec la MDA, sauf dans la fenêtre 3, dans laquelle la plupart des pixels retenus appartiennent au support de la lampe. Cela montre que la méthode de sélection des pixels par seuillage peut être plus précise et moins dépendante de la taille de la fenêtre en ce qui concerne la détermination du seuil.

La figure 3.8 présente le nombre de pixels retenus en se fondant sur la méthode de

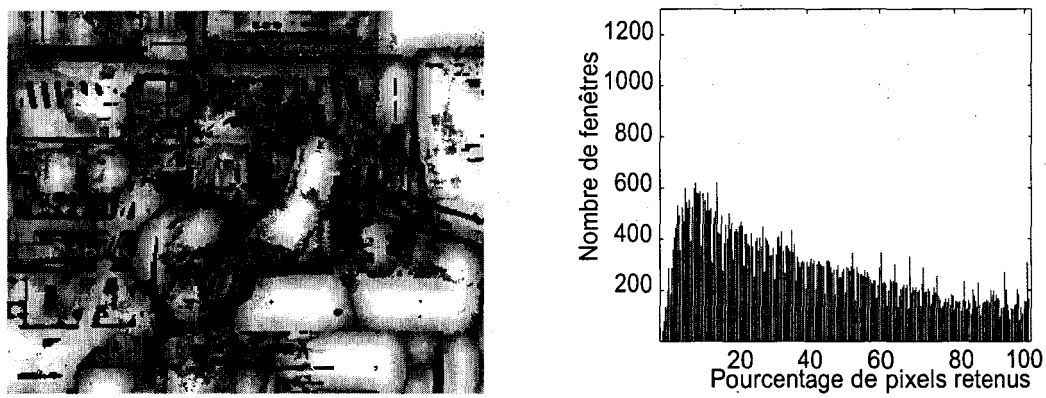


(a) Fenêtre 1 (b) Fenêtre 2

(c) Fenêtre 3 (d) Fenêtre 4

FIG. 3.7 : Les pixels retenus avec la méthode de sélection par seuillage.

sélection par seuillage et l'histogramme correspondant.



(a) Le nombre des pixels retenus (b) Distribution du pourcentage de pixels retenus

FIG. 3.8 : Le nombre de pixels retenus par la méthode de seuillage.

Le niveau de gris utilisé dans les zones homogènes est plus clair par rapport au niveau

de gris utilisé avec les critères de similarité précédents. Cela montre que le nombre de pixels retenus dans ces zones de l'image est plus important.

L'histogramme visualisé dans la figure 3.8.(b) indique que certains pixels suppriment la plupart de pixels de la fenêtre et peuvent rester isolés. Cela peut entraîner de mauvais appariements. Par contre, un grand nombre de pixels dans l'image utilisent la totalité de pixels de la fenêtre. Ces pixels sont placés dans les zones homogènes. Cela permet d'assurer un meilleur comportement dans les zones homogènes de l'image.

Cette méthode semble mieux adaptée pour déterminer le seuil utilisé dans la sélection des pixels que celles qui ont été décrites dans les paragraphes précédents. Cependant, elle peut difficilement être implantée dans une architecture spécialisée de calcul du fait de sa nature itérative.

3.2 Comparaison des résultats

Afin de vérifier le bon comportement de notre algorithme, nous avons calculé les cartes de disparité pour des images de synthèse utilisées couramment par la communauté scientifique. Dans ce cas, on dispose d'une carte de disparité réelle, définissant la vérité terrain, pour chaque pixel de l'image, ce qui permet de quantifier le comportement des algorithmes de mise en correspondance.

Le comportement de notre algorithme a été comparé à celui d'autres algorithmes de mise en correspondance par corrélation. Nous n'avons retenu que les techniques qui peuvent être implantées dans une architecture spécialisée de calcul. Les algorithmes retenus sont : l'algorithme qui utilise une fenêtre rectangulaire de taille fixe (SDA), l'algorithme de Hirschmüller (HIR), l'algorithme SMW et l'algorithme Census. Ces algorithmes ont été décrits dans le chapitre précédent.

La méthode de comparaison utilisée est celle proposée dans [Scharstein et Szeliski 2002]. En utilisant la carte de disparité réelle, les pourcentages de mauvais appariements, soit dans l'image complète, soit au voisinage des discontinuités, soit dans les zones homogènes, ont été calculés. On considère qu'un appariement est erroné quand la disparité estimée est différente d'au moins $\pm$ 1 pixel de la disparité réelle. Les occultations n'ont pas été prises en compte. Étant donné qu'aucun des algorithmes que nous avons sélec-

tionnés ne réalise une détection implicite des occultations, nous n'avons pas comptabilisé les mauvais appariements dans ces zones particulières de l'image.

Cette méthode de comparaison permet de comparer les algorithmes selon un critère quantitatif, mais elle ne permet pas la description qualitative des propriétés de chaque méthode. Une comparaison qualitative dépend directement de l'application envisagée, par exemple, si l'information obtenue de la carte de disparité est utilisée pour la manipulation d'objets avec une pince, une carte de disparité avec des contours bien définis est nécessaire, même si elle présente du bruit dans les zones avec une texture homogène. Dans ce cas, la méthode de comparaison quantitative peut indiquer un mauvais comportement alors qu'une méthode de comparaison qualitative peut indiquer le contraire.

Hélas, il n'y a pas de méthode standard pour faire une comparaison qualitative des cartes de disparité. Ainsi, nous avons tenté de souligner le comportement des algorithmes retenus dans les zones difficiles à mettre en correspondance, par exemple dans les zones qui contiennent de petits objets. La forme des contours détectés a également été considérée.

Comme les algorithmes comparés ont des comportements différents sur le bord des images, nous n'avons pas comptabilisé $w_x + s_{max}$ pixels à gauche et à droite de l'image et w_y pixels dans les parties supérieure et inférieure pour tous les algorithmes. Aucun post-traitement n'a été réalisé sur les images de disparité. En revanche, nous avons appliqué un filtre moyenneur 3×3 aux images de synthèse avant de déterminer les correspondances, afin d'atténuer le bruit présent dans les images.

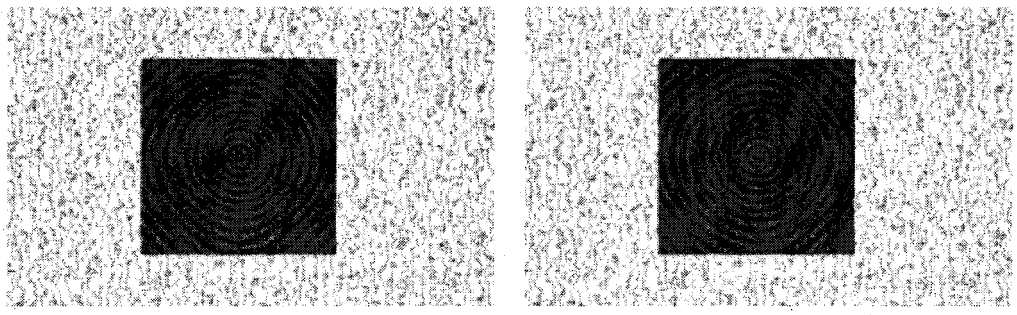
Les images de synthèse utilisées — tout comme la plupart des images réelles — présentent des zones homogènes. C'est la raison pour laquelle nous avons utilisé des fenêtres de corrélation de grandes tailles pour comparer les algorithmes.

Dans la section 3.2.1, nous présentons les résultats obtenus, puis les commentons dans la section 3.2.2.

3.2.1 Résultats obtenus

La figure 3.9 présente l'image de synthèse appelée "Carré". Elle contient deux objets plats non inclinés situés à des distances différentes des deux caméras. Le niveau de gris et la texture de chaque objet sont complètement différents. Cependant, le carré central présente une texture répétitive. Ainsi, la difficulté dans cette image correspond à calculer la

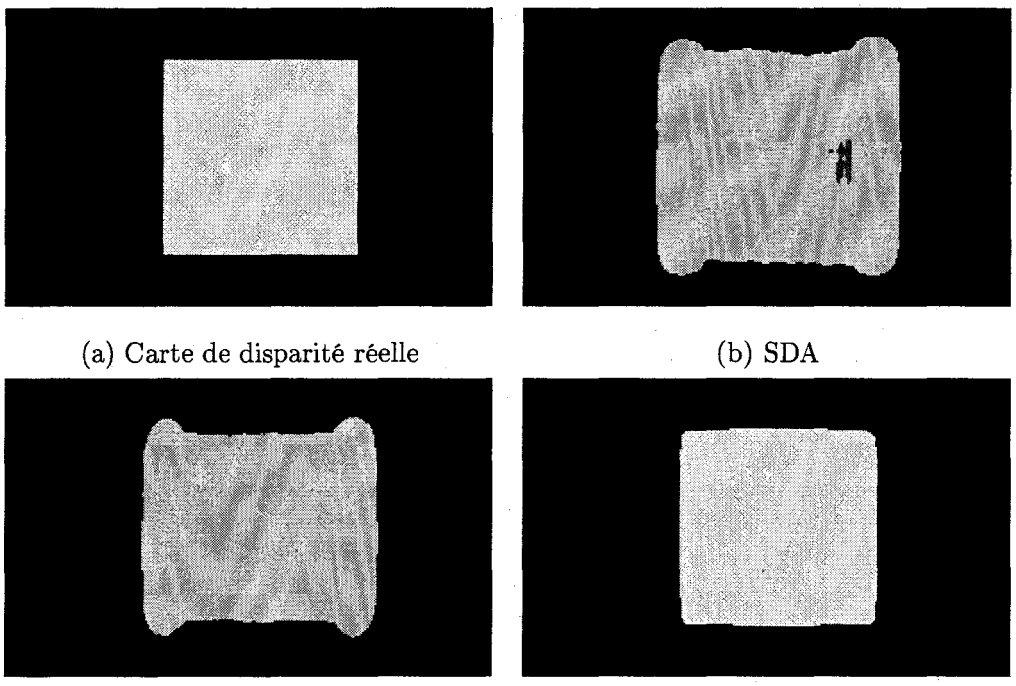
disparité correcte sur les bords du carré central en tenant compte des textures répétitives.



(a) Image gauche (b) Image droite

FIG. 3.9 : Image "Carré".

Les figures 3.10 et 3.11 présentent les cartes de disparité obtenues en utilisant une fenêtre de taille $w_x = w_y = 13$.



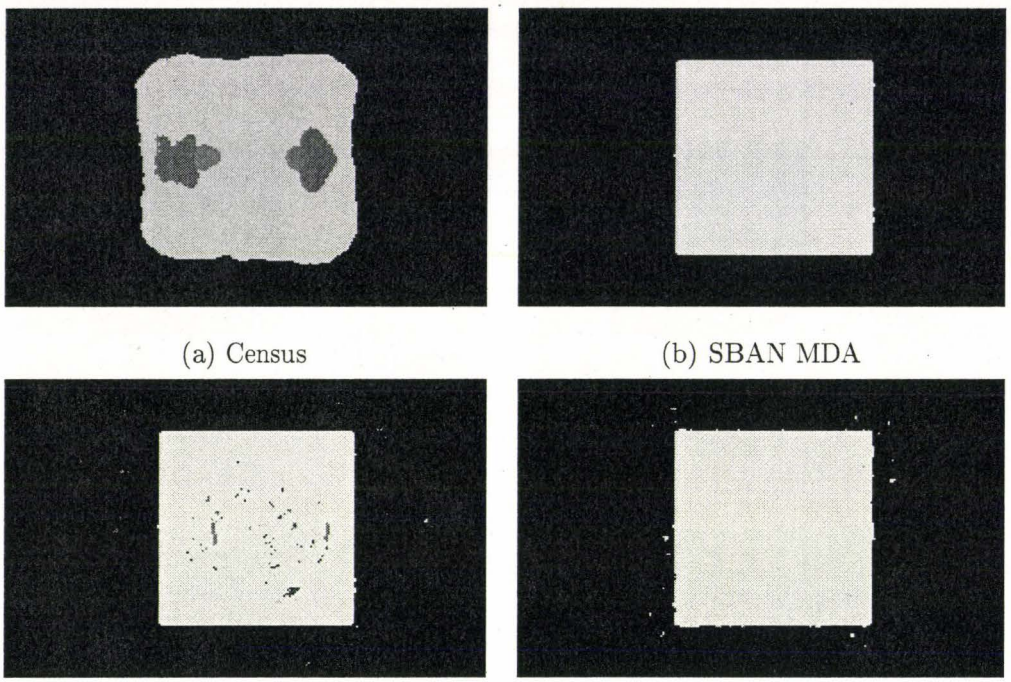
(a) Carte de disparité réelle (b) SDA

(c) HIR (d) SMW

FIG. 3.10 : Comparaison des cartes de disparité.

Le tableau 3.1 indique les pourcentages de mauvais appariements obtenus lorsqu'on utilise des fenêtres de différentes tailles.

La figure 3.12 présente l'image "Head and lamp" de l'Université de Tsukuba. Cette



(a) Census (b) SBAN MDA

(c) SBAN Connexité (d) SBAN Seuillage

FIG. 3.11 : Comparaison des cartes de disparité.

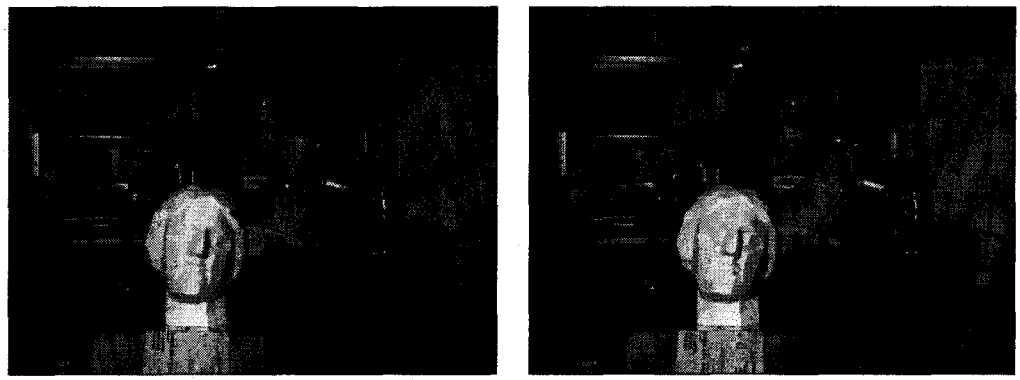
Algorithme	Total (%)			Discontinuités (%)			Homogènes (%)		
w	7	10	13	7	10	13	7	10	13
SDA	6,5	7,0	8,8	53,5	53,6	53,6	3,5	6,3	9,9
HIR	3,0	3,3	5,0	41,6	44,3	49,9	1,6	2,9	4,6
SMW	0,4	0,5	0,6	10,9	12,6	14,4	0,0	0,0	0,0
Census	7,3	6,2	5,2	39,3	41,3	38,6	0,2	0,6	1,3
SBAN SDA	0,3	0,3	0,3	9,1	7,8	7,3	0,0	0,0	0,0
SBAN Connexité	1,2	0,9	0,8	9,1	7,8	7,2	0,5	0,2	0,2
SBAN Seuillage	0,8	0,5	0,5	18,4	12,1	10,4	0,3	0,2	0,2

TAB. 3.1 : Pourcentage d'erreur obtenu dans l'image "Carré".

image contient des objets non inclinés situés à différentes distances. L'image présente aussi de grandes zones homogènes et de petits objets, tels le support de la lampe.

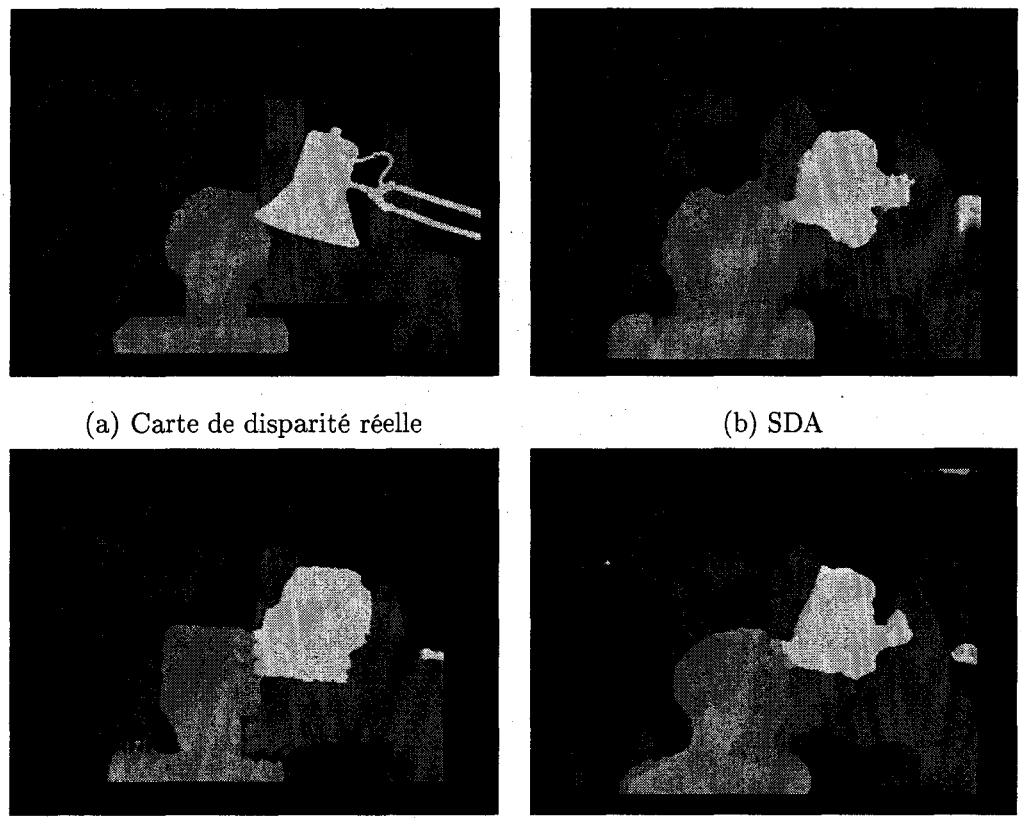
Les figures 3.13 et 3.14 présentent les cartes de disparité obtenues avec les algorithmes retenus en utilisant une fenêtre de taille $w_x = w_y = 13$.

Le tableau 3.2 indique les pourcentages de appariements obtenus lorsqu'on utilise des



(a) Image gauche (b) Image droite

FIG. 3.12 : Image "Head and lamp".



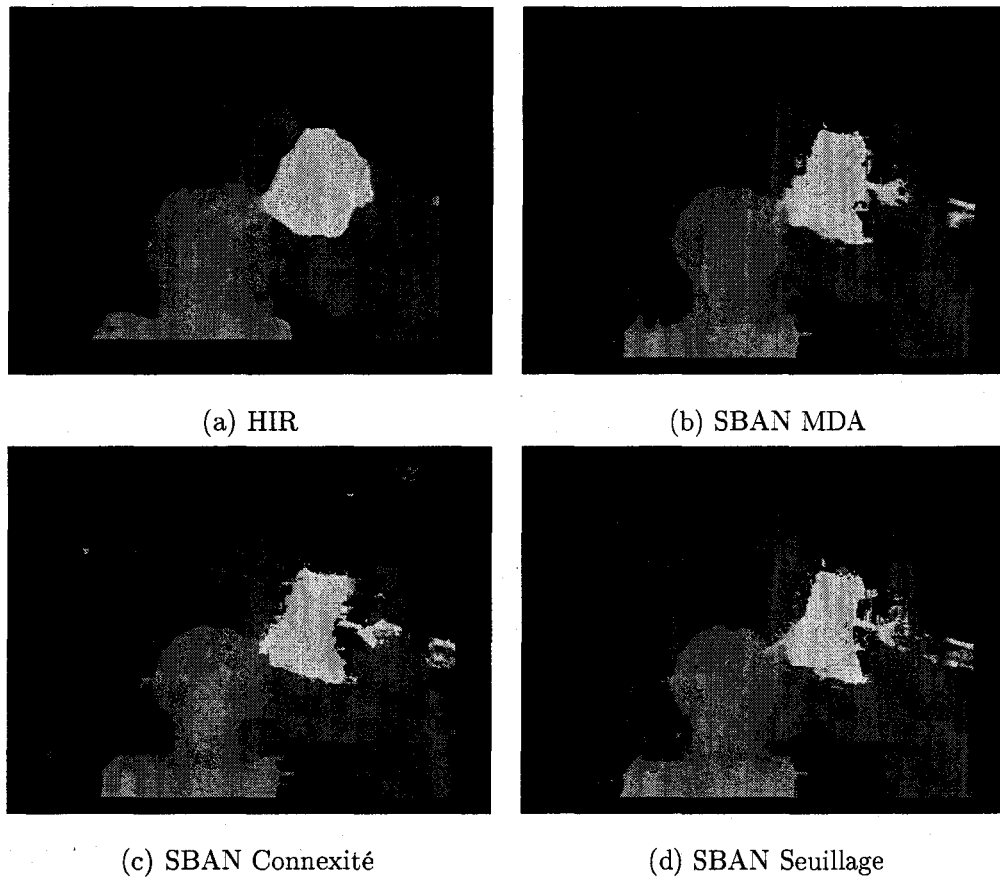
(a) Carte de disparité réelle (b) SDA

(c) SMW (d) Census

FIG. 3.13 : Comparaison des cartes de disparité.

fenêtres de différentes tailles.

La figure 3.15(a) présente l'image "Map", dans laquelle il y a deux objets inclinés. Comme dans l'image "Carré", la difficulté réside dans le fait qu'il s'agit de calculer la



(a) HIR (b) SBAN MDA

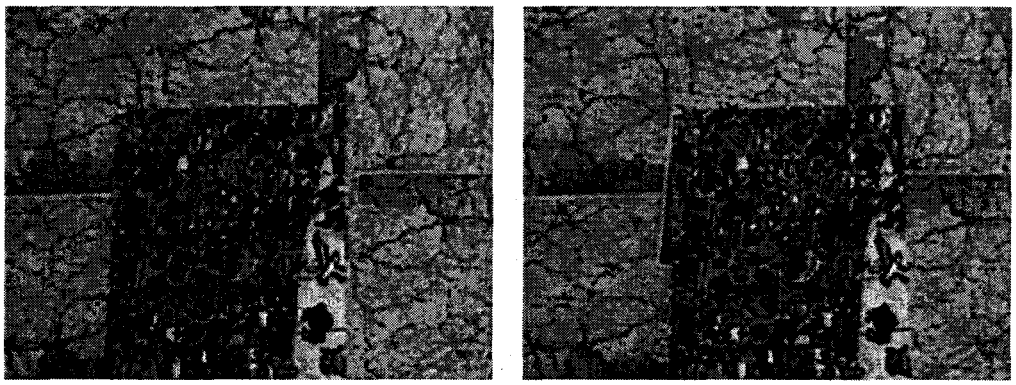
(c) SBAN Connexité (d) SBAN Seuillage

FIG. 3.14 : Comparaison des cartes de disparité.

Algorithme	Total (%)			Discontinuités (%)			Homogènes (%)		
	w								
	7	10	13	7	10	13	7	10	13
SDA	10,0	10,1	12,8	33,9	39,0	42,0	8,8	8,8	10,6
HIR	7,2	9,0	10,6	32,4	37,8	39,7	5,2	5,7	7,4
SMW	7,9	7,6	8,5	25,0	31,4	33,6	9,1	4,7	4,3
Census	7,4	6,4	7,1	26,8	29,2	32,3	7,1	4,0	4,1
SBAN MDA	7,0	5,7	5,9	19,0	21,4	25,1	8,9	5,1	3,6
SBAN connexité	8,4	6,9	6,5	17,1	19,2	21,5	11,6	7,9	5,9
SBAN Seuillage	8,2	7,5	6,9	21,5	22,8	23,1	9,7	7,8	5,9

TAB. 3.2 : Pourcentage d'erreur obtenue dans l'image "Head and lamp".

disparité correcte sur les bords des objets en tenant compte des surfaces inclinées.



(a) Image gauche (b) Image droite

FIG. 3.15 : L'image "Map".

Les figures 3.16 et 3.17 montrent les cartes de disparité obtenues avec les algorithmes retenus.

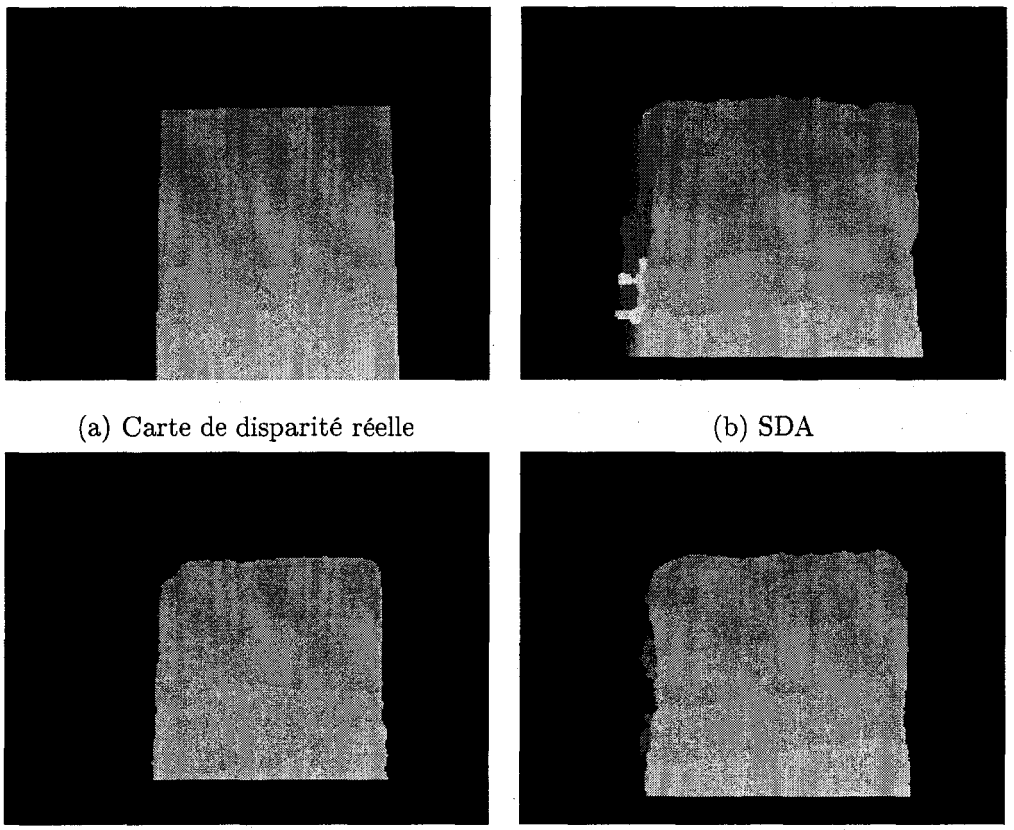
Le tableau 3.3 indique les pourcentages d'erreur obtenus avec les algorithmes comparés, en utilisant des fenêtres de différentes tailles.

Algorithme	Total (%)			Discontinuités (%)			Homogènes (%)		
	w								
	7	10	13	7	10	13	7	10	13
SDA	6,0	8,8	9,2	18,7	29,5	30,7	12,6	14,4	26,9
HIR	1,3	1,7	2,3	11,8	14,0	21,8	0,0	0,0	0,0
SMW	2,0	3,0	4,4	16,6	22,5	35,4	0,0	0,0	0,0
Census	2,1	1,6	1,6	8,9	10,4	14,0	6,6	4,1	6,4
SBAN MDA	1,3	1,7	2,0	10,9	12,6	14,7	0,0	0,0	0,0
SBAN Connexité	2,8	3,1	3,3	13,2	14,9	17,6	0,5	0,0	0,0
SBAN Seuillage	3,1	3,0	3,4	15,7	16,6	20,3	0,5	0,7	0,0

TAB. 3.3 : Pourcentage d'erreur obtenue dans l'image "Map"

3.2.2 Discussion des résultats

Le comportement de l'algorithme SDA a été décrit à de nombreuses reprises par la communauté scientifique qui travaille dans le domaine de la stéréovision. L'utilisation



(a) Carte de disparité réelle (b) SDA

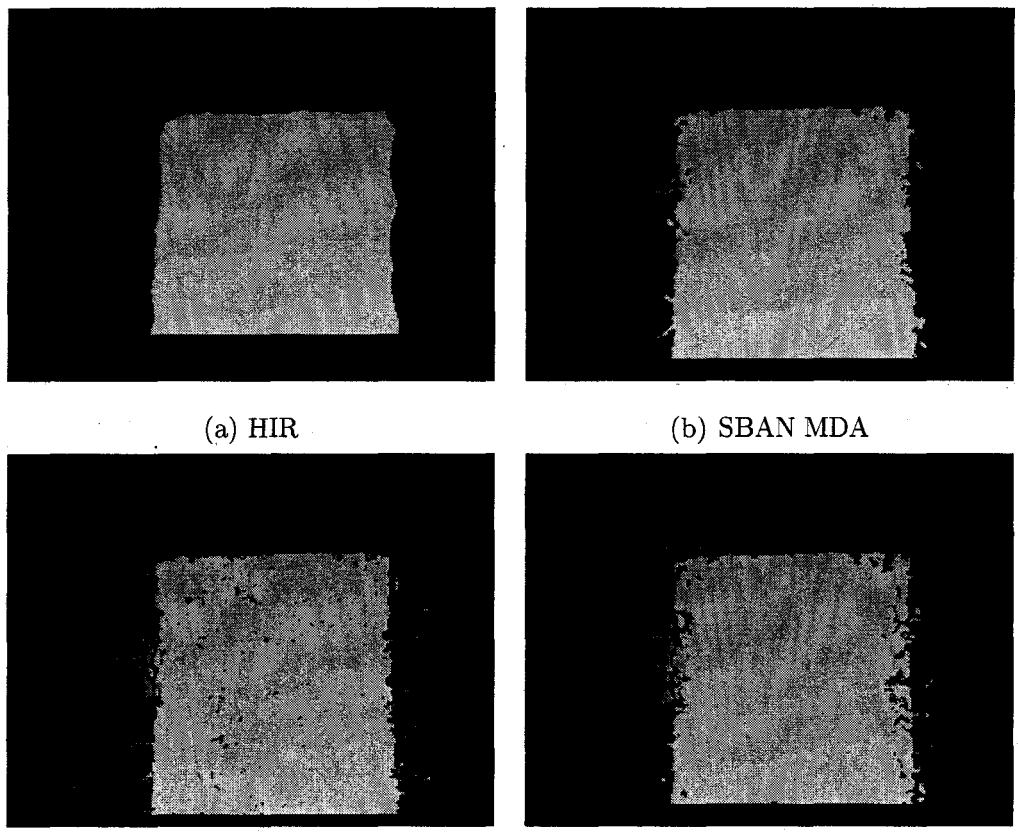
(c) SMW (d) Census

FIG. 3.16 : Comparaison des cartes de disparité.

d'une grande fenêtre est nécessaire dans les zones homogènes de l'image. En revanche, avec une grande fenêtre, les bords des objets ne sont pas détectés. On peut observer que la disparité obtenue sur les bords des objets ne correspond pas du tout à la disparité réelle. Les tableaux confirment le comportement de cet algorithme en indiquant que le pourcentage d'erreur au voisinage des discontinuités augmente avec la taille de la fenêtre.

Grâce à l'utilisation de plusieurs fenêtres, l'algorithme de Hirschmüller se comporte comme un algorithme adaptatif à proximité des discontinuités et comme un algorithme qui utilise une grande fenêtre dans les zones homogènes. Cela permet de réduire l'effet de moyennage sur le bord des objets et d'améliorer simultanément le comportement de l'algorithme dans les zones homogènes. Cependant, comme la fenêtre centrale est toujours utilisée, de mauvais appariements apparaissent à proximité des discontinuités.

Les tableaux montrent que les pourcentages de mauvais appariements obtenus avec l'algorithme HIR sont toujours plus faibles que les pourcentages obtenus avec l'algorithme



(a) HIR (b) SBAN MDA

(c) SBAN Connexité (d) SBAN Seuillage

FIG. 3.17 : Comparaison des cartes de disparité.

SDA. Cependant, le nombre de mauvais appariements reste élevé par rapport aux autres algorithmes.

L'algorithme SMW s'adapte bien dans les zones qui contiennent des objets rectangulaires. Cependant, dans la carte de disparité obtenue, les objets non rectangulaires sont déformés et ils présentent une forme rectangulaire qui ne correspond pas à la disparité réelle, par exemple, dans l'image "Head and lamp". L'algorithme présente aussi des inconvénients dans les coins des rectangles, lesquels sont arrondis dans la carte de disparité.

Avec une grande fenêtre de corrélation, l'algorithme Census ne présente pas d'avantages remarquables par rapport aux algorithmes précédents. De plus, il présente des inconvénients dans les zones comportant une texture répétitive, à cause de la diminution de l'information sémantique des pixels, qui sont décrits par un seul bit dans la transformée Census.

Il est important de noter qu'avec les algorithmes précédents, l'erreur aux discontinuités

grandit au fur et à mesure qu'on augmente la taille de la fenêtre de corrélation. Ce comportement s'explique par l'utilisation de pixels ayant des disparités différentes dans la fenêtre de corrélation.

Les cartes de disparité obtenues montrent que les contours sont mieux détectés par l'algorithme SBAN, tout en conservant un bon comportement dans les zones homogènes. Même dans les coins des objets, pour lesquels les autres algorithmes ont des problèmes d'appariement, la carte des disparités obtenue avec l'algorithme SBAN est de qualité largement supérieure. Grâce à l'utilisation d'un voisinage adaptatif, les petits objets, tels que le support de la lampe, ont été détectés.

On peut constater que le comportement de l'algorithme SBAN par rapport à la taille de la fenêtre est différent de celui des autres algorithmes. Plus on augmente la taille de la fenêtre initiale, plus l'erreur aux discontinuités et dans les zones homogènes diminue.

Le critère utilisé pour déterminer le seuil détermine en grande partie le comportement de l'algorithme SBAN. Le critère MDA est robuste mais son comportement dépend de la taille de la fenêtre. Certains critères, tels le seuillage et l'analyse de connexité, donnent de meilleurs résultats près des discontinuités. Cependant, comme nous l'avons remarqué précédemment, ces critères ont tendance à isoler certains pixels en diminuant de façon exagérée le voisinage utilisé pour le calcul de corrélation.

Dans l'image "Head and lamp", le niveau de gris du support de la lampe est très similaire au niveau de gris du fond de la scène. Les critères de similarité proposés ne permettent pas de les séparer et de mauvais appariements apparaissent dans le support de la lampe. Reste à trouver des critères de similarité plus adaptés, reposant par exemple sur l'exploitation de la couleur des pixels.

L'algorithme SBAN n'est pas du tout adapté aux images présentant des variations progressives de la disparité, comme l'image "Map". Ceci est confirmé par les scores médiocres apparaissant dans les tableaux comparatifs. D'autre part, dans cette image, l'objet principal est fortement texturé et il inclut des pixels dont les niveaux de gris sont totalement différents. Il faut noter que cela contredit notre supposition initiale, dans laquelle les variations du niveau de gris dans un objet sont faibles.

Dans l'algorithme SBAN, le nombre de pixels retenus dans le voisinage pour apparier un pixel peut être utilisé comme une mesure de confiance additionnelle. En effet, tous les

pixels qui utilisent un nombre de voisins inférieur à un seuil donné peuvent être considérés comme non fiables. Un post-traitement permettrait d'éliminer ces pixels erronés, en utilisant par exemple des opérateurs morphologiques modifiés qui tiendraient compte des valeurs du critère de confiance.

3.3 Conclusion

Nous avons présenté un algorithme adaptatif de stéréovision dense par corrélation appelé SBAN. Dans notre méthodes, certains pixels sont supprimés de la fenêtre de corrélation, ce qui est équivalent à modifier la taille et la forme de cette fenêtre. Les pixels à conserver dans le calcul de l'indice de corrélation sont sélectionnés en fonction de leur similarité avec le pixel central.

L'algorithme SBAN peut travailler sur des grandes fenêtres de corrélation, lesquelles sont nécessaires dans les zones homogènes de l'image. Grâce à la suppression de certains pixels, le comportement de l'algorithme est largement supérieur au comportement des algorithmes qui utilisent plusieurs fenêtres. Le voisinage utilisé s'adapte aux formes locales de la scène.

Plusieurs critères peuvent être utilisés pour sélectionner les pixels qui font partie du voisinage adapté. Nous avons montré que même un critère de similarité trivial comme l'égalité des niveaux de gris permet d'obtenir un excellent comportement de l'algorithme SBAN. Nous avons également montré que, sous certaines conditions, le niveau de gris n'est pas suffisant pour déterminer quels sont les pixels qui doivent intervenir dans le calcul. Reste à déterminer de critères de similarité mieux adaptés, même s'ils ne sont pas compatibles avec une implantation matérielle.

L'algorithme SBAN, comme les autres algorithmes décrits dans ce chapitre, peut être implanté dans une architecture de calcul spécialisée afin de traiter les images à la cadence vidéo. L'implantation de ces algorithmes dans le processeur STREAM, qui a été développé dans notre laboratoire, fait l'objet du chapitre suivant.

Chapitre 4

Implantation des algorithmes sur le processeur STREAM

Au Laboratoire d'Automatique, Génie Informatique et Signal (LAGIS), une architecture spécialisée de calcul a été développée, à savoir le Système Temps-Réel d'Extraction et d'Analyse du Mouvement (STREAM). Cette architecture, qui exploite un circuit reconfigurable de type FPGA de haute densité, est dédiée au traitement de séquences d'images numériques à la cadence vidéo.

Nous nous sommes intéressés à l'implantation sur ce système des algorithmes de stéréovision fournissant une carte dense des disparités. Parmi les algorithmes proposés dans la littérature, nous avons retenu uniquement les algorithmes décrits dans le chapitre précédent. Ces algorithmes ont été sélectionnés parce qu'ils n'utilisent que des calculs locaux et réguliers. Ils peuvent être implantés efficacement dans le processeur STREAM ou sur n'importe quelle autre architecture de calcul parallèle.

Dans la première partie de ce chapitre, nous décrivons l'architecture du processeur STREAM. Ensuite, nous présentons comment les algorithmes retenus ont été implantés dans ce processeur. Pour terminer le chapitre, nous comparons les ressources matérielles utilisées par chaque algorithme. Cet comparaison a fait l'objet d'une communication à l'occasion d'une conférence [Perez et Cabestaing 2003].

4.1 Description du processeur STREAM

Le processeur STREAM est une architecture dédiée de calcul conçue pour le traitement en temps-réel des séquences d'images [Cabestaing *et coll.* 1999; 1991]. Le processeur est composé de deux modules : un module d'acquisition et de numérisation des données vidéo et un module de traitement des données vidéo. La figure 4.1 présente le synoptique du processeur.

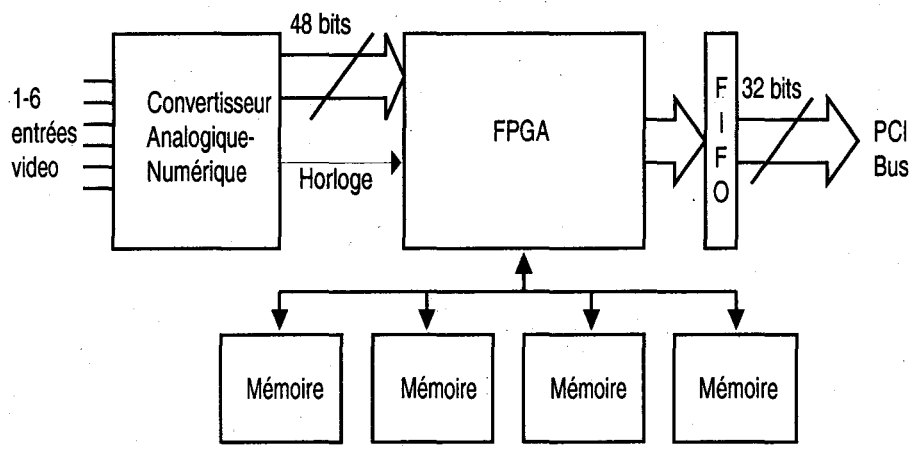


FIG. 4.1 : Architecture du processeur STREAM.

Le module de numérisation des données vidéo permet de gérer le fonctionnement des caméras connectées au système. Des signaux de cadencement, de synchronisation et de déclenchement sont envoyés aux caméras par ce module. Il assure également la numérisation et le formatage des signaux vidéo issus des caméras. Six caméras noir et blanc peuvent être connectées au système et numérisées sur 8 bits, ce qui crée un flot de données numériques sur 48 bits. Le flot de données, les signaux de synchronisation et les horloges sont transmises au module de traitement de données vidéo par une interface Camera-Link®. Cette interface est une liaison rapide qui permet le transfert d'une grande quantité de données à haute cadence. Un signal d'horloge, synchronisé avec les signaux de vidéo numérique, cadence les opérations dans le module de traitement et dans les éléments de mémorisation.

Le module de traitement de données a comme fonction principale de réaliser des opérations de bas niveau sur les images envoyées par le module de numérisation. Il est implanté

sous la forme d'une carte PCI, comme le montre la figure 4.2.

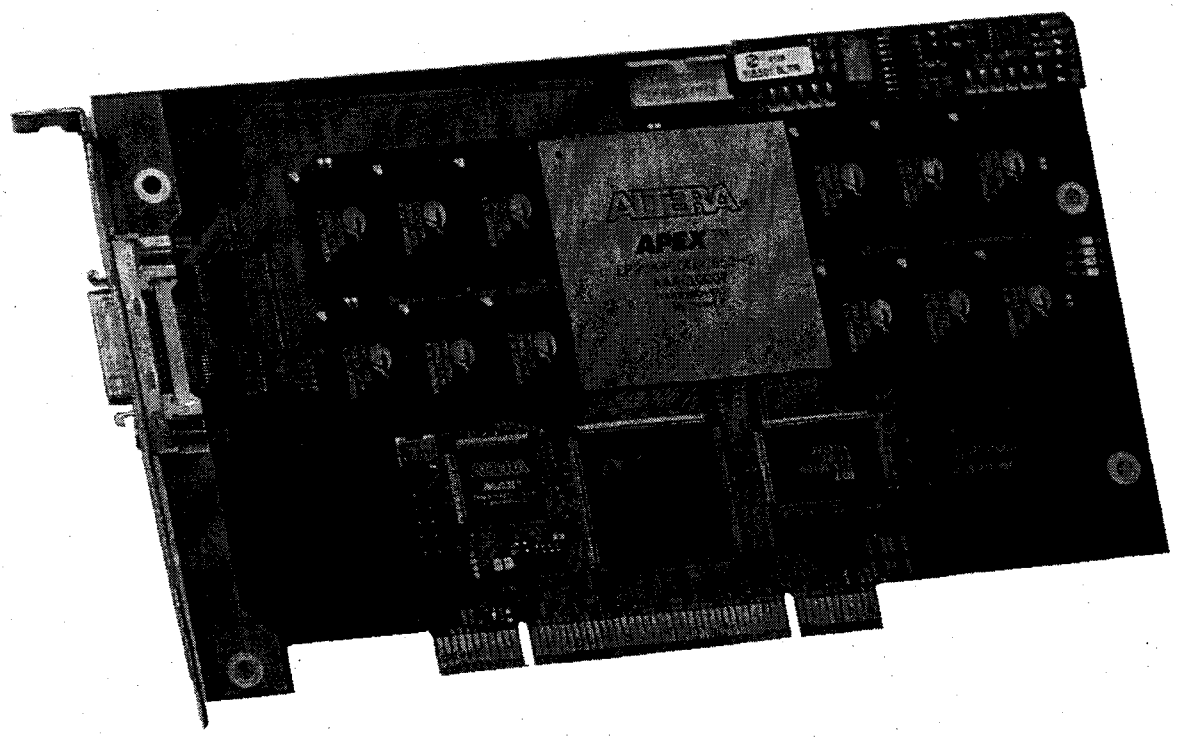


FIG. 4.2 : Le module de traitement de données.

Ce module intègre un circuit FPGA programmable de haute densité de chez Altera® disposant d'un million et demi de portes logiques. Quatre bancs de mémoire, accessibles indépendamment les uns des autres, sont connectés directement au FPGA pour stocker des donnés temporaires ou des images. Les résultats du traitement sont stockés dans une mémoire FIFO de grande capacité. Un contrôleur de bus PCI gère le transfert des données vers l'ordinateur.

L'algorithme à implanter peut être défini à l'aide d'un langage de description de composants de haut niveau, tel que VHDL. Les logiciels Quartus et MaxPlus permettent la simulation du comportement de l'implantation et la synthèse de l'algorithme. Après compilation, un fichier de code est obtenu lequel est téléchargé dans le circuit FPGA.

Plusieurs bibliothèques de fonctions ont été développées pour programmer le circuit FPGA, pour contrôler le transfert de données vers l'ordinateur et pour l'affichage des images sur l'écran dans le système d'exploitation Windows.

4.1.1 Traitement d'un flot de données

Le flot de données F créé par le processeur STREAM tire avantage du mode de balayage d'une image dans un système de transmission de vidéo comme le montre la figure 4.3. Chaque image (I) est balayée ligne (y) après ligne et pixel (x) après pixel. Ainsi, un seul pixel $P(I, y, x)$ est disponible à un instant donné dans le flot de données.

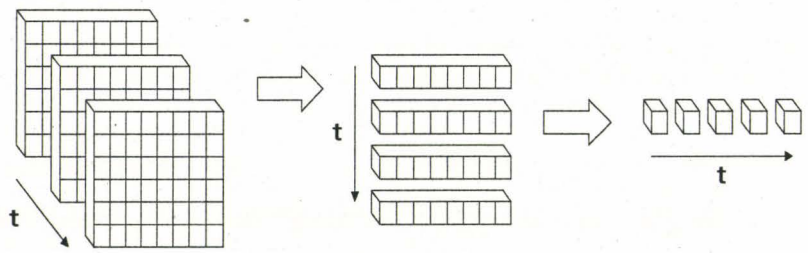


FIG. 4.3 : Création du flot de données.

Pour accéder au voisinage spatio-temporel d'un pixel, on insère dans le flot vidéo des éléments de retard, comme le montre la figure 4.4. Avec cette technique, on peut accéder aux pixels déjà balayés dans les images, mais pas aux pixels suivants. Il convient donc de retarder les calculs liés à un pixel particulier du flot jusqu'à ce que tous les pixels voisins aient été balayés.

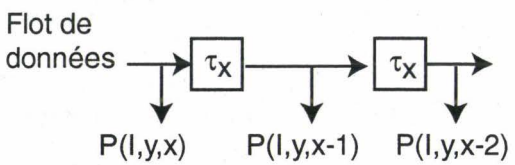


FIG. 4.4 : Utilisation de retards pour accéder au voisinage spatial d'un pixel.

Dans une architecture pipeline, les éléments de retard sont en général implantés au

moyen de bascules de type D disponibles dans le FPGA. Ces bascules peuvent être utilisées pour stocker un nombre limité de pixels. En revanche, pour stocker toute une ligne de pixels de l'image, l'utilisation d'une cascade de bascules D n'est pas la meilleure solution.

Le circuit FPGA intégré dans le STREAM contient des blocs appelés ESB (Embedded System Blocs), qui peuvent être configurés comme des mémoires FIFO, des mémoires RAM ou des mémoires CAM [1]. Dans notre cas, ces blocs sont utilisés pour stocker efficacement des lignes de pixels. Plusieurs configurations des blocs ESB sont disponibles, permettant d'implanter des tampons mémoire de différentes tailles et différentes profondeurs [2]. Par exemple, les configurations disponibles quand on utilise les blocs ESB pour implanter une mémoire RAM sont les suivantes : 128×16, 256×8, 512×4, 1024×2 ou 2048×1.

Soulignons que ces blocs sont plus efficaces qu'une succession de bascules quand il s'agit de mémoriser une ligne complète de pixels. Par contre, pour un nombre réduit de pixels, l'utilisation des blocs ESB n'est pas conseillée, car de nombreux bits de mémoire ne sont pas utilisés.

4.2 Implantation des algorithmes retenus

Comme nous l'avons montré précédemment, l'utilisation d'éléments de retard permet l'accès au voisinage d'un pixel, puisque ce dernier n'est présent dans le flot de données vidéo que durant une période d'horloge. Un algorithme qui utilise une fenêtre rectangulaire de taille $w_x \times w_y$ doit ainsi stocker au moins w_y lignes de pixels. Cela prend beaucoup de place dans le composant FPGA, malgré l'utilisation des ESB.

Dans une paire d'images rectifiées, le correspondant d'un pixel de l'image gauche se trouve dans la ligne de même indice dans l'image droite. Dans une architecture pipeline comme le STREAM, les deux pixels apparaissent dans deux flots différents, mais à des instants très proches étant donné que les lignes d'indices identiques dans les différentes images sont balayées simultanément.

Plusieurs méthodes ont été décrites dans la littérature pour tenter de simplifier l'implantation des algorithmes. Dans [Philipp *et coll.* 2002], les indices de corrélation sont

[1] http ://www.altera.com

[2] http ://www.altera.com/products/devices/apex/features/apx-architecture.html

calculés avec la somme des niveaux de gris de chaque colonne, plutôt qu'avec les niveaux de gris de chaque pixel. Dans [Woetzel et Koch 2004, Yang et Pollefeys 2003] une fenêtre de taille 1×1 est utilisée, pour pallier la limitation des ressources matérielles disponibles dans les systèmes exploités par les auteurs.

Ainsi, la mise en correspondance peut être réalisée en tenant compte d'un voisinage de taille $w_x \times 1$, c'est-à-dire contenant uniquement les voisins qui se trouvent dans la même ligne de l'image, les autres étant exclus de la fenêtre de corrélation. Cette technique permet de réduire considérablement le nombre d'opérations à réaliser et la quantité de ressources matérielles nécessaires à l'implantation.

Nous avons adopté cette simplification afin de réduire le nombre de ressources matérielles nécessaires à l'implantation des algorithmes. Nous avons donc opté pour une fenêtre de taille $w_x \times 1$. Il faut souligner que cette simplification nous a été imposée par les ressources limitées disponibles dans la version du processeur que nous utilisons actuellement. On pourrait implanter les versions bidimensionnelles des algorithmes retenus sans devoir apporter de modifications majeures aux techniques que nous allons présenter par la suite. Nous reviendrons sur ce point dans la conclusion de ce mémoire.

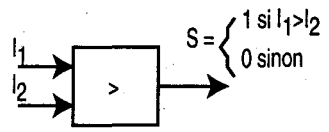
4.2.1 Description des modules utilisés

L'implantation d'un algorithme dans une structure de traitement de type flot de données nécessite la décomposition itérative des blocs de traitement en structures de plus en plus simples. L'algorithme est finalement décrit par un graphe hiérarchique faisant intervenir des modules élémentaires, qui réalisent une opération unique et en général très simple. La complexité de l'algorithme global est principalement traduite par celle de la structure hiérarchique qui le décrit, relativement peu par la complexité des modules de base.

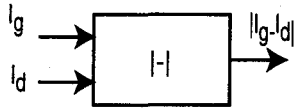
Chaque module de base est décrit à l'aide d'un langage de description de composants logiques en associant un ensemble de portes logiques. Un langage propre à Altera, appelé AHDL (Altera Hardware Description Langage), a été utilisé. Nous avons choisi ce langage, plutôt que le VHDL, car il permet l'utilisation de macro-commandes qui optimisent le nombre de portes logiques utilisées dans les circuits conçus par Altera. La description de l'implantation en AHDL n'est pas transposable directement à des circuits reconfigurables

d'autres fournisseurs. Cependant, hormis les macro-comandes spécifiques, la structure d'une description est très similaire à celle qui serait définie en utilisant le VHDL.

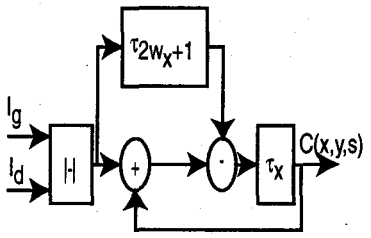
La figure 4.5 présente les modules de base que nous avons utilisés pour implanter tous les algorithmes retenus :



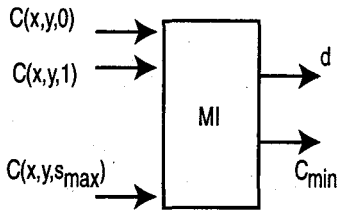
(a) Module de comparaison



(b) Module de différence absolue



(c) Module de corrélation, CI



(d) Module de sélection du minimum

FIG. 4.5 : Les modules de base.

La sortie du module de comparaison indique, par un niveau logique égal à 1, si la valeur de l'entrée I_1 est supérieure à la valeur de l'entrée I_2. Ce module est utilisé principalement dans l'implantation de l'algorithme Census.

Le module DA calcule la différence absolue des niveaux de gris présents sur les entrées I_g et I_d. Il est composé de deux soustracteurs, d'un comparateur et d'un multiplexeur.

Le module de corrélation calcule la somme des différences absolues dans une fenêtre de taille $w_x \times 1$. Le nombre de retards utilisés dans ce module détermine la largeur de la fenêtre de corrélation. Avec cette structure, l'indice de corrélation pour un pixel est calculé efficacement en profitant des calculs réalisés sur les deux pixels précédents dans les flots, donc situés à gauche des pixels courants dans les deux images [Faugeras *et coll.* 1993].

Les indices de corrélation calculés par les modules de corrélation sont envoyés au module de sélection du minimum MI. Le minimum est déterminé en comparant les indices de corrélation deux à deux, ce qui permet de diminuer le temps de propagation. Des

étages pipeline ont été insérés après chaque comparaison afin d'augmenter la cadence de traitement. L'indice de corrélation minimum et le déplacement correspondant, c'est-à-dire la disparité estimée, sont envoyés à la sortie du module.

4.2.2 Implantation matérielle de l'algorithme SDA

Le code suivant montre l'implantation logicielle de l'algorithme SDA en langage C. Des équation récursives sont utilisées, comme cela a été décrit dans le chapitre 2 :

FOR $(y = y_{min}; y < y_{max}; y++)$ {
    FOR $(x = x_{min}; x < x_{max}; x++)$ {
        FOR $(s = 0; s < s_{max}; s++)$ {
            C(x,y,s) $= C(x-1, y, s) + U(x, y, s)$
        }
        $d = arg_{min} C(x, y, s)$;
    }
}

Dans une architecture classique, donc séquentielle, toutes les itérations doivent être calculées les unes après les autres. On peut cependant constater qu'il n'y a pas de dépendance de données entre chaque itération. Ainsi chaque indice de corrélation, correspondant à chaque déplacement, peut être calculé séparément et en parallèle comme le montre le code suivant :

FOR $(y = y_{min}; y < y_{max}; y++)$ {
    FOR $(x = x_{min}; x < x_{max}; x++)$ DO IN PARALLEL {
        C(x,y,s) $= C(x-1, y, s) + U(x, y, s)$;
        $d = arg_{min} C(x, y, s)$;
    }
}

Dans une architecture parallèle, une unité de traitement est affectée au calcul de chaque indice de corrélation. Tous les modules travaillent donc simultanément, mais sur des données différentes. Cette technique permet de calculer les disparités à la cadence vidéo quand les modules fournissent un résultat à chaque top d'horloge.

L'implantation matérielle de l'algorithme SDA peut être déterminée à partir du code précédent. La figure 4.6 présente les modules utilisés dans l'implantation de l'algorithme SDA avec $s_{max} = 2$, l'extension à une valeur supérieure étant simplement réalisée en augmentant le nombre de modules.

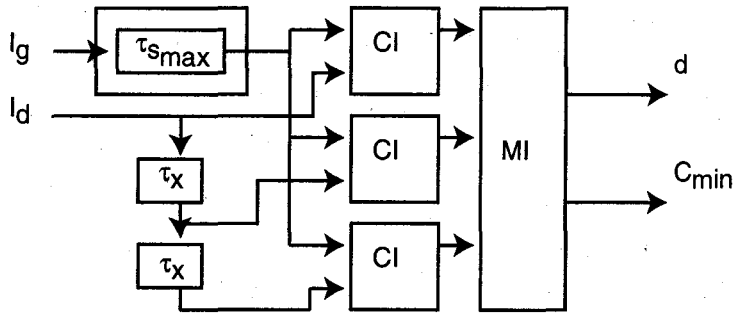


FIG. 4.6 : Implantation de l'algorithme SDA ($s_{max} = 2$).

Les pixels issus des deux caméras arrivent simultanément sur les entrées, des nouveaux pixels apparaissant à chaque top d'horloge envoyé par le module de numérisation. Cette même horloge cadence les opérations réalisées dans tous les modules de traitement. Des éléments de mémorisation sont insérés aux endroits adéquats pour assurer le décalage de la fenêtre de corrélation dans le flot de données qui correspond à l'image droite. Un module de corrélation CI est utilisé pour chaque déplacement s de la fenêtre de corrélation dans l'image droite. Tous les modules travaillent en parallèle et fournissent un résultat après chaque top d'horloge. Le module MI détermine le minimum parmi tous les indices de corrélation calculés. L'indice de corrélation minimum et le déplacement qui correspond à ce minimum sont envoyés à la sortie du module.

4.2.3 Implantation de la vérification directe-inverse

Comme nous l'avons montré dans le chapitre 2, la vérification directe-inverse peut être déterminée en utilisant les indices de corrélation calculés pour les pixels précédents de la même ligne. Cela peut s'implanter de façon efficace à partir des modules utilisés pour

l'implantation de l'algorithme SDA, auxquels il suffit d'adjoindre quelques composants, principalement des retards. La figure 4.7 présente l'implantation de l'algorithme SDA avec vérification directe-inverse.

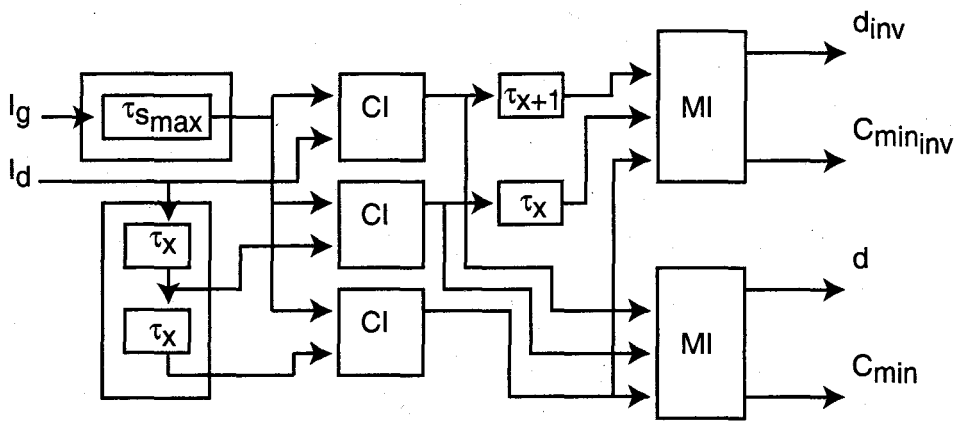


FIG. 4.7 : Implantation de l'algorithme SDA avec vérification directe-inverse ($s_{max} = 2$).

Les éléments ajoutés sont des retards de différentes profondeurs qui stockent les indices de corrélation calculés pour les pixels précédents, ainsi qu'un module MI qui détermine le minimum parmi les indices de corrélation et, par conséquent, la disparité inverse.

4.2.4 Implantation de l'algorithme de Hirschmüller

L'algorithme de Hirschmüller (HIR), dans une implantation avec des fenêtres unidimensionnelles, utilise simplement la fenêtre de corrélation centrale et plusieurs fenêtres supports de dimension $w_x \times 1$, situées sur la même ligne. Dans le cas le plus simple, il ne reste que deux fenêtres support, qui se chevauchent comme le montre la figure 4.8.

Il faut noter que l'indice de corrélation de la fenêtre support 1 correspond à l'indice de corrélation calculé pour le pixel $P(x - w_x - 1, y)$ et l'indice de corrélation de la fenêtre de support 2 correspond à l'indice de corrélation calculé pour le pixel $P(x + w_x + 1, y)$. A partir de cette observation, l'implantation de l'algorithme de Hirschmüller se fait de manière efficace en stockant les indices de corrélation calculés précédemment. La figure 4.9 présente le module utilisé pour l'implantation de l'algorithme HIR.

Ce module est composé de deux modules de mémorisation de taille $w_x + 1$ chacun,

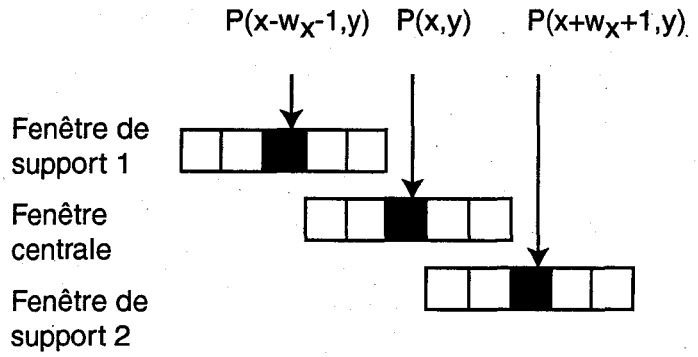


FIG. 4.8 : Fenêtres de support dans l'algorithme HIR.

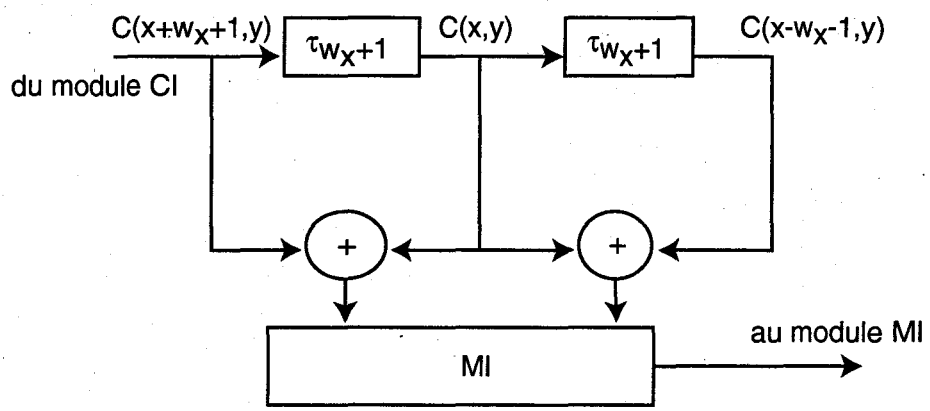


FIG. 4.9 : Module utilisé dans l'implantation de l'algorithme de Hirschmüller .

lesquels sont utilisés pour mémoriser les indices de corrélation calculés sur les pixels précédents. Deux additionneurs calculent la somme des indices de corrélation déterminés sur la fenêtre centrale et sur les fenêtres support. Dans l'implantation de l'algorithme SDA, ce module peut être inséré entre les modules de corrélation et le module qui détermine le minimum des indices de corrélation. Ainsi, la même structure de base permet l'implantation des deux algorithmes.

4.2.5 Implantation de l'algorithme SMW

L'algorithme SMW utilise plusieurs fenêtres dans lesquelles le pixel à apparier est placé en différentes positions. Avec une fenêtre rectangulaire de taille $w_x \times 1$, le nombre de fenêtres utilisées est ramené à 3, comme le montre la figure 4.10. Dans ces fenêtres, le pixel à mettre en correspondance est indiqué en noir.

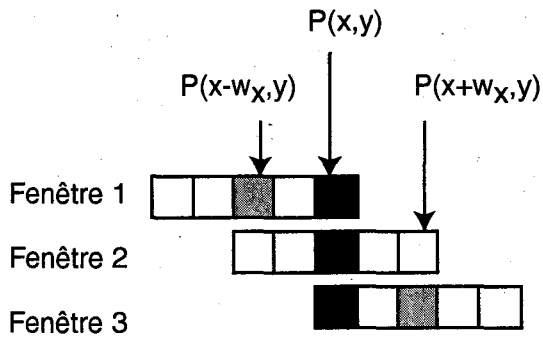


FIG. 4.10 : Fenêtres de support dans l'algorithme SMW.

Il faut noter que l'indice de corrélation de la fenêtre 1 correspond à l'indice de corrélation du pixel $P(x - w_x, y)$ et l'indice de corrélation de la fenêtre 3 correspond à l'indice de corrélation du pixel $P(x+w_x, y)$. Ainsi, l'implantation de l'algorithme SMW se fait de manière efficace en mémorisant les indices de corrélation calculés pour les pixels précédents. La figure 4.11 présente le module utilisé dans l'implantation de l'algorithme SMW.

Il est composé de deux modules de mémorisation de taille w_x utilisés pour mémoriser les indices de corrélation des pixels précédents et d'un module de comparaison pour choisir le plus petit indice de corrélation parmi les trois qui ont été calculés. Ce module peut être inclus juste après la sortie du module de corrélation dans l'implantation de l'algorithme SDA.

4.2.6 Implantation de l'algorithme Census

L'algorithme Census utilise une mesure non paramétrique pour rechercher la correspondance entre les pixels. En premier lieu, il faut déterminer la transformée Census pour

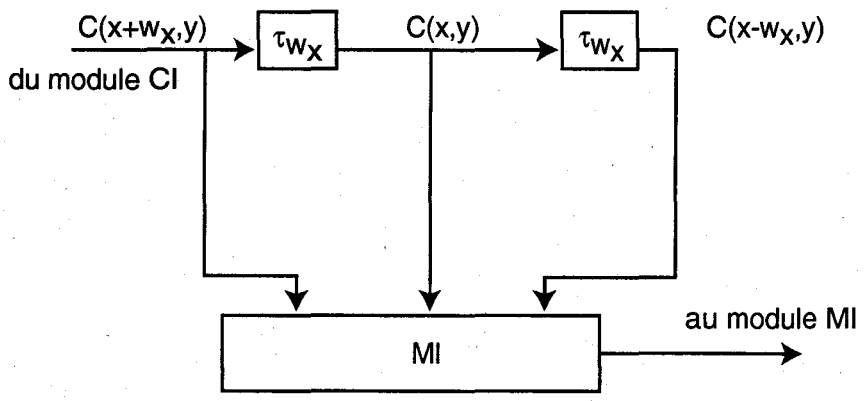


FIG. 4.11 : Implantation de l'algorithme SMW

chacun des pixels de la fenêtre. La figure 4.12 présente les modules utilisés pour le calcul de la transformée Census.

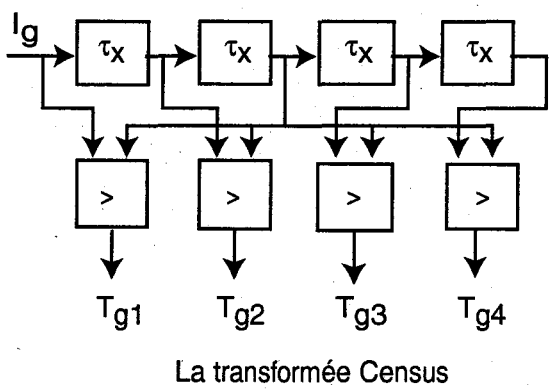


FIG. 4.12 : Module de calcul de la transformée Census (Cen).

Des modules de mémorisation ont été insérés afin d'accéder au voisinage du pixel de façon à calculer la transformée Census. Le pixel est placé au centre de la fenêtre et son

niveau de gris est comparé à celui des autres pixels de la fenêtre. A la sortie des modules de comparaison, l'ensemble des bits T_{gn} issus de chaque module correspond à la transformée Census du pixel central, où n est la taille de la transformée Census utilisée.

La similarité entre deux pixels est mesurée avec la distance de Hamming, c'est-à-dire, le nombre de bits différents dans les transformées Census correspondantes. La figure 4.13 présente les éléments nécessaires pour calculer la distance de Hamming. Dans cette figure, $\overline{XOR}$ désigne la fonction logique complément du "ou exclusif". Le calcul de la distance de Hamming est réalisé avec des additionneurs, déterminant la somme des valeurs égales à 1 à la sortie des portes logiques $\overline{XOR}$.

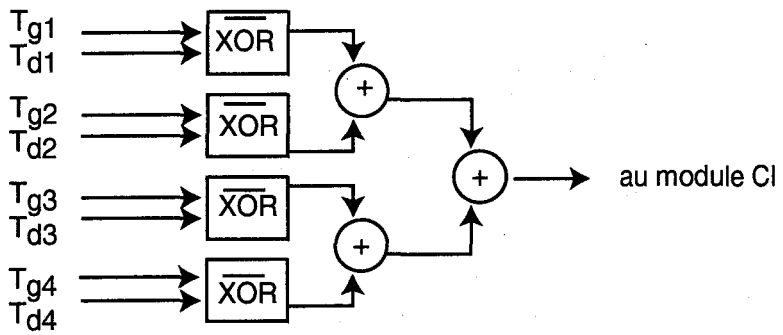


FIG. 4.13 : Module de calcul de la distance de Hamming (Ham).

Le nombre de bits issus du module de calcul de la distance de Hamming est inférieur à celui obtenu à la sortie du module qui calcule les différences absolues dans l'algorithme SDA. Ainsi, les modules de corrélation traitent également des données moins volumineuses, ce qui permet la réduction du nombre de ressources matérielles nécessaires à l'implantation.

La figure 4.14 présente les modules utilisés dans l'implantation de l'algorithme Census. Deux modules de calcul de la transformée Census (Cen) sont nécessaires, un pour chaque image. Les bits correspondants à la transformée Census de chaque pixel sont envoyés aux éléments de retard pour assurer le décalage de la fenêtre dans l'image droite. La distance de Hamming est calculée dans les modules Ham. Le nombre de ces modules et des modules

de calcul des indices de corrélation dépend de la disparité maximale qui peut être évaluée. À nouveau, le module MI détermine le minimum parmi les indices de corrélation calculés et fournit la disparité.

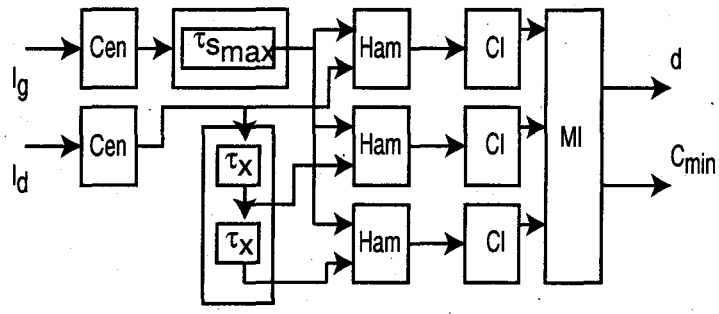


FIG. 4.14 : Implantation de l'algorithme Census ($s_{max} = 2$).

4.2.7 Implantation de l'algorithme SBAN

L'algorithme SBAN supprime de la fenêtre de corrélation les pixels qui ne sont pas similaires au pixel central. Dans le chapitre 3, nous avons décrit un critère de similarité basé sur la comparaison des niveaux de gris. Deux pixels sont similaires si leurs niveaux de gris sont égaux ou si l'écart entre leurs niveaux de gris est inférieur à un seuil. Le seuil est calculé en utilisant l'équation (4.1) :

$$T_g(x,y) = \frac{\sum_{i=-w_x}^{i=w_x} \sum_{j=-w_y}^{j=w_y} |I_g(x+i,y+j) - I_g(x,y)|}{(2w_x+1) \times (2w_y+1)} . \tag{4.1}$$

La figure 4.15 schématise le module qui calcule ce seuil.

Des éléments de mémorisation permettent d'accéder au voisinage du pixel central. Les modules DA calculent la différence absolue entre le niveau de gris du pixel central et le niveau de gris de chaque pixel de la fenêtre. Des additionneurs ont été utilisés pour calculer la somme des différences absolues dans la fenêtre.

Si la taille de la fenêtre est une puissance de deux, la division réalisée dans l'équation (4.1) peut être implantée par un simple décalage vers la droite du résultat obtenu à la

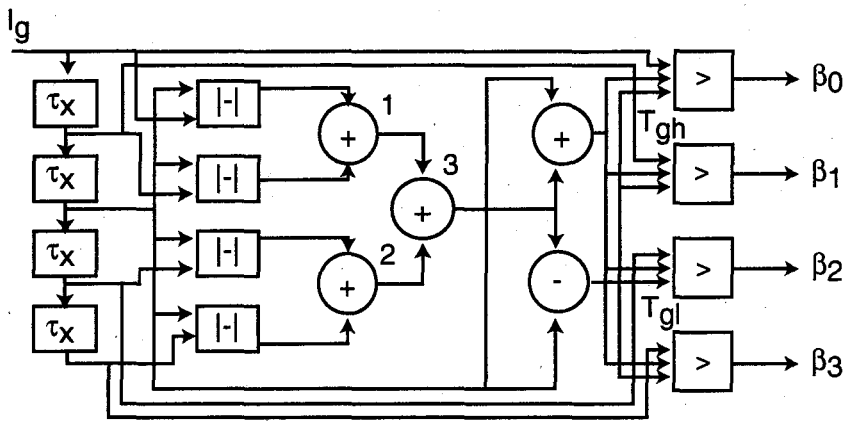


FIG. 4.15 : Module de calcul de la MDA (T_g).

sortie de l'additionneur 3. Cela correspond à prendre en compte uniquement les bits les plus significatifs du résultat.

Afin de simplifier l'implantation des modules suivants, deux valeurs additionnelles sont également calculées, qui correspondent aux seuils inférieur T_{gl} et supérieur T_{gh} par rapport au niveau de gris du pixel central $I_g(x, y)$:

$$T_{gh} = I_g(x, y) + T_g(x, y) \tag{4.2}$$

$$T_{gl} = I_g(x, y) - T_g(x, y). \tag{4.3}$$

Le module comparateur $>$ sélectionne les pixels qui interviennent dans le calcul de l'indice de corrélation. Il compare le niveau de gris d'un pixel de la fenêtre $P_g(x + i, y)$ avec les seuils T_{gl} et T_{gh} déterminés auparavant. Si le niveau de gris du pixel $P_g(x + i, y)$ est supérieur ou égal à T_{gl} et inférieur ou égal à T_{gh}, la sortie du comparateur est fixée à un. Elle vaut zéro dans le cas contraire. Une chaîne de bits β_n est ainsi obtenue, pour des valeurs de l'indice n comprises entre 1 et w_x. Dans cette chaîne, les bits égaux à un indiquent des pixels à utiliser tandis que les bits égaux à zéro indiquent les pixels à supprimer.

Dans l'algorithme SBAN, le voisinage utilisé pour mettre en correspondance un pixel peut être complètement différent du voisinage utilisé pour le pixel qui le précède. C'est la principale originalité de cette méthode adaptative. En revanche, la technique de calcul

récursive proposée dans [Faugeras *et coll.* 1993] ne peut pas être exploitée. En conséquence, un indice de corrélation doit être calculé indépendamment pour chaque pixel. La figure 4.16 décrit le module de corrélation utilisé dans l'implantation de l'algorithme SBAN.

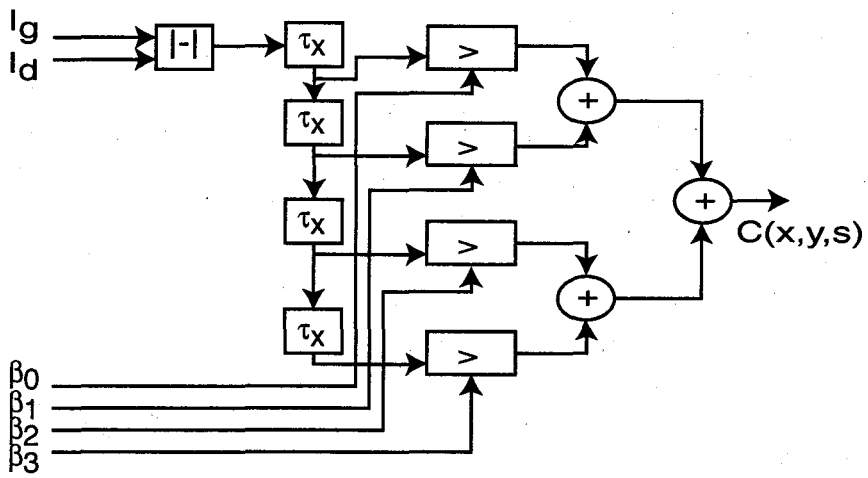


FIG. 4.16 : Le module de corrélation dans l'algorithme SBAN (CIS).

Un module $|-|$ calcule les différences absolues entre les pixels des images gauche et droite. Des éléments de mémorisation sont utilisés pour mémoriser les différences absolues. Le module comparateur $>$ laisse passer uniquement les différences absolues qui correspondent aux pixels conservés dans la fenêtre. Quand un pixel est éliminé de la fenêtre, la sortie du comparateur correspondant est égale à zéro. Ces pixels sont marqués en utilisant les bits de sélection β_n déterminés dans le module T_g. Des additionneurs ont été utilisés pour calculer la somme des différences absolues uniquement pour les pixels sélectionnés.

La figure 4.17 présente les modules utilisés dans l'implantation de l'algorithme SBAN. Des éléments de retard ont été insérés pour assurer le décalage de la fenêtre dans l'image droite. Le module T_g détermine les pixels qui doivent intervenir dans les calculs. Des modules CIS ont été insérés pour calculer les indices de corrélation correspondant à chaque décalage. Comme toujours, le module MI détermine le décalage qui minimise l'indice de corrélation.

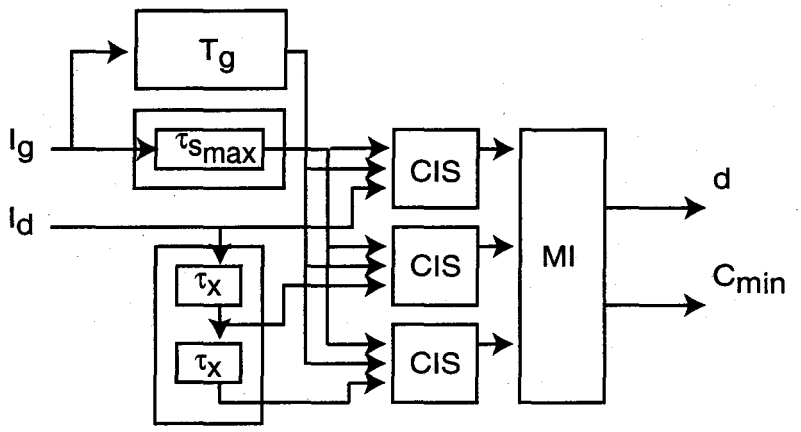


FIG. 4.17 : Implantation de l'algorithme SBAN ($s_{max} = 2$).

4.3 Comparaison des ressources utilisées

Le circuit FPGA intégré dans le processeur STREAM contient 51.480 éléments logiques et 442.368 bits de mémoire, ce qui correspond à environ un million et demi de portes logiques. Nous avons déterminé le nombre des ressources matérielles utilisées dans l'implantation de chaque algorithme. Les algorithmes ont été implantés en utilisant une fenêtre de taille 16×1, pour estimer une disparité maximale $s_{max} = 32$. Ces paramètres ont été retenus par rapport aux éléments logiques utilisés pour le contrôle de la carte, ce qui nécessite environ 50 % d'éléments logiques.

Le tableau 4.1 présente les résultats de ces implantations, en terme de ressources matérielles utilisées par chaque algorithme. Le nombre d'éléments logiques et le nombre de bits de mémoire nécessaires ont été utilisés pour la comparaison.

Algorithme	éléments logiques	bits mémoire
SDA	2 633 (5%)	4 532 (1%)
HIR	4 423 (8 %)	14 772 (3 %)
SMW	4 745 (9 %)	14 772 (3 %)
Census	7 188 (13 %)	3 924 (1%)
SBAN	18 328 (35 %)	180 (< 1%)

TAB. 4.1 : Ressources utilisées dans l'implantation des algorithmes.

Les structures FIFO, qui assurent le stockage des données intermédiaires dans les modules de corrélation, sont les mieux adaptées à une implantation dans les blocs ESB. Ces structures interviennent dans les implantations des algorithmes SDA, HIR, SMW et Census. Malheureusement, l'implantation de l'algorithme SBAN ne peut tirer partie de ces structures de mémoire. C'est la raison pour laquelle le nombre de bits de mémoire utilisés dans l'implantation de l'algorithme SBAN est toujours relativement faible.

Le nombre de ressources matérielles nécessaires à l'implantation de l'algorithme HIR est supérieur à celui utilisé pour l'implantation de l'algorithme SDA. Cela s'explique parfaitement puisque des éléments de mémorisation, des additionneurs et des comparateurs additionnels sont nécessaires dans l'implantation de l'algorithme HIR.

Malgré l'utilisation d'un nombre inférieur de composants additionnels dans l'implantation de l'algorithme SMW, les ressources matérielles nécessaires sont supérieures à celles utilisées dans l'implantation de l'algorithme HIR. La complexité de la fonction de comparaison dans le module intervenant dans l'algorithme SMW est à l'origine de ce problème. Dans l'algorithme SMW, chaque entrée est comparée avec les deux autres pour déterminer le minimum, tandis que dans l'algorithme HIR, une seule comparaison doit être réalisée.

Il faut noter que les deux algorithmes utilisent des nombres équivalents de bits de mémoire, malgré la différence de taille des éléments de mémorisation. Cela montre que les bits de mémoire ESB sont effectivement utilisés par blocs de taille prédéterminée et que dans l'implantation de l'algorithme SMW certains bits de mémoire ne sont pas réellement utilisés dans l'algorithme. Malgré tout, ces bits sont inexploitables pour l'implantation d'autres fonctions.

L'algorithme Census nécessite plus de ressources matérielles que les autres algorithmes, du fait de la complexité de la transformée Census. En effet, bien que le résultat soit un bit unique, les comparateurs intervenant dans la transformée utilisent autant de portes logiques que les additionneurs permettant le calcul de la différence absolue entre deux pixels. Par contre, le nombre de bits de mémoire diminue de façon notable, car les données traitées dans les modules sont codées sur moins de bits.

Enfin, les ressources utilisées pour l'implantation de l'algorithme SBAN sont les plus élevées en comparaison avec toutes les autres techniques. En effet, les modules de calcul des indices de corrélation prennent beaucoup de place dans le circuit FPGA, du fait que

les techniques de calcul récursif sont inutilisables. Cependant, la meilleure qualité de la carte de disparité obtenue justifie l'utilisation des ressources matérielles additionnelles. Le nombre de bits de mémoire est réduit par rapport aux autres implantations, du fait que les éléments de mémorisation d'un seul pixel ne sont pas implantés dans les blocs ESB.

4.4 Conclusion

Les algorithmes que nous avons sélectionnés peuvent être implantés de manière efficace dans la carte STREAM. Nous avons déduit l'implantation de la plupart des algorithmes de celle qui a été proposée pour l'algorithme SDA. Dans chaque cas, quelques composants additionnels sont nécessaires pour transformer le schéma de base.

L'utilisation des grandes fenêtres de corrélation est conseillée pour l'algorithme SBAN, principalement pour traiter des images présentant des zones uniformes. Pour cet algorithme, les ressources matérielles utilisées sont plus importantes que celles qui sont requises par les autres algorithmes. Reste à déterminer une technique d'implantation qui permet de réduire le nombre de ressources utilisées, tout en conservant l'efficacité de la méthode.

Le LAGIS a participé activement à la recherche de solutions techniques, exploitant la vision artificielle, qui permettent d'améliorer la sécurité dans les systèmes de transport. Notre algorithme original de stéréovision peut être utilisé dans ce cadre. Cela fait l'objet du chapitre suivant de ce mémoire.

Chapitre 5

Application à la détection d'obstacles

Depuis de nombreuses années, le Laboratoire d'Automatique, Génie Informatique et Signal (LAGIS), avec ses partenaires scientifiques et industriels, a consacré une partie de ses travaux de recherche à l'amélioration de la sécurité de la conduite des véhicules terrestres. Cet objectif est atteint grâce à la perception de l'environnement du véhicule opérée par un système d'assistance à la conduite, lequel envoie des informations pertinentes au conducteur, principalement lorsqu'il détecte une situation dangereuse. C'est le cas lorsqu'un obstacle est repéré sur la trajectoire du véhicule, soit sur la chaussée pour un véhicule automobile, soit sur les voies pour un véhicule guidé. Afin de détecter les obstacles potentiels situés à l'avant du véhicule, plusieurs systèmes de stéréovision ont été développés et testés dans notre laboratoire.

Les premiers travaux, qui ont fait l'objet de plusieurs thèses de Doctorat, ont été menés dans le cadre du projet Eureka PROMETHEUS qui a pris fin en 1994. Bruyelle a proposé un système stéréoscopique composé de deux caméras linéaires [Bruyelle 1994], dont les images ont été traitées par l'algorithme original de mise en correspondance proposé par Burie [Burie 1995]. Les caméras linéaires, qui ne comportent qu'une seule ligne de pixels mais qui compensent cette limitation par une cadence de fonctionnement élevée, se sont avérées très efficaces pour la détection des obstacles. Sur la base du même banc stéréo linéaire, Ruichek a développé un algorithme de mise en correspondance de primitives exploitant une technique d'optimisation globale par réseau de neurones de Hopfield [Ruichek 1997].

Pour des raisons de coût et de facilité d'intégration, les caméras linéaires ont par la suite

été remplacées par des des caméras matricielles standards. Ces travaux ont été réalisés dans le cadre d'un projet PREDIT 2 baptisé STATUE, qui est décrit plus précisément dans la suite de ce chapitre. Dooze a appliqué les techniques de mise en correspondance développées par Ruichek sur plusieurs lignes extraites des deux images matricielles [Dooze 2001]. Une méthode de mise en correspondance globale, utilisant cette fois des algorithmes génétiques, a été proposée ensuite par Issa [Issa 2004].

Dans toutes ces études, afin de diminuer le temps de calcul, des techniques de mise en correspondance de primitives étaient appliquées sur un faible nombre de lignes. En effet, même en employant les processeurs les plus rapides disponibles à l'époque, il s'avérait impossible de traiter la totalité des lignes des images du couple stéréoscopique à la cadence vidéo. En exploitant les algorithmes de stéréovision dense, notamment l'algorithme SBAN, fonctionnant à la cadence vidéo sur le processeur STREAM, cette limitation peut être contournée.

Dans ce chapitre, nous présentons les résultats obtenus par l'algorithme SBAN en termes de détection d'obstacles. Pour l'instant, le processeur STREAM n'ayant pas été intégré sur véhicule, ces résultats ont été calculés hors-ligne, sur des images acquises lors des expérimentations ou sur des images synthétiques. Dans une première partie, consacrée au projet STATUE, nous traitons les images acquises par deux caméras installées à l'avant d'une rame du métro automatique VAL. Ensuite, dans une partie consacrée au projet RaViOLi (Radar et Vision Orientables, Lidar), nous traitons des images de synthèse, simulant celles qui seront fournies par le banc stéréo mono-caméra développé dans notre laboratoire.

5.1 Projet STATUE

Le projet STATUE a débuté en janvier 1999 et s'est terminé en décembre 2001. Ce projet, qui a fait intervenir sept partenaires dont le LAGIS, était inscrit dans le cadre du PREDIT 2. L'objectif principal de STATUE était d'évaluer l'intérêt d'intégrer de nouvelles technologies de communication train-infrastructure dans un véhicule guidé. Notamment, il s'agissait d'évaluer les techniques de transmission d'images vidéo depuis le véhicule vers le poste de contrôle-commande (PCC), afin de permettre à un opérateur humain de

prendre le relais des automatismes quand le système fonctionne en mode dégradé.

Une application directe de cette technique est l'optimisation de l'accostage de deux rames. Lorsqu'un véhicule automatique tombe en panne sur la voie, un autre véhicule similaire vient l'accoster, puis le pousse vers la prochaine station. Dans les systèmes actuels, l'opérateur qui contrôle le véhicule durant l'accostage travaille en aveugle. D'autre part, la position du véhicule en panne sur les voies est connue de façon très approximative. De ce fait, la procédure consiste à approcher du train en panne à très faible vitesse jusqu'à ce que les deux véhicules entrent en contact, ce qui peut prendre un temps non négligeable. On comprend alors l'intérêt d'envoyer des images de la scène située à l'avant du véhicule vers le PCC durant la phase d'accostage.

5.1.1 Détection d'obstacles à l'avant du VAL

Puisque des caméras sont installées à l'avant du train, il est intéressant d'exploiter les images qu'elles acquièrent, même lorsque le véhicule fonctionne en mode automatique. Par exemple, un traitement adéquat des images permet de détecter la présence d'éventuels obstacles sur les voies. Normalement, ces dernières sont protégées et le véhicule est muni d'un chasse-corps, mais des objets volumineux ou dangereux peuvent parfois être déposés sur les voies par des vandales.

Le VAL est équipé d'un détecteur de chocs situé à l'avant de la rame. Il s'active dès qu'un obstacle a été percuté, entraînant un arrêt d'urgence. Lorsque le véhicule est arrêté, il faut faire intervenir une équipe pour vérifier l'état de la rame, ainsi que le type d'obstacle percuté, avant de reprendre l'exploitation. Ce mode opératoire est coûteux et les durées d'immobilisation du véhicule sont généralement très longues. En utilisant les caméras vidéo et la transmission d'images vers le PCC, un opérateur peut déterminer à distance le type d'obstacle qui a été percuté par la rame, ce qui constitue déjà une amélioration notable de l'efficacité de gestion des collisions.

Naturellement, cette méthode n'est pas la plus adaptée car les obstacles devraient être détectés avant que la rame ne les percute. C'est sur cet aspect particulier du projet STATUE qu'ont travaillé les chercheurs du LAGIS et du LEOST (Laboratoire Électronique Ondes et Signaux pour les Transports) de l'INRETS. Dans le système développé durant le projet STATUE, les techniques de stéréovision sont exploitées afin de déterminer la

distance des objets détectés sur les voies. Un système stéréoscopique composé de deux caméras matricielles a été installé à l'avant d'une rame expérimentale, comme le montre la figure 5.1.

FIG. 5.1 : Le système stéréoscopique

Ces deux caméras acquièrent des images de la scène située à l'avant du véhicule, à une distance suffisante pour que la détection d'un obstacle sur les voies puisse être prise en compte dans le pilotage du véhicule. De ce fait, elles ne surveillent pas la zone située à proximité du véhicule. Pour pallier cet inconvénient, une troisième caméra a été ajoutée lors des dernières expérimentations. Elle est placée au milieu des deux caméras de stéréovision, son axe optique étant plus incliné vers les voies. Le système fournit donc trois images des voies, comme le montre la figure 5.2.

5.1.2 Détection d'obstacles dans le projet STATUE

La définition du terme obstacle dépend largement de l'application visée et de l'environnement dans lequel évolue le véhicule. De ce fait, différentes techniques ont été décrites

FIG. 5.2 : Les trois images acquises par le système

dans la littérature pour détecter les obstacles dans un système de vision artificielle. Une revue de ces méthodes et des techniques de traitement associées est disponible dans [Bertozzi et Broggi 1998]. Dans le cas du projet STATUE, comme le véhicule évolue sur une voie ferrée, on considère qu'un obstacle est un objet situé entre les rails et de hauteur importante, c'est-à-dire qui dépasse de la surface réglée définie par les voies [Dooze 2001].

Il s'agit donc dans un premier temps de déterminer cette surface réglée, qui est constituée de l'ensemble des segments de droite reliant un point du rail de gauche au point correspondant du rail de droite. Pour ce faire, Dooze a utilisé les images fournies par la caméra centrale, dans lesquelles il détecte les rails en les modélisant par des contours actifs. Cette technique simple, qui utilise l'image des voies, a été préférée à celle qui exploiterait leur topographie réelle, qui a été définie lors de la construction de la voie ferrée. Ce choix se justifie par le fait que le VAL n'est pas localisé précisément par rapport à son environnement, ce qui interdit l'utilisation directe d'une carte du site.

Ensuite, les images fournies par les caméras gauche et droite sont traitées afin d'estimer une carte des disparités. La méthode utilisée est éparse car elle met en correspondance des primitives extraites des images, en l'occurrence des contours. Les contours sont rehaussés au moyen de l'opérateur de Deriche, puis marqués afin de définir les primitives. La mise en correspondance est recherchée par la méthode de Ruichek, au moyen d'un réseau de neurones de Hopfield qui minimise une fonction d'énergie incluant différents termes [Ruichek *et coll.* 1998].

Afin de simplifier la procédure, les chercheurs ont considéré que la voie ferrée est plane. Dans ce cas, la surface réglée qui définit la hauteur maximale d'un objet présent entre les

voies sans qu'il ne constitue un obstacle est également plane. Ce plan est utilisé comme référence d'altitude, tout objet situé au dessus constituant un obstacle potentiel. Cela est illustré par la figure 5.3, dans laquelle h_o est la hauteur d'un obstacle présent sur la voie et h_r la hauteur du rail.

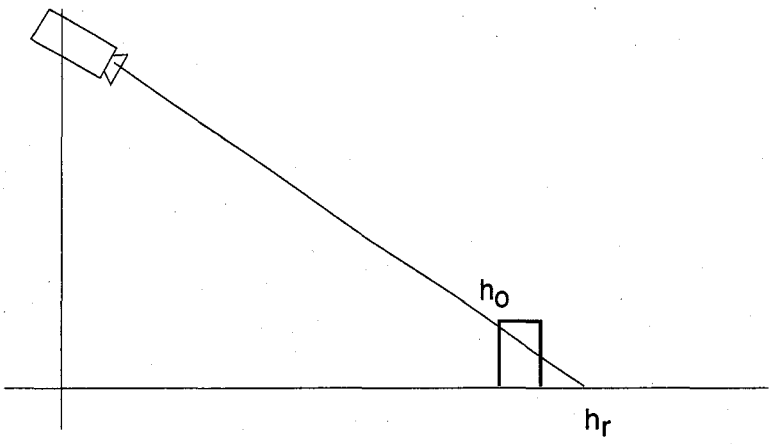


FIG. 5.3 : Détection d'obstacles

Le trait oblique représenté sur cette figure indique le chemin optique d'un rayon issu de la scène, lequel se projette sur une ligne particulière de l'image. Quand il n'y a pas d'obstacle sur les voies, on vérifie aisément que la distance entre la caméra et le point de la scène dont est issu ce rayon reste constante. De ce fait, la disparité estimée pour les pixels homologues correspondants reste également constante. Par contre, quand un obstacle est présent sur la voie, ce même chemin optique l'intercepte à une distance plus faible. La disparité estimée pour les pixels homologues correspondant à un point de l'obstacle, toujours situés dans la même ligne des images, est alors plus élevée.

En utilisant cette propriété, on peut déterminer a priori la disparité qui est obtenue pour chaque ligne des images en absence d'obstacle. Cette disparité est faible pour les lignes situées en haut des images, lesquelles correspondent à une zone éloignée du véhicule, et augmente progressivement avec le numéro de ligne. Après avoir calibré le banc stéréo installé sur le VAL, Dooze a obtenu la relation suivante :

$$d_r(y) = \frac{y - 106.6}{2.3}\,, \tag{5.1}$$

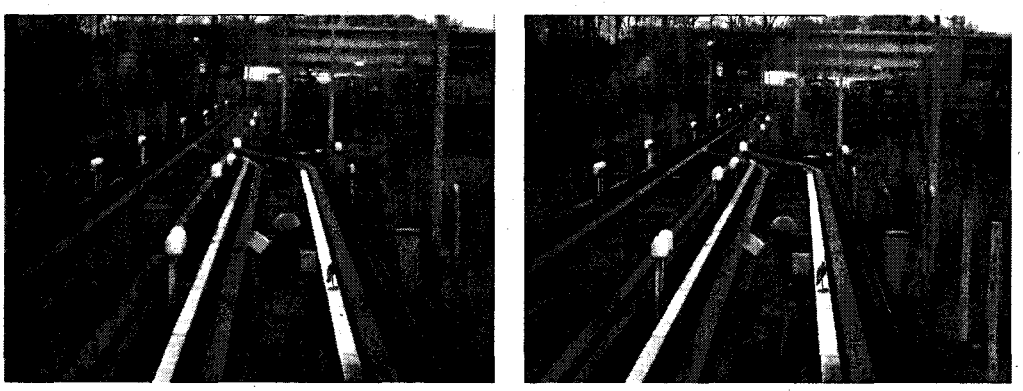
qui permet de calculer la disparité $d_r(y)$ correspondant à l'image des rails pour chaque ligne y des images.

Afin de détecter les obstacles, on compare la disparité estimée pour chaque couple de pixels homologues à cette disparité limite. Si elle est supérieure, on considère que le point correspondant dans la scène appartient à un obstacle. Cette comparaison est réalisée ligne par ligne. D'autre part, les points situés à l'extérieur de la zone délimitée par les rails ne sont pas marqués en tant qu'obstacle.

5.1.3 Utilisation de l'algorithme SBAN

Un algorithme de stéréovision dense comme le SBAN, une fois implanté dans le processeur STREAM afin de fonctionner à la cadence vidéo, peut être avantageusement utilisé dans cette application. Nous présentons ici des résultats préliminaires, obtenus sur des images enregistrées lors d'une expérimentation menée sur le site d'essai de TRANSPOLE, exploitant actuel du VAL dans la métropole Lilloise.

La figure 5.4 présente une paire d'images extraite de cette séquence. Trois obstacles ont été placés sur les voies, à savoir une boîte en carton placée à proximité du rail gauche, une autre à proximité du rail droit et un ballon placé à mi-distance entre les rails. Les boîtes sont plus proches du véhicule que le ballon. Un quatrième obstacle, un pigeon empaillé, a été placé sur le rail de droite. On note également la présence d'une personne à une distance plus importante, qui ne constitue pas un obstacle puisqu'elle est située à l'extérieur de la zone dangereuse.



(a) Image gauche (b) Image droite

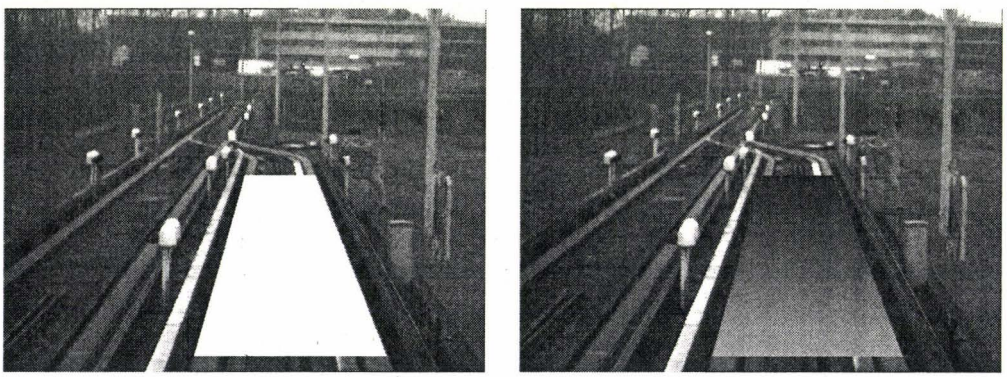
FIG. 5.4 : Paire d'images de la voie comportant des obstacles

La figure 5.5.(a) présente la zone à surveiller comprise entre les rails, laquelle a été déterminée en suivant la procédure proposée par Dooze. Cette zone est limitée à gauche et à droite par deux lignes droites d'équations respectives :

$$y = -3.58 \times x + 1648.2; \tag{5.2}$$

$$y = 2.55 \times x - 1009.9; \tag{5.3}$$

La figure 5.5.(b) présente la disparité limite calculée pour chaque ligne de l'image. Nous reprenons les mêmes paramètres que ceux qui ont été utilisés par Dooze dans sa thèse, notamment la relation de l'équation 5.1 qui définit la disparité limite pour chaque ligne de l'image.

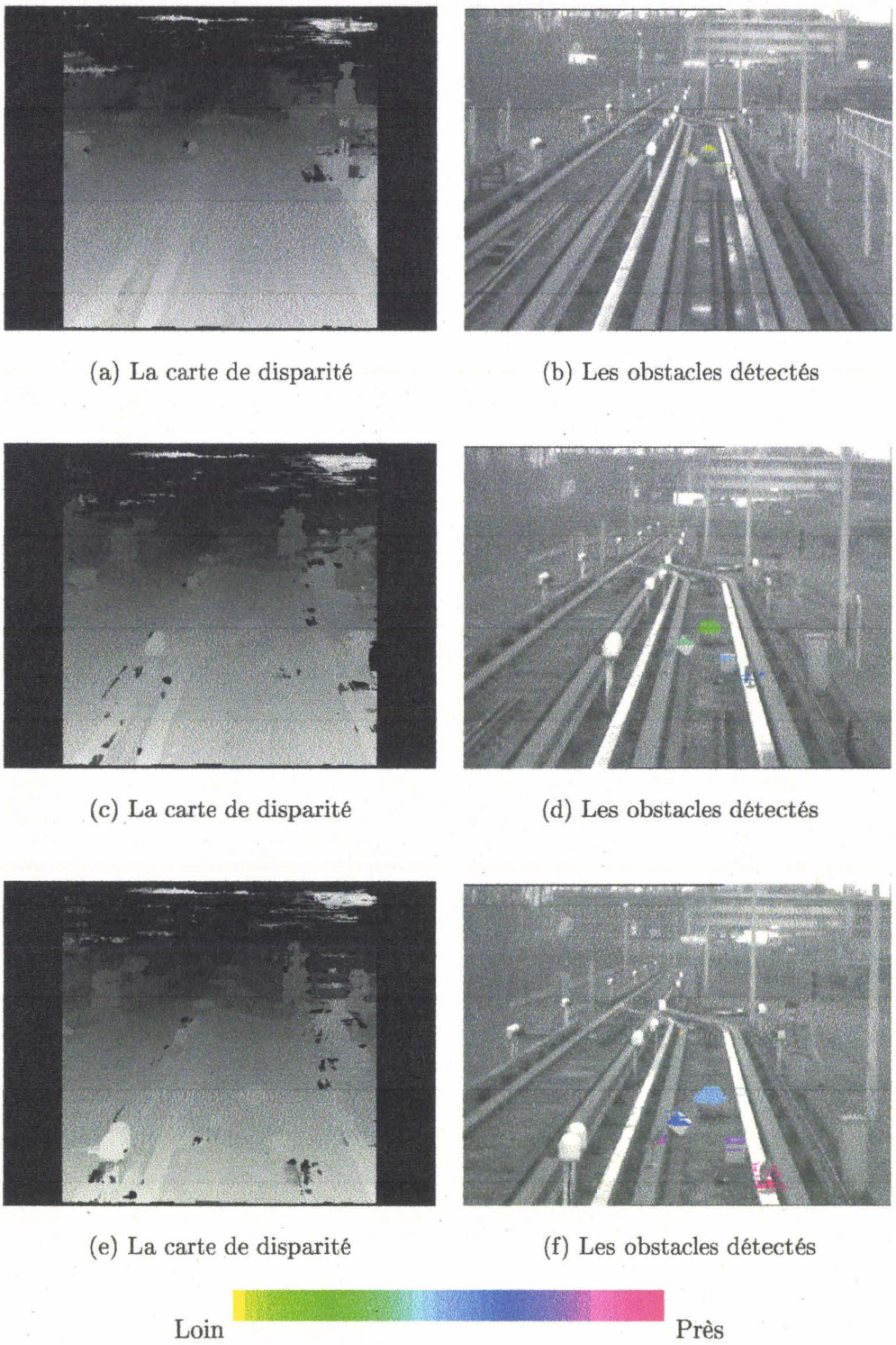


(a) Zone à surveiller (b) Disparité estimée pour chaque ligne

FIG. 5.5 : Zone de surveillance située entre les rails

Les figures 5.6.(a),5.6.(c) et 5.6.(e) présentent les cartes de disparité obtenues avec l'algorithme SBAN pour différentes images extraites de la séquence. Une grande fenêtre de taille 40×10 a été utilisée afin d'éliminer au maximum les problèmes d'appariement dans les rails, qui ont un niveau de gris homogène. Les figures 5.6.(b), 5.6.(d) et 5.6.(f) présentent les obstacles détectés au moyen de la procédure décrite dans la section précédente. Un dégradé de couleurs donne une indication de la distance des obstacles détectés dans la scène. La couleur jaune désigne les objets éloignés du véhicule et la couleur rouge les objets proches.

Les cartes de disparité obtenues sont précises, même dans les zones présentant une faible variation du niveau de gris, grâce à l'utilisation d'une fenêtre de corrélation de



(a) La carte de disparité (b) Les obstacles détectés

(c) La carte de disparité (d) Les obstacles détectés

(e) La carte de disparité (f) Les obstacles détectés

FIG. 5.6 : La détection d'obstacles

grande taille. Cet exemple montre l'intérêt d'utiliser l'algorithme SBAN sur une grande fenêtre, laquelle est nécessaire en présence de larges zones homogènes. Cependant, grâce

à la possibilité d'adapter le voisinage utilisé pour la mise en correspondance, les petits objets peuvent être détectés. Les obstacles sont ainsi détectés efficacement, y compris dans l'image de la figure 5.6.(a), dans laquelle la disparité correspondant aux objets et aux rails sont très similaires. Cette figure montre que le ballon, représenté en jaune, est plus éloigné du véhicule que les autres objets, représentés en vert. Les images suivantes montrent que les obstacles se rapprochent de la rame, ce qui se vérifie par la couleur utilisée pour les représenter.

De fausses détections peuvent apparaître dans les images, notamment quand les disparités calculées pour les pixels sont très proches de la disparité limite déterminée par le plan des rails. Dans l'application définitive, il faudra introduire une procédure de validation des détections, tirant parti de plusieurs informations additionnelles.

Par exemple, la procédure de validation des détections peut utiliser le nombre de pixels connexes pour indiquer la pertinence de l'obstacle détecté. Le degré de pertinence accordé à chaque appariement peut également être utilisé dans la procédure de validation d'obstacles, comme cela a été décrit dans le chapitre 2. Enfin, une technique de validation couramment utilisée est la persistence de la détection, c'est-à-dire le fait qu'elle apparaît dans un certain nombre d'images successives de la séquence.

5.2 Projet RaViOLi

Depuis 2003, notre laboratoire est impliqué dans un projet baptisé RaViOLi, acronyme de Radar et Vision Orientables, Lidar. Ce projet, qui se terminera en avril 2005, est financé par la Région Nord-Pas de Calais et le FEDER dans le cadre du Contrat de Plan État-Région. Son objectif est également d'améliorer la sécurité de la conduite en percevant l'environnement du véhicule, cette fois à longue distance, au moyen d'un ensemble de capteurs. Il fait intervenir neuf partenaires régionaux, sept laboratoires de recherche et deux industriels.

Les capteurs développés ou utilisés dans RaViOLi sont originaux du fait qu'il est possible d'orienter leurs axes de perception vers une zone bien définie de la scène. Habituellement, les capteurs installés sur un véhicule ont un axe de perception — axe optique pour les caméras, direction principale du faisceau pour les radars — qui est fixe et orienté

selon l'axe du véhicule. Quand il s'agit de percevoir l'environnement à longue distance, cette configuration n'est pas du tout adaptée. Pour s'en convaincre, il suffit d'imaginer les performances d'un conducteur humain qui ne pourrait pas modifier l'orientation de son regard lorsqu'il approche d'un virage.

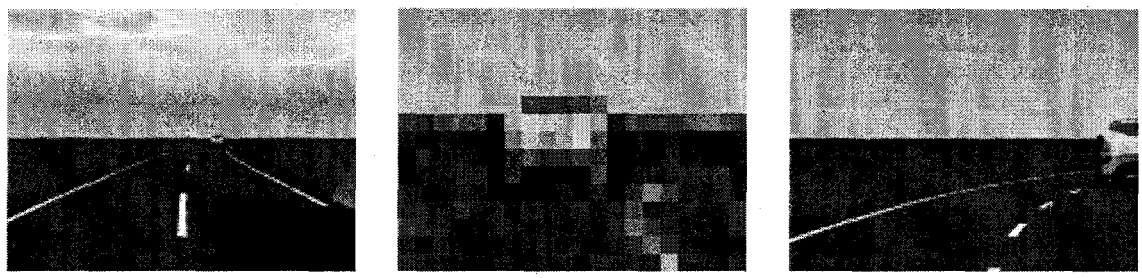
Le projet est composé de trois volets qui s'attachent chacun à un aspect particulier de la perception à longue distance. L'un est dédié au développement d'un banc stéréo mono-caméra à longue portée, dont les axes optiques sont orientables. Ce capteur original est décrit plus précisément par la suite. Un deuxième volet est consacré à la détection d'obstacles au moyen des capteurs actifs de type radar et lidar. L'objectif principal de ce volet est d'évaluer et de comparer les différentes technologies de radar à balayage. Enfin, un volet étudie les techniques de fusion bas niveau des données issues de ces multiples capteurs, en vue d'une implantation en temps-réel des traitements.

5.2.1 Capteur stéréoscopique mono-caméra actif à axes orientables

De nombreux systèmes stéréoscopiques dédiés à la détection des obstacles potentiels à l'avant d'un véhicule ont été décrits dans la littérature. Cependant, très peu de chercheurs ont abordé le problème de l'orientation de l'axe optique des caméras constituant ces bancs stéréo.

Le problème posé par des caméras dont l'axe optique est fixe et orienté selon l'axe du véhicule apparaît clairement sur la figure 5.7. Si on utilise des caméras équipées d'objectifs grand angle, on obtient une vue globale de la scène, comme le montre l'image (a). En revanche, les obstacles situés à une distance importante sont décrits par un faible nombre de pixels (cf. figure 5.7.(b)), ce qui interdit leur identification. A l'opposé, si on utilise un objectif de longue distance focale, les obstacles potentiels peuvent sortir du champ de vision dans certaines situations, par exemple à l'abord d'un virage (cf. figure 5.7.(c)).

On comprend ainsi l'intérêt de changer l'orientation de l'axe optique des caméras, afin d'observer une zone particulière de la scène. Pour ce faire, on peut les installer sur une platine motorisée. Cette solution comporte d'autres inconvénients, notamment dans le cas des bancs stéréo. Si on veut conserver un calibrage parfait du banc stéréo, il faut faire pivoter le système complet, dont la masse, donc l'inertie, est non négligeable. Si on



(a) Grand angle (b) Détail du véhicule (c) Télé-objectif

FIG. 5.7 : Perception à longue distance, axe optique fixe

fait pivoter les deux caméras, il faut utiliser des systèmes d'asservissement de la position angulaire extrêmement précis afin de conserver un calibrage adéquat.

Pour pallier ces problèmes, un système stéréoscopique dont l'axe optique est orientable a été développé dans notre laboratoire [Duvieubourg *et coll.* 2004]. Ce système est composé de deux miroirs, d'un prisme et d'une seule caméra, comme le montre la figure 5.8. Le couple d'images est obtenu grâce à l'utilisation des miroirs latéraux, qui reflètent les rayons vers le prisme puis vers l'une ou l'autre moitié du capteur d'image de la caméra. L'image gauche est obtenue dans la moitié gauche de l'image réelle, tandis que l'image droite est obtenue dans la moitié droite.

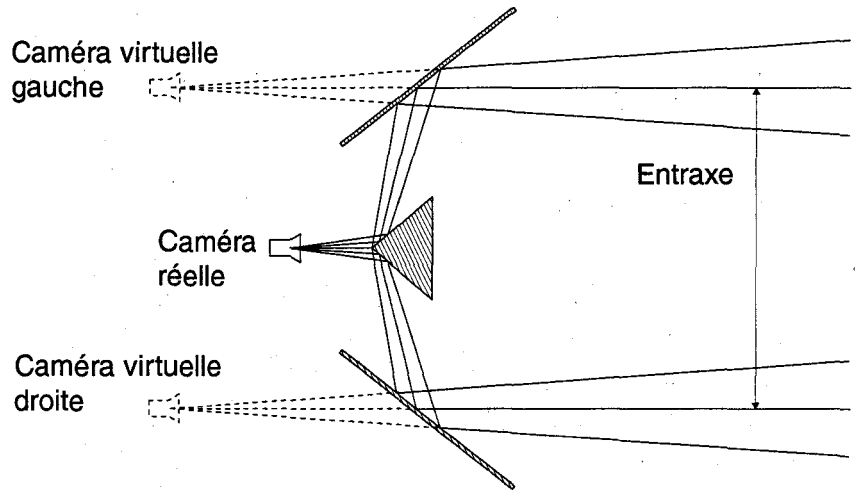


FIG. 5.8 : Le capteur stéréoscopique mono-caméra actif avec prisme

Des systèmes similaires ont été proposés auparavant [Lee *et coll.* 1999, Mathieu et

Devernay 1995]. Cependant, l'originalité du nouveau système réside dans la possibilité de faire pivoter uniquement le prisme central afin de suivre la zone d'intérêt. La figure 5.9 présente le prototype de ce capteur stéréoscopique mono-caméra, qui intègre les miroirs latéraux, le prisme et la caméra. Durant le calibrage, la position et l'orientation des miroirs latéraux sont ajustées manuellement au moyen de platines graduées. L'objectif principal du calibrage est d'obtenir deux caméras virtuelles dont les axes optiques sont parallèles. Le prisme est monté sur une platine qui pivote grâce à un moteur à pas. Quand le système est calibré, les axes optiques restent parallèles même après rotation du prisme.

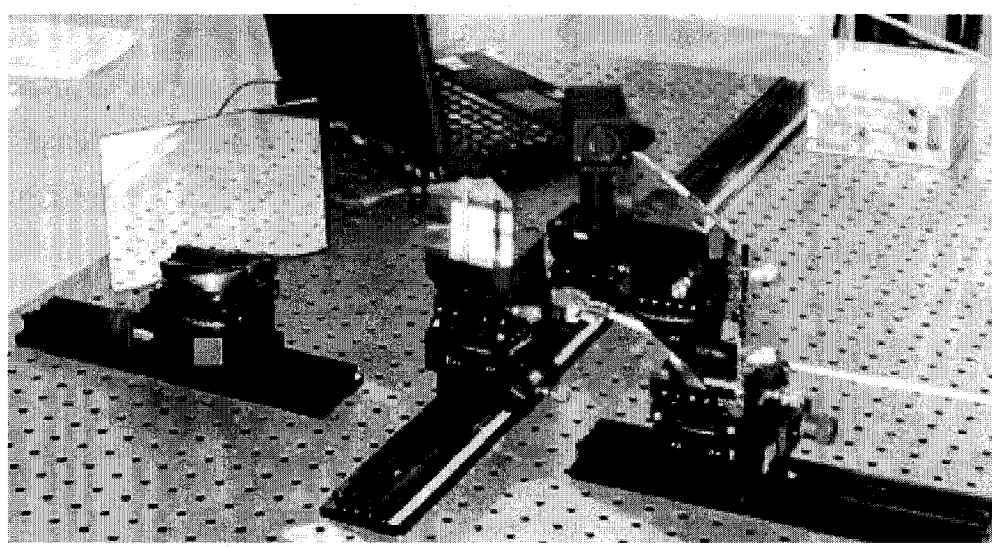


FIG. 5.9 : Photographie du prototype

Afin d'obtenir des images d'une scène routière sans installer le banc stéréo sur un véhicule, son fonctionnement a été simulé au moyen d'un logiciel de synthèse d'images, PovRay [1]. La figure 5.10 présente une image synthétique obtenue par cette simulation. La moitié gauche de l'image correspond à l'image gauche du banc stéréo, équivalent à celle qui serait obtenue par la caméra virtuelle de gauche, et la moitié droite correspond à l'image droite. Les images présentées ont subi un post-traitement qui consiste à corriger les distorsions géométriques introduites par le système optique du banc stéréo mono-caméra.

[1] http ://www.povray.org/

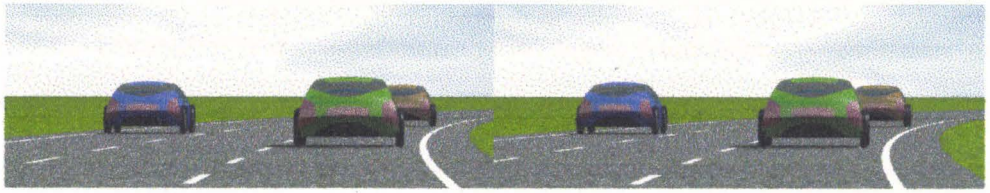


FIG. 5.10 : Scène synthétique

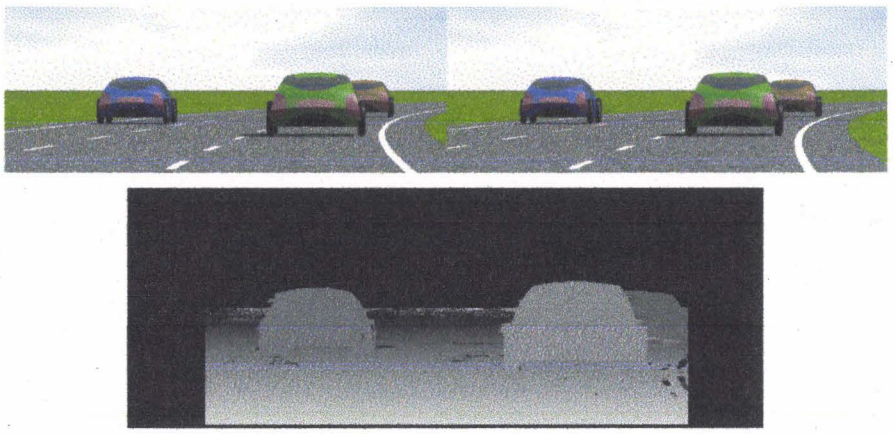
5.2.2 Utilisation de l'algorithme SBAN

Les algorithmes de stéréovision dense présentés dans cette thèse, notamment l'algorithme SBAN, seront utilisés afin de détecter les obstacles dans les images acquises par le banc stéréo. Cette phase expérimentale du projet n'a pas encore été menée à bien au moment de la rédaction de ce mémoire. De ce fait, afin de vérifier les performances de notre algorithme, nous avons travaillé sur des séquences d'images synthétiques.

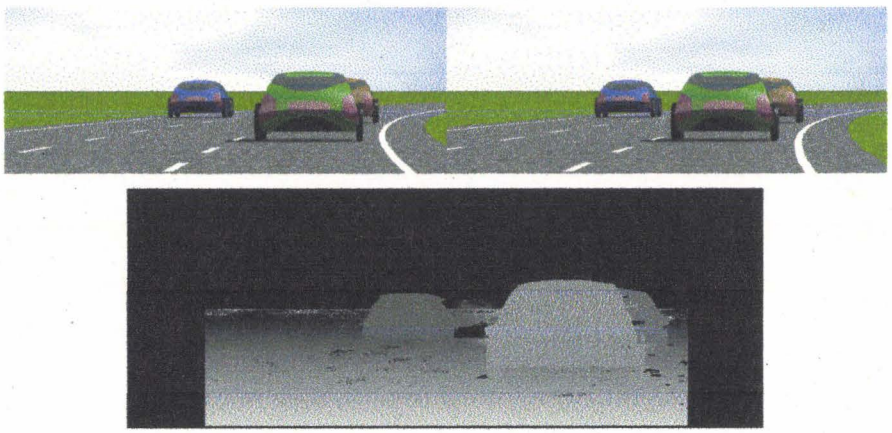
Les figures 5.11.(a), 5.11.(b) et 5.11.(c) présentent trois paires stéréoscopiques extraites d'une séquence simulant une situation de conduite sur une autoroute à trois voies. Quatre voitures se déplacent à différentes vitesses sur cette autoroute. Le capteur stéréoscopique mono-caméra est installé dans la quatrième voiture, qui n'apparaît donc pas dans les images. La première voiture, de couleur bleue, circule à grande vitesse sur la voie de gauche et dépasse les autres véhicules. La deuxième voiture, de couleur verte, est située sur la voie de droite. Elle se rapproche de la troisième voiture, de couleur beige, laquelle circule sur la même voie mais à plus faible vitesse. La quatrième voiture, qui circule légèrement plus vite et également sur la voie de droite, se rapproche donc progressivement des voitures verte et beige.

Dans cette figure, nous présentons également les cartes de disparité obtenues avec l'algorithme SBAN. Une fenêtre de corrélation de taille 41×5 et un déplacement maximal de la fenêtre $s_{max} = 70$ ont été utilisés comme paramètres d'entrée du logiciel de traitement d'images. La distance des objets par rapport au banc stéréo est indiquée par un niveau de gris, le noir correspondant aux objets éloignés et le blanc aux objets proches.

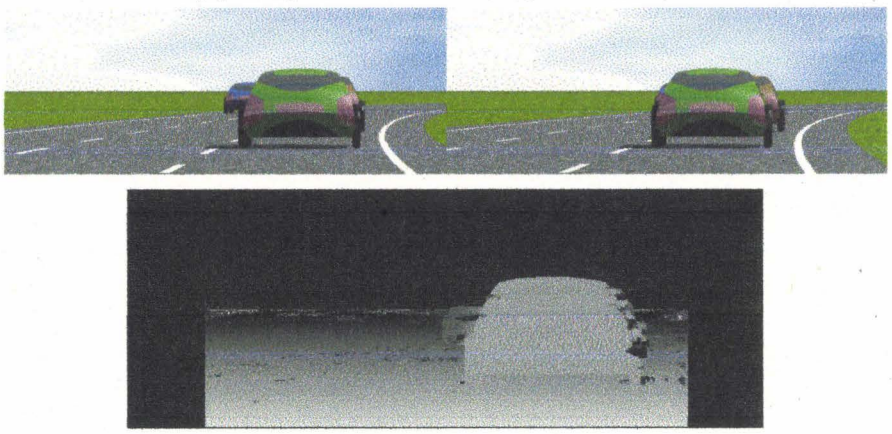
On constate que les cartes de disparité obtenues sont très précises. Elles font toutes apparaître clairement une rampe de disparité correspondant à la chaussée, et des zones de disparité à peu près constante correspondant aux véhicules. On vérifie également que les détails, comme les roues des véhicules, sont clairement détectés. Le niveau de gris des



(a) Image 1 et carte de disparité associée



(b) Image 25 et carte de disparité associée



(c) Image 50 et carte de disparité associée

FIG. 5.11 : Images extraites de la séquence et cartes de disparité associées

objets détectés donne une indication de la distance de l'obstacle potentiel. Par exemple, pour le véhicule vert, le niveau de gris qui s'éclaircit indique que cet obstacle se rapproche.

Dans les trois images, le véhicule vert occulte le véhicule beige, lequel est situé à une distance plus importante. Le voisinage adaptatif utilisé par l'algorithme SBAN permet néanmoins de séparer efficacement ces deux véhicules dans les cartes de disparité, tout au moins sur les images 1 et 25. Dans l'image 50, le véhicule beige qui est très occulté n'est pas parfaitement détecté. Le véhicule bleu, qui circule sur la voie de gauche, est parfaitement détecté dans les trois images, même quand il est partiellement occulté par le véhicule vert. Ces images montrent l'intérêt d'utiliser notre algorithme qui permet la détection de petits objets qui ne seraient pas détectés avec les algorithmes standards de mise en correspondance.

De mauvais appariements sont cependant observés, pour la plupart dans des zones de l'image qui correspondent au fond de la scène et à la chaussée. Dans ces parties de la scène, des textures présentant des hautes fréquences spatiales sont utilisées pour synthétiser les éléments de façon réaliste. Quand la résolution de l'image est limitée, par exemple vers la ligne d'horizon, l'échantillonnage fait apparaître des textures aléatoires qui ne sont pas identiques dans les deux images. Dans ces conditions, la fonction de corrélation, qui est calculée sur une fenêtre de taille importante, présente de nombreux minimums, dont l'absolu ne correspond pas à la disparité correcte.

5.3 Conclusion

Dans ce chapitre, nous avons montré comment l'algorithme SBAN peut être utilisé pour détecter les obstacles situés à l'avant d'un véhicule équipé d'un banc stéréo. Deux applications différentes ont été évoquées, l'une qui concerne un véhicule guidé circulant sur voie ferrée, l'autre un véhicule automobile.

Les résultats fournis par l'algorithme SBAN sont très satisfaisants, bien qu'ils aient été obtenus hors-ligne et par voie logicielle, soit sur des images enregistrées, soit sur des images de synthèse. Ceci ne constitue pas une limitation pour les systèmes futurs, puisque cet algorithme peut être implanté sur une architecture spécialisée, comme la carte STREAM, afin de fournir une carte dense des disparités à la cadence vidéo.

Conclusion et perspectives

Les algorithmes de mise en correspondance par corrélation sont les mieux adaptés à une implantation dans une architecture spécialisée de calcul, ce qui permet un traitement des images à la cadence imposée par la vidéo. Cependant, la taille et la forme idéales de la fenêtre de corrélation restent difficiles à déterminer, alors que l'influence de ces paramètres est primordiale sur le comportement temporel de l'algorithme. La plupart des méthodes adaptatives décrites jusqu'alors ne peuvent pas être facilement implantées dans une architecture spécialisée de calcul. Les méthodes à base de fenêtres multiples, bien qu'elles soient implantables, ne parviennent pas à traiter certaines situations complexes apparaissant dans les images réelles.

Pour pallier ces inconvénients, nous avons proposé un nouvel algorithme qui aborde de façon différente le problème du choix de la taille et de la forme de la fenêtre de corrélation. Il utilise une fenêtre de grande dimension, condition nécessaire pour traiter correctement des images qui contiennent des zones homogènes. Cependant, dans les zones de l'image qui correspondent à plusieurs objets de la scène, l'algorithme supprime de la fenêtre certains pixels qui ne doivent pas intervenir dans la procédure de mise en correspondance. L'algorithme se comporte de façon adaptative, en ajustant la taille et la forme du voisinage utilisé pour calculer la corrélation.

Seuls les pixels similaires au pixel situé au centre de la fenêtre sont conservés lors du calcul de corrélation. Nous avons proposé et testé plusieurs critères de similarité et avons comparé les performances de notre algorithme à celles des algorithmes similaires déjà décrits dans la littérature. Nous avons montré qu'avec un critère de similarité trivial comme la similitude du niveau de gris, notre algorithme est au moins aussi performant que ces méthodes.

L'algorithme proposé peut être implanté dans une architecture spécialisée de calcul

afin de traiter les images en temps-réel. Nous avons décrit l'implantation de plusieurs algorithmes de stéréovision dense sur le processeur STREAM, système de traitement dédié développé dans notre laboratoire. Nous avons montré que l'algorithme SBAN peut également être implanté sur cette architecture, bien qu'il nécessite un nombre plus élevé d'éléments logiques du fait de sa complexité.

Enfin, nous avons présenté une application potentielle de notre algorithme dans le cadre de la détection d'obstacles situés à l'avant d'un véhicule équipé de caméras. Dans la première application, les caméras sont installées à l'avant d'une rame du métro automatique VAL. Dans la deuxième application, un stéréoscope mono-caméra est installé à l'avant d'un véhicule automobile. Les résultats obtenus, pour l'instant hors-ligne, montrent que notre algorithme peut avantageusement être utilisé dans ces applications.

Le travail décrit dans ce mémoire nous a permis de proposer des améliorations par rapport aux techniques existantes, notamment en ce qui concerne l'adaptation de la fenêtre de corrélation. Cependant, les performances de cet algorithme pourraient encore être améliorées, en utilisant notamment d'autres critères de similarité ou plusieurs critères simultanément. La pertinence des appariements réalisés pourrait être vérifiée au moyen d'un degré de confiance, permettant par exemple de sélectionner l'extremum le plus adapté de la fonction de corrélation. Enfin, nous avons vu que certains algorithmes sont implantés de façon très efficace quand on utilise des voisinages réduits à une ligne de l'image. Ces différents points, qui ouvrent de nouvelles voies de recherche, sont détaillés par la suite.

C.1 Autres critères de similarité

L'algorithme SBAN supprime de la fenêtre de corrélation les pixels qui n'appartiennent pas au même objet que le pixel à mettre en correspondance. La sélection d'un pixel exploite sa similarité avec le pixel situé au centre de la fenêtre de corrélation. Nous avons proposé différents critères de similarité basés principalement sur la comparaison des niveaux de gris.

Nous avons constaté que, dans certaines situations, la comparaison des niveaux de gris n'est pas suffisante pour sélectionner correctement les pixels. De ce fait, des appariements incorrects sont déterminés, ce qui diminue la précision de la carte des disparités.

De critères de similarité plus efficaces doivent être déterminés, exploitant par exemple la couleur des pixels ou les textures présentes dans la fenêtre de corrélation. Nous avons évalué le comportement de notre algorithme lorsque la similarité est déterminée en fonction de la teinte des pixels. Ce travail, qui n'a pas été décrit dans ce mémoire, a été présenté récemment à la conférence ICCVG'2004 [Perez *et coll.* 2004a].

D'autre part, les critères de similarité que nous avons proposés respectent les contraintes imposées par l'implantation de l'algorithme sur une architecture dédiée de calcul. Il est probable que d'autres critères de similarité, plus complexes mais plus efficaces, pourraient être proposés si on supprime cette contrainte.

C.2 Critères multiples pour la corrélation

Dans les algorithmes de mise en correspondance par corrélation, la disparité est déterminée par le déplacement qui minimise l'indice de corrélation. Nous avons déjà mentionné que la minimisation de l'indice de corrélation ne suffit pas à garantir un bon appariement dans toutes les situations, notamment dans les zones homogènes ou celles qui contiennent des textures répétitives.

Dans ces situations, une analyse locale ne permet pas une détermination correcte de la disparité, ce qui ramène l'algorithme SBAN au même niveau que les autres méthodes locales analysées dans ce mémoire. Pour pallier cet inconvénient, il faudrait introduire dans la fonction de corrélation des termes non locaux, tenant compte par exemple de la position spatiale des pixels ou de la proximité d'un contour, ou des contraintes globales, telles la contrainte d'unicité ou la contrainte d'ordre.

Par exemple, on pourrait utiliser l'expression suivante du critère de corrélation afin de tenir compte de la distance entre les pixels considérés et les contours les plus proches dans les images gauche et droite :

$$\begin{aligned} C(x,y,s) &= k_1 \sum |I_g(x+i,y+j) - I_d(x+i+s,y+j)| \times \beta(x,y) \\ &+ k_2 \sum |l_g(x+i,y+j) - l_d(x+i+s,y+j)| \end{aligned} \tag{C.4}$$

où k_1 et k_2 sont des coefficients utilisés pour pondérer les critères et $l_g(x+i,y+j)$ et $l_d(x+i+s,y+j)$ désignent les distances au contour le plus proche pour les pixels respectivement de l'image gauche et de l'image droite.

C.3 Pertinence de l'appariement

Les différences de réglage des caméras, les réflexions, les différences de perspectives, la présence de zones homogènes ou de textures répétitives dans l'image sont à l'origine de mauvais appariements. La minimisation de l'indice de corrélation ne garantit pas que l'appariement réalisé soit correct. Ainsi, chaque appariement doit être accompagné d'un indice de confiance qui permet d'évaluer sa pertinence. Diverses mesures de confiance de l'appariement ont été proposées dans la littérature. La plupart tient compte de l'allure de la fonction de corrélation ou de la distance entre les deux pics les plus importants apparaissant dans cette fonction.

Nous proposons d'utiliser le nombre de pixels utilisés dans la fenêtre, donc considérés comme similaires au pixel central, dans l'expression du degré de pertinence de l'appariement. Par exemple, dans une zone homogène, tous les pixels sont similaires au pixel central, donc le pic dans la fonction de corrélation sera peu marqué. En revanche, dans une zone qui inclut une discontinuité, le nombre de pixels similaires au pixel central sera plus faible, ce qui peut indiquer un pic relativement significatif.

Le degré de pertinence ne peut pas être déterminé uniquement sur la base du nombre de pixels similaires présents dans la fenêtre, car de nombreux contre-exemples mettent en défaut les cas simples évoqués ci-dessus. En revanche, il doit être possible de combiner cette information aux autres informations issues de l'analyse de la fonction de corrélation afin de proposer un degré de pertinence plus significatif.

C.4 Voisinage 1D ou 2D ?

Dans le chapitre consacré à l'implantation des algorithmes, nous avons remarqué qu'il était plus simple d'implanter les algorithmes lorsqu'on utilise une fenêtre unidimensionnelle, réduite à une simple ligne de l'image. Cela se comprend aisément, puisqu'on évite ainsi de mémoriser les lignes précédentes des images, ce qui économise des bascules ou des bits de mémoire dans le circuit.

D'un point de vue plus théorique, on peut se poser la question du bien-fondé de l'utilisation de voisinages 2D dans les techniques de corrélation. La recherche de l'homologue d'un pixel est essentiellement un problème à une dimension, puisque ce dernier appartient

obligatoirement à la droite épipolaire correspondante. Quand les images sont rectifiées, les épipolaires sont ramenées sur des lignes horizontales de même indice dans les deux images.

Supposons qu'on recherche l'homologue $P_d(x_d, y_1)$ dans l'image droite d'un pixel $P_g(x_g, y_1)$ de l'image gauche. Quand on utilise un voisinage 1D pour calculer l'indice de corrélation, il arrive fréquemment que ce dernier ne présente pas de maximum significatif. C'est pourquoi la plupart des auteurs utilisent une fenêtre 2D, qui intègre des pixels $P_g(x, y_2)$ et $P_d(x, y_2)$, avec y_2 différent de y_1, qui améliorent la corrélation.

Les informations apportées par ces pixels situés sur des lignes différentes permettent de compenser le manque d'information dans la ligne en cours de traitement. Elles sont incorporées systématiquement dans l'indice calculé pour le pixel de la ligne y_1, sans tenir compte *a priori* de leur pertinence. Dans certaines situations, elles sont superflues et peuvent même entraîner des erreurs de mise en correspondance.

Pour ces différentes raisons, nous pensons qu'il conviendrait d'ajouter ces informations a posteriori, en étendant le voisinage 1D original aux lignes voisines, uniquement si cela s'avère nécessaire. Dans ce cas, la recherche de l'homologue resterait un traitement unidimensionnel, suivi éventuellement d'une phase d'enrichissement par fusion d'informations issues des lignes voisines. Ces aspects seront explorés dans la thèse de Sébastien Lefebvre, qui a débuté en septembre 2004.

Bibliographie

Barbaro, M. et Raffo, L. (2001). A low-power CMOS silicon retina for feature extraction in real-time, embedded systems. Dans *27th European solid-state circuits conference, ESSCIRC'01*, Villach, Austria.

Batlle, J., Marti, J., Ridao, P., et Amat, J. (2002). A new FPGA/DSP-based parallel architecture for real-time image processing. *RTI, Real time Imaging*, 8(5) :345–356.

Bertozzi, M. et Broggi, A. (1998). GOLD : a parallel real-time stereo vision system for generic obstacle and lane detection. *IEEE Transactions on Image Processing*, 7(1) :62–81.

BMVC'98 (1998). *Proceedings of the Ninth British Machine Vision Conference, BMVC'1998*, Southampton, UK.

Borga, M. et Knutsson, H. (1998). An adaptive stereo algorithm based on canonical correlation analysis. Dans *Proceedings of the 5th International Conference on Image Processing, ICIP'98*, Chicago, IL, USA.

Boykov, Y., Veksler, O., et Zabih, R. (1999). Fast approximate energy minimization via graph cuts. Dans *Proceedings of the 7th International Conference on Computer Vision, ICCV'99*, volume 1, pages 377–384, Kerkyra, Greece.

Boykov, Y., Veksler, O., et Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11) :1222–1239.

Broggi, A., Bertozzi, M., Gregoretti, F., Passerone, R., Sansoé, C., et Reyneri, L. (1997). A dedicated image processor exploiting both spatial and instruction-level parallelism.

Dans *Proceedings of the IEEE International Workshop on Computer Architectures for Machine Perception, CAMP'97*, volume 1, pages 106–115, Boston, USA.

Brown, M., Burschka, D., et Hager, G. (2003). Advances in computational stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(8) :993–1008.

Bruyelle, Jean-Luc (1994). *Conception et réalisation d'un dispositif de prise de vue stéréoscopique linéaire - Application à la détection d'obstacles à l'avant des véhicules routiers*. Thèse de Doctorat, Université des Sciences et Technologies de Lille.

Burie, Jean-Christophe (1995). *Mise en correspondance d'images linéaires stéréoscopiques, application à la détection d'obstacles à l'avant des véhicules routiers*. Thèse de Doctorat, Université des Sciences et Technologies de Lille.

Cabestaing, F., Bonnet, P., Deparis, J.P., et Postaire, J.G. (1991). Detection of moving objects in real time, the STREAM processor. Dans *Proceedings of the International Conference on Computer Architectures for Machine Perception, CAMP'91*, pages 279–284, Paris, France.

Cabestaing, F., Yang, R., Bruyelle, J.L., et Postaire, J.-G. (1999). Real-time preprocessing of image sequences : Application to intrusion detection in secured areas. Dans *Proceedings of the International Conference on Quality Control by Artificial Vision, QCAV'99*, pages 101–106, Trois-Rivières, Québec, Canada.

CAMP'00 (2000). *Proceedings of the 5th IEEE International Workshop on Computer Architectures for Machine Perception, CAMP'2000*, Padova, Italy.

Choudhary, A., Das, S., Ahuja, N., et Patel, J.H. (1990). A reconfigurable and hierarchical parallel processing architecture : performance results for stereo vision. Dans *Proceedings of the 10th International Conference on Pattern Recognition, ICPR'90*, volume 2, pages 389–393, Atlantic City, NJ, USA.

Cohen, L., Vinet, L., Sander, P., et Gagalowicz, A. (1989). Hierarchical region based stereo matching. Dans *Proceedings of the Conference on Computer Vision and Pattern Recognition, CVPR'89*, volume 1, San Diego, CA, USA.

Cozzi, A., Crespi, B., Valentinotti, F., et Worgotter, F. (1997). Performance of phase-based algorithms for disparity estimation. *Machine Vision and Applications*, 9 :334–340.

Crespi, B., Cozzi, A. G., Raffo, L., et Sabatini, S. (1998). Analog computation for phase-based disparity estimation : continuous and discrete models. *Machine Vision and Applications*, 11 :83–95.

CVPR'03 (2003). *Proceedings of the Conference on Computer Vision and Pattern Recognition, CVPR'2003*, Wisconsin, USA.

CVPR'99 (1999). *Proceedings of the Conference on Computer Vision and Pattern Recognition, CVPR'99*, Fort Collins, CO, USA.

Darabiha, A., Rose, J., et MacLean, W.J. (2003). Video-rate stereo depth measurement on programmable hardware. Dans CVPR'03 [2003], pages 203–210.

Devernay, F. (1997). *Vision stéréoscopique et propiétés différentielles des surfaces*. Thèse de Doctorat, Ecole Polytechnique.

Devernay, F., Bantiche, O., et Coste-Maniére, E. (2002). Structured light on dynamic scenes using standard stereoscopy algorithms. *Rapport de recherche*, 1(4477).

Dooze, D. (2001). *Conception et réalisation d'un stéréoscope bimodal à portée variable : Application à la détection d'obstacles à l'avant de véhicules guidés automatisés*. Thèse de Doctorat, Université des Sciences et Technologies de Lille.

Duvieubourg, L., Ambellouis, S., et Cabestaing, F. (2004). Single-camera stereovision setup with orientable optical axes. Dans ICCVG'04 [2004].

ECCV'94 (1994). *Proceedings of the European Conference on computer Vision, ECCV'94*.

Edelman, S. et Weinshall, D. (1989). Computational vision : a critical review. *Rapport de recherche*, AIM-1158 :34.

Faugeras, O., Hotz, B., Mathieu, H., Viéville, T., Zhang, Z., Fua, P., Théron, E., Moll, L., Berry, G., Vuillemin, J., Bertin, P., et Proy, C. (1993). Real-time correlation-based stereo : Algorithm, implementations and applications. *Rapport de recherche*, 1(2013).

Faugeras, O., Luong, Q., et Maybank, S. (1992). Camera self-calibration : Theory and experiments. Dans *Proceedings of the Second European Conference on Computer Vision, ECCV'92*, volume 1, pages 321–334.

Franke, U. et Joos, A. (2000). Real-time stereo vision for urban traffic scene understanding. Dans *Proceedings of the IEEE Intelligent Vehicles Symposium, IVS'2000*, volume 1, pages 273–278, Dearborn, MI, USA.

Fusiello, A., Roberto, V., et Trucco, E. (2000a). Symmetric stereo with multiple windowing. *International Journal of Pattern Recognition and Artificial Inteligence*, 8(14) :1053–1066.

Fusiello, A., Trucco, E., et Verri, A. (2000b). A compact algorithm for rectification of stereo pairs. *Machine Vision and Applications*, 12(1) :16–22.

Geiger, D., Ladendorf, B., et Yuille, A. (1995). Occlusions and binocular stereo. *International Journal of Computer Vision*, 14 :211–226.

Gong, M. et Yang, Y. (2002). Genetic-based stereo algorithm and disparity map evaluation. *International Journal of Computer Vision*, 47(1) :63–77.

Gonzalez, R., Cancelas, J., Alvarez, J., Fernandez, J., et Alvarez, I. (1999). Dynamic programming stereo vision algorithm for robotics applications. Dans *Proceedings of the 12th Conference on Vision Interface, VI'1999*, Trois Rivieres, Quebec.

Goulermas, J.Y. et Liatsis, P. (2001). Hybrid symbiotic genetic optimisation for robust edge-based stereo correspondence. *Pattern Recognition*, 34 :2477–2496.

Hartley, R. (1999). Theory and practice of projective rectification. *International Journal of Computer Vision*, 35(2) :115–127.

Heikkila, J. et Silven, O. (1997). A four-step camera calibration procedure with implicit image correction. Dans *Proceedings of the Conference on Computer Vision and Pattern Recognition, CVPR'97*, volume 1, pages 1106–1112, San Juan, PR, USA.

Herzen, B. V. (1997). Signal processing at 250 mhz using high-performance FPGAs. Dans *Proceedings of the 1997 ACM fifth international symposium on Field-programmable gate arrays*, pages 62–68, Monterey, Ca. ACM Press.

Hirschmuller, H. (2001). Improvements in real-time correlation-based stereo vision. Dans *Proceedings of IEEE workshop on Stereo and Multi-Baseline Vision*, pages 141–148, Kauai, Hawaii.

Horaud, Radu et Monga, Olivier (1995). *Vision par ordinateur, outils fondamentaux*, pages 39–68. Hermes.

Huang, J. et Liu, H. (1997). Stereo vision using a microcanonical mean field annealing neural network. *Network : Computation in Neural Systems*, 8 :87–104.

ICCVG'04 (2004). *Proceedings of the International Conference on Computer Vision and Graphics, ICCVG'2004*, Warsaw, Poland.

Intille, S. et Bobick, A. (1994). Disparity-space images and large occlusion stereo. Dans ECCV'94 [1994].

Ishikawa, H. (1999). Multi-scale feature selection in stereo. Dans CVPR'99 [1999].

Ishikawa, H. et Geiger, D. (1998). Occlusions, discontinuities, and epipolar lines in stereo. Dans *Proceedings of the European Conference on Computer Vision, ECCV'98*, Freiburg, Ger.

Issa, H. (2004). *Mise en correspondance stéréoscopique par algorithmes génétiques : nouvaux codages.* Thèse de Doctorat, Université des Sciences et Technologies de Lille.

Issa, H., Vieren, C., Ruichek, Y., et Khoudour, L. (2003). Aide à l'exploitation des systèmes de transport guidées par stéréoscopie : une nouvelle approche par algorithme génétique. Dans *Proceedings of the Technological Innovation for Land Transportation, TILT'2003*, volume 1, pages 305–312, Lille, France.

Kanade, T., Kano, H., Kimura, S., Yoshida, A., et Oda, K. (1995). Development of a video-rate stereo machine. Dans *Proceedings of IEEE workshop on Stereo and Multi-Baseline Vision*, pages 95–100.

Kanade, T. et Okutomi, M. (1991). A stereo matching algorithm with an adaptive window : Theory and experiment. Dans *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, volume 16, pages 920–932, Sacramento, CA, USA.

Koch, R. (1993). Automatic reconstruction of buildings from stereoscopic image sequences. Dans *Proceedings of the EUROGRAPHICS93, EUROGRAPHICS'93*, Barcelona, Spain.

Konolige, K. (1997). Small vision systems : Hardware and implementation. Dans *Proceedings of Eighth International Symposium on Robotics, ISR'97*, pages 111–116, Hayama, Japan.

Koschan, A., Rodehorst, V., et Spiller, K. (1996). Color stereo vision using hierarchical block matching and active color illumination. Dans *Proceedings of the 13th International Conference on Pattern Recognition, ICPR'96*, volume 1, pages 835–839, Vienna, Austria.

Lan, Z. et Mohr, R. (1995). Robust matching by partial correlation. *Rapport de recherche*, 1(RR-2643).

Lecoat, F., Pissaloux, E., Bonnin, P., Garié, T., Durbin, F., et Tissot, A. (1997). A parallel algorithm for a very fast 2D velocity field estimation. Dans *Proceedings of the 4th International Conference on Image Processing, ICIP'97*, volume 2, pages 179–182, Santa Barbara, CA, USA.

Lee, D., Kweon, I., et Cipolla, R. (1999). A biprism-stereo camera system. Dans CVPR'99 [1999], pages 1082–1088.

Lhuillier, M. (1998). Efficient dense matching for textured scenes using region growing. Dans BMVC'98 [1998], pages 700–709.

Lotti, J. et Giraudon, G. (1994). Correlation algorithm with adaptive window for aerial image in stereo vision. Dans *Proceedings of the Image and Signal Processing for Remote Sensing, EUROPTO'94*, volume 1, pages 701–703, Rome, Italy.

Maher, M., Deweerth, S., et Mahowald, M. (1989). Implementing neural architectures using analog VLSI circuits. *IEEE Transactions on Circuits and Systems*, 36(5) :643–652.

Mahowald, M. (1994). Analog VLSI chip for stereocorrespondence. Dans *Proceedings of the IEEE International Symposium on Circuits and Systems, ISCAS'94*, volume 6, pages 347–350, London, UK.

Marr, D. et Poggio, T. (1979). A computational theory of human stereo vision. *Proceedings Royal Society London*, 204(1) :301–328.

Martin, J. et Crowley, J.L. (1995). Experimental comparison of correlation techniques. Dans *Proceedings of the 3rd International Symposium on Intelligent Robotic Systems '95.*, page 287.

Martínez, J., Costa, E., Herreros, P., Sánchez, X., et Baldrich, R. (2003). A modular and scalable architecture for PC-based real-time vision systems. *RTI, Real time Imaging*, 9(2) :99–112.

Mathieu, H. et Devernay, F. (1995). Système de miroirs pour la stéréoscopie. Rapport technique 172, INRIA, Sophia Antipolis. Projet Robotvis.

Matsumoto, Y., Shibata, T., Sakai, K., Inaba, M., et Inoue, H. (1997). Real-time color stereo vision system for a mobile robot based on field multiplexing. Dans *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA'97*, Albuquerque, NM.

Mellor, J.P., Teller, Seth, et Lozano-Pérez, Tomás (1996). Dense depth maps from epipolar images. *Rapport de recherche.*

Moini, A. (1999). *Vision chips.* Kluwer.

Moravec, K., Harvey, R., et Bangham, J. (1998). Improving stereo performance in regions of low texture. Dans BMVC'98 [1998].

Mulligan, J., Isler, V., et Daniilidis, K. (2002). Trinocular stereo : a real-time algorithm and its evaluation. *International Journal of Computer Vision*, 1(1) :51–61.

Ohm, J., Gruneberg, K., Hendriks, E., Izquierdo, E., Kalivas, D., Karl, M., et Papadimatos, D. (1998). A realtime hardware system for stereoscopic videoconferencing with viewpoint adaptation. *Image Communication*, 14(1) :147–171.

Ouali, M., Laurgeau, C., et Ziou, D. (1999). Dense disparity estimation using gabor filters and image derivatives. Dans *Proceedings of the 2th International Conference on 3-D Imaging and Modeling, 3DIM'99*, volume 1, pages 483–490, Ottawa, Canada.

Perez, M. et Cabestaing, F. (2003). A comparison of hardware resources required by real-time stereo dense algorithms. Dans *Proceedings of the 6th IEEE International Workshop on Computer Architectures for Machine Perception, CAMP'03*, New Orleans, USA.

Perez, M., Cabestaing, F., et Postaire, J.G. (2003). STREAM, un procesador para tratamiento de secuencias de imágenes : aplicación a la estereo visión densa. Dans *Avances en Ciencias de la Computacion, ENC'03*, volume 1, pages 77–82, Apizaco, Mexico.

Perez, M., Cabestaing, F., et Postaire, J.G. (2004a). A SBAN stereovision algorithm using hue as pixel similarity criterion. Dans ICCVG'04 [2004].

Perez, M., Colot, O., et Cabestaing, F. (2004b). A new adaptive window area-based method for stereo matching. Dans *Proceedings of the 11th International Conference on Image Processing, ICIP'2004*, volume 1, page 1, Singapore.

Philipp, R., Reddy, V., Cummings, R., et Lewis, M. (2002). Architecture for an VLSI stereo vision system. Dans *Proceedings of the IEEE International Symposium on Circuits and Systems, ISCAS'02*, volume 3, pages 691–694, Scottsdale, USA.

Pissaloux, E., Lecoat, F., Bonnin, P., Tissot, A., et Durbin, F. (1999). Design and optimization of a parallel architecture dedicated to image matching. Dans *Proceedings of the 6th International Conference on Image Processing, ICIP'99*, volume 2, pages 783–786, Kobe, Japan.

Pissaloux, E., Lecoat, F., Tissot, A., et Durbin, F. (2000). An adaptive parallel system dedicated to projective image matching. Dans *Proceedings of the 7th International Conference on Image Processing, ICIP'2000*, volume 2, pages 507–510, Vancouver, BC.

Porr, B., Cozzi, A., et Worgotter, F. (1998). How to hear visual disparities : real-time stereoscopic spatial depth analysis using temporal resonance. *Biological Cybernetics*, 78 :329–336.

Quénot, G. (1996). Image matching using dynamic programming : Application to stereovision and image interpolation. *Image Communication*, pages 265–270.

Ruichek, Y. (1997). *Stéréovision linéaire par réseaux de neurones de Hopfield. Application à la détection d'obstacles à l'avant des véhicules routiers*. Thèse de Doctorat, Université des Sciences et Technologies de Lille.

Ruichek, Y., Postaire, J. G., et Bruyelle, J.-L. (1998). A neural approach for obstacle detection with a linear stereoscopic sensor. *Journal of Mathematical and Computer Modelling*, 27(9-11) :215–228.

Saito, H. et Mori, M. (1995). Application of genetic algorithms to stereo matching of images. *Pattern Recognition*, 16 :815–821.

Scharstein, D. (1994). Matching images by comparing their gradient fields. Dans *Proceedings of International Conference on Pattern Recognition*, pages 572–575, Jerusalem, Israel.

Scharstein, D. et Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1) :7–42.

Scherer, S., Andexer, W., et Pinz, A. (1998). Robust adaptive window matching by homogeneity constraint and integration of descriptions. Dans *Proceedings of the 14th International Conference on Pattern Recognition, ICPR'98*, volume 1, pages 777–780, Brisbane, Australia.

Schreer, O., Brandenburg, N., Plakas, C., Trucco, E., et Kauff, P. (2001). A combination of census transform and a hybrid block and pixel recursive disparity analysis approach for real-time videoconferencig applications. Dans *Proceedings of the International Conference on Augmented, Virtual Environments and three-dimensional imaging 01*, Mikonos, Greece.

Stefano, L., Marchionni, M., Mattoccia, S., et Neri, G. (2002). A fast area-based stereo matching algorithm. Dans *Proceedings of the 15th Conference on Vision Interface, VI'2002*, Calgary, Canada.

Stefano, L. et Mattoccia, S. (2000). Fast matching for the VIDET system using a general purpose processor with multimedia extensions. Dans CAMP'00 [2000].

Stefano, L. et Mattoccia, S. (2002).. Real-time stereo within the VIDET project. *RTI, Real time Imaging*, 8(5) :439–453.

Sun, C. (2002). Fast stereo matching using rectangular subregioning and 3d maximum-surface techniques. *International Journal of Computer Vision*, 47(1) :99–117.

Tao, H., Sawhney, H., et Kumar, R. (2001). A global matching framework for stereo computation. Dans *Proceedings of the 8th International Conference on Computer Vision, ICCV'2001*, volume 1, pages 532–539, Vancouver, BC.

Taraglio, S. et Zanela, A. (2000). A practical use of cellular neural networks : The stereo-vision problem as an optimisation. *Machine Vision and Applications*, pages 245–251.

Tsai, R.Y. (1987). A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Transactions on Robotics and Automation*, 3(4) :323–344.

Urmson, C. et Dias, M. (2002). Stereo vision based navigation fur sun-synchronous exploration. Dans *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS'97*, EPFL, Switzerland.

van der Mark, W. et Groen, F.C.A. (2001). Stereo based navigation in unstructured environments. Dans *Proceedings of the IEEE Instrumentation and Measurement Technology Conference, IMTC'01*, volume 1, pages 193–198, Budapest, Hungary.

Veksler, O. (2002). Stereo matching by compact windows via minimum ratio cycle. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12) :1654.

Wal, G., Hansen, M., et Piacentino, M. (2000). The ACADIA vision processor. Dans CAMP'00 [2000], pages 31–40.

Williamson, T. A. (1998). A high-performance stereo vision system for obstacle detection. *Rapport de recherche*, 1(CMU-RI-TR-98-24).

Woetzel, J. et Koch, R. (2004). Real-time multi-stereo depth estimation on GPU with approximative discontinuity handling. Dans *Proceedings of the 1st European Conference on Visual Media Production, CVMP'04*, Savoy Place, London.

Woodfill, John et Herzen, Brian Von (1997). Real-time stereo vision on the PARTS reconfigurable computer. Dans *Proceedings of the IEEE Symposium for Custom Computing Machines, CCM'97*, pages 201–210.

Yang, R. et Pollefeys, M. (2003). Multi-resolution real-time stereo on commodity graphics hardware. Dans CVPR'03 [2003], pages 211–217.

Yang, R., Pollefeys, M., Yang, H., et Welch, G. (2003). A unified approach to real-time, multi-resolution, multi-baseline 2D view synthesis and 3D depth estimation using commodity graphics hardware. *International Journal of Image and Graphics*.

Yuan, T. et Subbarao, M. (1998). Integration of multiple-baseline color stereo vision with focus and defocus analysis for 3D shape measurement. Dans *Proceedings of the SPIE, Photonics West 98*, volume 3520, San Jose, CA, USA.

Zabih, R. et Woodfill, J. (1994). Non-parametric local transforms for computing visual correspondence. Dans ECCV'94 [1994], pages 151–158.

Zhang, L., Curless, B., et Seitz, S. (2002). Rapid shape acquisition using color structured light and multi-pass dynamic programming. Dans *Proceedings of the 1st International Symposium on 3D Data Processing Visualization Transmission, 3DPVT'02*, volume 1, pages 24–36, Padova, Italy.

Zhang, Y. et Kambhamettu, C. (2002). Stereo matching with segmentation-based cooperation. Dans *Proceedings of the Seventh European Conference on Computer Vision, ECCV'02*, Copenhagen, Denmark.

Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11) :1330–1334.

Zitnick, C. et Kanade, T. (1999). A cooperative algorithm for stereo matching and occlusion detection. *Rapport de recherche*, 1(CMU-RI-TR-99-35).