Architectures pour la stéréovision passive dense temps réel : application à la stéréo-endoscopie

Abdelelah Naoulou

Année 2006

THESE

Présentée en vue de l'obtention du titre de

DOCTEUR DE L'UNIVERSITE PAUL SABATIER

Par :

Abdelelah NAOULOU

ARCHITECTURES POUR LA STEREOVISION PASSIVE DENSE TEMPS REEL : APPLICATION A LA STEREO-ENDOSCOPIE

Soutenue le 05 Septembre 2006 devant le jury :

Directeur de thèse :	J.L. Boizard
Co-directeur de these:	J.Y. Fourniols
Rapporteurs :	B. Zavidovique R. Zapata
Examinateurs :	M. Devy L. Soler D. Estève A. Cazarré

Thèse préparée au sein du groupe Microsystèmes et Intégration de Systèmes (MIS) du Laboratoire d'Analyse et d'Architecture des Systèmes LAAS-CNRS, 7 Av. du Colonel Roche, 31077 Toulouse Cedex 4.

Remerciements

Ce travail de thèse commencé en septembre 2003 et financé par l'université d'Alep (Syrie), a été effectué au Laboratoire d'Analyse et d'Architecture des Systèmes du Centre National de la Recherche Scientifique (LAAS-CNRS), et plus précisément au sein du groupe Microsystèmes et Intégration des Systèmes (MIS).

En premier lieu, je tiens à remercier Monsieur Malik Ghallab, Directeur du laboratoire, pour m'y avoir accueilli.

J'aimerais adresser ma profonde gratitude à M. Boizard et M. Fourniols, mes directeurs de thèse. Pour leur disponibilité, générosité, patience et qualité scientifique qui ont fortement contribué à mener à bien nos travaux et à faire de mes années de thèse une période agréable et enrichissante.

Je remercie aussi Monsieur Michel Devy, Directeur de Recherche, pour son aide dans la partie stéréovision.

Je tien à remercier Messieurs Bertrand Zavidovique et René Zapata pour accepter de rapporter et évaluer ce travail.

Et c'est avec beaucoup de plaisir que je souhaite remercier à présent tous les membres du LAAS-CNRS que j'ai côtoyé au cours de ces années et tout particulièrement :

- Monsieur Alain Cazarre, le responsable du DEA CCMM, la première personne qui m'a accordé sa confiance pour démarrer mes études supérieures en France.
- Monsieur Daniel Estève, notre grande personnalité scientifique à qui je témoigne un profond respect.
- Monsieur Christian Ganibal, Pierre et Corine du service 2I.
- Monsieur Frédéric Lerasle du groupe RIA.
- Hassan, Sylvaine, Younes et Joan pour leurs conseils pendant ces trois années de thèse.

- Cyril, Fabrice, Pierre et Gustavo mes voisins de bureau.

Je souhaite remercier très sincèrement tous mes proches (mes parents, mes frères et sœurs, mes beaux parents, et mes beaux frères et sœurs) qui m'ont accordé leur confiance et beaucoup de patience.

Enfin, Je ne peux pas oublier de bien remercier mon âme sœur et ma petite Zainab pour leur patience en m'attendant toujours et pour le confort qu'elles m'ont assuré pendant ces années.

Résumé

L'émergence d'une robotique médicale en chirurgie laparoscopique destinée à automatiser et améliorer la précision des interventions nécessite la mise en œuvre d'outils et capteurs miniaturisés intelligents dont la vision 3D temps réel est un des enjeux.

Bien que les systèmes de vision 3D actuels représentent un intérêt certain pour des manipulations chirurgicales endoscopiques précises, ils ont l'inconvénient de donner une image 3D qualitative plutôt que quantitative, laquelle nécessite un appareillage spécifique rendant l'acte chirurgical inconfortable et empêche le couplage avec un calculateur dans le cadre d'une chirurgie assistée.

Nous avons développé dans la cadre du projet interne « PICASO » (Plate-forme d'Intégration de CAméras multiSenOrielles) dont les enjeux scientifiques concernent le conditionnement de capteurs intégrés et le traitement et la fusion d'images multi spectrales, un dispositif de vision 3D compatible avec les temps d'exécution des actes chirurgicaux. Ce système est basé sur le principe de la stéréoscopie humaine et met en œuvre des algorithmes de stéréovision passive dense issus de la robotique mobile. Dans cette thèse nous présentons des architectures massivement parallèles, implémentées dans un circuit FPGA, et capables de fournir des images de disparité à la cadence de 130 trames/sec à partir d'images de résolution 640x480 pixels. L'algorithme utilisé est basé sur la corrélation Census avec une fenêtre de calcul de 7 x 7 pixels. Celui-ci a été choisi pour ses performances en regard de sa simplicité de mise en œuvre et la possibilité de paralléliser la plupart des calculs. L'objectif principal de cet algorithme est de rechercher, pour chaque point, la correspondance entre deux images d'entrées (droite et gauche) prises de deux angles de vue différents afin d'obtenir une "carte de disparités" à partir de laquelle il est possible de reconstruire la scène 3D.

Pour mettre en œuvre cet algorithme et tenir les contraintes « temps réel » nous avons développé des architectures en « pipeline » (calcul des moyennes, transformation Census, recherche des points stéréo-correspondants, vérification

droite-gauche, filtrage...). L'essentiel des différentes parties qui composent l'architecture est décrit en langage VHDL synthétisable. Enfin nous nous sommes intéressés à la consommation en termes de ressources FPGA (mémoires, macro-cellules) en fonction des performances souhaitées.

Abstract

The emergence of medical robotics in laparoscopic surgery intended to automate and to improve the precision of the interventions requires the application of tools and intelligent miniaturized sensors to which real-time 3D vision belongs.

Although the current 3D vision systems represent certain interest for precise laparoscopic surgical manipulations, they have the inconvenience to give a qualitative 3D image rather than quantitative, which requires a specific equipment making the surgical act uncomfortable and prevents the coupling with a computer in the computer aided surgery.

We have developed in the internal project PICASSO (Platform of Integration of CAMERAS multiSenOrielles) the scientific interests in which concern the adaptation of integrated sensors, the treatment and the fusion of multi spectral images and a 3D vision device compatible with the execution times of the surgical acts. This system is based on the principle of the human stereoscopy and implements passive dense stereovision algorithms. In this thesis we present massively parallel architectures, implemented in FPGA circuit, and able to supply disparity images at the cadence of 130 trams per second for image resolution of 640x480 pixels. The adopted algorithm is based on the Census correlation with a calculation window of 7x7 pixels. This one was chosen as the performances compared to its simplicity of application and the parallelism possibility. The main objective of this algorithm is to look for, for every point, the correspondence between two images of right and left taken by two different angles of sight to obtain a "disparities map" from which it is possible to reconstruct the 3D scene.

To implement this algorithm and hold the "real-time" constraints we've developed architectures in "pipeline" (calculation of means, Census transformation, searching stereo-corresponding points, check right-left, filtering). The architecture steps are described essentially in VHDL language. Finally we are interesting in the consumption in terms of FPGA resources (memories, macro-cells) according to the wished performances.

Table des Matières

Chapitre1

Introduction et contexte de l'étude

---

1 INTRODUCTION ET CONTEXTE DE L'ETUDE

1.1 Introduction

La vision stéréoscopique permet à l'homme de se déplacer et d'appréhender son environnement : détecter un objet, reconnaître un visage dans une scène et ressentir les émotions des gens perceptibles au travers de leur expression sont quelques exemples de fonctions dévolues à la vision et quotidiennement exécutées de manière implicite. Bien que celles-ci soient réalisées avec un minimum d'effort, elles traduisent lorsqu'on tente de les analyser l'existence d'un ensemble de processus extrêmement complexes.

Dans les trois dernières décennies, les chercheurs ont essayé de doter de capacité de vision les systèmes automatisés tels les robots afin de les rendre plus intelligents et d'élargir ainsi l'éventail de leurs capacités. Cependant beaucoup de chemin reste à faire : la façon dont le cerveau humain traite l'information visuelle est très mal connue et les tâches de traitement d'image couramment exécutées par les systèmes automatisés sont très lourdes en temps calcul. En développant des systèmes de vision, les concepteurs sont essentiellement confrontés à deux possibilités : implémenter les algorithmes de vision sous forme logicielle et les exécuter sur des processeurs standards (ou des DSP) ou les implémenter dans des architectures matérielles spécialement conçues pour l'application. Bien que les architectures matérielles spécifiques puissent offrir des vitesses de traitement beaucoup plus élevées et puissent exploiter beaucoup plus facilement le parallélisme que l'on rencontre fréquemment dans les algorithmes de traitement d'images, celles-ci sont restées longtemps ignorées car les ressources nécessaires ne sont pas toujours disponibles à un prix raisonnable et le délai de conception est relativement long. Ces différents aspects ont contribué au choix fréquent de la première option, à savoir le

développement d'algorithmes tournant le plus rapidement possible sur des processeurs standards.

Depuis quelques années, une troisième solution pour les concepteurs de systèmes de vision a vu le jour du fait de la croissance très rapide des circuits logiques programmables en termes de vitesse, de ressources disponibles et de performances des outils de développement. Les circuits logiques programmables ont également été utilisés avec succès dans des applications autres que celles liées au traitement d'images (filtrage numérique multivoies, FFT, formation de voies dans les antennes acoustiques large bandes [Carron_04]). Ces circuits permettent aux développeurs de configurer économiquement et rapidement leur structure interne en fonction des caractéristiques de l'algorithme de traitement choisi car ils éliminent les phases les plus lourdes et les plus coûteuses que l'on rencontre lors de la conception de circuits intégrés spécifiques tels les ASIC, à savoir leur fabrication. Ils permettent également une réduction significative des temps de mise au point puisqu'il est très facile et très rapide pour le concepteur de modifier le design, le recompiler et reprogrammer le composant.

Le but des travaux présentés dans ce mémoire est d'évaluer les performances des circuits FPGA pour les applications de vision 3D temps réel basées sur le principe de la stéréoscopie humaine et leur application dans le cadre de la stéréo-endoscopie en chirurgie laparoscopique.

1.2 Historique

Ce travail de thèse s'inscrit dans la cadre du projet interne « PICASO » (Plate-forme d'Intégration de CAméras multiSenOrielles) dont les enjeux scientifiques concernent d'une part le conditionnement de capteurs intégrés, et d'autre part le traitement et la fusion d'images multi spectrales. Les principales finalités de ce projet sont :

- la réalisation d'un démonstrateur regroupant les savoir faire du LAAS en algorithmique de la perception et en intégration de systèmes. Dans cette phase il

est prévu le développement d'une plate-forme matérielle temps réel dédiée à la stéréovision et basée sur de nouvelles architectures de composants, type « System On Chip » ou « System On Programmable Chip », entièrement portables puisque décrites en langage VHDL. Cette plate-forme exploitera au moins deux types de détecteurs lui conférant une vision multi longueur d'onde (Infra Rouge dans la bande 9-12_µm et visible dans la bande 0,4-1_µm).

- L'intégration en assemblage 3D, avec traitement temps réel des données par implantation en hardware des algorithmes de traitement (stéréo et fusion d'images).
- L'évaluation de ce prototype dans le cadre de plusieurs applications, notamment pour la détection d'obstacles (statiques ou mobiles) depuis un véhicule pour améliorer la sécurité dans les transports et en stéréoendoscopie dans le cadre de la chirurgie laparoscopique assistée par ordinateur. Notons que pour cette dernière le LAAS a développé dans un précédent projet un robot chirurgical, l'Endoxirob, auquel il est envisagé d'associer la stéréovision.

Par ailleurs le prototype PICASO doit avoir une architecture modulaire afin d'adapter sa configuration à l'application visée et devra pouvoir être exploité dans les deux modes suivants:

- traitement périodique, en mode *Temps réel* (fréquence visée > 100 images/sec),
- traitement sur requête d'un calculateur client en mode *Serveur d'images*.

Dans sa version complète, adaptée pour des applications courte portée, ce système comportera trois parties :

- une partie *Capteur* : il s'agit en fait de plusieurs capteurs couvrant chacun des bandes spectrales différentes (Infra Rouge dans la bande 9-12 microns, noté FIR, et visible, dans la bande 0,4-1microns, noté VIS) et leurs têtes optiques associées.

- une partie *Traitement* à base de circuits FPGA et mémoires, sur lesquels pourront s'exécuter des algorithmes de traitement du signal, de stéréovision, de fusion multi spectrale....
- une partie *Communication* avec un calculateur hôte : typiquement une liaison USB2 dans sa version haute vitesse.

La sortie de ce système sera :

- en mode *Temps réel*, une information dépendant de l'application (par exemple, position, vitesse et caractéristique des obstacles détectés).
- en mode *Serveur d'images*, une image 3D, dans laquelle pour chaque pixel seront disponibles la position (X,Y,Z) et une information de réflectance T.

Pour des applications longue portée, la distance entre les centres optiques des caméras doit être plus importante; dans ce cas, le prototype PICASO ne contiendra que les parties *Traitement* et *Communication* et sera connecté via des liaisons *Camera Link*, à des caméras déportées.

1.3 Objectifs et limites des travaux

Les travaux présentés dans ce mémoire prennent naturellement appui sur un ensemble de thèmes de recherche développés au laboratoire, à savoir :

- les algorithmes de stéréovision dense et de filtrage d'images
- les méthodes de calibration des caméras

dans le cadre du groupe RIA (Robotique et Intelligence Artificielle) et :

- l'intégration des systèmes
- les méthodologies de conception
- le robot chirurgical « Endoxirob »

dans le cadre du groupe MIS (Microsystèmes et Intégration de Systèmes).

Par ailleurs l'obtention d'images 3D en temps réel à partir de techniques de stéréovision dense a nécessité :

- une connaissance intime des algorithmes utilisés pour la stéréo-correspondance ainsi que des influences respectives des paramètres de réglage (moyennage, dimensions des fenêtrages, filtrage, …).
- une bonne maîtrise du fonctionnement des capteurs d'images matriciels (CCD, CMOS), des caméras vidéo et des protocoles de communication associés.
- Une bonne connaissance des solutions technologiques disponibles à ce jour (DSP, ASIC, FPGA).

Dans ce contexte le travail de thèse a consisté essentiellement à :

- concevoir et développer un banc générique paramétrable permettant d'évaluer les différents algorithmes potentiellement utilisables dans le cadre de la vision 3D temps réel et ayant des degrés de liberté mécaniques suffisants pour effectuer aisément sa calibration.
- évaluer les différents protocoles de communication utilisés en vidéo numérique.
- Evaluer les différentes solutions possibles (logicielles, matérielles) ainsi que les outils de développement associés.
- Coder en langage VHDL et simuler les différentes architectures retenues pour garantir les performances « temps réel » imposées par le cahier des charges.
- Valider ces résultats sur une maquette à base de circuits FPGA en termes de qualité d'images et de performances temps réel.
- Concevoir et développer une version du banc initial adaptée à la problématique de la vision stéréo-endoscopique.

1.4 Présentation du manuscrit

Il nous a semblé nécessaire avant d'aborder le vif du sujet, de donner quelques exemples de méthodes utilisées dans le cadre de la vision 3D, d'exposer de manière détaillée les principes de la stéréovision passive ainsi que les différentes techniques d'appariement qui lui sont associées. Ces différents points sont présentés dans le chapitre 2.

Par ailleurs les travaux de thèse se sont déroulés selon trois axes principaux :

- la conception d'un banc générique pour évaluer rapidement les algorithmes d'appariement ainsi que les architectures matérielles candidates. C'est l'objet du chapitre 3.
- Le portage des architectures retenues sur une plate-forme à base de circuits FPGA et l'interfaçage avec les caméras et le calculateur hôte. C'est l'objet du chapitre 4.
- La conception d'un banc dédié à la stéréo-endoscopie. C'est l'objet du chapitre 5.

Enfin une conclusion fait le bilan des travaux réalisés et brosse quelques perspectives de recherches futures.

1.5 Références bibliographiques

[Carron_04] H.Carron, 'Architectures dédiées à l'acquisition en temps réel de signaux acoustiques d'échosondage large bande', thèse au LAMI, France(Toulouse), 2004.

Chapitre2

La vision tridimensionnelle

---

2 LA VISION TRIDIMENSIONNELLE

2.1 Introduction

Ce chapitre a pour but de présenter brièvement dans un premier temps quelques méthodes dédiées à la vision tridimensionnelle sans contact puis dans un deuxième temps et de manière plus détaillée la méthode dite de « stéréovision passive dense », ou « pixel-based stereovision » que nous avons retenue en vue de son adaptation et de son transfert sur une architecture dédiée. Avant de décrire l'algorithme utilisé dans ce contexte, nous présentons le modèle classique d'une caméra, la méthode de calibration des paramètres apparaissant dans ce modèle et de ce fait, les notations mathématiques utiles pour la suite. Dans la partie suivante la modélisation du banc stéréoscopique est présentée sommairement : il s'agit surtout de justifier l'opération de rectification des images acquises par les deux caméras du banc. Puis par la suite nous décrivons l'algorithme proprement dit, avec deux versions : une version séquentielle et une version multi-résolution. Enfin, les équations de la reconstruction 3D sont rappelées.

2.2 Etat de l'art en vision tridimensionnelle

Dans cette partie nous ne présenterons que les méthodes susceptibles de fonctionner dans un contexte « temps réel ».

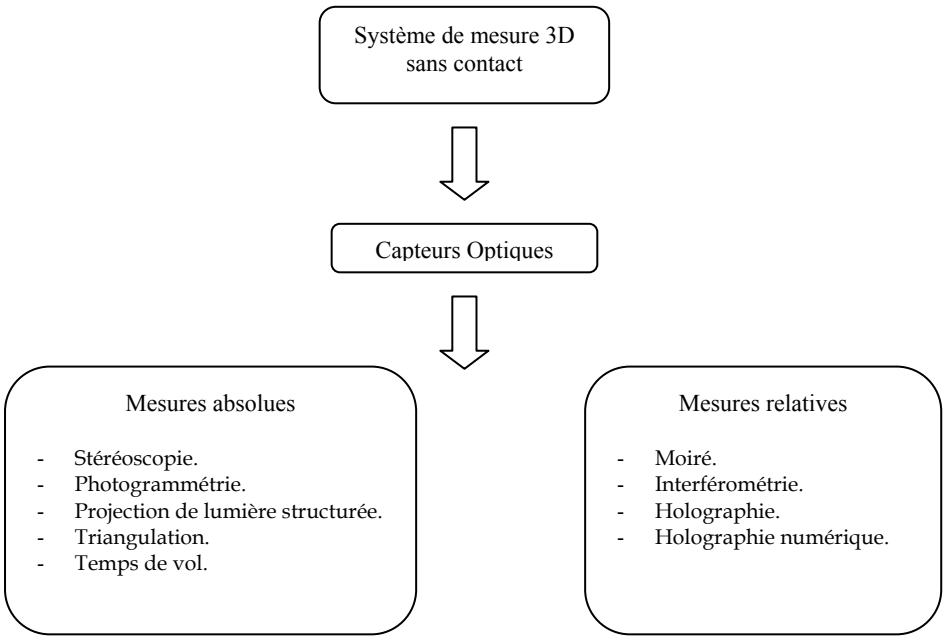


Figure 2-1 : Les techniques de vision tridimensionnelle.

La figure 2-1 donne un aperçu des différentes techniques utilisées. Parmi celles-ci on peut noter les deux types de classes : absolues ou relatives.

2.2.1 Techniques de mesure relative de distance

2.2.1.1 Technique de moiré

Le phénomène de moiré apparaît lorsque deux réseaux formés de traits équidistants alternativement opaques et transparents se superposent. Les figures de moiré peuvent être générées par différents types de réseau, par exemple, des réseaux circulaires, des réseaux linéaires, etc. En général, les réseaux utilisés en moiré sont transparents, avec une transmittance donnée par des fonctions carrées. Si deux réseaux sont illuminés avec des angles d'incidences différents ils génèrent dans l'espace des plans de lumière alternativement sombres et clairs. En plaçant un objet dans le champ ainsi obtenu on fait apparaître sur toute la surface de celui-ci une carte topographique tel qu'on peut le voir sur la figure 2-2, [Marshall_98].

Cette technique est bien adaptée pour suivre en temps réel les déformations d'objets. Les angles de projection ainsi que les caractéristiques des réseaux déterminent l'espacement entre deux franges successives.

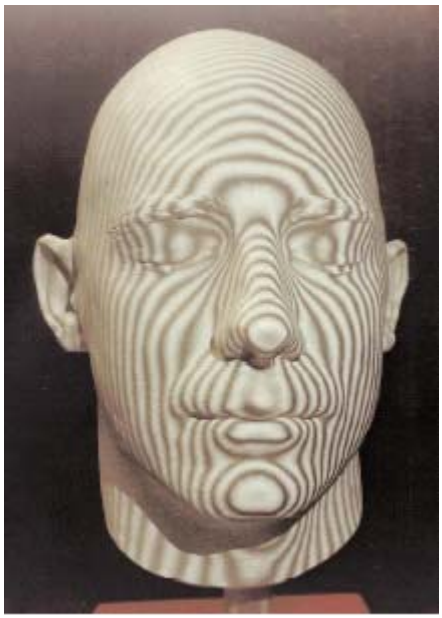


Figure 2-2 : Exemple de franges de moiré.

2.2.1.2 Techniques interférométriques

Le principe de l'interférométrie (figure 2-3) repose sur la mesure de la différence de chemin optique entre deux ondes. Associée à un dispositif à double balayage, il est possible de reconstruire les dimensions 3D d'un objet. Cependant cette technique est réservée à la mesure de très petites dimensions (<1mm).

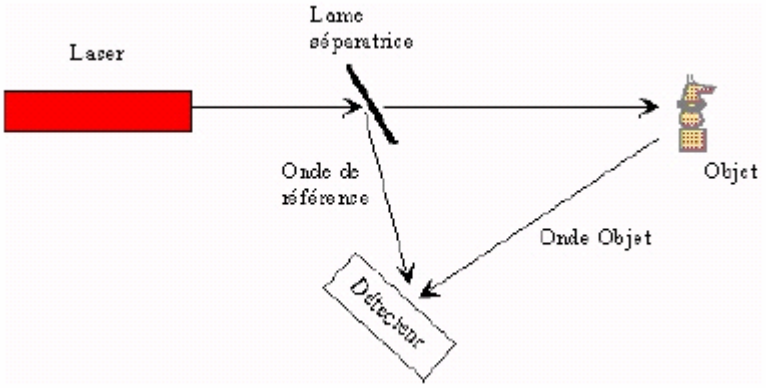


Figure 2-3 : Principe de l'interférométrie.

2.2.2 Méthodes de mesure absolues

2.2.2.1 Mesure du temps de vol

La technique du temps de vol (figure 2-4) repose sur la mesure du temps de propagation d'un faisceau lumineux monochromatique, cohérent et collimaté. Un dispositif à double miroir permet le balayage complet de l'objet à analyser. L'évolution récente des MEMS a permis une intégration très poussée de ce type de technique telle qu'on peut le voir dans la caméra TRIDICAM développée au laboratoire [Camon_01].

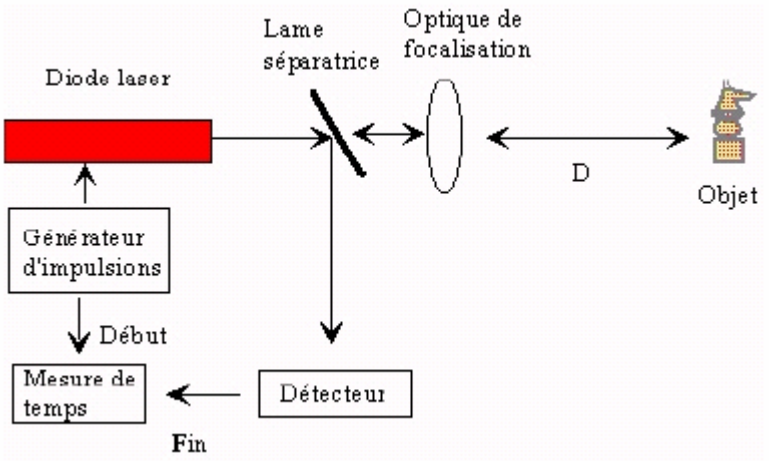


Figure 2-4 : Principe de la technique du temps de vol.

La source laser émet une impulsion lumineuse à l'instant *t*, qui est partiellement réfléchie par l'objet et qui est ensuite détectée au retour au temps $t + T$. Ce signal lumineux est converti en un signal électrique. L'équation de base est :

v x T = 2D où D est la distance entre l'objet et le détecteur, et v la vitesse de propagation du signal.

Les capteurs à temps de vol sont généralement destinés à évaluer de grandes distances (typiquement 10m) et ont été développés à l'origine pour la détection d'obstacles par des véhicules automatiques [Lewis_77, Besl_88].

2.2.2.2 Méthodes par projection de lumière structurée

Les systèmes de projection de lumière structurée comportent une ou plusieurs caméras et un laser. La projection du faisceau laser sur l'objet peut être réalisée par

différents moyens tels que la ligne [Boizard_85], la grille, la matrice de point, voire par balayage XY à partir de miroirs (figure 2-5). Dans ce dernier cas la caméra peut être remplacée par un dispositif à base de photodiodes délivrant deux courants proportionnels aux coordonnées X et Y du point image.

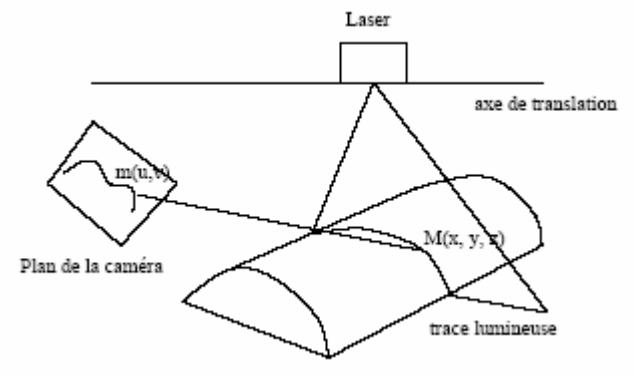


Figure 2-5 : Projection d'un plan de lumière cohérente.

2.2.2.3 Méthodes stéréoscopiques

La stéréoscopie est une technique qui permet d'obtenir des informations tridimensionnelles sur les coordonnées des points d'un objet à partir d'une ou plusieurs caméras localisées en différentes positions (figure 2-6).

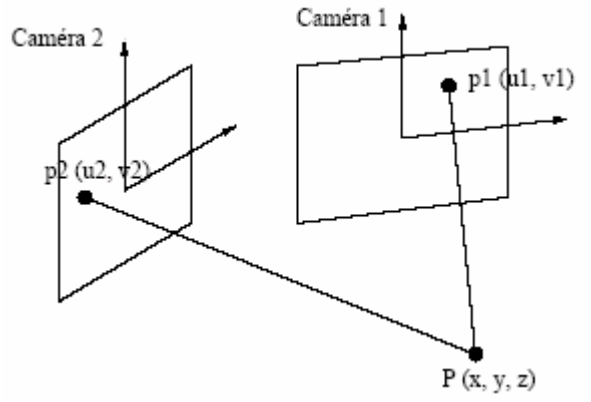


Figure 2-6 : Principe de la vision stéréoscopique.

La reconstruction tridimensionnelle des points à partir de leurs points images acquis dans *N* plans images (*N* correspond au nombre de caméras) repose sur les deux étapes suivantes :

- La calibration, consistant à estimer la matrice de transformation des points tridimensionnels en points bidimensionnels, étape réalisée pour les *N* caméras,

- La résolution d'un système de 2*N* équations du type $MiX - Ui = 0$, où l'inconnue est le point *X* de l'objet, où *Mi* est la matrice de calibration de la caméra *i* et où *Ui* sont les coordonnées images de *X*.

Ces techniques représentent le système de la vision humaine, en utilisant l'appariement entre les pixels homologues dans deux ou plusieurs images. Une fois ces pixels déterminés, on calcule les coordonnées du point correspondant dans l'espace par triangulation. Ces systèmes ont fait l'objet d'une importante littérature [Toscani_87, Luong_92, Champleboux_91, Ayache_89, Besançon_89, Ballard_82].

2.3 Bilan des différentes méthodes

Parmi les différentes méthodes présentées seules celles fournissant une mesure absolue sont adaptées. Nous avons retenu la méthode stéréoscopique et écarté la technique de mesure par temps de vol, trop imprécise pour notre application et plutôt réservée à l'analyse de scènes de dimensions importantes, ainsi que celles basées sur la projection de sources de lumière structurée car plus complexes à mettre en œuvre.

Dans les paragraphes suivant nous nous attacherons à présenter de manière plus détaillée la méthode stéréoscopique et les conséquences qui découlent de ce choix.

2.4 Principe de la stéréovision passive

La stéréovision passive est basée sur l'acquisition de deux images (gauche et droite) de la même scène. Dans ce cas un point P(X, Y, Z) peut être observé sur les deux images gauche et droite avec deux coordonnées différentes. La différence entre les deux coordonnées est utilisée ensuite pour calculer les coordonnées 3D du point P par une méthode simple de triangulation. En conséquence une des phases essentielle de cette méthode est la recherche des points stéréo-correspondants, laquelle peut être simplifiée si les systèmes de prise de vue sont alignés comme représentés dans la figure 2-7 et se limitant ainsi à une recherche ligne à ligne.

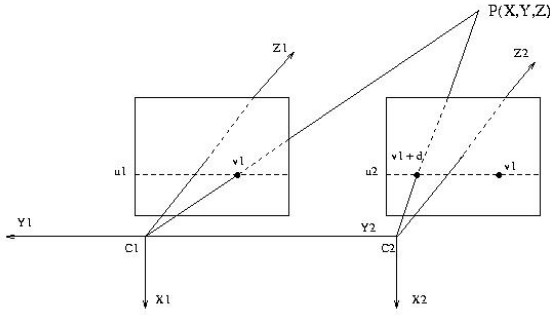


Figure 2-7 : Le Principe de la stéréovision passive.

Le synoptique global de la stéréovision passive est présenté sur le figure 2-8. En utilisant les paramètres de calibrage (les paramètres intrinsèques des deux systèmes de prise de vue, la transformation gauche-droite), les images initiales sont corrigées pour s'affranchir de l'erreur de positionnement et de la déformation des objectifs puis rectifiées. Les lignes épipolaires de chaque image sont ainsi mises en correspondance. Chaque pixel (u,v) de l'image gauche est alors comparé aux pixels de la même ligne dans l'image droite (voir le figure 2-7) dans un intervalle (u, v-Dmax) - (u,v). La différence entre les coordonnées de colonne v des Pixels assortis s'appelle la disparité D. La disparité est nulle pour deux Pixels sur lesquels un point 3D projeté est situé à une distance infinie. La disparité maximale Dmax est choisie en fonction de la distance minimale des points 3D que le système doit détecter.

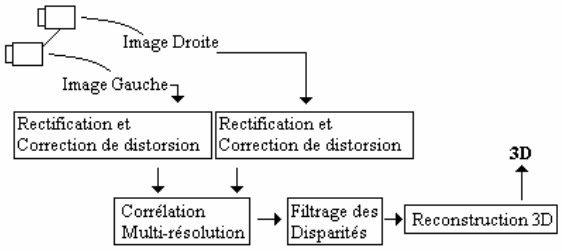


Figure 2-8 : Synoptique global de la stéréovision passive.

2.5 Modèle d'une caméra

Nous rappelons dans cette première section, le modèle classique que nous adoptons pour une caméra, à savoir un modèle à projection perspective directe tenant compte des distorsions optiques (indispensable pour notre application, vu la valeur des focales).

2.5.1 Les repères caméra, image et monde.

D'abord, définissons les différents repères mis en jeu pour exprimer le modèle d'une caméra. Le repère caméra CXYZ, repère 3D lié à la caméra, plus précisément à la lentille placée devant la matrice du capteur d'image représentée en figure 2-9 : origine C sur le centre optique de l'objectif, axes CX et CY choisis respectivement parallèles aux colonnes et aux lignes de la matrice, axe CZ choisi orthogonal au plan image, orienté vers la scène. En pratique, il correspond à l'axe optique de l'objectif.

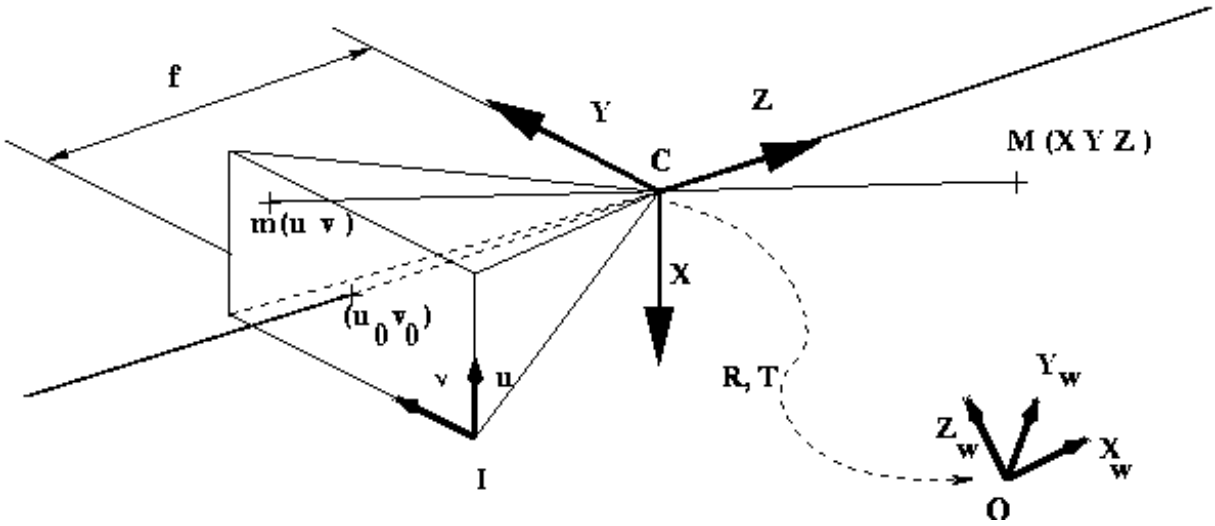


Figure 2-9 : Modèle géométrique de la camera.

La matrice du capteur est donc un rectangle, portée par un plan orthogonal à l'axe CZ, appelé ci-après plan image. Dans la vraie caméra, le plan image est à l'opposé de la scène perçue, donc vers les Z négatifs. De manière classique, mais en respectant le modèle mathématique de projection, il est souvent représenté devant, à une distance f de l'origine C. Cela permet dans les équations d'éviter l'inversion de l'image sur la matrice.

Les coordonnées (u,v) d'un point sur le plan image, sont exprimées dans un repère image (I,u,v), repère 2D défini de la manière suivante :

- Origine I dans le coin supérieur gauche.
- Axe I_u parallèle aux colonnes de la matrice.

- Axe I_v parallèle aux lignes de cette matrice.

Ces coordonnées (u,v) sont exprimées en unité pixel, donc ne tiennent pas compte de la taille des cellules CCD sur la matrice : elles correspondent aux indices (ligne, colonne) d'un point de cette matrice.

Le modèle sténopé est une simplification du modèle de l'optique de l'objectif ; il suppose qu'un point de la scène M(X,Y,Z) se projette sur le plan image, selon un rayon optique qui passe par le centre de l'objectif C. Dans ce modèle, la distance f est la focale de l'objectif : le plan image a donc l'équation (z = f) dans le repère caméra.

Cette représentation d'une caméra néglige donc la présence possible de plusieurs lentilles dans l'objectif, ou le fait que des rayons traversent la lentille sur toute sa surface. D'autres modèles plus complexes (modèles à lentille mince ou à lentille épaisse) existent, mais ne sont pas nécessaires pour notre application.

Pour que le repère caméra corresponde à un repère orthogonal, ce modèle suppose un montage mécanique parfait de la caméra, à savoir

- l'orthogonalité entre l'axe optique et le plan support de la matrice (photosensible),
- l'orthogonalité entre lignes et colonnes dans la matrice.

En générale, ces propriétés sont vérifiées. Il est possible de rajouter des paramètres dans le modèle d'une caméra pour tenir compte de tous les cas, mais cela complexifie les équations et peut rendre instable l'estimation des paramètres du modèle.

En pratique l'expression des coordonnées d'un point 3D dans le repère de la caméra n'est pas adaptée pour une application donnée. Il faut prendre en compte un repère scène ou monde noté $OX_wY_wZ_w$ en figure 2-10. Plusieurs repères 3D peuvent être introduits selon le niveau de traitement :

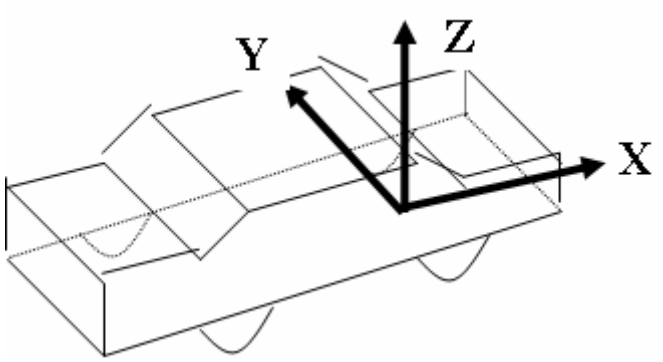


Figure 2-10 : Repère lié au véhicule pour une application vers l'avant.

- Pour une caméra montée dans un véhicule, la connaissance 3D, extraite initialement dans un repère caméra, sera ensuite exprimée vis-à-vis d'un repère lié au véhicule, défini par exemple, pour une application à l'avant du véhicule par :
    - Le plan OXY, plan du sol.
    - L'origine O, la projection au sol du milieu de l'essieu avant.
    - L'axe OX, orthogonal à l'axe des essieux, orienté vers l'avant.
    - L'axe OY, orienté vers la gauche du véhicule.
    - L'axe OZ, orthogonal au sol, orienté vers le haut.
- Le véhicule étant mobile, cette connaissance peut être ensuite exprimée dans un repère fixe lié à la route. Ce repère peut être défini comme le repère lié au véhicule lors de la première acquisition ; il peut être glissant pour ne retenir qu'une portion limité de l'environnement.

Ces différents repères 3D sont liés deux à deux par des transformations qui décrivent des déplacements euclidiens (rotation, translation) ; cette transformation est dite rigide si la situation relative entre les deux repères qu'elle lie est fixe (typiquement, la transformation entre repère caméra et repère véhicule). Elle est alors estimée hors-ligne lors d'une phase d'étalonnage.

Elle correspond à une localisation quand un des repères est mobile : localisation d'un obstacle vis-à-vis du véhicule, du véhicule vis-à-vis de la route….

Une transformation entre deux repères A et B est généralement représentée par une transformation homogène, matrice 4x4 notée T_{AB}, qui permet de calculer les coordonnées (X_B, Y_B, Z_B) d'un point N exprimées dans le repère B, à partir des coordonnées (X_A, Y_A, Z_A) de ce même point N exprimées dans le repère A.

Ces coordonnées sont exprimées sous la forme de vecteurs de 4 éléments notés

$\tilde{N}_A = (X_A\ Y_A\ Z_A\ 1)\ t$ et $\tilde{N}_B = (X_B\ Y_B\ Z_B\ 1)\ t$.

Elles sont appelées coordonnées homogènes. Avec cette transformation, nous avons

$\tilde{N}_A = T_{AB}\ \tilde{N}_B$

Rappelons la forme d'une matrice décrivant une transformation entre repères euclidiens :

$$T_{AB} = \begin{pmatrix} . & . & . & . \\ . & R_{AB} & . & t_{AB} \\ . & . & . & . \\ 0 & 0 & 0 & 1 \end{pmatrix}$$ où R_{AB} est une matrice de rotation 3 x 3 (normalisée, orthogonale), et t_{AB} est un vecteur de translation 3 x 1. Les colonnes de R_{AB} correspondent aux coordonnées des vecteurs unitaires des axes x, y, z du repère B exprimées dans le repère A. Le vecteur t_{AB} correspond aux coordonnées de l'origine du repère B exprimées dans le repère A.

La composition des transformations correspondra à des produits de matrices. Si trois repères A, B et C sont définis, si les transformations entre A et B d'une part, entre B et C d'autre part, sont connus respectivement par des matrices 4 x 4 T_{AB} et T_{BC}, alors la transformation, T_{AC} entre les repères A et C, est fournie par la matrice :

$T_{AC} = T_{AB}\ T_{BC}$

Lorsqu'une application met en jeu un grand nombre de repères, un graphe de repères orienté permet de représenter les transformations connues entre repères ; les

nœuds correspondent aux repères A, B, C…. une arête entre deux nœuds A et B traduit la connaissance de la transformation T_{AB}. Calculer la transformation entre deux repères quelconques revient à rechercher un chemin dans le graphe entre les deux nœuds correspondants et à composer les transformations associées aux arêtes de ce chemin.

Le modèle de la caméra incorpore donc, une transformation rigide entre le repère caméra et un repère de référence. Pour rester général, ce repère de référence est appelé repère monde.

Il correspondra ensuite au repère du corps solide sur laquelle la caméra est montée. Le point M a donc des coordonnées (X,Y,Z) dans le repère caméra, et des coordonnées (X_w, Y_w, Z_w) dans le repère monde.

2.6 Modèle géométrique d'une camera

2.6.1 Modèle à projection perspective directe

Dans cette section, nous décrivons le modèle mathématique d'une caméra, à savoir la fonction de transfert entre les coordonnées pixel (u,v) d'un point du plan image et les coordonnées 3D (X_w, Y_w, Z_w) exprimées dans le repère monde, des points de la scène qui se projettent en (u,v).

Le modèle "pinhole" ou sténopé; il ne considère que les rayons optiques qui passent par le centre optique (d'où l'appellation 'trou d'épingle').

Si nous notons (X,Y,Z) les coordonnées 3D du point M de la scène dans le repère de la caméra, et (x,y,f) les coordonnées métriques dans ce même repère, de la projection de M sur le plan image, alors la relation de la projection perspective donne :

$$\frac{X}{x} = \frac{Y}{y} = \frac{Z}{z}$$

On passe donc des coordonnées 3D de M aux coordonnées métriques de sa projection sur le plan image par la transformation suivante :

$$x = f\frac{X}{Z} \quad ; \quad y = f\frac{Y}{Z}$$

En fait, sur le plan image, les coordonnées sont exprimées en unité pixel, afin de prendre en compte l'acquisition de l'image: la fonction de luminance n'est connue que pour des positions discrètes du plan image, positions correspondant aux centres des cellules photosensibles. La discrétisation des points du plan image est représentée sous la forme d'une grille rectangulaire (cf. Figure 2-11 pour les caméras en résolution VGA: 480 lignes, 640 colonnes).

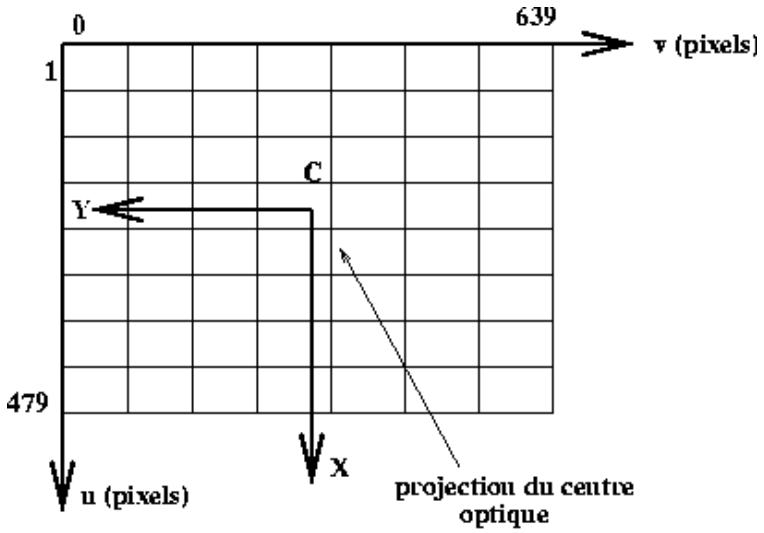


Figure 2-11 : Discrétisation du plan image.

Décrivons la transformation dans le plan image, des coordonnées métriques (x,y) en coordonnées pixel (u,v) dans la grille rectangulaire (tableau de pixels en figure2-11). Pour des valeurs entières, u et v correspondent à des indices ligne et colonne, et la luminance en un tel point du plan image est mesurée par la caméra. Mais la projection d'un point quelconque de la scène, a des coordonnées u et v réelles. La fonction de luminance en ce point du plan image, sera obtenue par interpolation à partir des valeurs de luminance acquises par la matrice.

Cette transformation exploite des caractéristiques de la caméra : (u_0, v_0) en pixels, les coordonnées pixel de la projection du centre optique.

$x = \frac{u - u_0}{k_u}$; $y = \frac{v - v_0}{k_v}$, où k_u et k_v en pixels/m (les facteurs de discrétisation en colonne et ligne).

La transformation des coordonnées 3D dans le repère caméra (X,Y,Z) en coordonnées pixel (u,v) dans le plan image:

$$u = u_0 + \alpha_u \frac{X}{Z}$$
$$v = v_0 + \alpha_v \frac{Y}{Z}$$

, u_0, v_0 (pixels) et $\alpha_u = k_u f$; $\alpha_v = k_v f$ (pixels) sont appelés les paramètres intrinsèques de la caméra.

En utilisant les coordonnées homogènes 3D, et les coordonnées image dans l'espace projectif associé au plan image (d'où le facteur s), ces équations peuvent s'écrire sous forme matricielle :

$$\begin{pmatrix} s_u \\ s_v \\ s \end{pmatrix} = \begin{pmatrix} \alpha_u & 0 & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = MatI \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

La matrice (3 x 4) contenant les paramètres de la caméra, est notée MatI, matrice intrinsèque. Cette équation montre que le facteur (s) correspond à la profondeur du point 3D dans la scène, exprimée dans le repère de la caméra.

En pratique, cette relation indique qu'on ne peut pas déterminer les coordonnées 3D (X,Y,Z) d'un point à partir des indices(u,v) de sa projection dans une image; par contre, on peut associer à un point de l'image, une droite 3D (rayon) qui porte le point 3D correspondant. La connaissance des paramètres intrinsèques de la caméra permet de calculer ce rayon, à partir de deux équations de plan. Dans le repère de la caméra :

$\alpha_u X - (u - u_0) Z = 0$ et $\alpha_v Y - (v - v_0) Z = 0$

Pour avoir un modèle géométrique complet de la caméra, il faut prendre en compte le repère monde, dans lequel l'utilisateur exprime les coordonnées 3D (X_w Y_w Z_w) des points perçus dans l'image. Il faut donc décrire le déplacement (rotation R et translation T) entre deux repères, par une transformation homogène qui lie les coordonnées exprimées dans ce repère de référence (X_w Y_w Z_w) avec celles exprimées dans le repère caméra (X Y Z):

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = R \begin{pmatrix} X_w \\ Y_w \\ Z_w \end{pmatrix} + T \quad \rightarrow \quad \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} R & T \\ 0 & 1 \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} = MatE \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$, où R est une matrice 3 x 3 de rotation et T un vecteur 3 x 1 de translation. La transformation homogène regroupant R et T est une matrice 4 x 4 notée MatE. Ces paramètres qui décrivent la situation de la caméra dans le repère monde, sont appelés paramètres extrinsèques de la caméra. La transformation homogène est notée E, matrice extrinsèque.

Le modèle complet de la caméra est donc:

$$\begin{pmatrix} s_u \\ s_v \\ s \end{pmatrix} = \begin{pmatrix} \alpha_u & 0 & u_0 & 0 \\ 0 & \alpha_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} R & T \\ 0 & 1 \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$

La relation entre un pixel 2D (u,v) de l'image et le point 3D correspondant (X_w Y_w Z_w) s'écrit donc:

$$\begin{pmatrix} s_u \\ s_v \\ s \end{pmatrix} = Mat\,\mathrm{Pr}oj \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix} = MatI \bullet MatE \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$

MatProj, MatI et MatE contiennent les paramètres suivants:

- MatProj, matrice 3 x 4 de projection perspective: 11 paramètres indépendants seulement car si MatProj est solution, alors λ MatProj l'est aussi (vu le facteur s).
- MatI, matrice intrinsèque : 4 paramètres α_u, α_v, u, v.
- MatE, matrice extrinsèque : 6 paramètres indépendants.
- 3 paramètres indépendants pour la rotation (angles d'Euler, angles de Bryant ou vecteur rotation): les 9 coefficients de la matrice R sont exprimés par rapport à ces 3 paramètres indépendants.
- 3 paramètres indépendants pour la translation.

Notons que MatProj contient 11 paramètres indépendants, alors que le produit MatI MatE n'en contient que 10. Il est possible donc d'introduire un 11ième paramètre pour décrire une caméra. Plusieurs auteurs [Horaud_95] introduisent un 5ième paramètre intrinsèque, appelé Skew, supposé décrire la non orthogonalité de la matrice photosensible.

En pratique, l'ajout de ce paramètre supplémentaire n'améliore pas le modèle, et au contraire, rend plus instable la procédure de calibrage. Il est donc négligé.

2.6.2 Méthode de calibrage du modèle sténopé

Il existe plusieurs protocoles de calibrage ou d'étalonnage des paramètres du modèle sténopé d'une caméra. Il faut d'abord disposer de nombreux quintuplets (u,v,X_w,Y_w,Z_w), chacun mettant en relation un point du monde 3D et sa projection sur le plan image. Ces quintuplets sont obtenus généralement grâce à des images acquises sur une mire de calibrage, connue. Le repère monde est initialement le repère de la mire : l'étalonnage donnera donc la situation de la caméra vis-à-vis de la mire.

La méthode classique, présentée notamment dans des articles de Toscani et Faugeras [Faugeras_86, Faugeras_87], procède en deux étapes :

- Estimation d'abord des 12 coefficients m_{ij} de la matrice MatProj par moindres carrés (système linéaire) avec une contrainte, pour prendre en compte le fait que seuls 11 coefficients sont indépendants.
- Calcul analytique ensuite des paramètres intrinsèques et extrinsèques, à partir de l'identification des termes de MatProj et du produit matriciel MatI MatE.
- Conditionnement des paramètres obtenus en résultats (pour la matrice de rotation R).

Pour éviter cette phase de conditionnement et obtenir une matrice de rotation viable sans post-traitement, le calibrage du modèle sténopé peut aussi se fonder sur une méthode d'optimisation non linéaire, afin d'estimer directement les 10 paramètres recherchés (4 intrinsèques et 6 extrinsèques), la rotation étant exprimée par 3 paramètres indépendants. Dans ce cas, il faut donner à la procédure des estimées initiales des inconnues.

Les valeurs estimées des facteurs k_u et k_v peuvent être obtenues à partir de la taille de la matrice et du nombre de pixels dans le tableau image. Vu le choix sur le repère de la caméra (CX parallèle aux colonnes, CY parallèle et opposé aux lignes), le facteur k_v est négatif, tandis que le facteur k_u est positif.

De même, les estimées initiales de (u_0, v_0) sont calculées en considérant que la matrice est centrée par rapport à l'axe optique. Donc la projection du centre optique est estimée par le centre de l'image.

Par exemple, soit une caméra ayant les caractéristiques suivantes :

- Matrice 1/2pouce, donc de longueur 6,4mm et de largeur 4.8mm,
- Image de 480 lignes de 640 pixels,
- Optique 2.1mm.

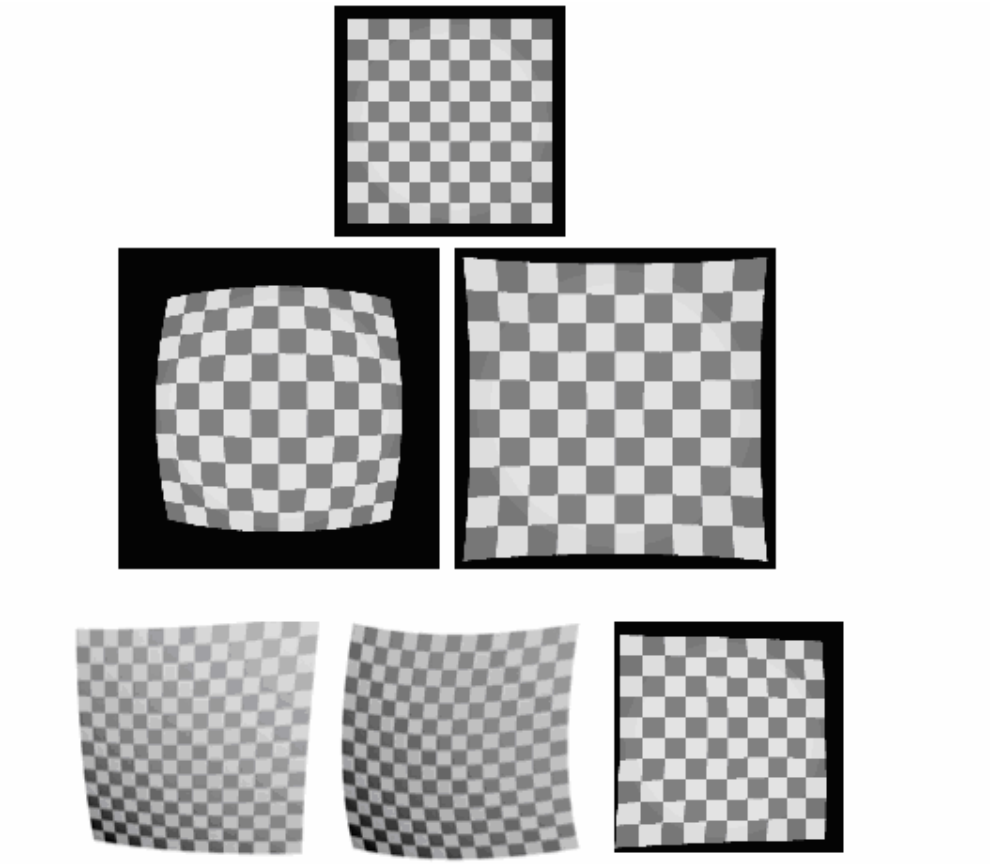
Les valeurs estimées de α_u, α_v, u_0 et v_0 sont:

α_u = (2.1x10-3) 480 / (4.8x10-3) = 210 ; α_v = (2.1x10-3) 640 / (6.4x10-3) = 210 ; u_0 = 240 ; v_0 = 320.

A partir de cette estimée initiale des paramètres intrinsèques, celle des paramètres extrinsèques peut être calculée pour chaque image de la mire, par une procédure de localisation (ou calcul de pose) de la mire à partir de l'ensemble des quintuplets issus de cette image, en exploitant par exemple, la procédure de Tsai [Tsai_87] ou de Puget et Skordas [Puget_90].

2.6.3 Modèle avec distorsions radiales

Les optiques présentent des aberrations géométriques, non décrites dans le modèle sténopé présenté ci-dessus. En fait, l'objectif contient généralement plusieurs lentilles, dont les caractéristiques sont calculées pour appliquer au mieux une projection perspective directe ; mais dans les optiques de faible coût, ou avec une focale très courte (grand angle), le modèle sténopé n'est pas respecté et les rayons optiques subissent des déviations dans l'objectif.



en haut, image non distordue
au milieu, distorsions radiales en coussinet (à gauche) et en barillet (à droite)
en bas, distorsion de décentrage (à gauche) et prismatique (à droite)

Figure 2-12 : Exemples de distorsions sur une image de synthèse.

Les distorsions géométriques se décomposent en plusieurs types : les plus importantes sont le décentrage et les distorsions radiale et tangentielle (aussi appelée prismatique).

Les distorsions de décentrage proviennent d'un mauvais alignement des lentilles à l'intérieur de l'objectif monté sur la caméra.

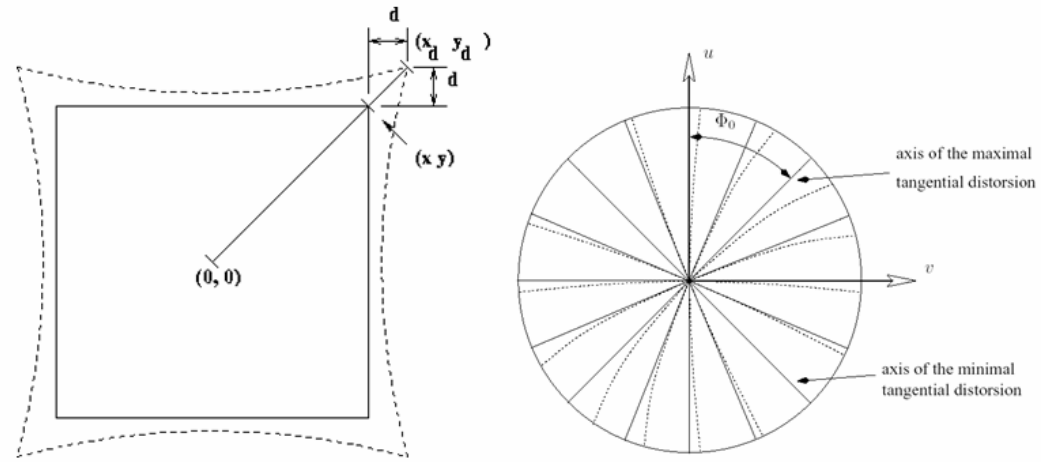


Figure 2-13 : Distorsions radiales et tangentielles.

La distorsion radiale provient de défaut de courbure sur les lentilles de l'objectif. Elle se manifeste dans le plan image, comme une translation (d d) le long du rayon (voir figure 2-13 à gauche) joignant la projection du centre optique (non loin du centre de l'image) et le pixel considéré. L'amplitude de la translation est proportionnelle à la distance entre ces deux points.

La distorsion prismatique tangentielle provient d'un défaut de parallélisme entre le plan image et les plans des lentilles sur l'objectif. Cela provoque en un pixel, une translation le long de la tangente au cercle centré sur la projection du centre optique et passant par ce pixel. (Voir figure 2-13 à droite). La norme de cette translation varie en fonction de la position du pixel sur le cercle ; il existe un diamètre sur lequel la distorsion est nulle, tandis que sur le diamètre orthogonal, elle est maximale.

La figure 2-12 présente des exemples de distorsions. Des images distordues sont obtenues en appliquant des modèles de distorsion à une image de synthèse d'un damier. En haut, nous présentons l'image sans distorsion, au milieu à gauche une distorsion en coussinet, à droite une distorsion en barillet, en bas à gauche une distorsion de décentrage et à droite une distorsion prismatique (dans les deux cas, torsions dans l'image).

En général, pour une caméra standard, pour une application ne nécessitant pas une précision importante (typiquement, détection d'obstacles depuis un véhicule), les distorsions dites de décentrage et tangentielle sont négligeables : de plus, leur prise en compte rend plus instables les processus de calibrage. Nous avons validé lors de nombreux calibrages, que les paramètres décrivant ces distorsions, sont très faibles, et donc non significatifs.

Sur le plan image, on distingue alors

- Les coordonnées idéales (x, y), qui correspondent à la projection perspective parfaite.
- Les coordonnées distordues (x_d, y_d) (cf. Figure2-13 à gauche), obtenues à partir des coordonnées idéales, en modélisant la distorsion.

Rappelons la transformation entre coordonnées 3D (X,Y,Z) dans le repère caméra, et les coordonnées métriques idéales (x, y) sur le plan image, avec le modèle idéal pinhole :

$$x = f\frac{X}{Z}; y = f\frac{Y}{Z}$$

Le pixel réel, image du point 3D (X, Y, Z), aura en fait les coordonnées distordues. Exprimons la transformation entre coordonnées métriques idéales (x, y) et coordonnées métriques distordues (x_d, y_d) par distorsion radiale:

$$x_d = x\,[1 + k_1 r^2 + k_2 r^4 + k_3 r^6]$$

$$y_d = y\,[1 + k_1 r^2 + k_2 r^4 + k_3 r^6]$$

, avec k_1, k_2, k_3 … coefficients de distorsion et $r^2 = x^2 + y^2$ est la distance au point principal.

Le nombre de coefficients nécessaires pour décrire une caméra donnée, est déterminé lors de la procédure de calibrage. Il existe un nombre optimal, en fonction d'une focale donnée de l'objectif.

Généralement, pour des focales grandes (plus de 12mm) de bonne qualité, les distorsions peuvent être négligées. L'ordre 1 suffit pour des focales 8.5mm de bonne qualité.

Dans les applications robotiques, avec des focales 4.8mm, on se satisfait de 3 coefficients. Pour des focales très courtes au contraire, de meilleurs résultats peuvent être obtenus en introduisant d'autres coefficients (par exemple il y 5 coefficients de distorsions pour décrire des focales 2.6mm).

Prendre un ordre trop élevé provoque une sur-paramétrisation, ce qui peut entraîner des problèmes lors de l'estimation.

Notons que c'est dans ces équations liant coordonnées distordues (x_d, y_d) et coordonnées idéales (x, y), que peuvent être introduites les distorsions tangentielles, avec différents ordres (2 paramètres à l'ordre 2) (voir les équations dans [Li_95]).

Pour exprimer le modèle de la caméra prenant en compte les distorsions, il faut encore transformer les coordonnées métriques en coordonnées pixel :

$$x_d = \frac{u_d - u_0}{k_u} \quad ; \quad y_d = \frac{v_d - v_0}{k_v}$$

$$x = \frac{u - u_0}{k_u} \quad ; \quad y = \frac{v - v_0}{k_v}$$

La transformation entre coordonnées 3D (X Y Z) dans le repère caméra, et les coordonnées pixel distordus (u_d v_d), s'obtient en combinant les différentes équations.

En posant $K_1 = k_1 f^2$, $K_2 = k_2 f^4$, $K_3 = k_3 f^6$, on obtient le modèle non linéaire suivant :

$$u_d = u_0 + \left(\alpha_u \frac{X}{Z}\right)\left[1 + k_1\left(\frac{X^2}{Z^2} + \frac{Y^2}{Z^2}\right) + k_2\left(\frac{X^2}{Z^2} + \frac{Y^2}{Z^2}\right)^2 + k_3\left(\frac{X^2}{Z^2} + \frac{Y^2}{Z^2}\right)^3\right]$$

$$v_d = v_0 + (v - v_0)\left[1 + \left[1 + k_1\left(\frac{X^2}{Z^2} + \frac{Y^2}{Z^2}\right) + k_2\left(\frac{X^2}{Z^2} + \frac{Y^2}{Z^2}\right)^2 + k_3\left(\frac{X^2}{Z^2} + \frac{Y^2}{Z^2}\right)^3\right]\right]$$

Rappelons qu'il faut rajouter à cette équation, la transformation des coordonnées 3D (X, Y, Z) dans le repère caméra, en (X_w, Y_w, Z_w) exprimées dans le repère monde. Ce modèle comporte donc 19 paramètres : 4 intrinsèques (α_u, α_v, u_0, v_0), 3 coefficients de distorsion (K_1, K_2, K_3) et 12 extrinsèques (dont 6 seulement sont indépendants).

2.6.4 Méthode de calibrage du modèle avec distorsions

Notons que ce modèle est non linéaire. La matrice de projection MatProj n'apparaît plus de manière explicite. Donc en pratique, il faut utiliser la méthode d'optimisation non linéaire évoquée pour le calibrage des paramètres du modèle sténopé.

Généralement, pour l'estimation initiale des paramètres, il est souvent procédé dans un premier temps, à l'étalonnage des paramètres du modèle sténopé, donc en considérant nuls, les coefficients K_1, K_2, K_3 … Les résultats de ce premier calibrage sont ensuite fournis comme estimées initiales à l'étalonnage complet.

Notons que la prise en compte des distorsions nécessite un protocole de calibrage plus rigoureux. En particulier, il faut que les quintuplets (u, v, X, Y, Z) fournis en entrée, couvrent tout le plan image, et notamment les bordures puisque les distorsions radiales seront plus fortes sur les bords. Avec une seule image acquise sur une mire de calibrage 3D, cela se révèle très difficile ; c'est la raison pour laquelle sont apparues des méthodes de calibrage sur mires planes, donc prenant en entrée de nombreuses images sur une mire déplacée devant la caméra.

Généralement ces positions de mire ne sont pas référencées vis-à-vis d'un seul repère monde ; pour chaque image, il faut donc estimer des paramètres extrinsèques particuliers. Si la procédure exploite (NbImages) images de la mire, il conviendra d'estimer 4 paramètres intrinsèques + 3 coefs de distorsion + 6 NbImages paramètres extrinsèques, soit (6 NbImages + 7) paramètres.

Si la mire comporte (NbIndices) indices visuels, et en considérant que tous ces points sont extraits dans les NbImages images, ces paramètres devront être étalonnés à partir de 2(NbIndices x NbImages) équations (2 équations par point, une pour u, une pour v).

2.6.5 Correction des distorsions

De nombreuses méthodes de Vision Artificielle se fondent sur le modèle sténopé parfait, donc non perturbé par les distorsions géométriques ; ce sont toutes les méthodes exploitant soit la projection de points 3D sur le plan image, soit la rétroprojection d'un pixel de ce plan image sur la scène, selon le rayon optique.

La projection ou la rétroprojection ne sont valides que sur des images non distordues. Avant de les appliquer, il convient donc de corriger les distorsions, ou de vérifier qu'elles peuvent être négligées. Pour corriger une image, il faut trouver la correspondance entre les coordonnées pixel distordues (u_d, v_d) et les coordonnées idéales (u, v). Ces deux coordonnées sont liées par les équations

$x_d = x\,[1 + k_1 r^2 + k_2 r^4 + k_3 r^6]$

$y_d = y\,[1 + k_1 r^2 + k_2 r^4 + k_3 r^6]$

, avec $r^2 = x^2 + y^2$ et $x_d = \frac{u_d - u_0}{k_u}$; $y_d = \frac{v_d - v_0}{k_v}$ qui montrent que pour un

$x = \frac{u - u_0}{k_u}$; $y = \frac{v - v_0}{k_v}$

pixel (u, v) de l'image corrigée, sa position dans l'image distordue (u_d, v_d) peuvent être calculées directement.

Par contre, pour un pixel (u_d, v_d) de l'image distordue, sa position dans l'image corrigée, est donnée par les racines d'équations polynomiales, ici de degré 7.

Après acquisition, la fonction de luminance est connue uniquement pour les coordonnées entières distordues (u_d, v_d). Le but de la correction des distorsions, consiste à synthétiser la fonction de luminance pour des coordonnées entières idéales (u, v).

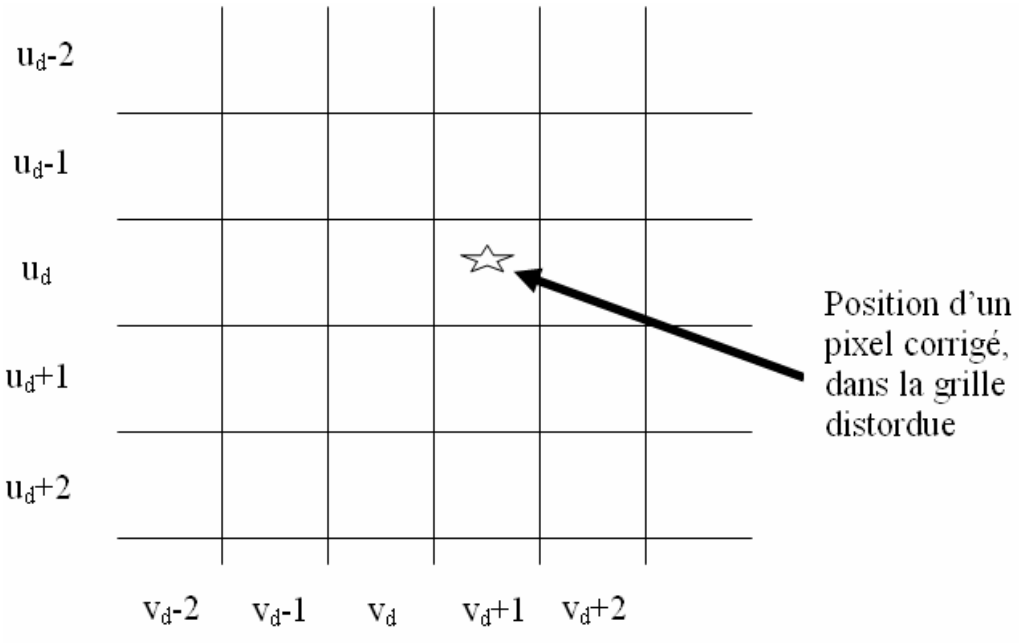


Figure 2-14 : Correction des distorsions.

Pour ce faire, pour chaque pixel (u, v) de l'image corrigée, avec u et v indices entiers, les coordonnées (u_d, v_d) sont calculées par les équations rappelées ci-dessus. Selon le type de distorsion, le résultat peut donner des coordonnées non entières, éventuellement négatives ou plus grandes que (d_u - 1) ou (d_v - 1). Quand elles sont dans l'intervalle [0, d_u - 1] et [0, d_v - 1], la valeur de la fonction de luminance en ce pixel (u, v) (pour la couleur, le traitement de la mosaïque doit être fait au préalable), est obtenue par interpolation à partir des luminances connues pour les pixels de l'image distordue entourant (u_d, v_d) (voir figure2-14). Cette interpolation peut être simplement bilinéaire, mais peut aussi utiliser des fonctions B-splines cubiques ou quintiques.

Plusieurs stratégies existent, les valeurs de u et v peuvent évoluer seulement dans les limites [0, d_u - 1], [0, d_v - 1] (premier exemple en figure 2-15), ce qui généralement conduit à une réduction du champ perçu dans l'image corrigée (les bords de l'image d'origine ne sont pas pris en compte) et à un grossissement pour la zone corrigée. Ces valeurs sont prises entre des limites prédéterminées pour que tout le champ perçu dans l'image d'origine soit corrigé (deuxième exemple en figure 2-15).

La correction de distorsion provoque donc une interpolation et pour éviter des interpolations successives, cette fonction est souvent associée à d'autres traitements qui nécessitent aussi une telle interpolation à partir de l'image initiale (réduction, construction d'une pyramide d'images, rectification…).

Notons que la correction est un bon moyen de juger de la qualité du calibrage: on montre le résultat de ce traitement en figure 2-15. Notons, dans ces deux cas, qu'il s'agit d'une distorsion en coussinet.

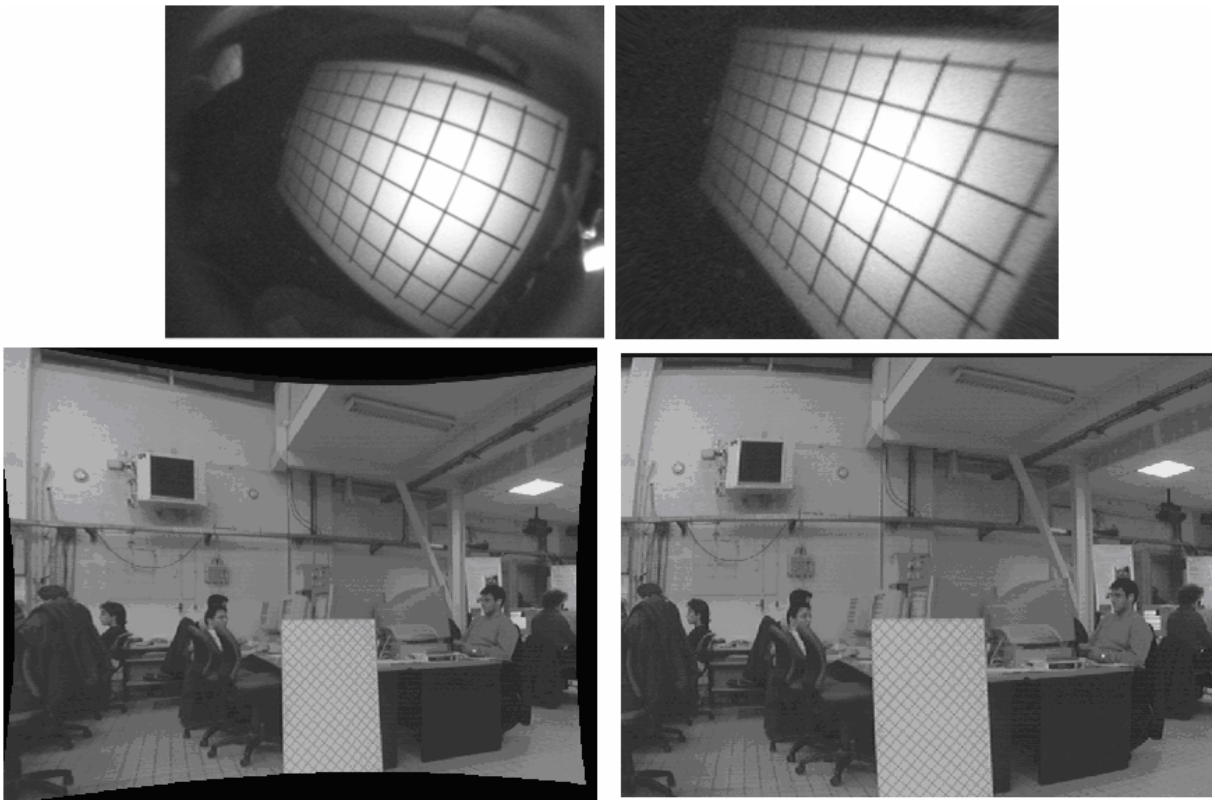


Figure 2-15 : exemple de correction des distorsions.

2.7 Modèle géométrique d'un banc de stéréovision

La stéréovision est une méthode d'acquisition d'informations 3D par vision. La stéréovision binoculaire s'apparente à la vision humaine. Avec une caméra, un pixel est l'image d'un point 3D qui se trouve sur une droite dans l'espace ; avec un système stéréoscopique à 2 caméras, à partir de 2 images d'une même scène, on peut retrouver la profondeur de la scène (l'information 3D), si on sait traiter :

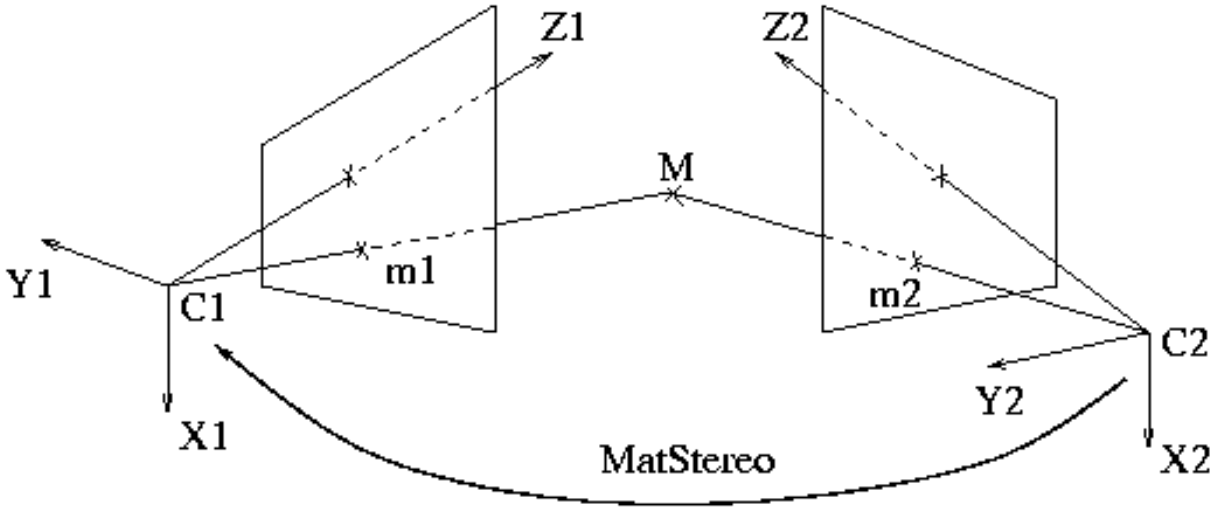


Figure 2-16 : la stéréovision binoculaire.

- Le calibrage du système stéréo;
- La mise en correspondance des pixels m_1 de l'image1 et m_2 de l'image2, qui sont les projections dans ces images, de mêmes points 3D M (voir figure 2-16);
- La reconstruction 3D: calculer les coordonnées 3D des points M, par intersection des 2 droites issues de m_1 et m_2.

2.7.1 Calibrage du banc de stéréovision

Le capteur stéréo est donc constitué de deux caméras, généralement assemblées de manière robuste; il est décrit par les modèles des deux caméras et par la transformation rigide MatStereo liant leurs repères.

L'espace tridimensionnel de la scène est muni de son repère monde R_w. Chacune des deux caméras possède son propre repère caméra: nous les appellerons repère caméra gauche R_1 et repère caméra droite R_2.

La figure 2-17 montre le graphe liant ces trois repères ; les arêtes de ce graphe correspondent aux transformations permettant d'exprimer un point dans un autre référentiel. Ces transformations MatStereo, E1 et E2 rigides sont exprimées par des matrices homogènes.

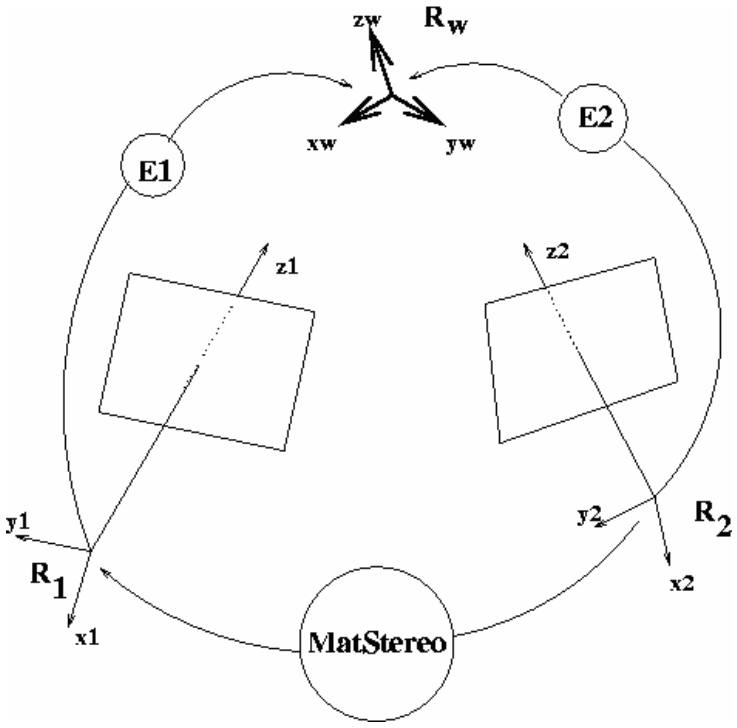


Figure 2-17 : Les trois référentiels tridimensionnels du capteur de stéréovision.

Rappelons nos notations :

- E1 est la transformation entre repère caméra gauche et repère monde. Elle contient une matrice de rotation 3 x 3 R_1 et un vecteur de translation T_1.
- E2 est la transformation entre repère caméra droite et repère monde. Elle contient une matrice de rotation 3 x 3 R_2 et un vecteur de translation T_2.

Si M est un point 3D exprimé dans le repère monde, M_1 désigne ce même point exprimé dans le repère caméra gauche, M_2 ce même point exprimé dans le repère caméra droit.

I_1 et I_2 sont les matrices des paramètres intrinsèques pour les deux caméras.

Rappelons les relations dues au modèle sténopé adopté pour les deux

$$\begin{pmatrix} s_{u1} \\ s_{v1} \\ s \end{pmatrix} = I_1 \begin{pmatrix} X_1 \\ Y_1 \\ Z_1 \\ 1 \end{pmatrix} \quad ; \quad \begin{pmatrix} X_1 \\ Y_1 \\ Z_1 \\ 1 \end{pmatrix} = E_1 \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

caméras :

$$\begin{pmatrix} s_{u2} \\ s_{v2} \\ s \end{pmatrix} = I_2 \begin{pmatrix} X_2 \\ Y_2 \\ Z_2 \\ 1 \end{pmatrix} \quad ; \quad \begin{pmatrix} X_2 \\ Y_2 \\ Z_2 \\ 1 \end{pmatrix} = E_2 \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

, qui peut aussi s'écrire:

$$\begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix} = K_1 \begin{pmatrix} X_1 / Z_1 \\ Y_1 / Z_1 \\ 1 \end{pmatrix} \qquad \begin{pmatrix} u_2 \\ v_2 \\ 1 \end{pmatrix} = K_2 \begin{pmatrix} X_2 / Z_2 \\ Y_2 / Z_2 \\ 1 \end{pmatrix}$$

, avec K_1 et K_2 matrices 3 x 3 égales à :

$$K_1 = \begin{pmatrix} \alpha_{u1} & 0 & u_{01} \\ 0 & \alpha_{v1} & v_{01} \\ 0 & 0 & 1 \end{pmatrix} \qquad K_2 = \begin{pmatrix} \alpha_{u2} & 0 & u_{02} \\ 0 & \alpha_{v2} & v_{02} \\ 0 & 0 & 1 \end{pmatrix}$$

C'est une expression des paramètres intrinsèques, variantes des matrices I_1 et I_2, dont nous verrons l'utilité ci après.

Prises séparément, ces équations sont non inversibles: un pixel (u_1, v_1) ou (u_2, v_2) est l'image d'un point qui est sur une droite donnée. Néanmoins, on a:

$$\begin{pmatrix} X_2 \\ Y_2 \\ Z_2 \\ 1 \end{pmatrix} = E_2 \bullet E_1^{-1} \begin{pmatrix} X_1 \\ Y_1 \\ Z_1 \\ 1 \end{pmatrix}$$

Cette transformation MatStereo, caractéristique du banc stéréo, est notée de la manière suivante:

$$MatStereo = E_2 \bullet E_1^{-1} = \begin{pmatrix} . & . & . & t_x \\ . & RotStereo & . & t_y \\ . & . & . & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

RotStereo (matrices 1 x 3) est la matrice de rotation liant les deux caméras, t_x, t_y et t_z sont les composantes de la translation liant les deux caméras.

Le calibrage d'un banc stéréoscopique, consiste à estimer la transformation MatStereo. Cette opération se fait très facilement, en calibrant séparément les deux caméras du banc, en exploitant des images de mires disposées dans les mêmes positions. Si le repère monde est identifié lors de cette étape au repère de la mire, le calibrage des deux caméras fournit les transformations E_1 et E_2, matrices des paramètres extrinsèques. L'estimation de MatStereo est donc obtenue par les équations ci-dessus.

Notons qu'on n'utilise dans ces équations, que le modèle sténopé, sans distorsions : la stéréovision nécessite la correction au préalable des distorsions dans les images acquises par les caméras.

2.7.2 Géométrie épi-polaire

Lorsqu'un point M de la scène est visible simultanément par les deux caméras, sa projection dans les images nous donne deux points, notés m_1 pour la caméra gauche, et m_2 pour celle de droite. Par construction géométrique, le correspondant m_2 du point m_1 de l'image1, se trouve sur une droite dans l'image2, appelée droite épi-polaire correspondant au point m_1.

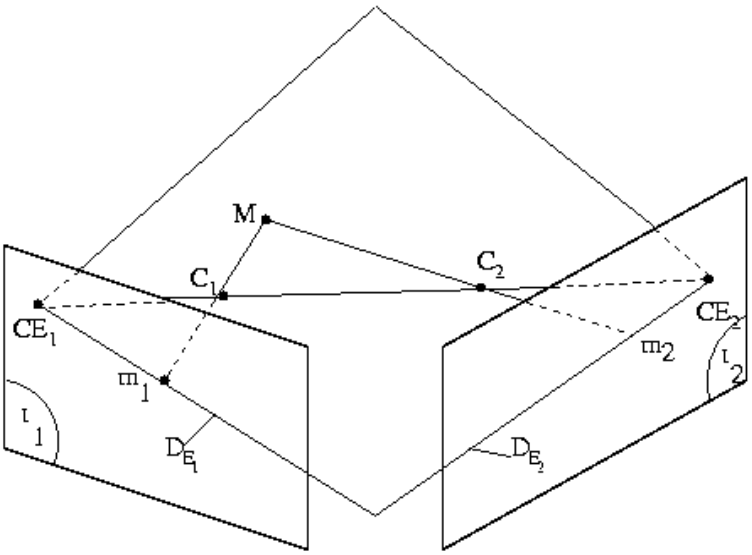


Figure 2-18 : géométrie épi-polaire.

Cette droite est définie par l'intersection du plan support du triangle (C_1, C_2, m_1) et du plan support de l'image2 (cf. figure 2-18). Toutes les droites épi-polaires dans l'image2 (resp. image1) convergent vers un point, qui est l'intersection de la droite portant le segment (C_1, C_2) et du plan support de l'image2 (resp. image1).

Ce point est appelé, le centre épi-polaire de l'image2 CE2 (resp. image1 CE1). Les droites épi-polaires dans une image, forment donc un faisceau, issu du centre épi-polaire.

Les coordonnées pixel dans l'image2 du centre épi-polaire CE2 sont données par la projection de $C_1 = (b_x, b_y, b_z)$ dans l'image2.

Le calibrage permet de calculer la droite épi-polaire dans l'image2 (resp. image1), correspondant à un pixel (u_1, v_1) de l'image1 (resp. (u_2,v_2) de l'image2): l'équation de cette droite est

$$\begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix} \bullet F \bullet \begin{pmatrix} u_2 \\ v_2 \\ 1 \end{pmatrix} = 0$$

On montre (voir [Horaud_95]) que la matrice F s'exprime de la manière suivante en fonction d'une matrice E :

$F = K_2^{-1} E K_1^{-1}$, où on retrouve les paramètres intrinsèques des deux caméras. La matrice E est définie par rapport à la matrice MatStereo qui décrit la géométrie du banc stéréo:

$$E = \begin{pmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{pmatrix} RotStereo$$

Ces matrices E et F sont connues sous les noms suivants :

- F est la matrice fondamentale.
- E est la matrice essentielle.

2.7.3 Rectification des images

L'opération de rectification d'images (voir figure 2-19) permet de rendre les épi-polaires horizontales, ou encore de renvoyer les centres épi-polaires à l'infini :

Avant rectification, le correspondant de m_1 de I_1 est un pixel m_2 de I_2 situé sur la ligne épi-polaire associée à m_1.

Après rectification, le correspondant d'un point m'_1 de I'_1 est un pixel m'_2 de I'_2 situé sur la même ligne que m'_1.

La rectification des images a donc pour finalité de définir des « caméras virtuelles » dites caméras rectifiées dont les plans image sont totalement alignés, de calculer les positions de ces caméras rectifiées par rapport aux caméras réelles, puis de synthétiser deux images droite et gauche qu'elles permettraient d'acquérir, à partir des vraies images. Pour un pixel m'_1 (u'_1, v'_1) d'une image rectifiée, il s'agit de calculer les coordonnées correspondantes (u_1, v_1) dans l'image réelle, puis de récupérer la luminance en ce point. Les coordonnées u_1 et v_1 étant réelles, cette

opération exploite une interpolation dans l'image réelle, dans notre cas, une interpolation bilinéaire.

Pour limiter la complexité des traitements précédents la stéréovision et le nombre d'interpolations, la rectification se fait dans le même parcours des images que la correction des distorsions : les caméras rectifiées respectent parfaitement le modèle sténopé. Elles ont les mêmes paramètres intrinsèques.

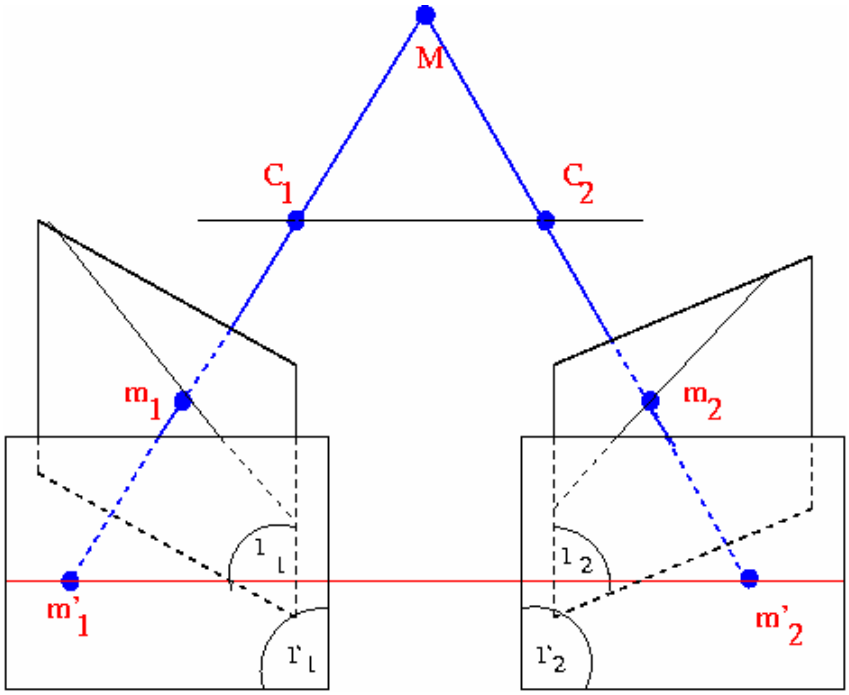


Figure 2-19 : l'opération de rectification.

Il existe plusieurs algorithmes de rectification qui diffèrent par la situation relative des deux caméras rectifiées. La solution adoptée est décrite en [Lasserre_95] : elle permet de minimiser les déformations entre images d'origine et images rectifiées. Caméras réelles et caméras rectifiées ont les mêmes centres optiques : l'algorithme calcule seulement des rotations permettant de confondre les plans image, puis d'aligner les lignes des images.

2.8 Algorithme de Stéréovision

Différentes entités image sont utilisables pour la mise en correspondance entre deux images stéréo: le point, le segment, la chaîne de contour, la région … Dans le

projet PICAS$O, nous nous intéressons à des appariements de pixels, car c'est la seule méthode qui donne une image dense de disparités et donc un maximum d'informations sur la scène à traiter.

2.8.1 Corrélation ou mise en correspondance entre images stéréoscopiques

Le but est de :

- Mettre en correspondance deux pixels 2D (droite et gauche) correspondants au même point 3D.
- Calculer la position 3D (la profondeur) de ce point, en exploitant la disparité (d sur la figure 2-20), c'est-à-dire la différence de position des deux points dans les images droite et gauche.

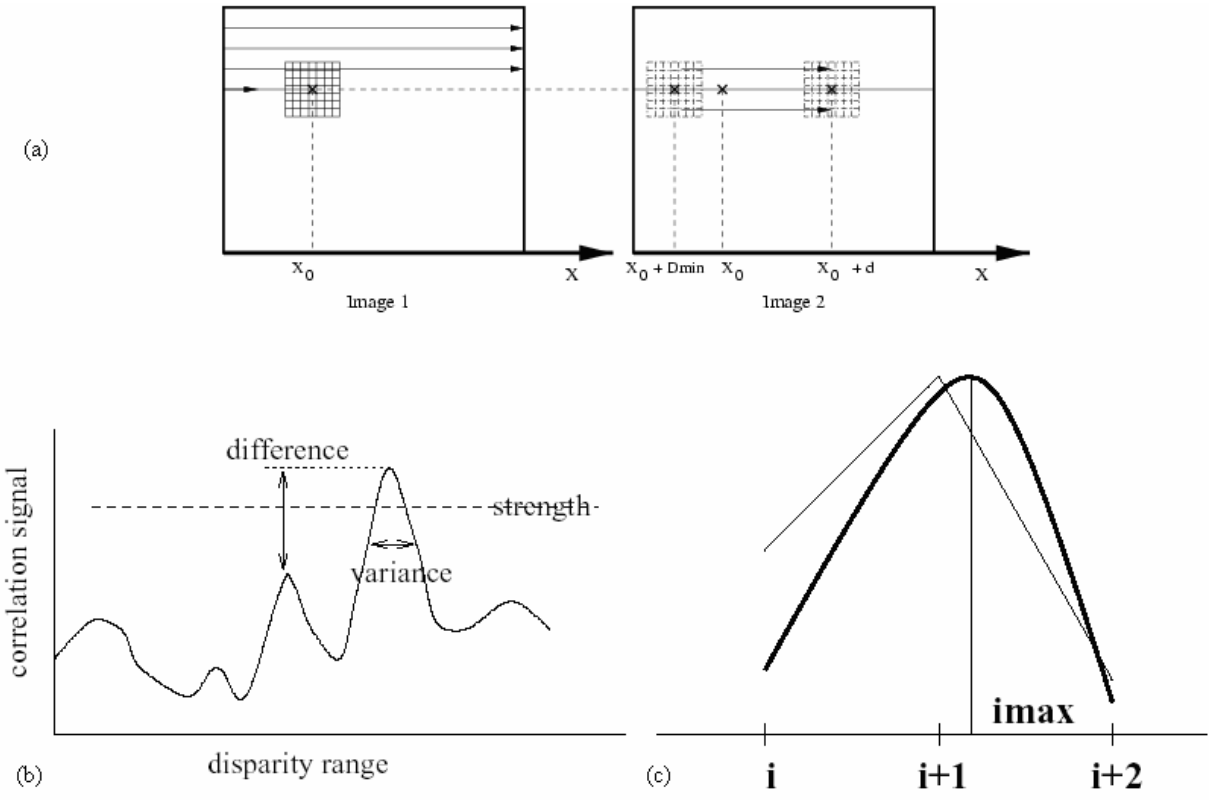


Figure 2-20 : Recherche du pixel stéréo-correspondant.

Avec cette méthode, pour garantir de bonnes performances, la rectification et la correction des distorsions sont indispensables pour une prise en compte directe de la contrainte épi-polaire. Vu ces opérations, le correspondant du pixel m'_1 de l'image

gauche rectifiée, est sur la même ligne dans l'image droite rectifiée (ce qui apparaît explicitement en figure 2-20.a : la réciproque est évidemment vraie.

Les lignes seront donc traitées successivement ; pour un pixel en position x_0 sur une ligne de l'image gauche, son correspondant sera recherché dans un intervalle de disparité $[D_{min}, D_{max}]$ sur la même ligne de l'image droite. Sur des images rectifiées, on a généralement $D_{max} = 0$ (point 3D à l'infini) et $D_{min} = - f$ (dist_min).

L'écart est maximal (en valeur absolue) pour une distance minimale des points appariés, distance dépendant de l'application et de la configuration géométrique du banc stéréo.

Pour savoir quel pixel dans cet intervalle est le correspondant de x_0, on utilise un critère de ressemblance local: la corrélation sur les niveaux de gris dans un voisinage carré, centré sur les pixels analysés. Ce voisinage est appelé fenêtre de corrélation. La taille de cette fenêtre est un paramètre important de l'algorithme : cela va habituellement de 3 x 3 à 15 x 15.

Différents critères numériques de corrélation ont été proposés comme critère de ressemblance entre deux fenêtres de corrélation, une prise autour du pixel m'_1 de l'image gauche, l'autre prise autour d'un pixel candidat pris sur la même ligne de l'image droite. Parmi ces critères on peut citer :

- SSD : Sum of Squared Differences, ou somme des différences quadratiques entre termes correspondants des deux fenêtres.
- SAD : Sum of Absolute Differences, ou somme de la valeur absolue des différences entre termes correspondants des deux fenêtres.
- ZSSD : Zero mean Sum of Squared Differences, identique à SSD sauf que, au préalable, pour chaque fenêtre de corrélation, la moyenne des termes est d'abord calculée et soustraite à chaque terme: la différence est donc faite entre les écarts à la moyenne des termes, et non entre les termes eux-mêmes. Ceci permet d'obtenir un critère invariant à des translations uniformes des luminances dans une des images.

- ZSAD: Zero mean Sum of Absolute Differences, identique à SAD, mais différence entre écarts à la moyenne. C'est donc un critère invariant aux variations uniformes de luminance dans une des images.
- NCC : Normalized Cross Correlation, le critère le plus utilisé, mais aussi le plus long à calculer. C'est la somme des produits entre les termes correspondants des fenêtres, normalisée par le produit des moyennes quadratiques calculées pour chacune des fenêtres.
- ZNCC : Zero mean Normalized Cross Correlation, identique à NCC mais corrélation croisée entre écarts à la moyenne.

Le critère ZNCC (voir formule ci-dessous) est le plus couramment utilisé. Il s'applique aux écarts à la moyenne dans chaque fenêtre, il est donc invariant aux variations uniformes de sensibilité entre les deux caméras, et par ailleurs, il est normalisé : le critère est compris entre 0 et 1, et il vaut 1 pour deux fenêtres identiques. Il est donc plus simple de définir un seuil de corrélation pour décider si deux fenêtres sont identiques ou différentes (par exemple, le score ZNCC doit être supérieur à 0.9 pour décider de l'appariement entre deux fenêtres).

$$C(x,y,d) = \frac{\sum_{(i,j)\in F}((I_g(x+i,y+j) - \overline{I_g(x,y)}) \cdot ((I_d(x+d+i,y+j) - \overline{I_d(x+d,y)})}{\sqrt{\sum_{(i,j)\in F}[I_g(x+i,y+j) - \overline{I_g(x,y)}]^2} \cdot \sqrt{\sum_{(i,j)\in F}[I_d(x+d+i,y+j) - \overline{I_d(x+d,y)}]^2}}$$

Avec : F est la fenêtre de corrélation centrée autour du pixel (x,y)

, $I_g(x,y)$ et $I_d(x,y)$ sont les intensités des images gauche et droite, à la position du pixel (x, y)

, $\overline{I_g(x,y)}$ et $\overline{I_d(x,y)}$ sont les moyennes des intensités des images gauche et droite sur la fenêtre de corrélation F

, d est le décalage horizontal sur x du pixel dans l'image droite, c'est la disparité.

Nous rappelons que, les deux images étant rectifiées, il n'y a pas de décalage vertical sur y, C(x, y, d) est la valeur du critère au pixel (x, y) pour une disparité d.

Ces critères numériques, en particulier le critère ZNCC, sont longs à calculer. Les critères SAD ou ZSAD sont plus simples, mais, généralement, ils s'avèrent peu robustes. Notons cependant que Kurt Konolige du SRI, expert du domaine, auteur d'un comparatif assez exhaustif des méthodes en 1997 (KONOLIGE97), utilise le critère SAD dans son propre algorithme.

Pour des applications temps réel, il a été proposé par Woodfill et Zabih [Zabih_94], un critère non numérique, plus qualitatif, appelé CENSUS. D'après la comparaison effectuée au LAAS par S. Gautama [Gautama_99], ce critère est plus robuste au bruit et plus efficace que les critères paramétriques sur des images peu texturées comme on en rencontre dans le milieu médical.

Au préalable, sur les images gauche et droite à comparer, une transformation est appliquée afin de remplacer la luminance en chaque pixel, par une chaîne de bits, dépendant de la comparaison de la valeur de ce pixel avec les autres pixels de la fenêtre de corrélation.

Par exemple, pour la fenêtre présentée en figure 2-21, la valeur 67 de luminance pour le pixel central est remplacée par la chaîne montrée à droite. Les bits de cette chaîne donnent les résultats des comparaisons de cette valeur (67) avec les valeurs de luminances des pixels de la fenêtre, parcourue ligne à ligne, de gauche à droite, et de haut en bas : le bit de comparaison vaut 1 si 67 est plus grand ou égal à la valeur du pixel traité, 0 sinon.

Par exemple, le premier bit est 0 car 120 > 67, le second 0 car 96 > 67, le troisième 1 car 65 < 67 ….

120	96	65	76	130
34	90	45	78	34
78	90	67	45	67
78	100	67	158	130
67	120	95	189	145

→ **Census = 00100101010010010010000**

Figure 2-21 : Transformation Census sur un pixel avec une fenêtre 5 x 5.

Une fois que ces transformations sont faites sur les deux images, le score utilisé pour comparer la chaîne associée à un pixel m'_1 de l'image gauche, avec celles associées aux pixels de la même ligne de l'image droite, est la distance de Hamming entre chaînes de bits, c'est à dire le nombre de bits identiques entre deux chaînes, ou le nombre de bit égal à 1 dans le OU-exclusif entre deux chaînes.

Dans la réalisation pratique, différentes solutions peuvent être essayées :

- Calcul de la transformation CENSUS sur 8 bits seulement (fenêtre 3x3), mais utilisation d'un filtre de moyennage sur les images CENSUS pour prendre en compte en chaque pixel, un voisinage plus large ; elle est plus simple à programmer, car les images CENSUS restent des tableaux d'octets.
- Calcul de la transformation CENSUS sur N bits (par exemple 168 bits pour une fenêtre 13x13) et comparaison directe des images CENSUS sans filtrage préalable. Pour une implémentation logicielle, cela pose le problème de gérer des images avec des pixels codés sur 5, 6, 7…. Octets, ce qui peut s'avérer assez peu performant en terme de temps calcul.

Pour trouver le correspondant du pixel x_0 de l'image gauche, le critère est calculé pour tous les pixels candidats de l'image droite, appartenant à l'intervalle de disparité. Pour chaque association possible (x_0, x_0+d), on a donc un score de corrélation. L'ensemble des scores donne une fonction représentée en figure 2-20.b. Le score ZNCC est compris entre 0 et 1 ; il est optimal (fenêtre de corrélation identique) quand il vaut 1. Le correspondant sera donc donné par le score maximum. Plusieurs tests heuristiques permettent d'éviter les faux appariements :

- Le score maximum doit être supérieur à un seuil (ressemblance minimale exigée).
- L'écart avec le deuxième score maximal doit être significatif.
- Le pic de corrélation doit être assez étroit.

Si ces critères sont satisfaits, alors l'algorithme calcule une valeur de disparité sub-pixel, en faisant une simple interpolation parabolique entre le score maximum et les deux voisins. (cf. Figure 2-20.c). La sortie de cette étape est une image flottante de disparité qui contient pour chaque pixel de l'image gauche:

- 0 s'il n'a pas de correspondant.
- La valeur de disparité avec le correspondant sinon.

2.8.2 Approche séquentielle : algorithme général.

La figure 2-22 suivante présente l'algorithme global de la méthode de stéréovision par corrélation, appliquée à une paire d'images droite et gauche.

Hors-ligne, une phase de calibrage réalisée généralement sur des images de mire, permet d'estimer les paramètres des caméras et du banc stéréoscopique (matrice essentielle). Ces paramètres permettent d'identifier la géométrie épi-polaire, de calculer des tables de pré-rectification et de les appliquer afin de rectifier les images et de corriger leurs distorsions géométriques (ici radiales).

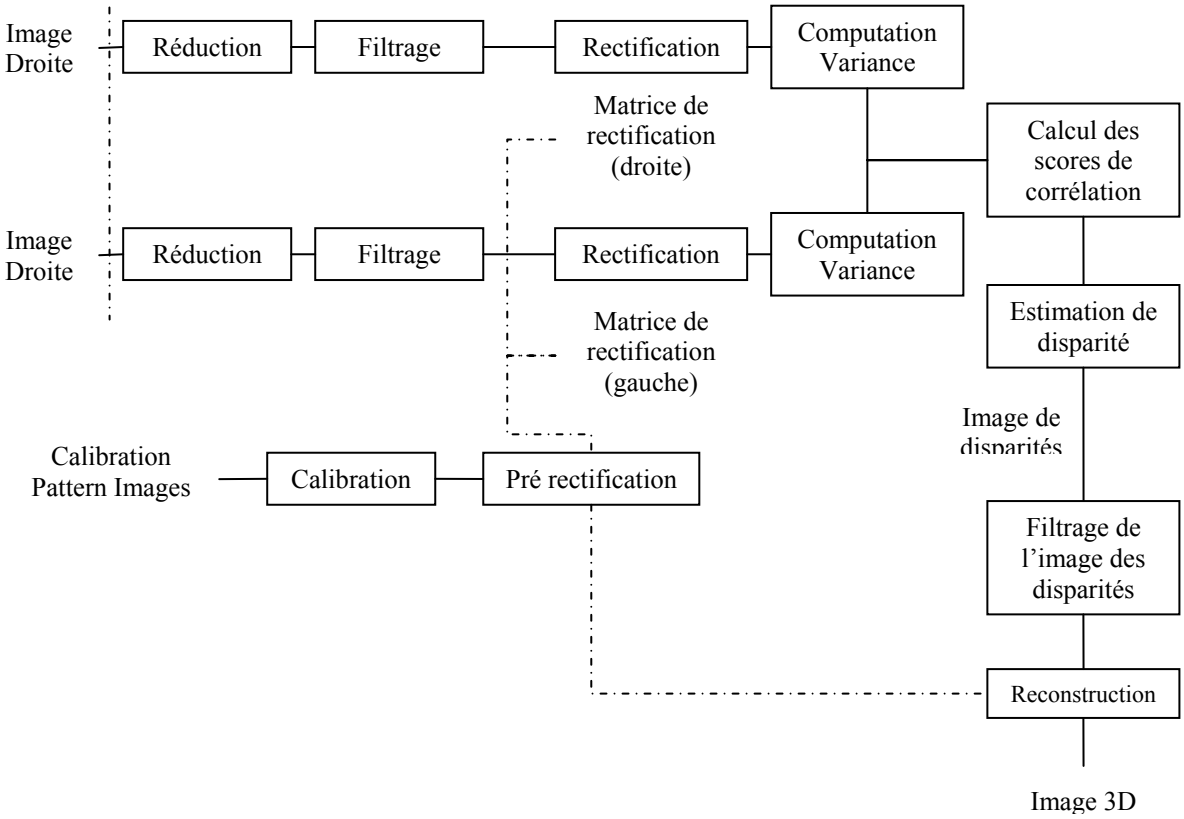


Figure 2-22 : Synoptique de la méthode de corrélation : méthode séquentielle.

En ligne, la méthode de stéréovision requiert trois phases :

- Tout d'abord, avant de rechercher des appariements entre pixels des deux images, plusieurs prétraitements sont réalisés séparément sur les deux images :
  - Réduction de la résolution par échantillonnage ou moyennage. Si l'image est en couleur, il convient d'appliquer cette réduction soit sur les pixels G (un sur deux vu la mosaïque Bayer), soit sur la luminance calculée au préalable sur tous les pixels par interpolation des pixels R, G et B.
  - Filtrage des images originales. Cette étape est optionnelle ; selon le contexte et le type d'environnement, il peut être intéressant de n'appliquer la corrélation que sur des pixels spécifiques, par exemple, les points de discontinuité de luminance.
  - Rectification et correction des distorsions. Cette phase exploite les tables de pré-rectification, de sorte qu'elle donne directement les coordonnées u_d et

v_d dans les images originales, correspondants à un pixel u et v dans les images rectifiées et corrigées.

- Pour le score ZNCC, calcul des moyennes et variances en chaque pixel des images droite et gauche, réduites, filtrées, rectifiées et corrigées. La variance permet d'une part de normaliser les scores (afin qu'ils soient compris entre 0 et 1), d'autre part de fournir un critère d'uniformité sur les fenêtres de corrélation. Donc ce pré-calcul permet :
  - d'éviter des calculs redondants.
  - d'effectuer un test préalable afin d'éviter de rechercher un correspondant pour un pixel qui est dans une zone totalement uniforme. En un pixel de l'image gauche, les scores ne seront calculés avec les pixels de l'image droite de la même ligne, que s'il existe une texture suffisante, donc si la variance en ce pixel est supérieure à un seuil donné.
- Pour le score Census, calcul de la transformée Census sur chacune de ces images. Avec la transformée, un test d'uniformité peut également être calculé pour éviter des recherches inutiles d'appariements.

Le calcul des scores de corrélation est l'étape la plus longue de la méthode. Différents traitements annexes permettent d'optimiser le temps nécessaire pour cette étape en évitant les calculs redondants.

Les traitements sur la disparité : recherche de la disparité optimale en chaque pixel de l'image gauche à partir des scores obtenus. Ce traitement est par principe local car la meilleure disparité est recherchée indépendamment pour chaque pixel. Il est complété par plusieurs vérifications à caractère global : on vérifie par exemple que la disparité optimale est identique lorsque la recherche se fait de la droite vers la gauche. Si la disparité est différente, l'appariement pour ce pixel est invalidé. Cette vérification est très rapide puisqu'elle exploite les mêmes scores de corrélation.

Le filtrage de l'image de disparité : cette image contient pour chaque pixel de l'image gauche, la valeur trouvée pour la disparité. Les pixels non appariés ont une disparité nulle. Le filtrage exploite des opérateurs morphologiques simples : il

permet d'invalider la disparité pour des pixels isolés (par exemple un pixel d'une scène qui a une disparité nettement différente de celle des pixels voisins).

La reconstruction 3D : pour chaque pixel qui a une disparité non nulle, on calcule les coordonnées 3D du point correspondant et l'imprécision associée (variances sur X, Y et Z). Ce calcul est déjà fait dans le repère de la caméra gauche rectifiée.

2.8.3 Optimisation de l'approche séquentielle : approche multi-résolution

La méthode décrite dans la section précédente peut être accélérée en exploitant un schéma de traitement multi-résolution. Le but est de réduire l'écart de disparité [D_{min}, D_{max}] en effectuant d'abord la recherche du correspondant à un niveau de résolution très faible (typiquement, sur une image 80 x 60), puis à propager les disparités pour les résolutions plus élevées.

Par exemple si l'écart de disparité est de 80 pixels :

L'algorithme séquentiel décrit dans la section précédente calculera 80 scores de corrélation à la résolution 640 x 480 puis cherchera le maximum.

L'algorithme exploitant une pyramide multi-résolution à 4 étages calculera 19 scores :

- 10 scores à la résolution 80 x 60 pour prédire une disparité
- 3 scores aux résolutions 160 x 120, 320 x 240 puis 640 x 480.

La multi-résolution s'appuie sur l'algorithme séquentiel de la corrélation "simple résolution". Le principe peut se décomposer comme suit :

- Réduction de la paire originale avec différents facteurs (cf. Figure 2-23).
- Corrélation de la paire d'images à la plus « faible » résolution.
- Corrélation successive en "remontant" la pyramide jusqu'au niveau de résolution original.

- Et éventuellement, recherche d'une résolution sub-pixel au niveau de résolution le plus élevé.

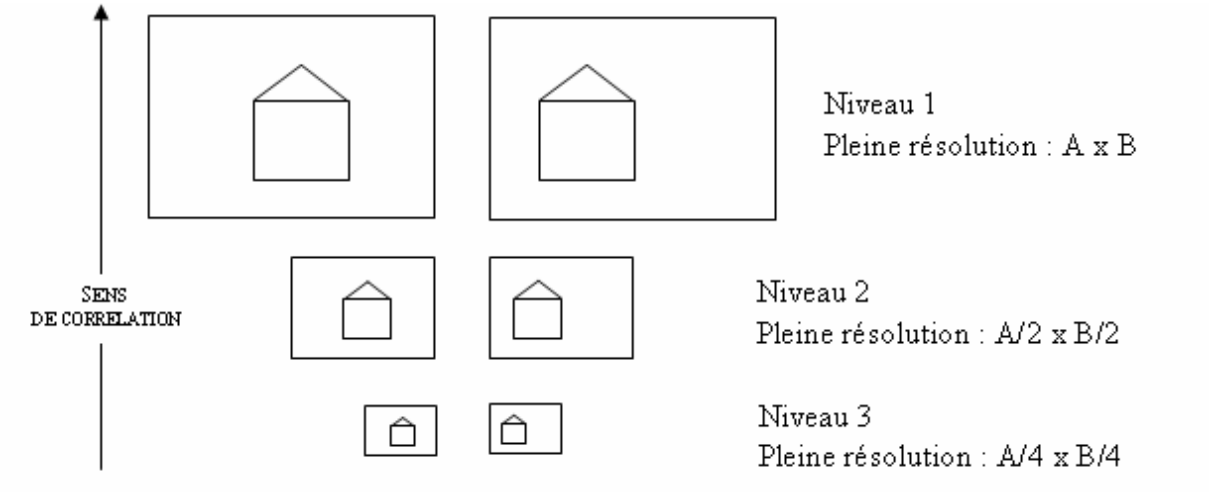


Figure 2-23 : Pyramide de résolution.

Il est possible d'effectuer les réductions avec différents algorithmes :

- Réduction par échantillonnage : en prenant 1 pixel sur n (n étant le facteur de réduction)
- Réduction par moyennage : pour chaque pixel de l'image réduite on calcule la moyenne des niveaux de gris sur les n x n pixels correspondants dans l'image initiale.

2.9 Reconstruction 3D à partir des images rectifiées

2.9.1 Calcul des coordonnées 3D d'un point

Nous donnons les équations de la reconstruction 3D, valables une fois que les images ont été rectifiées. Sur la figure 2-24, nous avons représenté les deux images alignées, un point P de la scène observée qui se projette en (u_1, v_1) sur l'image gauche et en (u_1, v_1+d) sur l'image droite. La disparité d est ici négative.

Les repères des deux caméras ont des axes X et Z parallèles, des axes Y confondus et les centres optiques éloignés d'une distance B (base du banc de

stéréovision). Après rectification, les paramètres intrinsèques des deux caméras sont identiques.

Les équations de projections dans les deux plans image sont les suivantes :

$$\frac{X}{x_1} = \frac{Y}{y_1} = \frac{Z}{F} \quad ; \quad \frac{X}{x_2} = \frac{Y+B}{y_2} = \frac{Z}{F}$$

La transformation des coordonnées métriques en coordonnées pixel plus la prise en compte de la disparité, donnent :

$$x_1 = \frac{u-u_0}{k_u} \quad ; \quad y_1 = \frac{v-v_0}{k_v}$$

$$x_2 = x_1 \quad ; \quad y_2 = \frac{v_2-v_0}{k_v} = \frac{v_1+d-v_0}{k_v}$$

En combinant les deux étapes précédentes, nous avons :

$$\frac{\alpha_u \cdot X}{u_1-u_0} = \frac{\alpha_v \cdot Y}{v-v_0} = Z$$

$$\frac{\alpha_u \cdot X}{u_1-u_0} = \frac{\alpha_v \cdot (Y+B)}{v_1+d-v_0} = Z$$

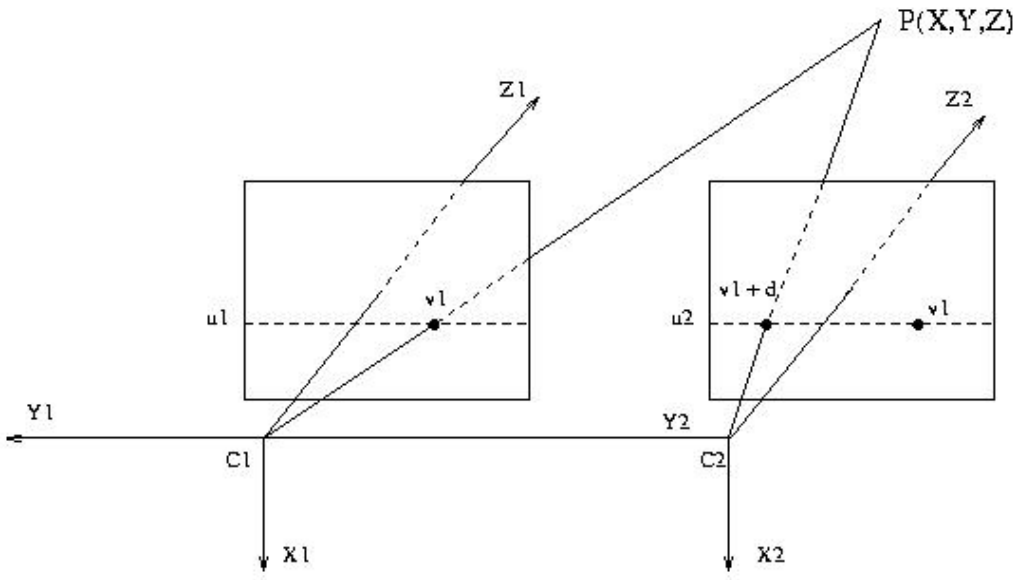


Figure 2-24 : Reconstruction à partir des images rectifiées.

Finalement, la reconstruction est obtenue par :

$$X = \frac{\alpha_v \cdot B \cdot (u_1 - u_0)}{\alpha_u \cdot d}$$

$$Y = \frac{B \cdot (v_1 - v_0)}{d}$$

$$Z = \frac{\alpha_v \cdot B}{d}$$

2.10 Évaluation des méthodes d'appariement

2.10.1 Mesures de corrélation standard

Les mesures de ressemblance que nous avons évoquées sont implémentées dans l'immense majorité des cas par des mesures de corrélation. De telles mesures intègrent les différences des intensités sur des voisinages rectangulaires (généralement carrés) des pixels considérés. Quelques mesures usuelles sont rappelées dans le tableau 2-1 ; elles mesurent la ressemblance d'un point (u_1,v_1) de l'image 1 de signal I_1 à celle d'un point (u_2,v_2) de l'image 2 de signal I_2 sur un masque carré (2n+1) x (2n+1) (une fenêtre, ou un patch).

Mesure	Expression
SAD	$\sum_{du=-n}^{du=+n} \sum_{dv=-n}^{dv=+n} \lvert I_2(u_2+du, v_2+dv) - I_1(u_1+du, v_1+dv) \rvert$ (3.1)
ZSAD	$\sum_{du=-n}^{du=+n} \sum_{dv=-n}^{dv=+n} \lvert I_2(u_2+du, v_2+dv) - \overline{I_2} - I_1(u_1+du, v_1+dv) + \overline{I_1} \rvert$ (3.2)
SSD	$\sum_{du=-n}^{du=+n} \sum_{dv=-n}^{dv=+n} (I_2(u_2+du, v_2+dv) - I_1(u_1+du, v_1+dv))^2$ (3.3)
ZSSD	$\sum_{du=-n}^{du=+n} \sum_{dv=-n}^{dv=+n} (I_2(u_2+du, v_2+dv) - \overline{I_2} - I_1(u_1+du, v_1+dv) + \overline{I_1})^2$ (3.4)

Tab 2-1 : Quatre mesures usuelles de corrélation.

Ces mesures sont assez intuitives : SAD correspond mathématiquement à la norme L1, évaluée entre les fonctions I1 et I2 en (2n + 1) x (2n + 1) points et SSD au carré de L2. Ce sont des mesures de l'énergie de la fonction de différence des intensités des masques.

ZSAD et ASSD sont centrées, par soustraction de la moyenne locale des intensités I1 et I2, sur les masques courants. Ceci permet d'annuler l'effet de changements d'intensité locaux, mais a pour inconvénient de donner de bons scores à tort à des masques très différents.

Avec ZSAD ou ZSSD, une fenêtre uniformément blanche correspond parfaitement à une fenêtre uniformément noire. Pour compenser l'effet des changements de luminosité il est plus correct de modifier globalement les intensités des images, puis d'appliquer des mesures non-centrées. Par exemple, si les pixels de l'image 1 ont une intensité de moyenne m1 et d'écart-type σ_1, et ceux de l'image 2 une intensité de moyenne m_2 et d'écart-type σ_2, alors on peut transformer les intensités I_2 de chaque pixel de l'image 2 selon la formule suivante :

$$I_2 \mapsto \frac{\sigma_1}{\sigma_2} I_2 + m_1 - m_2 \frac{\sigma_1}{\sigma_2}$$

La distribution des intensités des pixels de l'image 2 transformée aura ainsi la même moyenne m_1 et le même écart-type σ_1 que dans l'image 1.

2.10.2 Mesures de corrélation robustes

Z.D. Lan répertorie dans sa thèse [Lan_97] de nombreuses mesures de corrélation robustes aux occultations, dont celles de R. Zabih, et de D.N. Bhat. Il apporte également ses propres innovations.

Mesures de R. Zabih :

R. Zabih propose dans [Zabih_94] deux mesures de ressemblance non paramétriques. Ces mesures agissent sur des images transformées. La transformation Rank transforme chaque pixel en une valeur indiquant le nombre de pixels de son

voisinage 3 x 3 qui lui sont inférieurs. La transformation Census transforme chaque pixel en une chaîne de bits indiquant quels sont les pixels du voisinage qui lui sont inférieurs. Si pour construire la chaîne de bits, on considère ces 8 pixels dans l'ordre suivant : | 8 | 1 | 2 | / | 7 | - | 3 | / | 6 | 5 | 4 |, alors le masque | 84 | 119 | 157 | / | 85 | 232 | 226 | / | 246 | 250 | 175 | sera transformé en 11110011, car 119 < 232, 157 < 232, 226 < 232, 175 < 232, 250 ≥ 232, 246 ≥ 232, 85 < 232 et 84 < 232. La transformation Rank donne la valeur 6, car 6 bits sont à 1. Un exemple est donné en figure2-25.

Image originale en niveau de gris.

84	119	157	94	222
85	232	226	250	250
246	250	175	250	57
6	17	67	88	51
72	170	147	253	10

Image transformée par *rank*.

-	-	-	-	-
-	6	4	6	-
-	8	3	6	-
-	1	1	4	-
-	-	-	-	-

Image transformée par *census*.

--------	--------	--------	--------	--------
--------	11110011	11001001	11010111	--------
--------	11111111	00011100	00111111	--------
--------	00000010	00000010	01110010	--------
--------	--------	--------	--------	--------

Figure 2-25 : Exemples de transformations de R. Zabih.

La mesure de distance sur ces images transformées est une mesure SAD (dans le cas d'une transformation Rank), ou une distance de Hamming (dans le cas d'une transformation Census).

Il est clair que de telles mesures sont robustes à des occultations : si un pixel est occulté dans un masque, il y a une probabilité non-nulle pour que le masque transformé ne change absolument pas ; il suffit que la nouvelle valeur du pixel soit du même ordre que l'ancienne valeur (par rapport au pixel central). De plus, même si le masque transformé change, ce changement est minime vis-à-vis de la distance utilisée : les distances SAD ou de Hamming ne changeront au plus que de 1 unité. R. Zabih a réalisé un test sur deux images synthétiques, représentant 3 plans de

disparité fixe, vus de face (pas de déformation perspective). Il compare les cartes de disparité obtenues avec les mesures SSD, Rank + SAD, et Census + Hamming. Pour un point de l'image 1, on choisit comme appariement dans l'image 2 celui qui donne la distance minimale, sans autre vérification. Les contours obtenus sont bien mieux délimités dans le cas des mesures robustes. Le nombre de faux appariements est aussi considérablement réduit (tableau 2-2).

Mesure	Nombre de faux appariements
SSD	1385
rank + SAD	609
census + Hamming	407

Tab 2-2 : Tests comparatifs des mesures de R. Zabih, selon [Zabih_94].

Mesure de D.N. Bhat :

D.N. Bhat décrit une nouvelle transformation dans [Bhat_96]. Elle consiste à numéroter les 9 pixels d'un masque 3 x 3 dans l'ordre croissant. Si on numérote ces 9 pixels dans l'ordre suivant : | 1 | 2 | 3 | / | 4 | 5 | 6 | / | 7 | 8 | 9 |, alors le masque | 84 | 119 | 157 | / | 85 | 232 | 226 | / | 246 | 250 | 175 | sera transformé en 142396578, car les valeurs apparaissent dans cet ordre : (84; 85; 119; 157; 175; 226; 232; 246; 250). Ainsi, sur la même image brute qu'en figure 2-25, l'image transformée est celle donnée en figure 2-26.

---------	---------	---------	---------	---------
---------	142396578	312854679	921734568	---------
---------	789163245	789521346	967841235	---------
---------	456798312	456872139	963457128	---------
---------	---------	---------	---------	---------

Figure 2-26 : Transformation de D.N. Bhat.

Sur les images transformées, deux masques en correspondance doivent normalement présenter des pixels dans le même ordre. C'est toujours le cas si les variations des intensités des pixels sont suffisamment faibles d'une image à l'autre. En revanche, si les changements sont plus importants, l'ordre peut être perturbé ; cette perturbation doit être faible pour des masques en correspondance. La mesure

de ressemblance utilisée est optimale si la permutation observée est minimale (c.-à-d. proche de l'identité).

D.N. Bhat revendique des résultats meilleurs que ceux de R. Zabih : de 25 % à 50 % de faux appariements en moins.

Résultats :

En termes d'appariement RZSSD se révèle plus performante que les mesures Census ou Rank sur les régions occultées. Les faux appariements sont alors moins nombreux de 5 % à 20 % par rapport à ces deux méthodes.

Sur les régions non occultées, la meilleure mesure est le Census ou même simplement ZSSD. Aussi, l'utilisation recommandée est de réaliser une première phase d'appariement à l'aide de mesures standards (non robustes), de détecter sur cette base les régions possibles d'occultation et d'effectuer une seconde passe d'appariement robuste dans ces régions seulement [Blanc_98].

2.11 Conclusion

Après avoir brièvement passé en revue quelques méthodes adaptées à la vision 3D temps réel, nous avons présenté plus en détail la méthode de stéréovision passive dense (autant de points 3D que de pixels dans les zones perçues par les deux caméras). En théorie cette méthode permet d'obtenir une image 3D dense ; en pratique la densité dépendra de la texture présente dans les images. Cette texture dépend elle-même de la qualité des caméras et des objectifs, des conditions d'acquisition (notamment influence de l'effet de flou dû au mouvement des caméras) et de la scène. Cette méthode, basée sur la vision stéréoscopique humaine, nécessite l'extraction, dans deux images droite et gauche d'une même scène, de points stéréo-correspondants afin de construire l'image de disparité. Dans cet objectif nous avons présenté différents procédés d'appariement des pixels et avons retenu la méthode du Census pour sa robustesse et la relative simplicité de son algorithme. Nous avons également pris en considération la problématique de la calibration des caméras et la rectification des images sans lesquelles il est impossible d'obtenir une carte de

disparité convenable indispensable à toute reconstruction 3D. Dans le chapitre suivant nous nous intéresserons aux différentes technologies des capteurs d'images, aux interfaces vidéo disponibles sur le marché, aux plateformes de développement, à leurs avantages et inconvénients respectifs afin de concevoir un banc d'évaluation.

2.12Références bibliographiques

[Ayache_89] N. Ayache, 'Vision Stéréoscopique et Perception Multi sensorielle, Application à la robotique Mobile', Inter Edition Science Informatique, 1989.

[Ballard_82] D. Ballard et C. Brown, Computer Vision, Prentice-Hall, 1982.

[Besançon_89] J. Besançon, 'Vision par ordinateur en deux et trois dimensions', Eyrolles, 1989.

[Besl_88] P. Besl, 'Active optical range imaging sensors and applications', Machine Vision and Applications, pp. 1:127–152, 1988.

[Bhat_96] D.N. Bhat et S.K. Nayar, 'Ordinal measure for visual correspondence', Conference on Computer Vision and Pattern Recognition, pp. 351-357, San Francisco, California, USA, Juin 1996.

[Blanc_98] J. Blanc, 'Synthèse de nouvelles vues d'une scène 3d à partir d'images existantes', Thèse de doctorat, Institut National Polytechnique de Grenoble, Jan 1998.

[Boizard_85] 'Etude et Réalisation d'un Capteur de Vision Tridimensionnelle mettant en œuvre un photo détecteur CCD matriciel', Thèse de doctorat, Université Paul Sabatier, Juin 1985.

[Champleboux_91] G. Champleboux, 'Utilisation de fonctions splines pour la mise au point d'un capteur tridimensionnel sans contact', Thèse de doctorat à l'Université Joseph Fourier, Grenoble 1, 1991.

[Faugeras_86] O.D. Faugeras et G. Toscani, 'The Calibration Problem for Stereo', CVPR, 1986.

[Faugeras_87] O.D. Faugeras et G. Toscani, 'Camera Calibration for 3D Computer Vision', International Workshop on Machine Vision and Machine Intelligence, 1987.

[Gautama_99] S. Gautama, S. Lacroix et M. Devy, 'Evaluation of stereo matching algorithms for occupant detection', International Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems (RATFG-RTS 99), Corfu, Grèce, 1999.

[Horaud_95] R. Horaud et O. Monga, 'Vision par ordinateur: outils fondamentaux', HERMES, 1995.

[Konolige_97] K. Konolige, 'Small Vision Systems: Hardware and Implementation', 8th International Symposium on Robotics Research (ISRR), Hayama, Japan, Octobre 1997.

[Lan_97] Z.D. Lan, 'Méthodes robustes en vision : application aux appariements visuels', Thèse de doctorat, Institut National Polytechnique de Grenoble, Mai 1997.

[Lasserre_95] P. Lasserre et P. Grandjean, 'Stereo Vision Improvements', IEEE International Conference on Advanced Robotics, Barcelone, 1995.

[Lewis_77] R. Lewis et A. Johnson, 'A scanning laser range-finder for a robotic vehicle, International Joint Conference on artificial Intelligence, pp. 762–768, 1977.

[Li_95] M. Li et J.M. Lavest, 'Some aspects of zoom-lens camera calibration', Technical Report, Computational Vision and Active Perception Laboratory Royal Institute of Technology, Stockholm, Sweden, Number ISRN KTH/NA/P-95/03-SE, February 1995.

[Luong_92] Q. Luong, O. Faugeras, et S. Maybank, 'Camera self-calibration: Theory and experiments', Second European Conference on Computer Vision (ECCV'92), pp. 321–334, Springer Verlag, Berlin, 1992.

[Puget_90] P. Puget et T. Skordas, 'An Optimal Solution for Mobile Camera Calibration', 1st European Conference on Computer Vision, 1990.

[Toscani_87] G. Toscani, 'Système de calibration et perception du mouvement en vision artificielle', Thèse de Doctorat, Paris-Sud, 1987.

[Tsai_87] R.Y. Tsai, 'A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Self TV Cameras and Lenses', IEEE Journal of Robotics and Automation, vol. 3, No 3, 1987.

[Zabih_94] R. Zabih et J. Woodfill, 'Non-parametric local transforms for computing visual Correspondence', 3rd European Conference on Computer Vision, pp. 151-158, Stockholm, Sweden, Mai 1994.

Chapitre3

Conception d'un banc d'évaluation

---

3 CONCEPTION D'UN BANC D'EVALUATION

3.1 Introduction

Il nous a semblé pertinent, avant d'aborder la phase du portage de la stéréovision dans un environnement temps réel, de disposer d'un banc d'évaluation ergonomique utilisé en temps différé et à partir duquel il serait possible :

- de configurer et régler les deux caméras (ouverture du diaphragme, grossissement, netteté).
- d'enregistrer et d'archiver les images obtenues sous des formats divers pour des traitements ultérieurs (initialisation des mémoires du FPGA par exemple).
- d'évaluer rapidement sur différents types d'images les performances qualitatives et quantitatives des algorithmes de traitement.
- d'évaluer l'influence des paramètres apparaissant dans le calcul des images de disparité (format des données, taille des fenêtres de moyennage, de Census …).
- de tester différentes méthodes de rectification des images primaires et de filtrage de l'image de disparité.
- D'évaluer l'influence de la précision pour le codage des données

3.2 Le banc d'évaluation

3.2.1 Aspects fonctionnels et rôle

Ce banc a pour objectif de :

- visualiser en temps réel les images reçues depuis les caméras sur l'écran de l'ordinateur afin de procéder aux différents réglages.
- Sauvegarder les images pour les traiter en temps différé afin de calibrer les caméras et calculer les paramètres de position.
- Synchroniser les caméras droite et gauche.

- Evaluer l'influence des paramètres associés aux algorithmes de traitement

Il est aussi connecté à la caméra infrarouge développée au LAAS via une carte interface parallèle au format PCI. Les caractéristiques essentielles du banc en terme d'entrées-sorties sont résumées sur la figure 3-1.

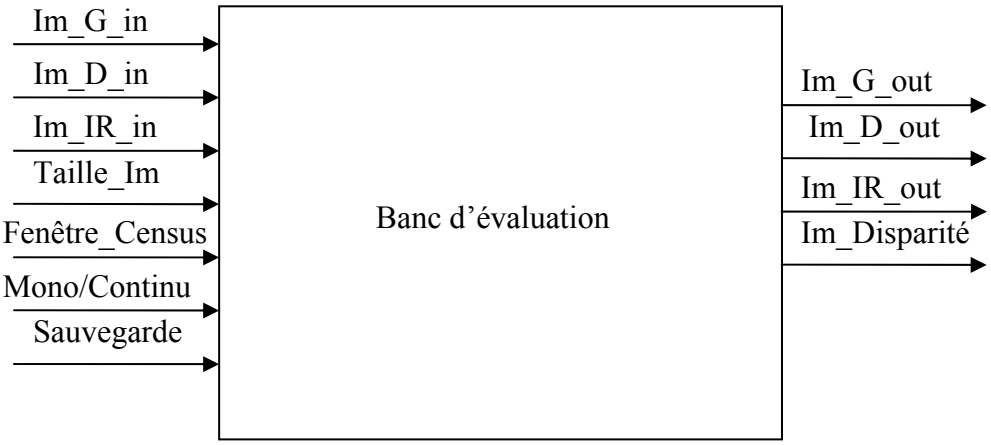


Figure 3-1 : Entrées-Sorties du banc d'évaluation.

Entrées :

- Im_G_in, Im_D_in : flux des images gauche et droite au format Camera Link.
- Im_IR_in : flux de l'image infrarouge au format 16 bits parallèles.
- Taille_Im : définit la taille de l'image de travail en nombre de pixels ligne/colonne (ex : 640x480).
- Fenêtre_Census : définit la taille de la fenêtre de calcul pour le Census (ex : 3x3, 5x5, …).
- Mono/Continu : acquisition d'une image et affichage ou acquisition et affichage en temps réel.
- Sauvegarde : enregistrement des images d'origine, des images moyennées, des images Census ou des images de disparité dans différents formats pour la partie visible.

Sorties :

- Im_G_out, Im_D_out, Im_IR_out, Im_disparité : affichage/sauvegarde des différentes images.

Les paragraphes qui suivent donnent les principaux points qui ont contribué au choix des éléments constitutifs du banc d'évaluation : plateforme de développement, choix des caméras et interfaces pour la vidéo numérique.

3.2.2 La plateforme de développement

Nous avons conservé l'environnement de développement LabWindows CVI de National Instruments précédemment utilisé dans le développement de la caméra infrarouge du LAAS-CNRS pour les raisons suivantes :

- Disponibilité de nombreuses cartes d'acquisition vidéo analogiques et numériques et leurs pilotes dans les standards les plus répandus et à des coûts raisonnables.
- Environnement de développement facile à mettre en œuvre disposant de boîtes à outils et fonctions de traitement d'images intégrées.
- Gestion aisée de différentes interfaces de communication si nécessaire (USB2, Ethernet, CAN, …) pour les évolutions futures.
- Programmation en langage C ou C++.
- Réutilisation de programmes en C développés dans le cadre du groupe de recherche RIA notamment pour la phase de calibration des caméras.

3.2.2.1 Caractéristiques de LabWindows/CVI

LabWindows/CVI de National Instruments est un environnement de développement consacré à l'origine au monde du test et de la mesure et qui a fait ses preuves dans le domaine industriel et de la recherche car il optimise l'acquisition, l'analyse et la présentation de données en ANSI C. Il permet de construire des systèmes d'instrumentation associant des applications et des cartes d'acquisition de données (DAQ) ainsi que du matériel PXI, GPIB, VXI, RS-232, d'origines diverses. LabWindows/CVI associe une approche du développement interactive et facile d'utilisation à la souplesse et à la puissance de la programmation ANSI C. Les bibliothèques et les outils de développement sont spécialement conçus pour les

développeurs de systèmes de tests automatisés, de bancs d'essai, d'acquisition de données et de contrôle/commande de processus [NI_98].

L'environnement se caractérise principalement par :

- des assistants de configuration matérielle générateurs de code et une interface aux matériels d'acquisition de données.
- une bibliothèque d'interface utilisateur étendue.
- des capacités de débogage puissantes.
- divers modes de représentation de graphes.
- des bibliothèques de mesure intégrées et une bibliothèque de fonctions structurée.
- Une pérennité des développements.

Par ailleurs il est possible de disposer d'un assistant d'Entrées/Sorties d'instruments permettant la génération de code de façon automatique pour le prototypage rapide de systèmes de contrôle d'instruments. Cet environnement se donc parfaitement adapté à nos besoins.

3.2.3 Les caméras CMOS et CCD

Nous décrirons ici simplement le principe de base des deux types de capteurs d'images qui équipent les caméras vidéo et donnerons leurs avantages et inconvénients respectifs.

3.2.3.1 Les capteurs CCD (Charge Coupled Devices)

Un capteur à couplage de charges se compose d'une série de capacités positionnées en réseau sur la surface isolée d'un semi-conducteur. Les puits de potentiel brutalement créés sous les électrodes emmagasinent des charges de porteurs minoritaires qu'il est possible de déplacer d'un condensateur au suivant en appliquant aux électrodes des tensions adéquates. Bien qu'utilisés initialement pour constituer des lignes à retard analogiques, c'est la détection d'image qui a été le point de départ de leur évolution rapide. Dans ce cas les informations ont été introduites dans le capteur par des moyens optiques, en exploitant les propriétés

photoélectriques du silicium. Les photons incidents créent des porteurs minoritaires qui sont intégrés dans les puits de potentiel sous les électrodes de transfert [Sze_81]. La quantité de porteurs ainsi accumulée est proportionnelle à l'intensité du flux lumineux incident et à la durée d'intégration (figure 3-2.a). Un mécanisme de transfert des charges accumulées, réalisé par un couplage des électrodes qui les maintient en phase, permet ensuite d'évacuer ces charges du premier puits vers le second, puis du second vers le troisième et ainsi de suite jusqu'à l'étage de sortie (figure 3-2.b), où elles sont converties en différence de potentiel.

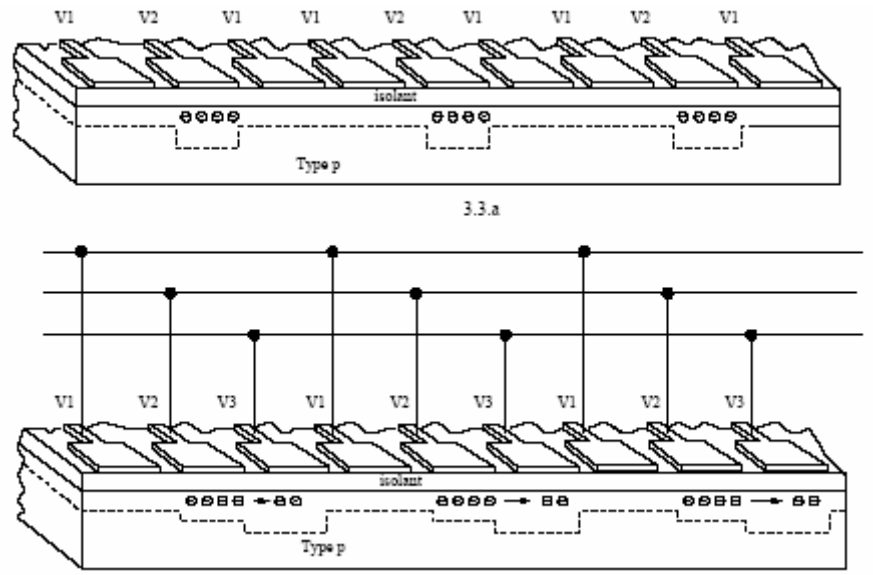


Figure 3-2 : Intégration (a) et transfert des charges (b).

On peut remarquer qu'un des inconvénients de ce dispositif est la lecture systématique de tous les points constituant le capteur d'images, ce qui peut être génant dans des applications de robotique mobile par exemple. Par ailleurs les signaux de commande de ces dispositifs, d'apparence numérique, sont en fait analogiques et polyphasés car de fortes contraintes existent sur leurs niveaux de tension, leur timing ainsi que sur leurs fronts de montée et descente pour assurer un fonctionnement optimal du circuit lors des phases d'intégration et de lecture [Sequin_75], [Nicollin_82]. Le capteur CCD reste cependant le meilleur choix lorsque des images de très bonne qualité sont recherchées ou lorsque les conditions d'éclairement sont faibles.

3.2.3.2 Les capteurs CMOS (Complementary Metal Oxide Semiconductors)

Depuis l'accroissement des parts de marché de la photo numérique grand public, ce type de capteur tend à remplacer petit à petit les capteurs CCD. Nous ne présenterons ici que des capteurs APS (Active Pixel Sensors) CMOS (figure 3-3.b) qui sont une amélioration du capteur CMOS (figure 3-3.a). Ceux-ci sont constitués d'une matrice de photodiodes ou de condensateurs MOS photosensibles constituant le capteur d'images et à chacun desquels est associé un amplificateur (d'où le terme « Active »). Un dispositif de multiplexage analogique permet d'aiguiller vers l'étage de sortie le signal correspondant à chaque photo-élément. Un des avantages de ces capteurs est qu'il est possible d'accéder à différents blocs ou points de la matrice image de manière aléatoire.

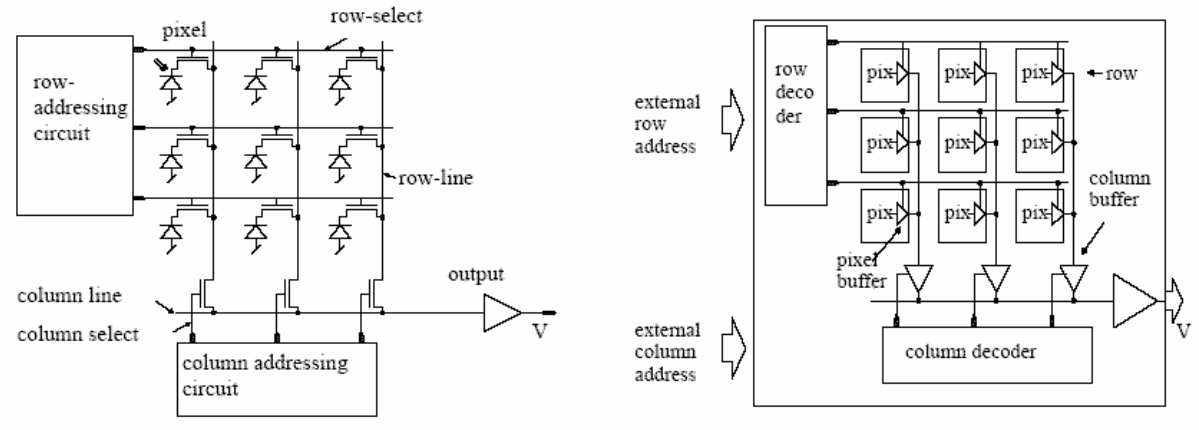


Figure 3-3 : Le capteur CMOS (a) et APS-CMOS (b).

D'autre part, contrairement au CCD qui nécessite un procédé de fabrication compliqué, la fabrication de ces capteurs utilise le même procédé que pour les puces d'ordinateur ce qui se traduit par une baisse spectaculaire des coûts de revient. Le coût de fabrication d'une plaquette CMOS est le tiers de celui d'une plaquette équivalente avec des dispositifs à couplage de charge. La technologie CMOS permet également l'intégration d'opérateurs analogiques (amplificateurs) ou numériques (adressage, fonctions avancées de traitement d'image, …) sur la même puce de semi-conducteur.

3.2.3.3 Bilan

Les avantages de la technologie CMOS par rapport au CCD sont les suivants :

- Coûts de fabrication moins importants : l'implantation des composants CMOS est aujourd'hui bien répandue et maîtrisée.
- Faible consommation d'énergie : l'architecture des capteurs permet une consommation 100 fois plus faible que pour les CCD (composants essentiellement capacitifs qui consomment pour le transfert des charges).
- Accès à des régions de pixels facilitant les approches d'appariements de pixel en mode multi-résolution.
- Plus grande rapidité pour obtenir l'image.
- Intégration sur le capteur de fonctions de haut niveau (générateurs de timing, CAN, traitements d'images…) qui rendent des services considérables en vision artificielle.
- Miniaturisation.
- Résolution d'image potentiellement très élevée

Néanmoins, leur gros inconvénient est d'avoir un facteur de remplissage d'emblée inférieur à celui des CCD. Il en résulte qu'ils sont moins adaptés pour les applications en faible lumière ou lorsque une grande qualité d'image est recherchée.

Dans le projet PICASSO, compte tenu des contraintes temps réel que nous nous étions fixées pour la partie détection d'obstacles, notre choix s'est porté sur l'utilisation de caméras à base de capteurs CMOS. Pour la partie stéréo-endoscopie, où les vitesses de traitement sont un peu moins critiques, nous avons privilégié la qualité des images et avons opté pour des capteurs CCD.

3.2.4 Les interfaces pour la vidéo numérique

Dans ce chapitre nous présentons les standards de communication les plus utilisés en vidéo numérique puis les choix que nous avons effectués en fonction de leurs avantages et inconvénients respectifs.

3.2.4.1 L'interface USB

L'interface USB (Universal Serial Bus : Bus série universel) est, comme son nom l'indique, basé sur une architecture de type série [Informatique_enc]. Il s'agit toutefois d'une interface entrée-sortie beaucoup plus rapide que les ports série standards. L'architecture qui a été retenue pour ce type de port est en série pour deux raisons principales :

- l'architecture série permet d'utiliser une cadence d'horloge beaucoup plus élevée qu'une interface parallèle (dans une architecture à haut débit, les bits circulant sur chaque fil arrivent avec des décalages, provoquant des erreurs).
- les câbles série coûtent beaucoup moins cher que les câbles parallèles.
- Les connecteurs sont plus simples et plus compacts.

Le standard USB 1.0 propose deux modes de communication :

- 12 Mb/s en mode haute vitesse.
- 1.5 Mb/s à basse vitesse.

La norme USB 2.0 permet elle d'obtenir des débits pouvant atteindre 480 Mbit/s.

Fonctionnement du bus :

L'architecture USB a pour caractéristique de fournir l'alimentation électrique aux périphériques qu'elle relie, dans la limite de 15 W maximum par périphérique. Elle utilise pour cela un câble composé de quatre fils (la masse GND, l'alimentation VBUS et deux fils de données appelés D- et D+) (figure 3-4).

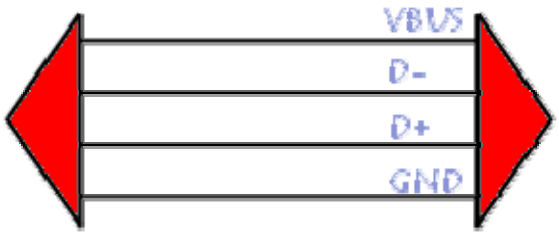


Figure 3-4 : le câble USB.

La norme USB permet le chaînage des périphériques, en utilisant une topologie en bus ou en étoile. Les périphériques peuvent alors être soit connectés les uns à la suite des autres, soit ramifiés.

Les ports USB supportent le Hot plug and play. Ainsi, les périphériques peuvent être branchés sans éteindre l'ordinateur (branchement à chaud, en anglais hot plug). Lors de la connexion du périphérique à l'hôte, ce dernier détecte l'ajout du nouvel élément grâce au changement de la tension entre les fils D+ et D-. A ce moment, l'ordinateur envoie un signal d'initialisation au périphérique pendant 10 ms, puis lui fournit du courant grâce aux fils GND et VBUS (jusqu'à 100mA). Le périphérique est alors alimenté en courant électrique et récupère temporairement l'adresse par défaut (l'adresse 0). L'étape suivante consiste à lui fournir son adresse définitive (c'est la procédure d'énumération). Pour cela, l'ordinateur interroge les périphériques déjà branchés pour connaître la leur et en attribue une nouvelle au nouveau périphérique, qui en retour s'identifie. L'hôte, disposant de toutes les caractéristiques nécessaires est alors en mesure de charger le pilote approprié. Ce type de bus, bien que simple dans son utilisation, est assez complexe à mettre en œuvre pour les développeurs.

3.2.4.2 L'interface Firewire (IEEE 1394)

Le bus IEEE 1394 a été mis au point à la fin de l'année 1995 afin de fournir un système d'interconnexion permettant de faire circuler des données à haute vitesse en temps réel. La société Apple lui a donné le nom commercial « Firewire » [Informatique_enc]. Sony lui a également donné le nom commercial de (i. Link), tandis que Texas Instrument lui a préféré le nom de Lynx. Il s'agit d'un port, équipant certains ordinateurs, permettant de connecter des périphériques (notamment des caméras numériques) à très haut débit. Il existe ainsi des cartes d'extension (généralement au format PCI ou Carte PC/PCMCIA) permettant de doter un ordinateur de connecteurs Firewire.

Les normes Firewire :

Il existe différentes normes Firewire permettant d'obtenir les débits suivants :

Norme	Débit théorique
IEEE 1394a	
IEEE 1394a-S100	100 Mbit/s
IEEE 1394a-S200	200 Mbit/s
IEEE 1394a-S400	400 Mbit/s
IEEE 1394b	
IEEE 1394b-S800	800 Mbit/s
IEEE 1394b-S1200	1200 Mbit/s
IEEE 1394b-S1600	1600 Mbit/s
IEEE 1394b-S3200	3200 Mbit/s

Tab 3-1 : les normes Firewire.

La norme IEEE 1394b est également appelée Firewire 2 ou Firewire Gigabit.

Fonctionnement du bus Firewire :

Le bus Firewire ou IEEE 1394 suit à peu près la même structure que le bus USB, si ce n'est qu'il utilise un câble composé de six fils (deux paires pour les données et pour l'horloge, et deux fils pour l'alimentation électrique) lui permettant d'obtenir un débit de 800 Mb/s (il devrait atteindre prochainement 1.6 Gb/s, voire 3.2 Gb/s à plus long terme). Ainsi, les deux fils dédiés à une horloge montrent la différence majeure qui existe entre le bus USB et le bus IEEE 1394, c'est-à-dire la possibilité de fonctionner selon deux modes de transfert :

- le mode de transfert asynchrone : Le mode de transfert asynchrone est basé sur une transmission de paquets à intervalles de temps variables. Cela signifie que l'hôte envoie un paquet de données et attend de recevoir un accusé de réception du périphérique. Si l'hôte reçoit un accusé de réception, il envoie le paquet de données suivant, sinon le paquet est à nouveau réexpédié au bout d'un temps d'attente.
- le mode isochrone : Le mode de transfert isochrone permet l'envoi de paquets de données de taille fixe à intervalles de temps réguliers. Un nœud, appelé Cycle Master est chargé d'envoyer un paquet de synchronisation

(appelé Cycle Start packet) toutes les 125 microsecondes. De cette façon aucun accusé de réception n'est nécessaire, ce qui permet de garantir un débit fixe. De plus, étant donné qu'aucun accusé de réception n'est nécessaire, l'adressage des périphériques est simplifié et la bande passante économisée permet de gagner en vitesse de transfert.

En terme de mise en œuvre, côté développeurs, le bus IEEE 1394 présente les mêmes inconvénients que l'USB et bien qu'il existe des circuits d'interface spécialisés, il nécessite l'utilisation d'un processeur pour la configuration et la gestion des transferts de données.

3.2.4.3 L'interface Camera Link

Ce nouveau type d'interface, destiné à la vision industrielle, a vu le jour en Octobre 2000. Mise au point par un consortium de fabricants de caméras et de cartes d'acquisition d'images industriels/scientifiques, dirigé par PULNix American, Inc., la norme Camera Link™ permet, dans sa configuration de base, une transmission des données sur 5 paires différentielles LVDS (4 data + 1 horloge) ainsi qu'un pilotage de la caméra et des communications série asynchrones (4 signaux + 2 lignes séries) avec le même câble.

Cette interface est basée sur la technologie « Channel Link » développée par National Semiconductors » qui consiste en un multiplexage 7 => 1 de 28 bits.

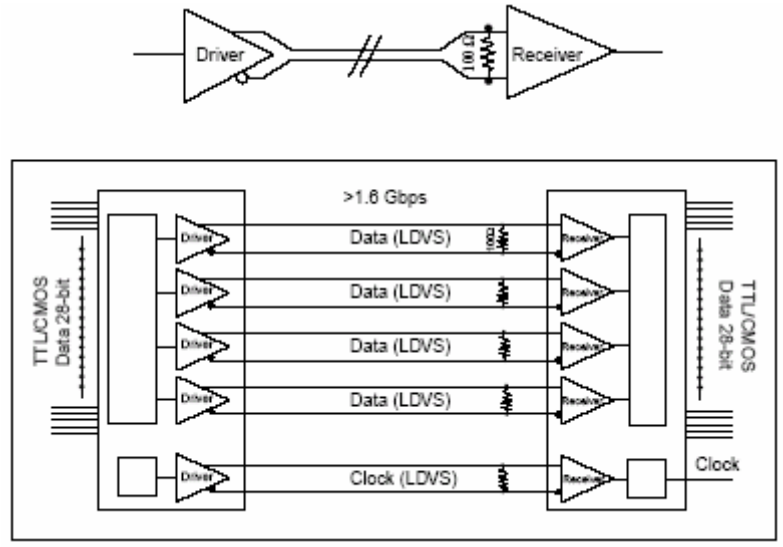


Figure 3-5 : technologie Channel Link.

Elle offre de nombreuses possibilités selon le nombre de composants « Channel Link » utilisés (configurations Base, Medium et Full) qui permettent d'utiliser des caméras monochromes ou couleurs, haute ou basse résolution sans dégrader la vitesse de transfert des données. La configuration dite de base qui prédomine le marché offre un débit de 2,3 Gbits/s et on peut théoriquement atteindre 4,76 Gbits/s dans la configuration Medium et 7,14 Gbits/s dans la configuration Full.

Comparativement aux interfaces précédentes, la connectique et les câbles de l'interface Camera Link sont plus coûteux et plus encombrants mais elle offre l'avantage d'être « transparente » pour le développeur : elle est purement matérielle et ne nécessite pas la mise en œuvre de microcontrôleurs pour son initialisation et sa gestion. C'est la raison pour laquelle nous l'avons retenue pour la première phase du projet PICASSO.

3.2.5 Constitution du banc

3.2.5.1 Choix des caméras

Ce sont des caméras référencées CV-A33 de la société JAÏ [JAI_v1]. Les principales caractéristiques sont :

- caméra monochrome à balayage progressif.
- capteur CMOS de résolution 659 x 493 pixels utiles.
- sortie vidéo numérisée sur 8 ou 10 bits au standard Camera Link.
- programmation par ligne série ou interface Camera Link de la durée d'exposition, mode de déclenchement, taille des fenêtres de scrutation et nombre d'images par seconde.
- synchronisation par trigger externe.
- fréquence pixel de 40 MHZ fixe.

3.2.5.2 Architecture générale retenue

L'architecture du banc est représentée sur la figure 3-6. Celui-ci se compose de :

- deux caméras JAÏ CV-A33 équipées au choix d'objectifs à focales variables ou fixes.
- deux cartes interfaces Camera Link au standard PCI réf : PCI 1428 de National Instruments.
- une caméra infrarouge à base d'une matrice de micro bolomètres.
- une carte interface parallèle à haut débit réf : PCI-HS32.
- un PC fonctionnant sous Windows XP Pro.
- l'environnement de développement LabWindows/CVI version 7 de National Instruments ainsi que les pilotes des différentes cartes.

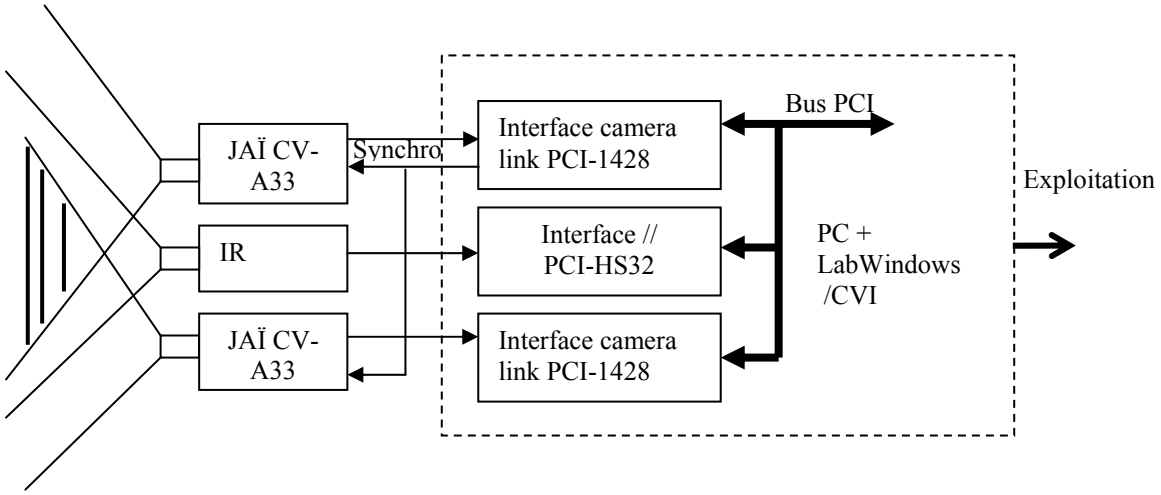


Figure 3-6 : Synoptique du banc d'évaluation PICASSO.

Notons que ce banc peut accueillir également des caméras vidéo au format Firewire, analogique (RS-170 ou S-VHS) ou parallèle via la carte interface adéquate disponible sur le marché.

Figure 3-7 : Le banc d'évaluation PICASSO.

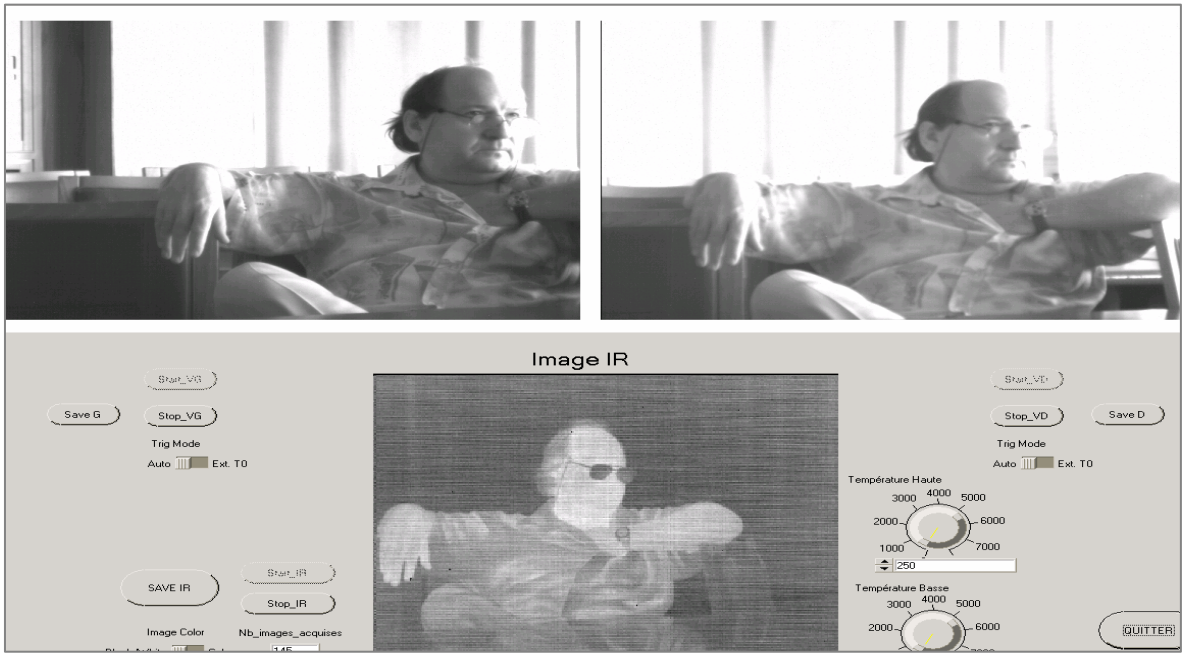


Figure 3-8 : exemple d'images visible et infrarouge.

3.3 L'implantation des algorithmes de stéréovision

L'algorigramme de la figure 3-9 représente l'enchaînement des fonctions pour l'obtention de la carte de disparité.

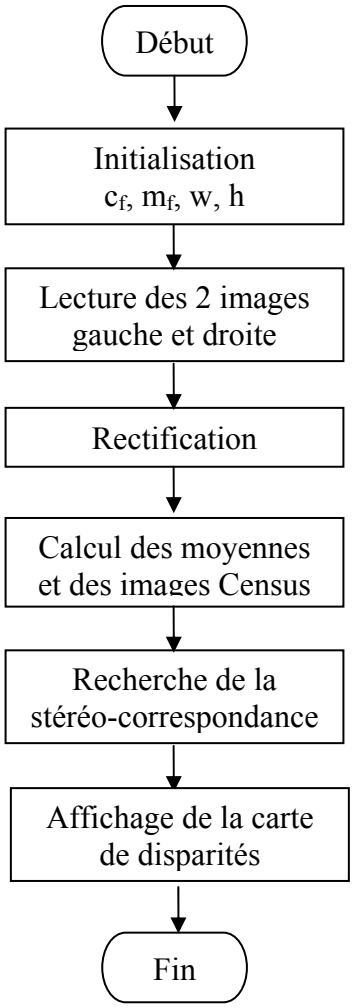


Figure 3-9 : l'algorithme de la stéréovision (corrélation census).

Le calcul des moyennes comprend actuellement deux étapes : le moyennage-ligne et le moyennage-colonne.

Le moyennage-ligne s'applique en scannant les lignes de l'image d'entrée (la gauche et la droite) pour faire une somme dans une fenêtre glissante de trois pixels successifs, comme montré dans l'algorithme de la figure 3-10.

Dans cette étape les deux boucles (i et j) scannent les images d'entrée et la boucle (k) une chaîne de trois pixels successifs.

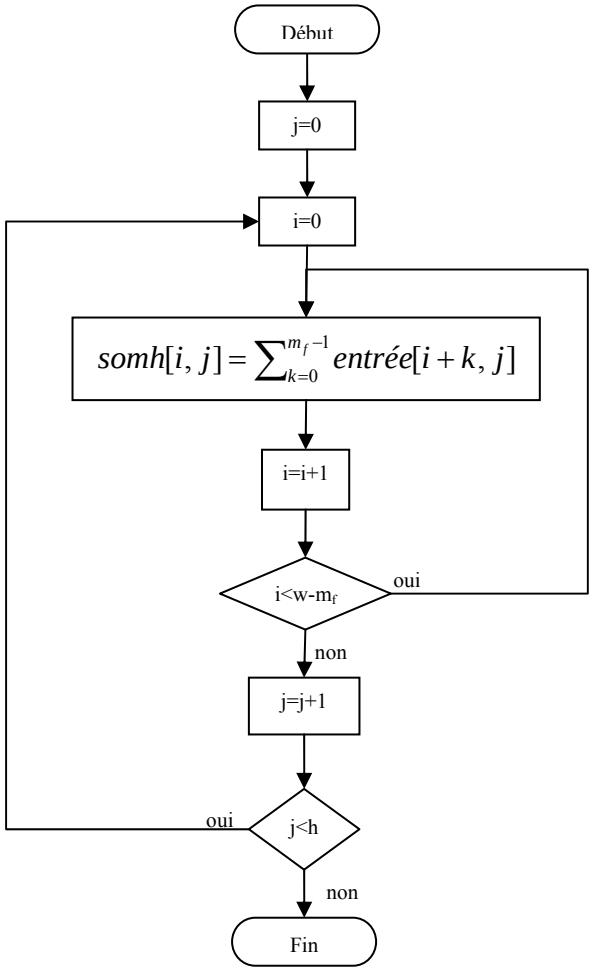


Figure 3-10 : l'algorithme du moyennage-ligne.

Le moyennage-colonne s'applique de la même manière en scannant les colonnes issues de l'image de l'étape précédente pour faire la somme dans une fenêtre glissante de trois pixels successifs (figure 3-11).

Dans cette étape, les deux boucles (i et j) scannent les deux images résultats de l'étape précédente (somhd et somhg) et la boucle (k) scanne verticalement sur une gamme de trois pixels.

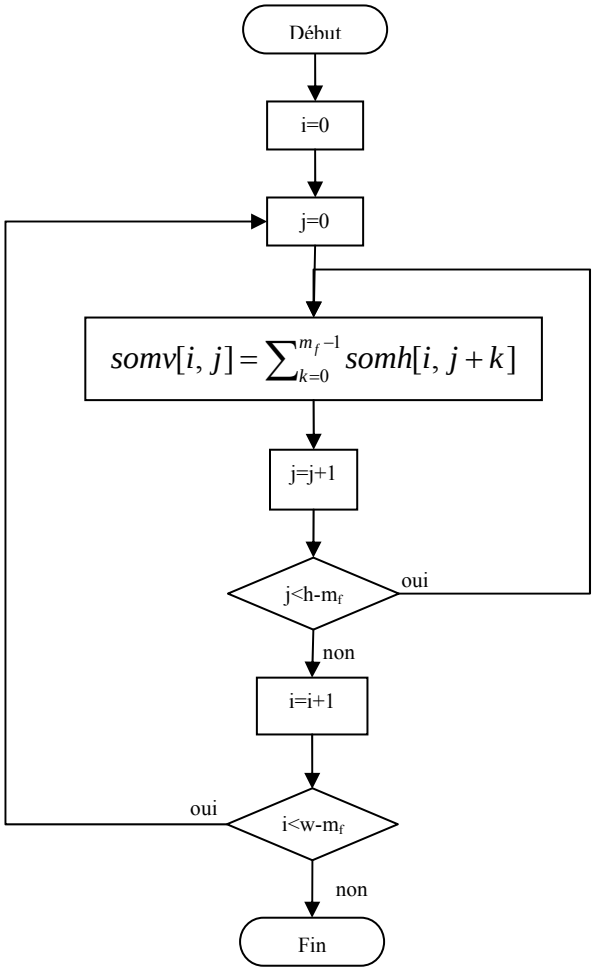


Figure 3-11 : l'algorithme du moyennage-colonne.

L'algorithme suivant décrit le processus de recherche de la stéréo-correspondance. Cet algorithme comprend trois étapes : la transformation Census, le calcul des scores et la recherche du meilleur score dont le rang correspond à la disparité recherchée (figure 3-12).

Les tableaux « somvg » et « somvd » sont les résultats de l'étape de moyennage.

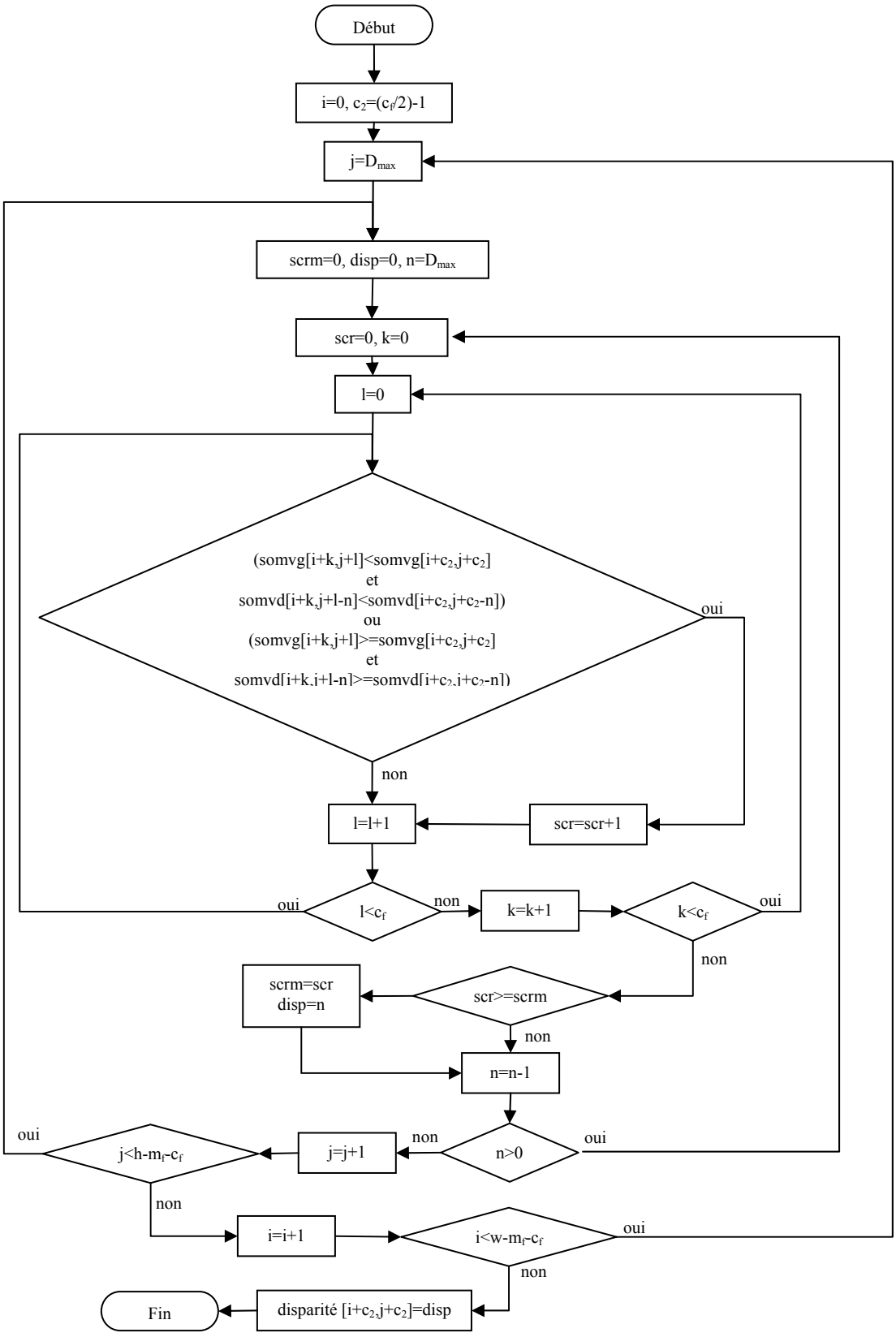


Figure 3-12 : l'algorithme de recherche de la stéréo-correspondance.

Les deux boucles (i et j) scannent les images d'entrée, la boucle (n) scanne la chaîne des pixels droite et les deux boucles (k et l) scannent la fenêtre glissante sur l'image Census (7x7 dans ce cas). L'algorithme présenté ci-dessus fournit un tableau des disparités qui peut être affiché en fausses couleurs ou en niveau de gris.

3.4 Influence des paramètres de fenêtrage

La taille de la fenêtre Census est un élément important pour le calcul de l'image de disparités : elle contrôle la qualité de l'appariement entre les images gauche et droite de la scène observée. En effet comme on peut le constater sur la figure 3-13, plus la fenêtre sera importante plus le critère de ressemblance sera fin et meilleurs seront les appariements.

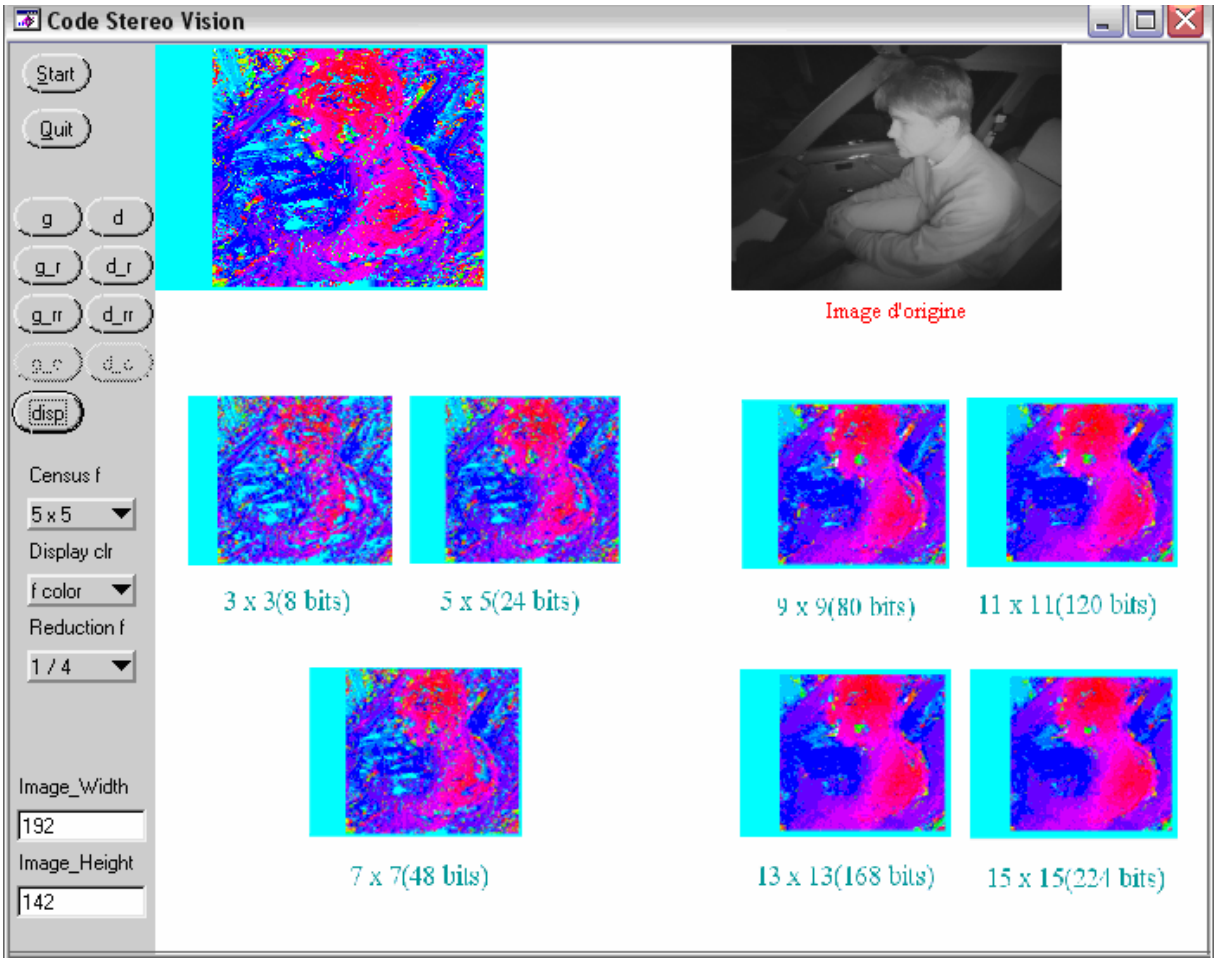


Figure 3-13 : l'interface du banc d'évaluation et exemples d'images de disparité en fonction de la taille de la fenêtre Census.

De manière générale sur les images peu texturées, comme on en rencontre dans le domaine médical, il est nécessaire d'augmenter la taille de ces fenêtres pour améliorer les appariements et réduire les artefacts. En contre partie cette augmentation entraine la manipulation de chaînes de données de taille pouvant être très importantes (8 bits pour des fenêtres 3x3, 224 bits pour des fenêtres 15x15) ce qui entraine un accroissement des temps calcul et de la taille des blocs mémoire utilisés. Un compromis est donc à trouver en fonction du type de scène observé.

3.5 Conclusions

Dans ce chapitre nous avons présenté un banc d'évaluation ergonomique et évolutif qui nous a permis :

- d'effectuer la calibration des caméras
- de déterminer les contenus des matrices pour la rectification des images
- d'évaluer l'influence des paramètres intervenant dans le processus d'appariement des pixels (taille des fenêtres, précision et codage des variables…)
- de faciliter le débogage de la plate-forme temps réel par comparaison des résultats obtenus
- de trouver un compromis entre la complexité du calcul, la texture des images et la qualité des images de disparité. Sur ce dernier point nous avons retenu des fenêtres Census de taille 7x7 qui sera notre base de travail pour l'implémentation matérielle.

3.6 Références bibliographiques

[JAI_v1] 'Digital Monochrome Quad Speed CMOS Progressive Scan Camera CV-A33CL', Manuel de fonction, Version 1.0.

[Informatique_enc] 'http://www.commentcamarche.net/'.

[NI_98] National Instruments, 'LabWindows/CVI User Interface Reference Manual', edition Fev 1998.

[Nicollin_82] E. Nicollian et J. Brews, 'MOS Physics and Technology', John Wiley and Son NY, 1982.

[Sequin_75] C. Sequin et M. Tompset, 'Charge Transfert Devices. Academic', 1975.

[Sze_81] S. Sze, 'Physics of Semiconductors devices', John Whiley and Son NY, 1981.

Chapitre4

Architectures Matérielles pour la Stéréovision Temps Réel

---

4 ARCHITECTURES MATERIELLES POUR LA STEREOVISION TEMPS REEL

4.1 Introduction

Ce chapitre est consacré au portage de l'algorithme de la stéréovision dense sur une plate-forme matérielle afin de répondre aux contraintes temps réel qui nous sont imposées. Nous présentons dans un premier temps un état de l'art du domaine, les contraintes qui ont guidé notre choix vers des solutions à base de circuits FPGA ainsi que les outils et langages de développement associés à cette technologie. Nous présentons également brièvement les caractéristiques de ces circuits en termes de granularité, de ressources embarquées et leur adéquation avec le problème posé. Enfin l'implémentation de chaque étape de la stéréovision est explicitée et évaluée en termes d'architecture, de ressources consommées et de performances temps réel.

4.2 Etat de l'art en stéréovision temps réel

La stéréovision en temps réel a été largement utilisée dans la navigation de robots, l'interaction homme-machine intelligente et d'autres systèmes autonomes. Dans cette partie nous donnons une vue d'ensemble des systèmes de vision stéréo en temps réel les plus appropriés basés sur des algorithmes locaux proposés dans la dernière décennie.

Ainsi un système stéréo basé sur un DSP a été développé à l'INRIA par Faugeras [INRIA_93] : il utilise un système de vision trinoculaire et la corrélation NCC (Normalised Cross Correlation). Il calcule une carte de profondeur de 64 x 64 pixels en environ 400 msec à partir d'une plate-forme équipée de quatre DSP Motorola 96002 (MD96). DEC-PRL développe une carte à base de circuits LCA Xilinx

XC3090 (PeRLe-1) et réduit les temps calcul d'un facteur 34 par rapport à la carte MD96.

[Kanade_96] a développé un système polynoculaire (jusqu'à 6 caméras) basé sur un DSP C40 et capable de produire une carte de profondeur au rythme de 30 images/sec avec une résolution de 256 x 240 pixels. Le calcul de correspondance était de type SSD (Sum of Squared Differences) et une valeur globale (SSSD) était calculée en ajoutant les contributions données par les différences entre plusieurs vues.

[Woodfill_97] a développé une carte FPGA au format PCI. Ce système, dont l'algorithme d'appariement est basé sur le Census, peut exécuter 42 images de disparités par seconde pour une paire stéréo de deux caméras de 320 x 240 pixels de résolution.

Un système basé sur une carte ADSP2181, développé au SRI [Konolige_97], fournit la stéréo-correspondance en temps réel pour des images de 160 x 120 pixels à un rythme de 8 images/sec. Ce système utilise deux images prétraitées via un filtre LOG et exploite un algorithme de correspondance de type SAD.

Corke a développé un autre système stéréo mettant en œuvre des circuits FPGA et basé sur la transformation Census [Corke_99] : il exécute 30 images/sec en résolution de 256 x 256 pixels.

La machine SAZAN [Kimura_99] se compose de deux cartes PCI et d'un système poly-oculaire d'images composé de neuf caméras d'une résolution de 320 x 240 pixels. Le système de stéréovision SAZAN combine plusieurs mesures SAD (SSAD) et fournit 20 images par seconde de taille 200 x 200 avec une gamme de disparité de 25 pixels.

Un système, proposé par [Gluckman_99], utilise deux caméras et deux miroirs paraboliques pour produire des mesures de profondeur temps réel omnidirectionnelles. Son algorithme de stéréo-correspondance utilise la mesure SAD et, avec des images panoramiques de 600 x 60 pixels et une gamme de disparité de 32

pixels, il obtient un rythme de 7 images/sec sur un processeur Pentium II cadencé à 300 MHz.

Sarnoff Corporation a développé un circuit dédié aux traitements vidéo [Van der Val_01] et capable d'exécuter la stéréo-correspondance sur des images de 320 x 240 pixels. Le module stéréo utilise la Somme de Différences Absolues comme mesure d'appariement.

Tanahashi a développé un système de vision omnidirectionnelle [Tanahashi_01], il utilise 60 caméras pour obtenir des mesures de profondeur omnidirectionnelles en temps réel.

Un système de stéréovision a été développé pour les aveugles par Areibi [Areibi_01], ce système traite 4 images/sec et utilise un algorithme d'appariement pixel to pixel (P2P). La carte de traitement utilisée est basée sur un FPGA Xilinx XS40.

D'autres systèmes de vision stéréo en temps réel à base de PC ont été récemment proposés : ils sont basés sur la mesure SAD, la correspondance bidirectionnelle et les capacités SIMD [Hirshmuller_02, Muhlmann_02, Di Stefano_02, Di Stefano_04].

[Navarro_02] a implanté, en analogique, une transformée de Census 3x3 sur un capteur d'image CMOS pour estimer les déplacements d'objets.

Yang a développé en langage C, en s'appuyant sur l'algorithme SSD (Sum of Squared Differences), un système stéréo produisant jusqu'à 8 images/sec [Yang_03].

[Darabiha_03] a mis en œuvre un système de stéréovision temps réel avec quatre FPGA. Celui-ci est capable de produire 30 images/sec pour une taille d'image de 256 x 360 en utilisant un algorithme multi-résolution, multi-orientation (local weighted phase correlation).

La machine de vision MSVM-II développée par [Jia_03] est un système stéréo très compact (10 x 10 cm) et multi-bases (jusqu'à 8 caméras). Le système prétraite des

images en corrigeant la distorsion des lentilles, en exécutant un filtrage LOG et trouve les points de correspondance en utilisant l'algorithme SSAD. Il est basé sur la mise en œuvre de FPGA fonctionnant à 80 MHz et fournit jusqu'à 30 images/sec à la résolution de 640 x 480 pixels et jusqu'à 50 images/sec à la résolution de 320 x 240 pixels avec une gamme de disparité de 64 pixels.

Successivement, les mêmes auteurs ont proposé MSVM-III [Jia_04] un système de vision stéréo trinoculaire. MSVM-III est basé sur des FPGA à 60 MHz et utilise l'algorithme SSAD avec une gamme de recherche de disparité de 64 pixels, il fournit 30 images/sec à la résolution de 640 x 480 pixels et jusqu'à 120 images/sec à la résolution de 320 x 240 pixels.

Le tableau ci-dessous résume les principaux travaux dans le domaine et les performances obtenues.

Année	*Auteur*	*Plateforme*	*Algorithme*	*Taille image*	*Images/sec*
93	*INRIA*	*DSP 96002*	*-*	*64x64*	*2.5*
96	*Kanade*	*DSP C40*	*SSD*	*200x200*	*25*
97	*Woodfill*	*FPGA*	*Census*	*320x240*	*42*
97	*SRI*	*DSP2181*	*SAD*	*160x120*	*8*
99	*Corke*	*FPGA*	*Census*	*256x256*	*30*
99	*Gluckman*	*Pentium II*	*SAD*	*600x60*	*7*
01	*Areibi*	*FPGA XS40*	*P2P*	*-*	*4*
03	*Yang*	*PC*	*SSD*	*-*	*8*
03	*Darabiha*	*FPGA*	*MRMO*	*256x360*	*30*
03	*Jia*	*FPGA*	*SSAD*	*640x480*	*30*
04	*Jia*	*FPGA*	*SSAD*	*320x240*	*120*

Tab 4-1 : Etat de l'art en stéréovision temps réel.

4.3 Les solutions de développement

Pour implémenter la stéréovision en temps réel nous avons globalement quatre possibilités :

- une implémentation logicielle (S1) exécutée sur une plate-forme native.
- une implémentation logicielle (S2) exécutée sur un processeur cible.
- une implémentation matérielle (S3) obtenue à partir d'un outil de transformation d'un langage informatique de haut niveau en langage VHDL synthétisable.
- une implémentation matérielle (S4) obtenue à partir d'une description structurelle.

La première solution consiste à traduire les algorithmes de la stéréovision dans un langage de haut niveau (basic, fortran, c, c++,…) ou éventuellement en langage assembleur, le compiler ou l'assembler et le faire tourner sur une plate-forme puissante munie de son Operating System (OS). Cette solution a l'avantage d'être simple, prend très peu de temps pour la développer mais ne répond pas à notre besoin ni en terme de rapidité ni en terme de portabilité. C'est la solution retenue, car la mieux adaptée, pour le banc d'évaluation.

La deuxième solution consiste à utiliser un cross-compilateur ou un cross-assembleur pour traduire le langage de haut niveau ou assembleur dans un exécutable pour une machine cible plus appropriée comme un DSP par exemple. Bien que cette solution soit embarcable, plus performante que la première car on peut exploiter au mieux les architectures internes des DSP et offre des temps de développement courts, elle s'avère incompatible avec notre critère de rapidité.

Depuis quelques années des outils permettant la traduction d'un logiciel décrit en langage de haut niveau en langage VHDL synthétisable et donc portable vers un environnement matériel câblé à base de FPGA ont fait leur apparition (solution 3). Ils offrent l'avantage des temps de développement raisonnables que l'on rencontre dans les solutions logicielles tout en offrant les avantages de la rapidité dévolue aux

architectures câblées. Cependant ces outils issus du génie logiciel exploitent peu les multiples possibilités de parallélisme que l'on peut rencontrer dans le traitement du signal et plus généralement dans le traitement d'images. La conséquence est que le gain en rapidité est peu important comparativement a une solution DSP.

Enfin la dernière solution permet une description intime des architectures et une exploitation au mieux des ressources et des possibilités de routage que l'on rencontre dans les circuits FPGA. Elle nécessite une analyse très fine des algorithmes à porter mais elle est la plus performante pour des applications où les temps d'exécution sont critiques ; en contrepartie les temps de développement sont plus longs. Malgré ce dernier aspect c'est cette solution que nous avons retenue.

4.4 Le portage de l'algorithme vers le monde parallèle

4.4.1 Etude détaillée de l'algorithme

L'implantation de l'algorithme de stéréovision sur le banc d'évaluation nous a permis de :

- mettre en évidence l'influence des paramètres de fenêtrage ainsi que les effets de la quantification des différentes grandeurs.
- Comprendre de manière intime les différentes étapes ou fonctions à réaliser pour exploiter au mieux le parallélisme.

4.4.1.1 La calibration

Cette étape, réalisée « hors ligne », permet de pré-calculer 4 tables de rectification inverses permettant de prendre en compte les imprécisions liées aux alignements mécaniques, les effets de distorsion des optiques et la réduction entre les images originales et les images rectifiées.

Principe du calcul des tables inverses :

A partir d'un pixel (u_{rect}, v_{rect}) dans l'image rectifiée et en utilisant les tables de rectification, on peut obtenir sa projection dans l'image originale :

- On détermine un pixel ($u_{origine}$, $v_{origine}$), qui correspond à cette projection.

- On affecte les éléments des tables inverses ainsi calculés :

  table_inverse_i [$u_{origine}$* $NbCol_{origine}$ + $v_{origine}$] = (u_{rect}) (table de « ligne »)

  table_inverse_j [$u_{origine}$* $NbCol_{origine}$ + $v_{origine}$] = (v_{rect}) (table de « colonne »)

  , où $NbCol_{origine}$ est le nombre de colonnes de l'image d'origine.

  - On crée deux tables inverses (respectivement pour les coordonnées « ligne » et « colonne » des pixels) pour rectifier les images originales gauches (resp. les images originales droites). Au total on doit créer quatre tables inverses.
  - Chaque table a la même taille que l'image originale.
  - Dans les tables de rectification inverses, pour chaque pixel de l'image originale, on stocke les coordonnées du pixel qui lui correspond dans l'image rectifiée.
  - Ces tables comportent des « trous » qui correspondent aux pixels de l'image d'origine qui n'ont pas de correspondant dans l'image rectifiée (plus petite que l'image d'origine).

  Choix de ($u_{origine}$, $v_{origine}$) à partir de la projection d'un pixel (u_{rect},v_{rect}) :

La projection du pixel (u_{rect},v_{rect}) dans l'image originale correspond à une « case » (coordonnées entières) des tables inverses. On doit donc « arrondir » ou « tronquer » la position ($u_{origine}$, $v_{origine}$) de la projection de (u_{rect},v_{rect}) dans l'image originale.

S'il n'y a pas de réduction de taille entre l'image originale et l'image rectifiée, on choisit pour ($u_{origine}$, $v_{origine}$) le pixel le plus proche de la projection de (u_{rect},v_{rect}) dans l'image originale (on « arrondit »).

S'il y a un taux de réduction de taille entre l'image originale et l'image rectifiée, le choix pour ($u_{origine}$, $v_{origine}$) dépend de la parité du taux de réduction :

- si le taux de réduction entre l'image originale et l'image rectifiée est impair (par exemple 3 x 3 dans la figure 4-1), on choisit pour ($u_{origine}$, $v_{origine}$) le pixel le plus proche de la projection de (u_{rect},v_{rect}).
- si le taux de réduction entre l'image originale et l'image rectifiée est pair (par exemple 4 x 4 dans la figure 4-2), on choisit pour ($u_{origine}$, $v_{origine}$) les valeurs entières de la projection (on « tronque »).

Ceci permet alors de centrer au mieux la fenêtre de calcul de l'intensité de (u_{rect},v_{rect}) (fenêtre de moyennage si on veut tenir compte du taux de réduction, cf. figure 4-3 et figure 4-4).

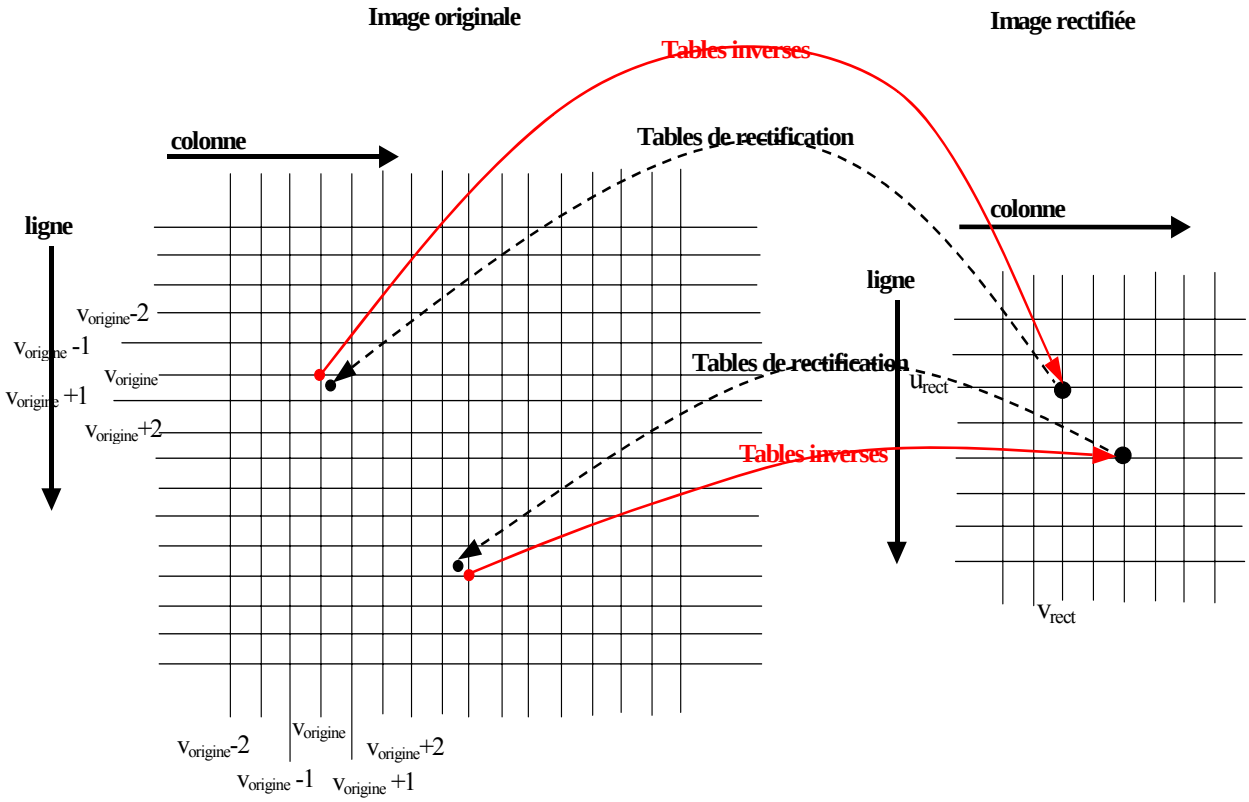


Figure 4-1 : Exemple avec taux de réduction de 3 x 3.

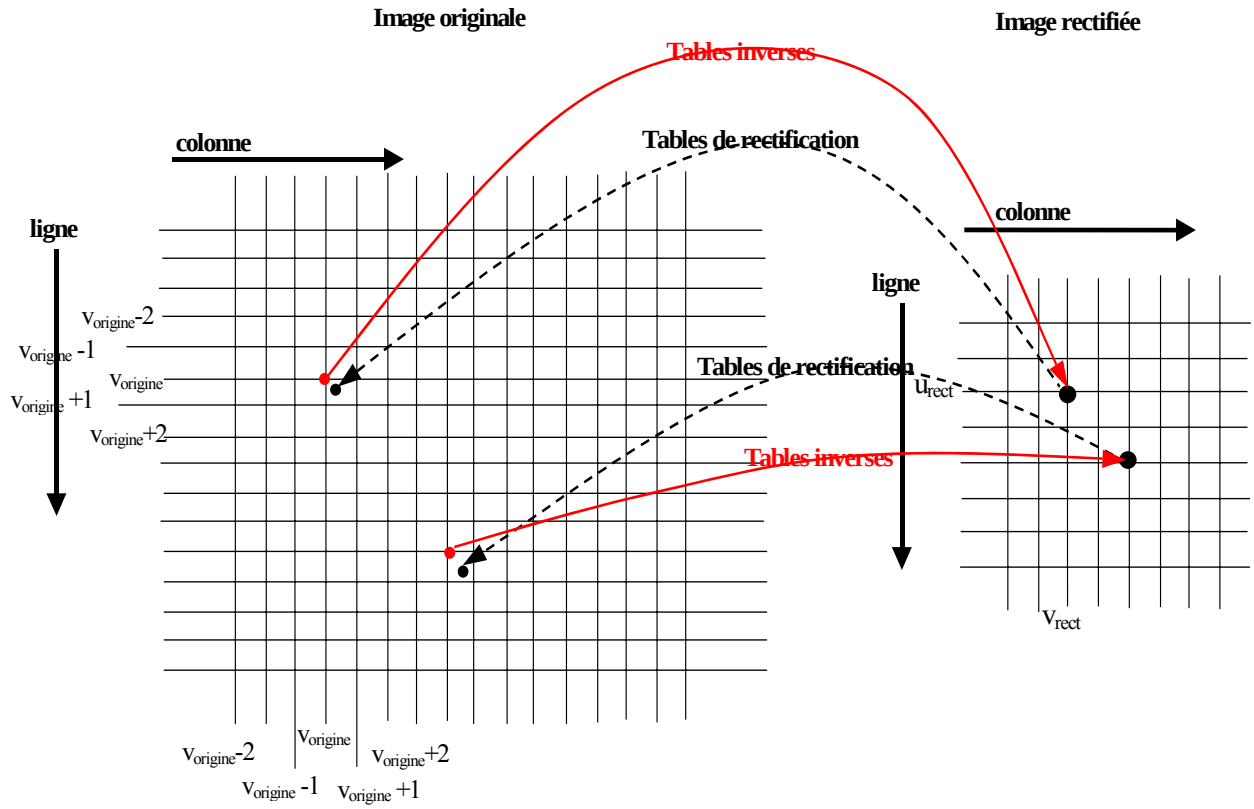


Figure 4-2 : Exemple avec taux de réduction de 4x4.

4.4.1.2 La rectification

Dans cette étape on rectifie les images originales en utilisant les tables inverses pré-calculées dans la phase de calibrage.

Calcul du projeté d'un pixel de l'image d'origine dans l'image rectifiée :

Pour chaque point de l'image d'origine :

- Si la table de rectification inverse indique l'existence d'un correspondant dans l'image rectifiée, on calcule l'intensité en niveaux de gris de ce point projeté.
- Si la table de rectification inverse n'indique pas l'existence d'un correspondant dans l'image rectifiée, on ne fait rien.

Calcul de l'intensité d'un pixel dans l'image rectifiée :

Pour calculer l'intensité en niveaux de gris de ce point projeté, 2 approches ont été considérées :

- On affecte directement l'intensité de ($u_{origine}$, $v_{origine}$) à (u_{rect}, v_{rect}).
- On prend en compte les effets de la réduction entre l'image originale et l'image rectifiée. Pour obtenir le niveau de gris d'un pixel dans l'image rectifiée, on fait alors la moyenne des niveaux de gris dans une fenêtre N×N (N dépendant du taux de réduction entre l'image originale et l'image rectifiée). Le positionnement de cette fenêtre autour de ($u_{origine}$, $v_{origine}$) dépend de la parité du taux de réduction. Par exemple, pour un taux de réduction impair on peut centrer la fenêtre sur ($u_{origine}$, $v_{origine}$). La fenêtre de moyennage est positionnée de telle sorte que ($u_{origine}$, $v_{origine}$) soit à la position [(N-1)/2, (N-1)/2] par rapport au point en haut à gauche de cette fenêtre (voir le cas N=3 figure 4-3) ; pour un taux de réduction pair, on ne peut pas centrer exactement la fenêtre de moyennage. La fenêtre de moyennage est positionnée de telle sorte que ($u_{origine}$, $v_{origine}$) soit à la position [(N/2 -1), (N/2 -1)] par rapport au point en haut à gauche de cette fenêtre (voir l'exemple du cas N=4 dans la figure 4-2).

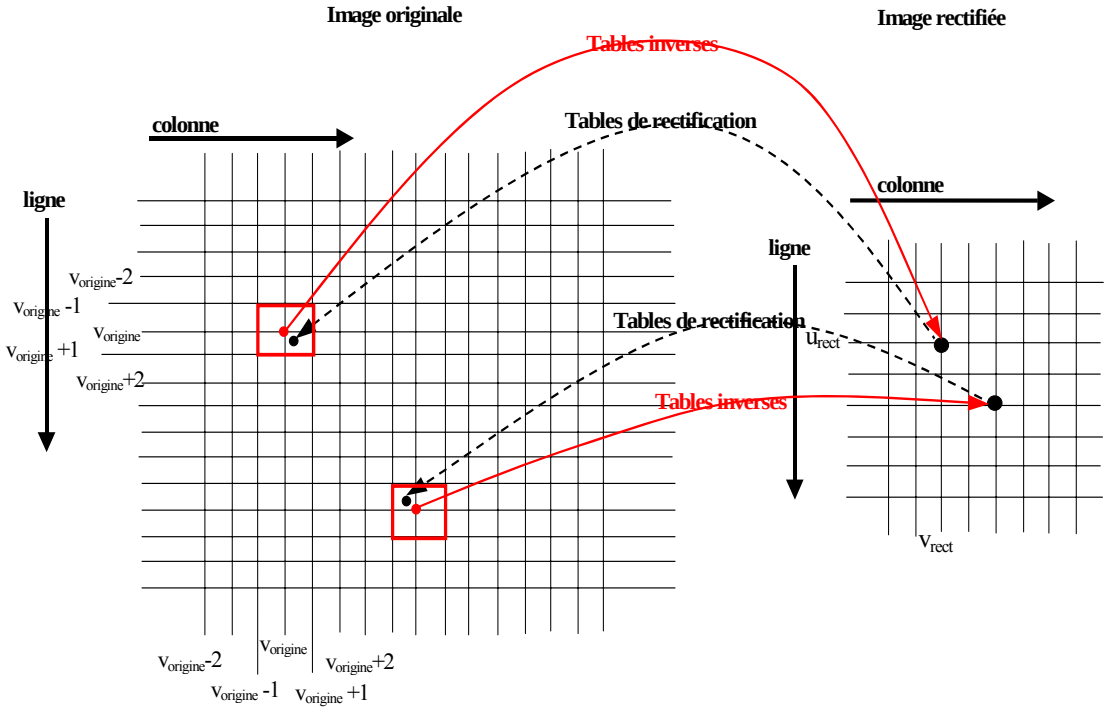


Figure 4-3 : calcul de la moyenne dans une fenêtre 3x3.

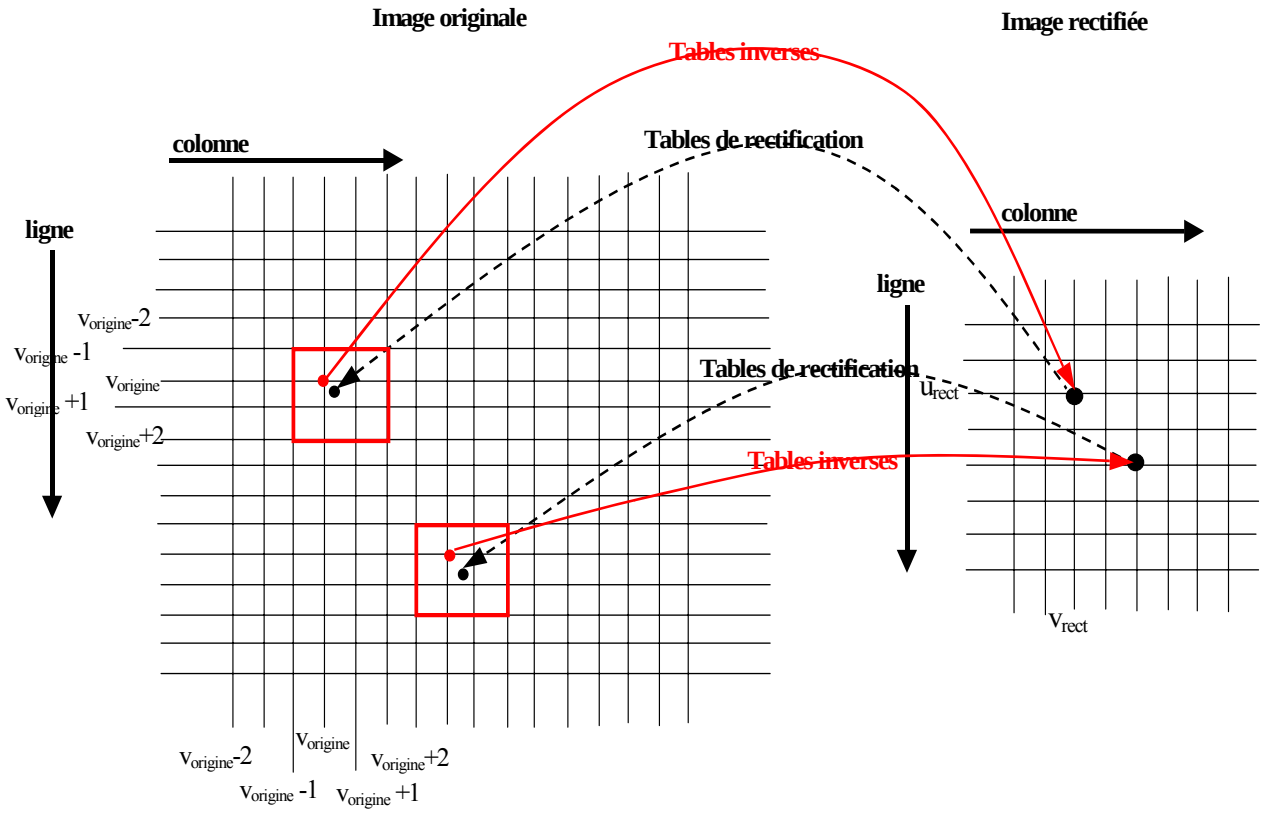


Figure 4-4 : calcul de la moyenne dans une fenêtre 4x4.

Dans cette approche, on doit attendre que tous les pixels voisins nécessaires dans l'image originale (par exemple, 16 pixels pour un taux de réduction = 16) soient disponibles pour calculer le niveau de gris d'un pixel de l'image rectifiée. C'est cette deuxième proposition que nous avons retenu.

4.4.1.3 La recherche de l'appariement

Calcul du Census (figure 4-5) :

- le pixel central est le pixel de référence (valeur de luminance = 67).
- la valeur 67 est remplacée par la chaîne Census (montrée à droite).
- Les bits de cette chaîne donnent les résultats des comparaisons de cette valeur 67 avec les valeurs de luminance des pixels de la fenêtre, parcourue ligne à ligne, de gauche à droite, et de haut en bas.
- le bit vaut 1 si 67 est plus grand ou égal à la valeur du pixel traité, 0 sinon. Par exemple, le premier bit est 0 car 120 > 67, le second 0 car 96 > 67, le troisième 1 car 65 < 67 ……

120	96	65	76	130
34	90	45	78	34
78	90	67	45	67
78	100	67	158	130
67	120	95	189	145

→ **Census = 00100101010010010010000**

Figure 4-5 : Transformation Census sur un pixel avec une fenêtre 5×5 (24 bits).

Rappel du calcul de la mesure de similarité :

La mesure donne seulement un score de ressemblance entier compris entre 0 et 8 dans le cas d'une fenêtre Census de 3 x 3 (de 0 à 8 bits identiques entre les deux codes).

Par exemple (figure 4-6) :

Le code Census du pixel x_0 (8 bits) : 01100110

Le code Census du pixel x_0+D_{min} : 10000110 (la distance de Hamming est 5)

Le code Census du pixel x_0+D_{min} +1 : 10011111 (la distance de Hamming est 2)

…… ……

Le code Census du pixel x_0+D_{max} : 00100110 (la distance de Hamming est 7)

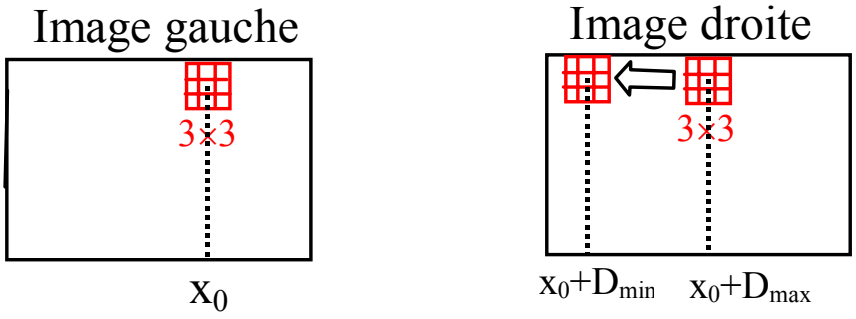


Figure 4-6 : recherche d'appariement avec un code Census 8 bits sur une fenêtre 3x3.

Cette solution n'est pas suffisante pour rechercher un maximum de ressemblance de manière satisfaisante (taille de la fenêtre trop faible). Deux solutions sont toutefois possibles pour améliorer cette méthode :

- 1ère solution : on calcule le code Census dans une fenêtre plus grande et on calcule ensuite la similarité entre ces codes Census droite et gauche.
- 2ème solution : on calcule d'abord les scores de ressemblance entre les codes Census droite et gauche en utilisant une fenêtre 3x3 puis on calcule la moyenne des scores sur une fenêtre de corrélation plus large que 3x3 (typiquement 13x13 en figure 4-7). Cette moyenne est une valeur flottante comprise entre 0 et 8. En la normalisant (en divisant par 8), on obtient un score compris entre 0 et 1 dont le maximum est recherché pour trouver la disparité.

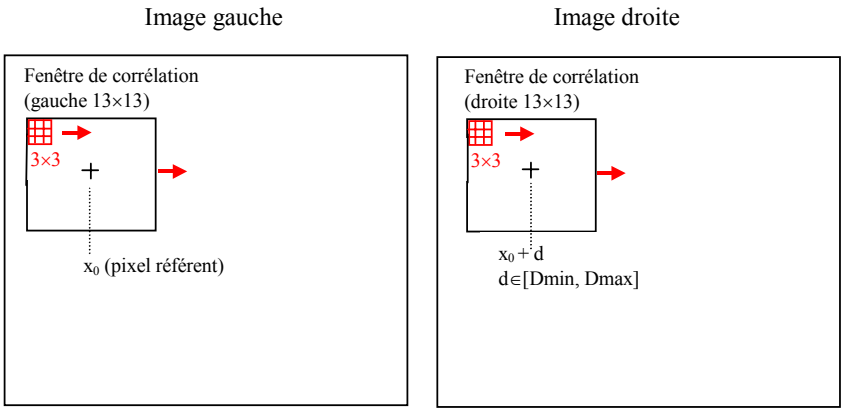


Figure 4-7 : Moyenne des scores dans une fenêtre de corrélation 13x13.

Dans l'algorithme actuel on effectue le moyennage sur les images rectifiées avec une fenêtre 3x3 puis on utilise la 1ère solution : on calcule les codes Census dans une fenêtre 7x7 (48 bits) et ensuite on recherche la similarité entre les codes Census droite et gauche.

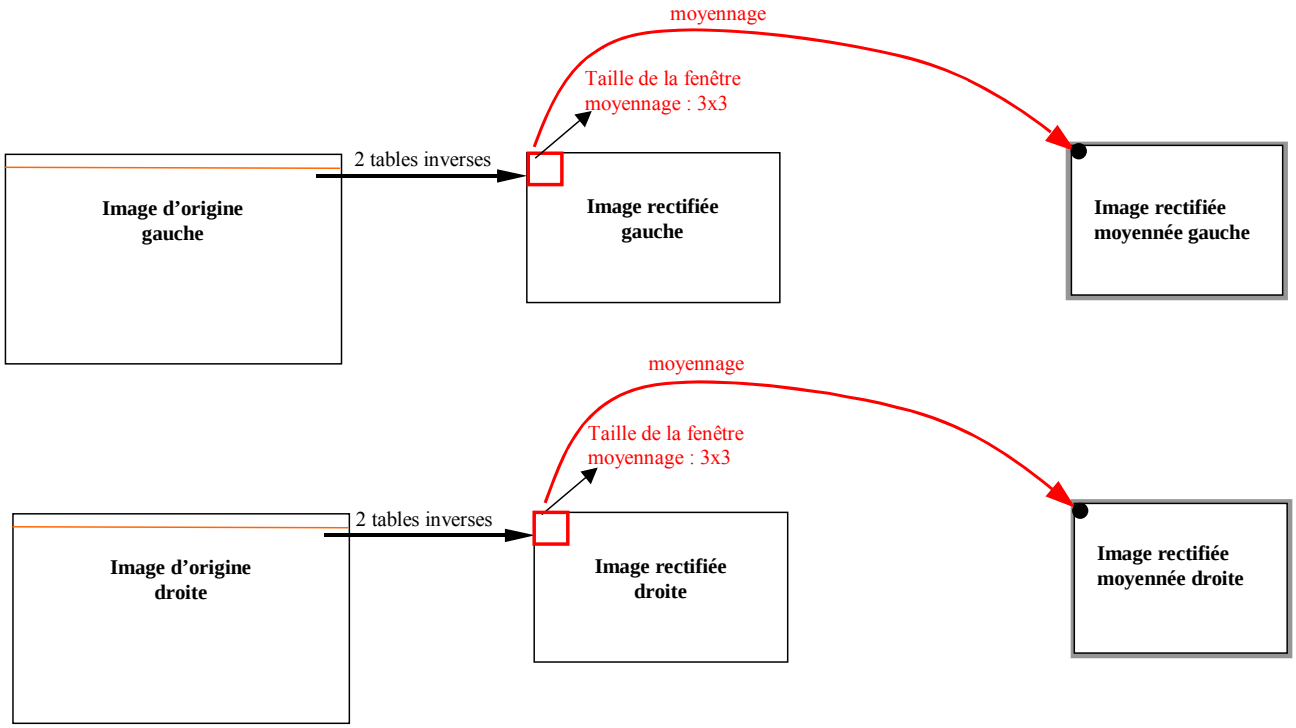


Figure 4-8 : rectification, puis moyennage de l'image rectifiée dans une fenêtre 3x3.

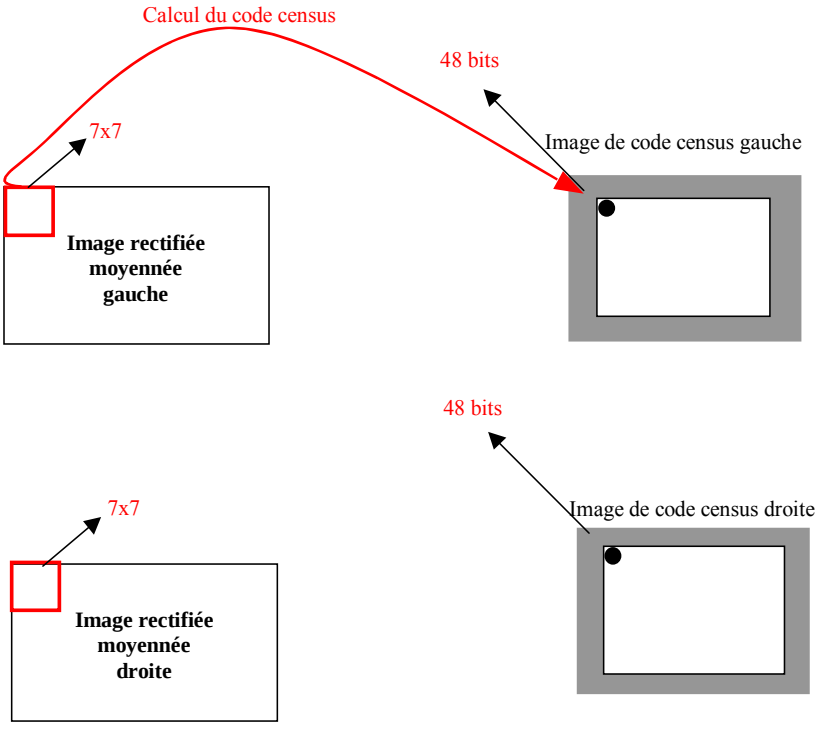


Figure 4-9 : Calcul du code Census dans une fenêtre 7x7.

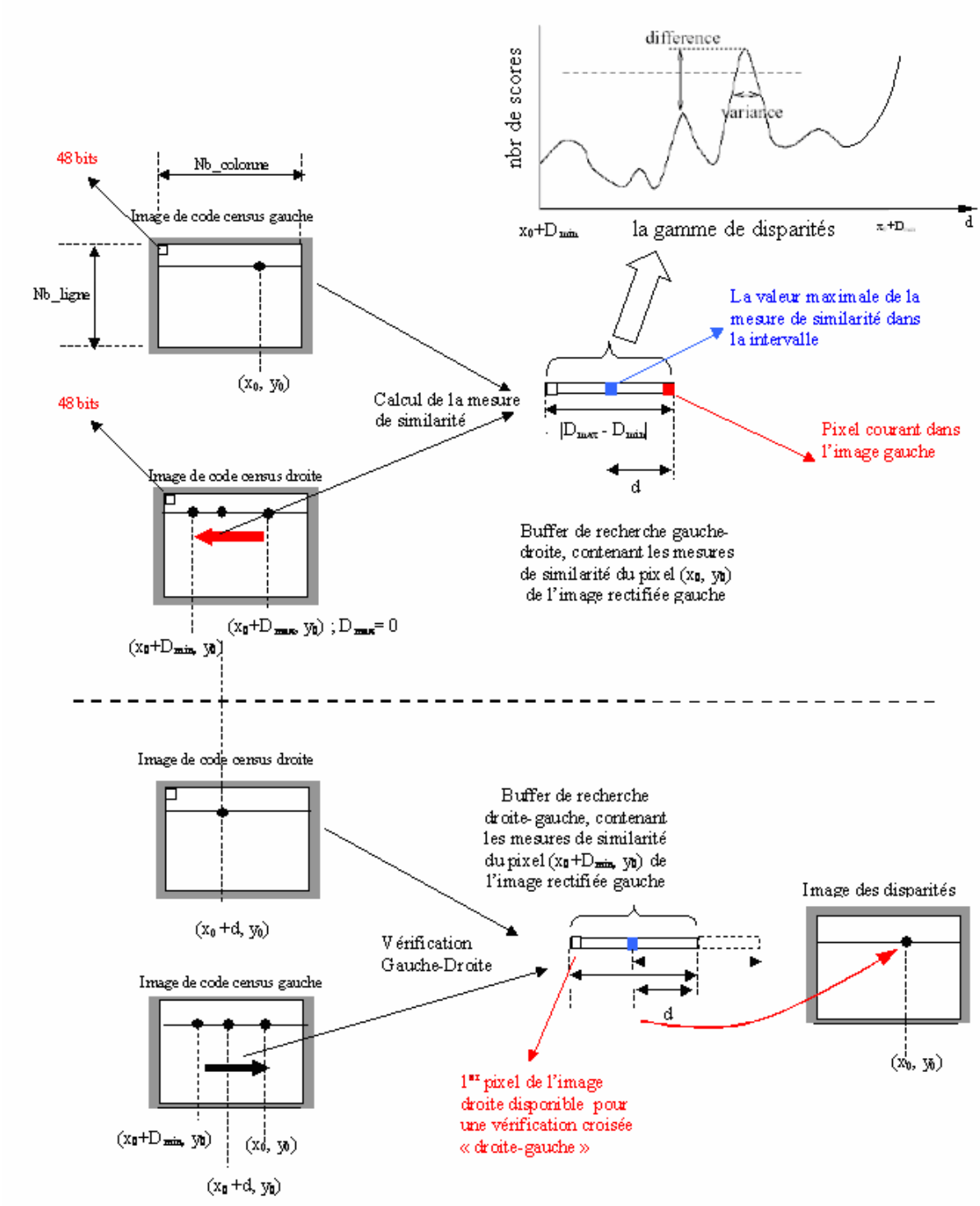


Figure 4-10 : Calcul de la mesure de similarité dans un intervalle [D_{min} , D_{max}] et calcul de la disparité d'un pixel de l'image gauche rectifiée.

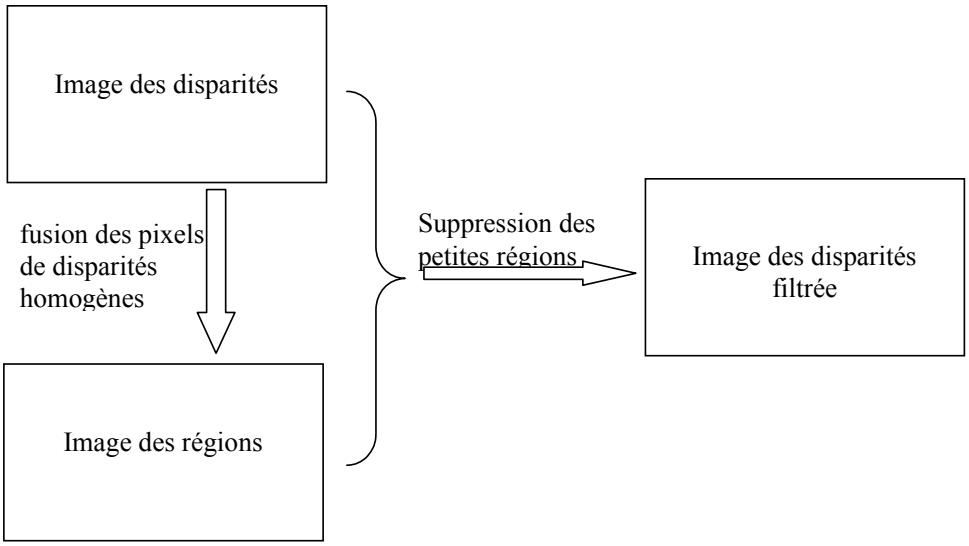
4.4.1.4 *La recherche des disparités*

Pour chaque pixel de l'image rectifiée gauche, on recherche la valeur maximale dans l'intervalle [D_{min} , D_{max}] de l'image droite. La distance *d* trouvée

correspond à la disparité du pixel. On la vérifie ensuite par la « vérification droite-gauche » (cf. figure 4-10).

4.4.1.5 Le filtrage de l'image des disparités

Processus à réaliser :



Cette étape nécessite de stocker :

- la table des correspondances « couleurs-étiquettes ».
- le tableau des cardinaux (nombre de pixels) des régions.

, et de réaliser un maximum d'opérations en temps réel, notamment :

- Algorithme de construction / fusion des régions.
- gestion de la table des correspondances couleurs-étiquettes.

On mémorise le statut des régions à filtrer en utilisant une étiquette « 0 », mais on ne peut statuer sur la région grise (à filtrer ou pas ?) avant de l'avoir entièrement décrite.

Il faut donc stocker aussi :

- l'image des disparités.
- l'image des régions.

, et faire le filtrage effectif des disparités en même temps que la reconstruction seulement après que toute l'image des disparités ait été traitée.

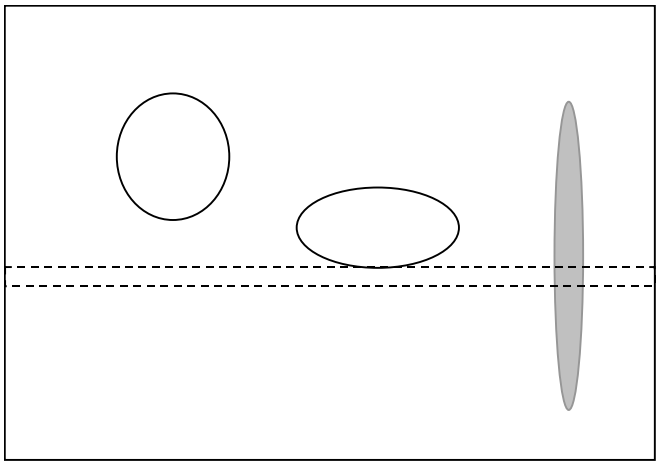


Figure 4-11 : filtrage des disparités.

4.4.1.6 La reconstruction 3D

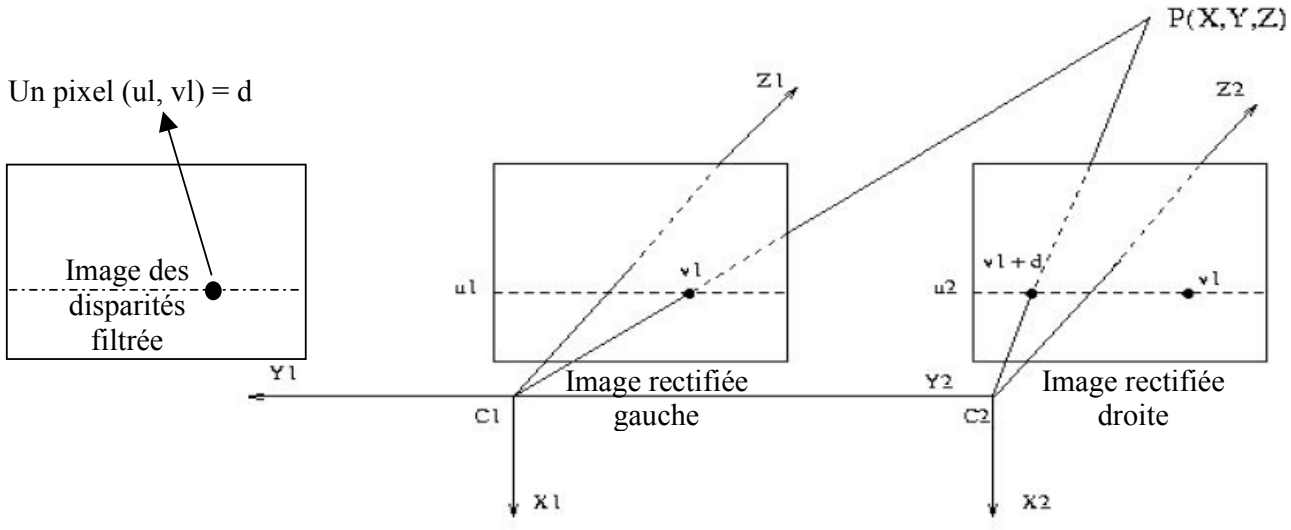


Figure 4-12 : Reconstruction 3D.

$$X = \frac{\alpha_v.B.(u_1 - u_0)}{\alpha_u.d}$$

$$Z = \frac{\alpha_v.B}{d} \qquad Y = \frac{B.(v_1 - v_0)}{d}$$

, avec :

(X, Y, Z) les coordonnées d'un point 3D dans la scène dans le repère de la caméra gauche rectifiée (le repère de référence),

(u_1, v_1) un pixel dans l'image gauche rectifiée (ou l'image de disparité filtrée),

(u_0, v_0) les paramètres intrinsèques des caméras rectifiées qui correspondent aux coordonnées du centre de l'image gauche rectifiée,

B la base du banc stéréo,

α_u, α_v les paramètres intrinsèques des caméras rectifiées.

4.4.2 L'algorithme de stéréovision dense « au fil de l'eau »

L'architecture globale pour l'implantation de l'algorithme est une architecture « pipeline » comprenant cinq étapes : la rectification, le moyennage 3 x 3, le calcul de Census 7 x 7, la mesure de l'appariement (le calcul des scores) et la recherche des disparités.

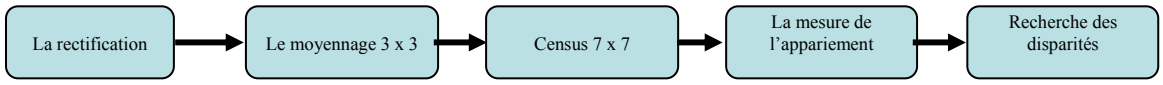


Figure 4-13 : Les étapes de la stéréovision dense

4.4.2.1 La rectification :

L'image rectifiée est une correction géométrique des positions des pixels de l'image originale déformée par l'objectif. Cette opération se fait en temps réel à partir des tableaux des paramètres pré-calculés et stockés en mémoire (SRAM ou Flash-Eprom).

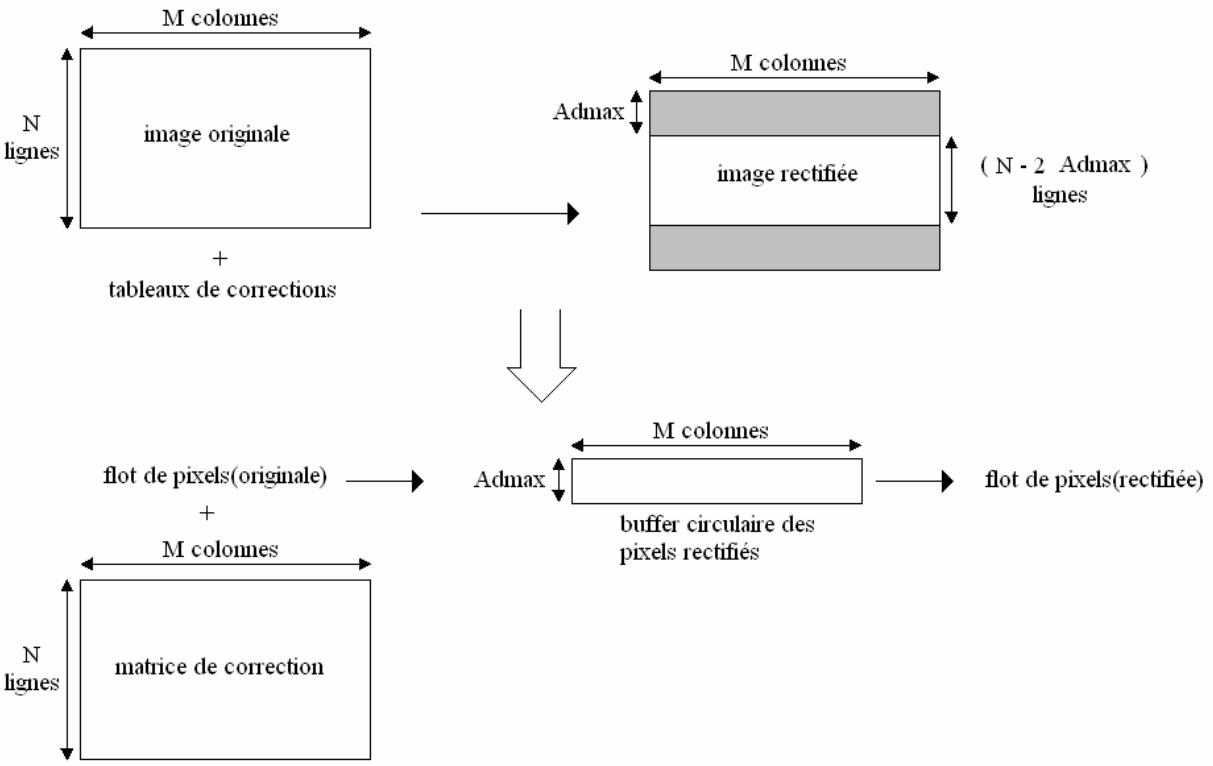


Figure 4-14 : Principe de la rectification en temps réel (pour une image).

Les pixels de l'image d'origine sont stockés au rythme de l'horloge dans un buffer circulaire ayant autant de lignes de mémoire que l'amplitude de la distorsion maximale (Ad_{max}) de l'image traitée. Chaque ligne du buffer circulaire est constituée d'une mémoire double-port permettant l'accès simultané en lecture-écriture. Dès que le buffer est rempli, la procédure de rectification est activée générant le flot de pixels de l'image rectifiée au rythme de l'horloge. Cette opération introduit par conséquent un retard pur dans le flot de traitement de :

$$W \, x \, Ad_{max} \, x \, T_{pixel}$$

, où W est la largeur de l'image traitée (M colonnes), T_{pixel} est la période pour un cycle d'horloge pixel.

4.4.2.2 Le moyennage 3 x 3:

L'image rectifiée est moyennée et stockée en utilisant une fenêtre glissante de taille 3 x 3 (voir figure 4-15).

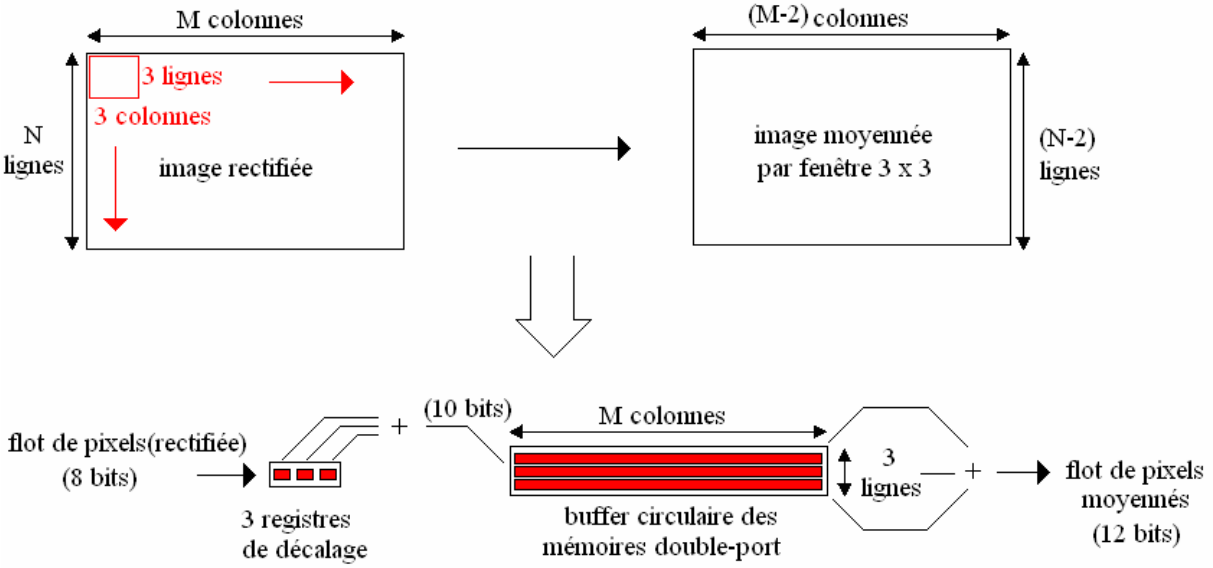


Figure 4-15 : Principe du moyennage 3 x 3.

Les pixels de l'image rectifiée traversent un bloc constitué de trois registres à décalage dont les sorties sont additionnées et stockées dans un buffer circulaire composé de trois mémoires double-port. Les sorties de ces trois mémoires sont également additionnées pour fournir à la fréquence pixel, une image de la valeur moyenne de la fenêtre considérée.

4.4.2.3 Calcul de l'image Census sur une fenêtre 7 x 7 :

L'image Census est obtenue en scannant l'image des moyennes par une fenêtre dont la taille est un élément essentiel pour faciliter l'appariement entre les images gauche et droite. Dans chaque fenêtre de l'image globale des moyennes, la valeur de l'intensité du pixel de référence (le pixel central de la fenêtre) est comparée avec celles des pixels voisins. Chaque pixel voisin est représenté par un bit dans le code Census.

Les pixels de l'image moyennée sont stockés au rythme de l'horloge pixel dans un buffer circulaire composé de sept mémoires double-port (figure 4-16). Les sorties de ces mémoires sont connectées à leur tour à une matrice composée de sept lignes et sept colonnes de registres à décalage (49 registres à décalage au total). Le rôle de cette matrice est de stocker les valeurs du pixel central et de ses pixels voisins pour la

fenêtre considérée. Ces valeurs sont ensuite traitées en temps réel pour fournir un flot de codes Census sur 48 bits.

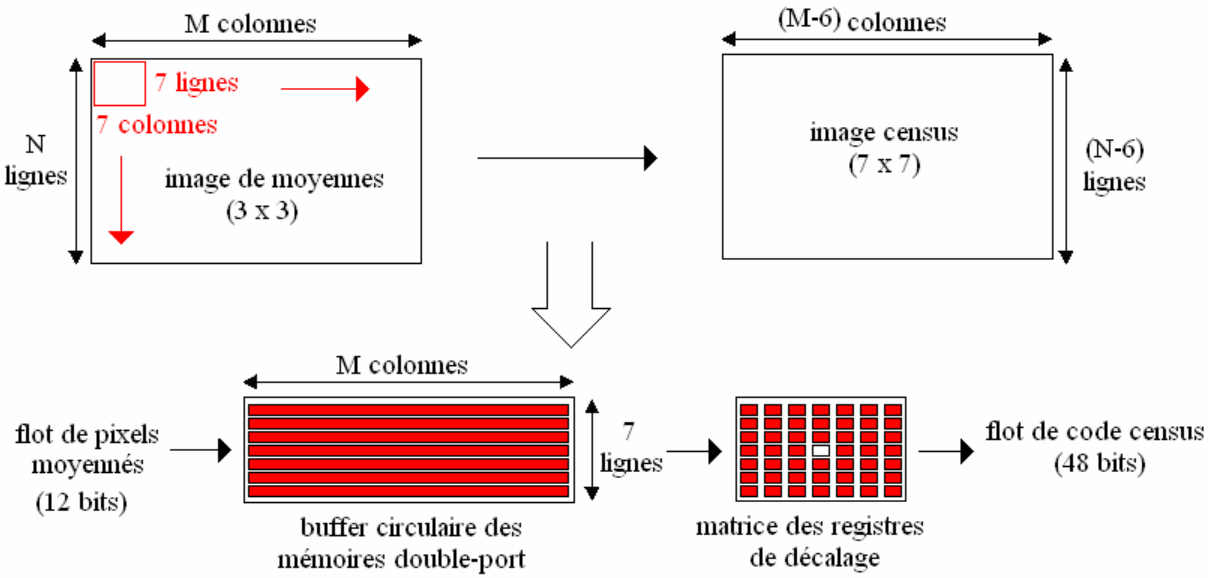


Figure 4-16 : Principe de la transformation Census à partir d'une fenêtre 7 x 7.

4.4.2.4 *Mesure de l'appariement et recherche des disparités :*

A partir des deux images Census (gauche et droite), on cherche pour chaque pixel gauche son meilleur correspondant dans une chaîne de pixels de l'image droite. Pour cela on calcule dans un premier temps pour chaque pixel candidat de l'image droite, le score associé à l'appariement : celui-ci correspond au nombre de bits identiques entre le pixel gauche et le pixel de droite concerné.

Dans un deuxième temps on effectue une recherche du meilleur score parmi les scores issus des pixels candidats.

Pour réaliser ces opérations, le flot des pixels de l'image droite traverse une chaîne de registres à décalage (D_{max} registres) préalablement initialisée à 0 (figure 4-17). Les scores sont calculés en appliquant simultanément la fonction XNOR entre le pixel gauche motif et les sorties de la chaîne de registres. On obtient les scores en comptabilisant, à partir de fonctions combinatoires et pour chaque pixel candidat, le nombre de comparaisons réussies lors de cette opération. Les D_{max} scores obtenus

sont ensuite appliqués à un bloc de tri fonctionnant à la fréquence pixel et dont le rôle est d'extraire le meilleur score et son rang.

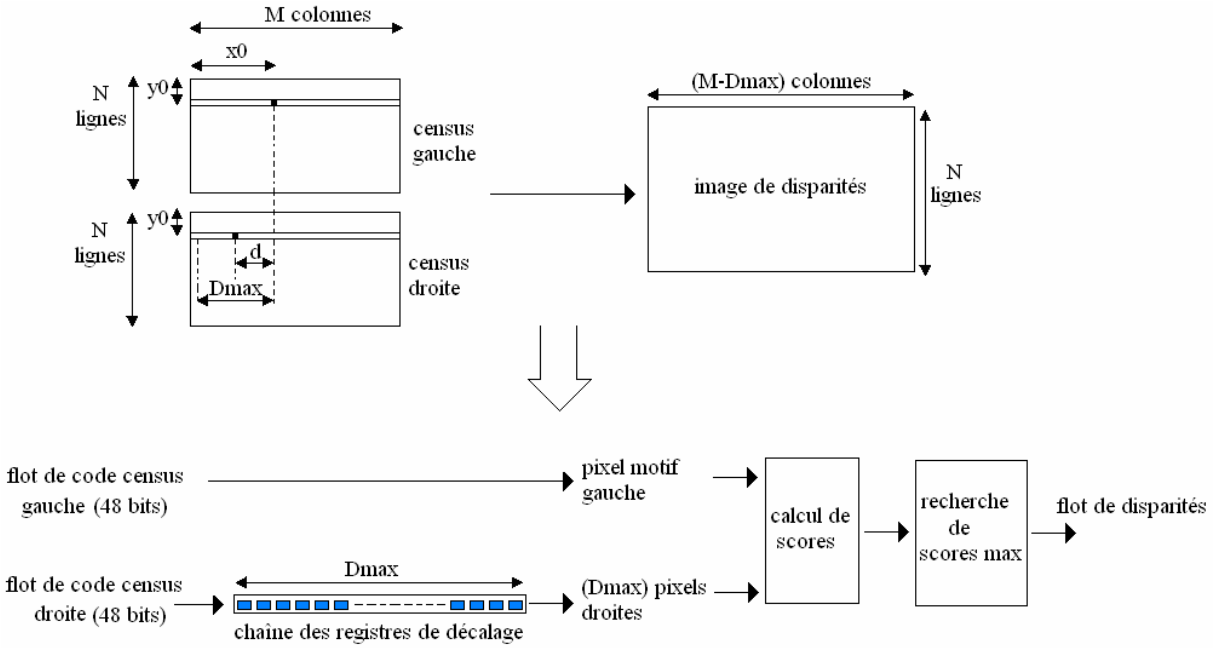


Figure 4-17 : Etapes pour la recherche des disparités.

4.4.2.5 Conclusions

Dans cette partie nous avons évalué la faisabilité d'un traitement temps réel à la cadence de l'horloge pixel. Celui-ci ne pourra être obtenu qu'à partir d'architectures massivement parallèles comme il est possible d'en implanter dans les circuits ASIC ou FPGA et à la condition d'utiliser, concernant les circuits FPGA, des composants dont la granularité est adaptée. Notons toutefois que l'obtention des images de disparité à ce rythme ne pourra se faire qu'avec un retard pur lié à :

- la distorsion apportée par le dispositif de prise de vue (DistoMax).
- la taille de la fenêtre de moyennage (3x3 dans notre cas).
- la taille de la fenêtre Census (7x7).

Ce retard pourra par ailleurs se trouver fortement augmenté en fonction des stratégies de filtrage d'image choisies.

4.5 Les circuits FPGA (Field Programmable Gate Array)

4.5.1 Historique

Le principe de la logique programmable remonte au début des années 1960, le concept ayant été proposé par Estrin [Estrin_63]. Mais ce n'est qu'au cours des années 1980 que les premières réalisations matérielles sont introduites sur le marché. L'apparition de ce type de circuit s'est d'abord faite au travers de circuits logiques programmables simples de type PAL (Programmable Array Logic), qui se programment comme des mémoires non volatiles de type ROM et sont utilisés pour implémenter des fonctions combinatoires simples, telles des décodeurs d'adresse ou des contrôleurs de bus.

Avec les évolutions en micro-électronique, différentes familles de circuits programmables ont vu le jour : les CPLD (Complex Logic Programmable Device), puis les FPGA (Field Programmable Gate Arrays), introduits par la société Xilinx en 1985. Avec l'apparition de circuits de plus en plus performants et reprogrammables à volonté, l'industrialisation et la commercialisation de ce type de circuits s'est faite à grande échelle. À l'heure actuelle, on compte une dizaine de fabricants, le marché étant nettement dominé par les sociétés Xilinx, Altera (circuits reprogrammables) et Actel (circuits non reprogrammables). Actuellement ces fabricants mettent sur le marché des circuits de type FPGA qui offrent désormais l'équivalent de dix millions de portes logiques programmables, à des fréquences de fonctionnement atteignant les 200MHz.

4.5.2 Architecture générale d'un FPGA

L'architecture d'un FPGA se décompose en deux types de ressources (figure 4-18) :

- les ressources de traitement incluant les mémoires, la logique, et les registres. Elles sont regroupées en blocs logiques de différents types,
- les ressources d'interconnexions programmables qui relient les blocs logiques entre eux.

La programmation d'un circuit reconfigurable consiste donc à spécifier la fonctionnalité de chaque bloc logique et à organiser le réseau d'interconnexion afin de réaliser la fonction demandée.

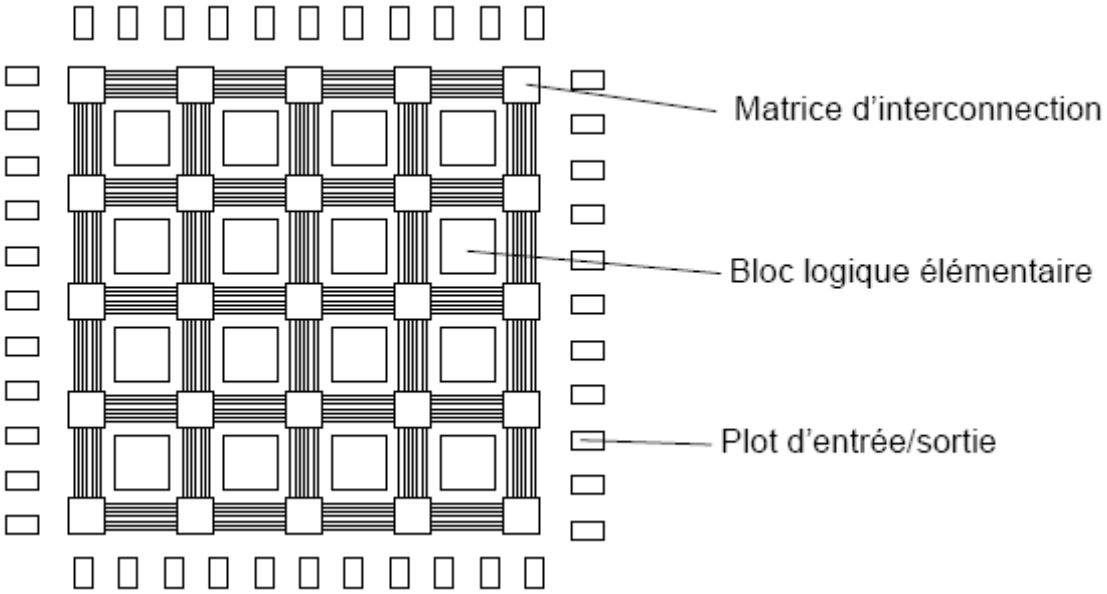


Figure 4-18 : Vue simplifiée d'un circuit programmable de type FPGA à organisation matricielle, et de ses différents composants internes.

4.5.2.1 Blocs logiques

On les distingue en général d'après leur granularité (en d'autre termes le niveau de complexité intégrable dans un seul bloc logique) et sont généralement répertoriés en trois catégories :

- Les blocs logiques à grain fin (assemblage de portes logiques et/ou d'éléments de mémorisation (registres ou bascules) [Actel_00]; (un exemple est présenté sur la figure 4-19.a),
- Les blocs logiques à grain moyen (fonctions tablées programmables (Look Up Tables), combinées à un ou plusieurs éléments de mémorisation). [Altera_00, Xilinx_98, Xilinx_94], (un exemple est présenté sur la figure 4-19.c),
- Les blocs logiques à gros grain (opérateur arithmétique et/ou logique combiné à un ensemble de registres organisé en file). Ce type d'architecture est très proche d'un chemin de données de processeur programmable [Lu_99, Marshall_99] (un exemple est présenté sur figure 4-19.b).

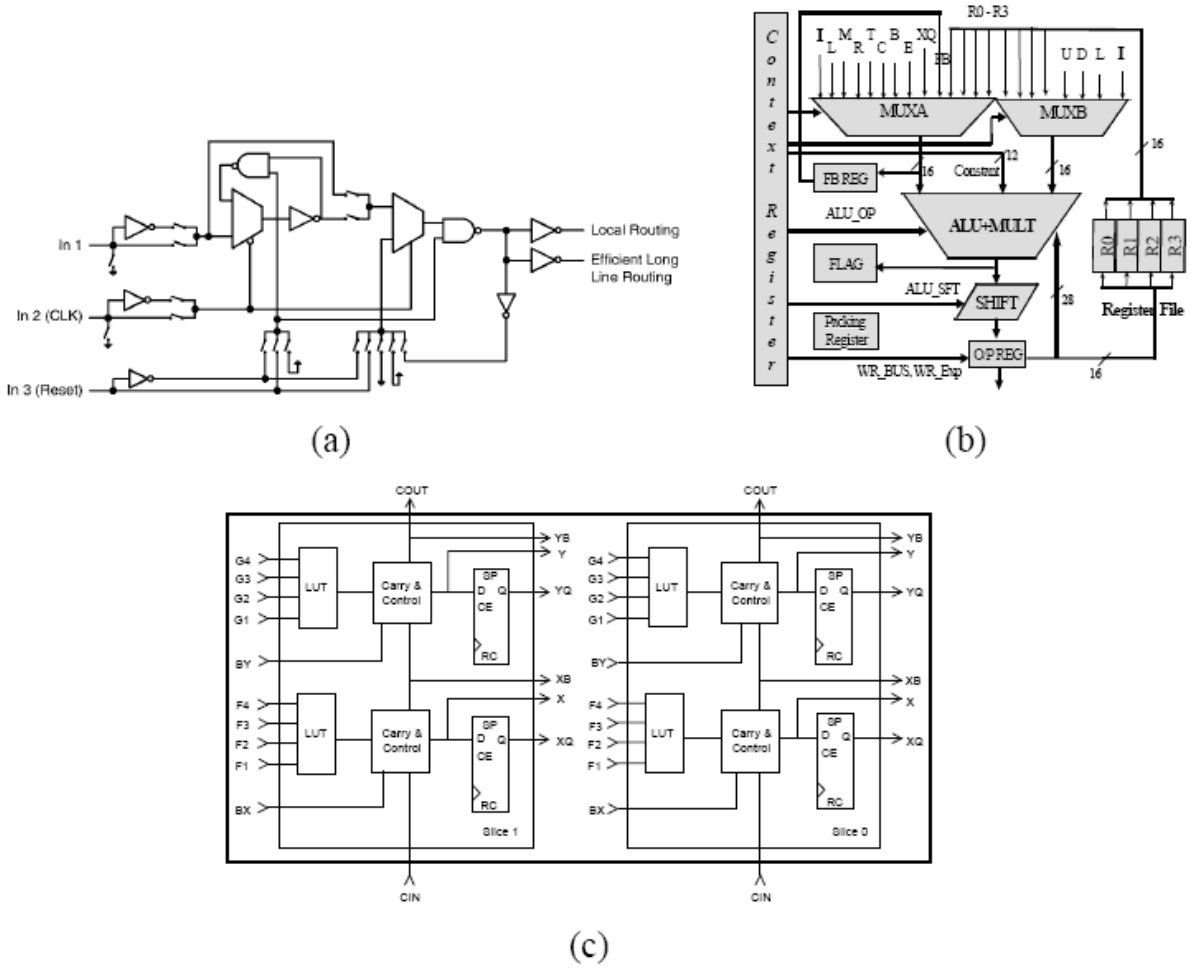


Figure 4-19 : Architectures de différents blocs logiques : (a) bloc logique à grain fin utilisé dans le circuit ProAsic de la société Actel composé de simples portes logiques ; (c) bloc logique à grain moyen du FPGA Virtex de la société Xilinx, composé de quatre look-up-tables ainsi que de ressources dédiées au traitement arithmétique (génération de retenue); (b) bloc logique du circuit Morphosys, composé d'une ALU et de registres.

Selon l'application le choix de la granularité implique un compromis entre flexibilité et performance. Une architecture à grain fin permet d'implanter une plus grande variété de circuits, puisque les blocs logiques peuvent émuler des fonctions logiques quelconques. Cependant cette versatilité se fait aux dépens des performances : un opérateur arithmétique implanté sous la forme d'une combinaison de blocs logiques à grain fin sera moins rapide, utilisera plus de ressources, et dissipera plus d'énergie qu'un bloc à gros grain. En revanche, l'implantation d'une machine à état complexe ou d'opérateurs arithmétiques inhabituels ne sera pas toujours efficace sur ces derniers. C'est pour ces différentes raisons que les circuits

programmables modernes offrent des architectures hétérogènes, pour les ressources de traitement comme pour les ressources de mémorisation.

En effet la plupart des familles de circuits récentes intègrent désormais des ressources mémoires à grain moyen (connues sous le terme de Blocs Mémoire Embarqués) qui offrent des capacités de quelque Kbits, et une interface (nombre de lignes d'adresses et largeur du chemin de données) configurable [Xilinx_98, Altera_00]. Par ailleurs, certaines architectures permettent d'utiliser les LUT présentes dans les blocs logiques comme des mémoires à grain fin (SRAM asynchrone, …). De fait, la combinaison de ces différents types de ressources permet la construction d'un système hiérarchisé de mémoires distribuées, dont l'organisation est gérée par le concepteur en fonction de l'application à implanter.

4.5.2.2 Le routage de l'application

Bien qu'une grande souplesse soit recherchée dans les possibilités de routage, celle-ci se heurte au compromis à accepter entre vitesse de propagation des signaux, consommation en énergie et surface de silicium utilisée. Aussi on distingue globalement trois techniques de routage :

- le routage programmable classique qui consiste en un ensemble de canaux séparant les blocs logiques entre eux. Ces canaux bénéficient de deux types de connexions : soit directement connectés aux blocs logiques, soit reliés entre eux au travers de matrices d'interconnexions. Cette technique offre un très bon potentiel de routage mais peut dégrader fortement les temps de propagation;
- le routage segmenté [Betz_99] pour lequel les canaux d'interconnexion sont de longueur variables et ne traversent pas nécessairement toutes les matrices d'interconnexion;
- le routage hiérarchique qui consiste en une organisation de façon à ce que les délais d'interconnexion évoluent linéairement avec le nombre de couches traversées par une connexion [DeHon_99].

4.5.3 Flot de conception

Les quatre étapes de conception classique d'une architecture de type reconfigurable peuvent se résumer à :

- la spécification et la synthèse.
- l'allocation des ressources.
- le placement et le routage.
- la génération du fichier de configuration.

4.5.3.1 Spécification et synthèse

Cette étape a pour but de fournir une description de l'architecture du circuit à implanter. Deux types de synthèse peuvent être distingués:

- La synthèse logique à partir d'une spécification structurelle qui décrit explicitement l'architecture comme un ensemble d'opérations combinatoires connectées au travers de registres,
- La synthèse comportementale, dans laquelle l'utilisateur décrit le comportement de l'architecture à un plus haut niveau d'abstraction, et où l'outil de synthèse génère une (ou plusieurs) architecture(s) satisfaisant les spécifications fournies par l'utilisateur.

On peut noter dans ce contexte l'utilisation incontournable aujourd'hui de langages de description structurels ou comportementaux tel que VHDL dont les avantages seront vus au chapitre 4.6.

4.5.3.2 Allocation des ressources

L'objectif de cette étape est d'associer à chaque primitive logique obtenue après synthèse, un bloc logique du circuit programmable.

Dans la plupart des cas, l'allocation consiste à trouver, en fonction des contraintes fournies par l'utilisateur, le meilleur compromis entre les critères de vitesse et de surface en utilisant des techniques de réplication et/ou fusion des primitives logiques.

Placement et routage :

Ces deux étapes prennent en charge le placement physique des blocs logiques puis la programmation du réseau d'interconnexion.

Le compromis est ici entre localité et congestion : afin de permettre des interconnexions plus courtes, il est préférable de placer les blocs reliés entre eux, proches les uns des autres, afin de minimiser le nombre de matrices d'interconnexion traversées. Toutefois, le placement d'un nombre élevé de blocs logiques fortement interconnectés, sur un espace limité, peut rapidement poser des problèmes de congestion. De fait, le réseau d'interconnexion n'est plus capable de router toutes les connexions efficacement, et les délais s'en trouvent dégradés.

4.5.3.3 Génération du fichier de configuration

Une fois que la description complète de l'implémentation est obtenue, un fichier de configuration contenant l'ensemble des informations relatives à l'implémentation, est généré. Celui-ci peut alors être utilisé pour la configuration du circuit programmable, soit à l'aide d'une mémoire non-volatile associée au circuit (E²PROM série ou parallèle), soit directement à partir d'une interface externe (processeur, bus, liaison parallèle, etc.)

4.5.4 Conclusions

La technologie logique programmable se positionne comme une solution intermédiaire entre processeurs programmables et circuits VLSI dédiés.

En regard de l'application « stéréovision temps réel » une évaluation des contraintes nous conduit d'une part à retenir le principe du circuit programmable, le langage de description VHDL et d'autre part à utiliser des composants disposants de ressources hétérogènes :

- granularité fine pour les nombreuses fonctions logiques nécessaires.
- possibilités de routage sans dégrader les temps de propagation.

- mémoire embarquée conséquente organisable en nombreux blocs de taille variable et capables d'être accédés simultanément pour les stockages intermédiaires.
- ressources de calcul pour la reconstruction 3D.
- processeur et périphériques embarquables pour la partie communication avec le monde extérieur.

4.6 Le langage VHDL

Développé sous l'impulsion du DoD (Department of Defense) dans les années 80 aux États-Unis, le langage de description VHDL (Very high speed integrated circuit Hardware Description Language) est devenu une norme IEEE sous la référence 1076 en 1987. Révisée en 1993 pour supprimer quelques ambiguïtés et améliorer la portabilité du langage, cette norme est vite devenue un standard en matière d'outils de description de fonctions logiques [Vhdl_98]. A ce jour, on utilise le langage VHDL pour :

- Concevoir des ASIC,
- Programmer des composants programmables du type PLD, CPLD et FPGA,
- Concevoir des modèles de simulations numériques ou des bancs de test.

4.6.1 Un langage de description

L'électronicien a toujours utilisé des outils de description pour représenter des structures logiques ou analogiques. Le schéma structurel que l'on utilise depuis si longtemps et si souvent n'est en fait qu'un outil de description graphique. Aujourd'hui, l'électronique numérique est de plus en plus présente et tend bien souvent à remplacer les structures analogiques utilisées jusqu'à présent.

Ainsi, l'ampleur des fonctions numériques à réaliser nous impose l'utilisation d'un autre outil de description. Il est en effet plus aisé de décrire un compteur ou un additionneur 64 bits en utilisant l'outil de description VHDL plutôt qu'un schéma. Le deuxième point fort du VHDL est d'être "un langage de description de haut niveau".

En fait, un langage est dit de haut niveau lorsqu'il fait le plus possible abstraction de l'objet auquel ou pour lequel il est écrit. Dans le cas du langage VHDL, il n'est jamais fait référence au composant ou à la structure pour lesquels on l'utilise. Ainsi, il apparaît deux notions très importantes :

- Portabilité des descriptions VHDL : c'est-à-dire, possibilité de cibler une description VHDL dans le composant ou la structure que l'on souhaite en utilisant l'outil que l'on veut en VHDL.
- Conception de haut niveau : c'est-à-dire qui ne suit plus la démarche descendante habituelle (du cahier des charges jusqu'à la réalisation et le calcul des structures finales) mais qui se “limite” à une description comportementale directement issue des spécifications techniques du produit que l'on souhaite obtenir.

4.6.2 Les limites actuelles

La norme qui définit la syntaxe et les possibilités offertes par le langage de description VHDL est très ouverte. Il est donc possible de créer une description VHDL de systèmes numériques non réalisable, tout au moins, dans l'état actuel des choses. Il est par exemple possible de spécifier les temps de propagations et de transitions des signaux d'une fonction logique, c'est à - dire créer une description VHDL du système que l'on souhaite obtenir en imposant des temps précis de propagation et de transition. Or les outils actuels de synthèses logiques sont incapables de réaliser une fonction avec de telles contraintes. Seuls des modèles théoriques de simulations peuvent être créés en utilisant toutes les possibilités du langage. La situation peut donc se résumer de la façon suivante (voir figure 4-20) :

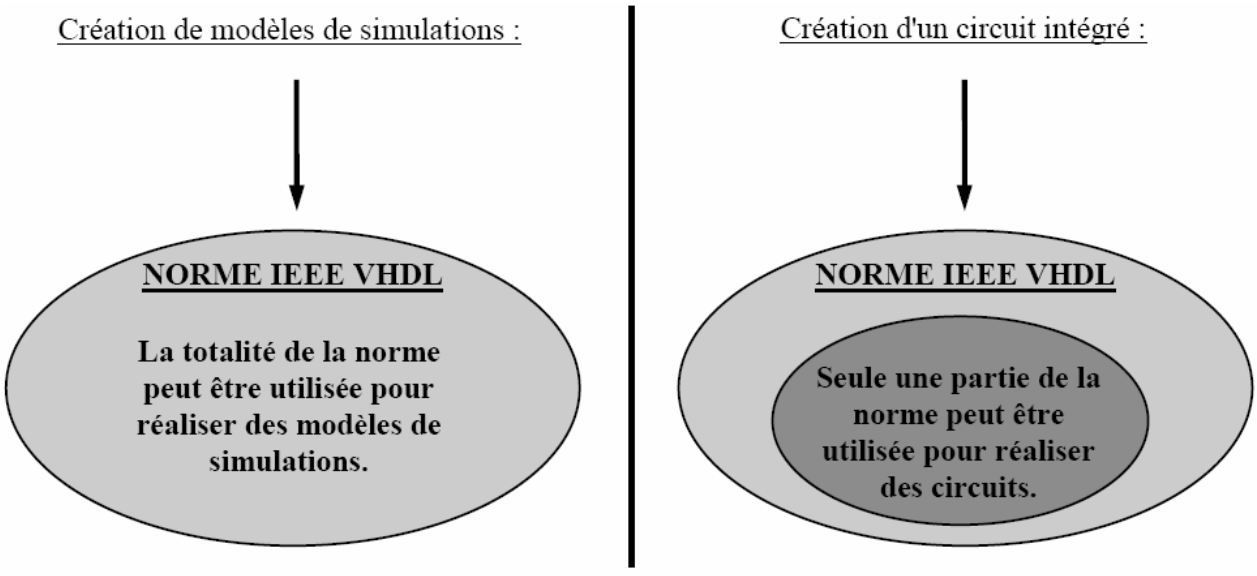


Figure 4-20 : les normes IEEE VHDL.

4.6.3 Structure d'une description VHDL

En VHDL, une structure logique est décrite à l'aide d'une entité et d'une architecture de la façon suivante :

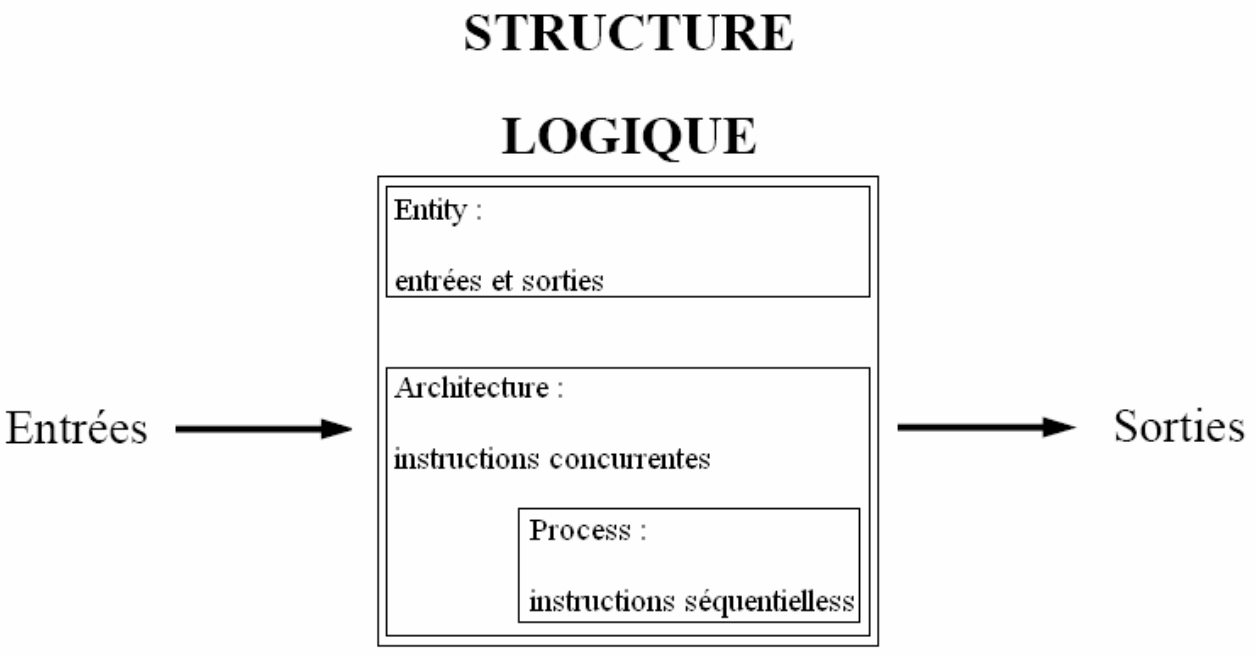


Figure 4-21 : la structure du code VHDL.

L'entité donne les informations concernant les signaux d'entrées et de sorties de la structure ainsi que leurs noms, leurs types et leurs directions (IN, OUT et

INOUT). Dans l'architecture, on décrit la structure logique en utilisant des instructions concurrentes pour les rapports directs entre les entrées et les sorties et des instructions séquentielles présentées par un processus qui sera traduit en tableau de vérité.

- L'architecture décrit le comportement de l'entité.
- Il est possible de créer plusieurs architectures pour une même entité. Chacune de ces architectures décrira l'entité de façon différente.

4.6.4 Description comportementale et structurelle

Il existe deux façons distinctes de décrire une structure. L'une dite « comportementale » et l'autre « structurelle ». Pour la description comportementale, le comportement de la structure est directement inscrit dans l'architecture à l'aide d'instructions séquentielles ou sous forme de flot de données. Alors que pour la description structurelle, on relie entre eux des composants préalablement décrits en VHDL.

Dans le langage VHDL, il est possible de créer des "package" et "package body" dont les rôles sont de permettre le regroupement de données, variables, fonctions, procédures, etc., que l'on souhaite pouvoir utiliser ou appeler à partir d'architectures. Il est ainsi possible de se créer une sorte de bibliothèque d'objets (constantes, variables particulières, fonctions ou procédures spécifiques, etc.) que l'on pourra appeler à partir de plusieurs descriptions. Cette possibilité est particulièrement intéressante lorsque l'on utilise, au sein d'un projet contenant plusieurs entités, des ressources (constantes, variables particulières, fonctions...) identiques.

4.6.5 La création de modèles paramétrables

C'est un aspect important à porter à l'avantage du langage : il permet le développement de modèles plus facilement capitalisables, favorise le « re-use » et permet une meilleure évolutivité des architectures décrites.

Exemple : (registre à décalage de « width » bits)

```
ENTITY reg_dec is
generic (width : integer :=20);
        port    (
                        clk, data, reset : in std_logic;
                        qout : out std_logic_vector ((width-1) downto 0);
                        );
end reg_dec;
```

4.6.6 Conclusion

Parce qu'on peut choisir entre description structurelle ou comportementale, créer des modèles génériques et effectuer en une fraction de temps la synthèse d'un circuit, le langage VHDL s'avère le mieux adapté pour la description des architectures dédiées à la stéréovision. Notons toutefois que le degré de paramétrisation reste limité à cause de l'impossibilité de décrire en VHDL certains blocs (notamment mémoires câblées en dur dans le composant).

4.7 Implémentation des différentes étapes de la stéréovision

4.7.1 La rectification des images droite et gauche

A cause des distorsions causées par les optiques des deux caméras ainsi que par les incertitudes de leur positionnement géométrique, une étape de calibration et de rectification est indispensable pour corriger ces phénomènes. La figure 4-22 illustre le principe de la rectification.

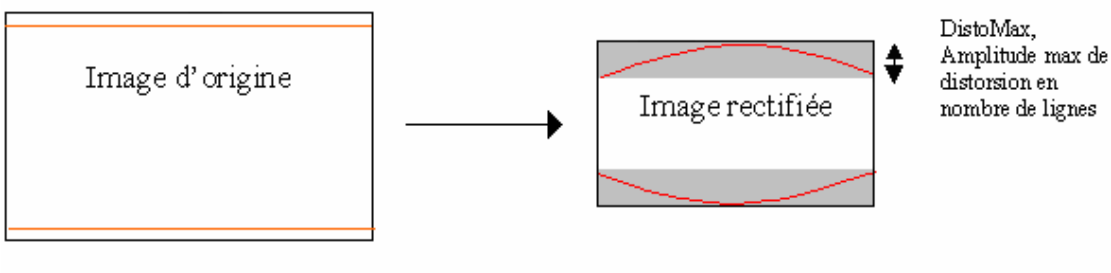


Figure 4-22 : Principe de la rectification.

Pour réaliser la rectification nous avons proposé deux méthodes : la première considère les nouvelles positions (adresses absolues) des pixels stockées dans des mémoires de correction et la deuxième ne prend en compte que les offsets (variation par rapport à l'adresse courante) des coordonnées des pixels.

Méthode 1:

- les ressources nécessaires

Cette méthode nécessite deux mémoires contenant les éléments de correction issus de la calibration des images droite et gauche et deux buffers circulaires de stockage intermédiaire. La taille de ces buffers dépend du niveau de distorsion apporté par les deux optiques : si l'on appelle DistoMax l'amplitude maximale de la distorsion en nombre de lignes (Ad_{max}) et W le nombre de pixels par ligne (la largeur de l'image traitée), alors la taille de chaque buffer devra être Ad_{max} x W mots. Les mémoires de correction, de la taille des images à traiter, contiennent les coordonnées U et V des pixels qui composeront l'image rectifiée.

- Principe de la rectification en temps réel

Les pixels reçus au rythme de l'horloge pixel (40 MHZ) sont stockés dans le buffer circulaire à des adresses définies par la mémoire de correction. Ils occupent ainsi la place qu'ils occuperaient si le système de prise de vue n'était pas affecté de distorsions. A la fin de son remplissage la première ligne du buffer est simultanément lue linéairement fournissant ainsi la première ligne de l'image rectifiée et écrite par la ligne correspondant à un multiple de Ad_{max}. Cette opération de lecture/écriture est en fait effectuée avec un décalage d'un pixel pour ne pas avoir à gérer les conflits d'accès à une même ressource. On peut donc considérer que la phase de rectification entraine un retard pur de valeur :

$$Ad_{max} \; x \;\; W \, x \, Tpixel$$

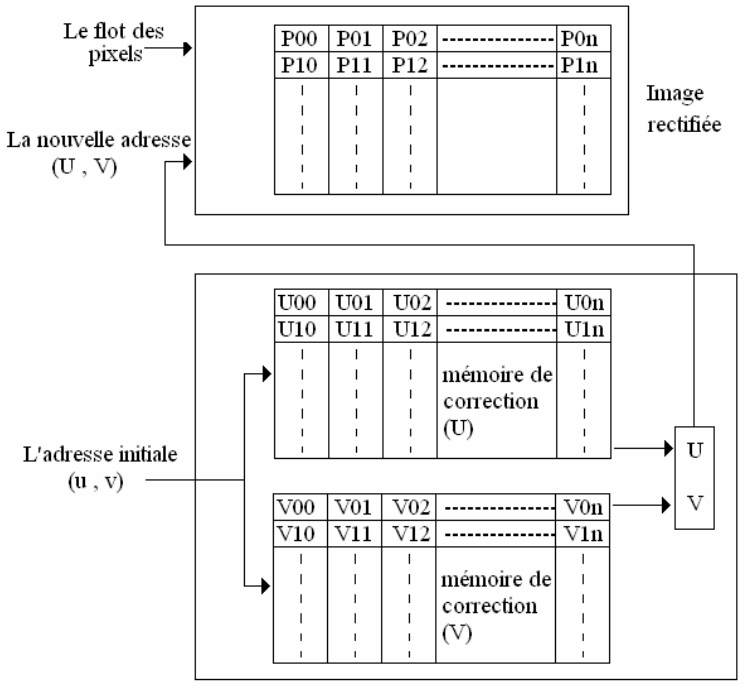


Figure 4-23 : Synoptique de la première méthode de rectification en temps réel.

Méthode 2:

- ressources nécessaires

Elle nécessite également l'utilisation de deux buffers circulaires de taille similaire à ceux utilisés dans la méthode 1. Concernant les mémoires de correction, celles-ci s'avèrent de capacités réduites du fait de la diminution de la taille des mots qui y sont stockés. En effet dans cette méthode seules sont stockées les variations de position d'un pixel par rapport à sa position courante. Il y a donc nécessité d'introduire un bloc de calcul pour déterminer la valeur de l'adresse corrigée.

- Principe de la rectification

Elle est similaire à la méthode précédente sauf que les adresses corrigées sont déterminées par le bloc de calcul à partir des adresses courantes augmentées ou diminuées des facteurs stockés dans les mémoires de correction.

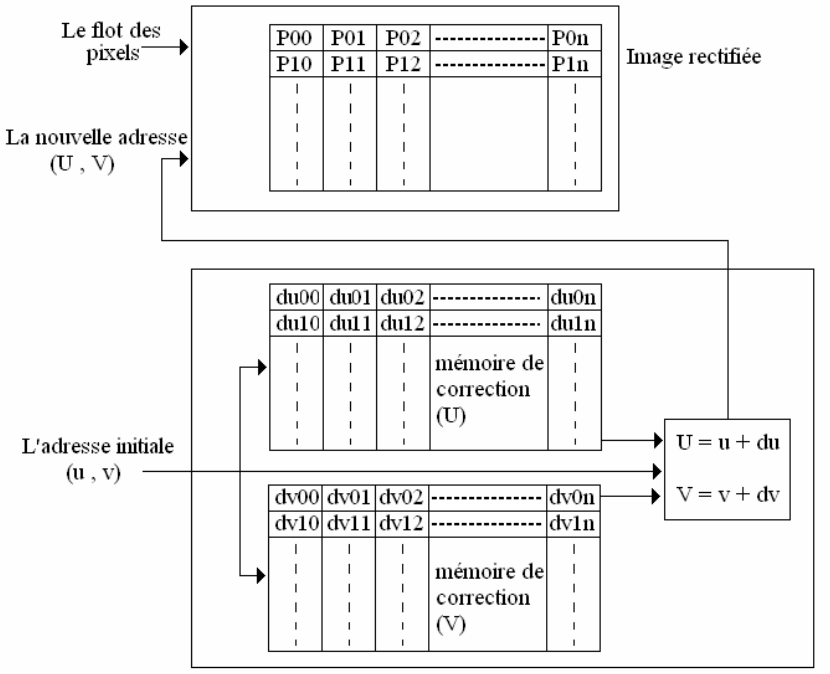


Figure 4-24 : Synoptique de la deuxième méthode de rectification en temps réel

4.7.2 La phase de moyennage

Le calcul des moyennes se fait à partir d'une fenêtre de neuf pixels comme représenté dans la figure 4-25 (trois lignes et trois colonnes) et a pour but de générer un léger flou dans l'image, lequel améliore l'opération de sub-pixelisation.

Ce calcul diminue légèrement la taille de l'image traitée puisque les deux lignes et les deux colonnes extrêmes, bien qu'elles puissent être conservées, sont supprimées pour des raisons de simplification de l'architecture de traitement. Par ailleurs pour éviter de traiter des nombres flottants, plus complexes à manipuler dans des architectures matérielles, nous avons choisi de travailler sur des nombres images des sommes des points plutôt que sur leur moyenne. Ainsi, pour un signal vidéo codé sur huit bits, le codage des moyennes s'effectuera sur 12 bits.

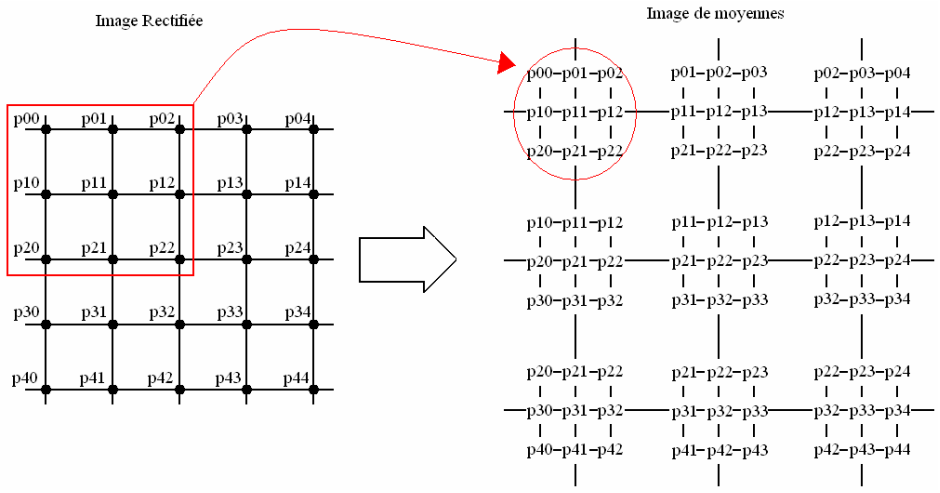


Figure 4-25 : principe du calcul des moyennes

Pour réaliser le calcul des moyennes en temps réel à partir d'un flot de pixels issus de l'étape de rectification, deux méthodes sont possibles.

- Soit on effectue le calcul en deux étapes d'addition et une étape de stockage intermédiaire: ainsi on calcule la somme horizontale de trois pixels successifs puis la somme verticale à partir des pixels adjacents comme représenté dans la figure 4-26. Cette méthode nécessite une mémoire de stockage intermédiaire et deux additionneurs parallèles. La mémoire est composée de quatre lignes et la taille de chaque ligne est celle de la largeur de l'image. Les deux additionneurs effectuent les opérations en parallèle sur des séries de trois pixels.

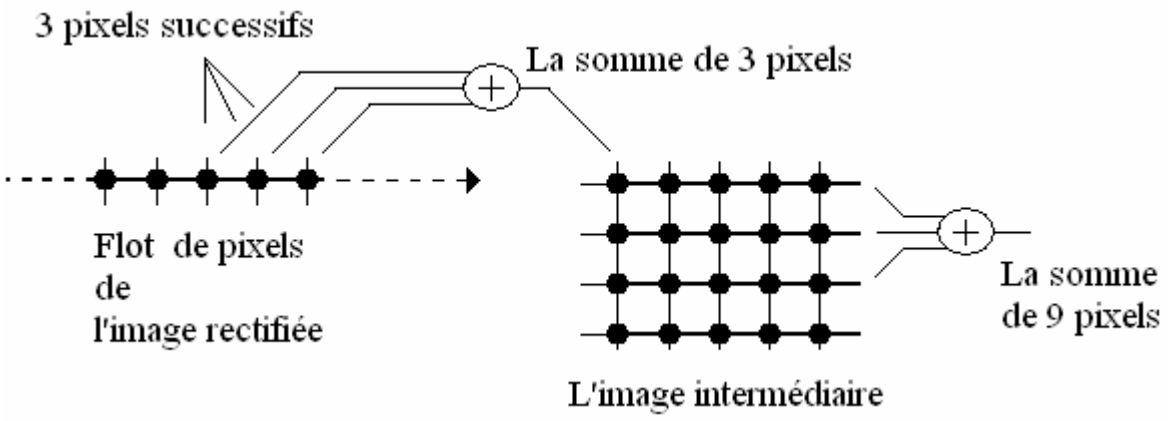


Figure 4-26 : Principe de la première méthode.

- Soit on calcule la moyenne à partir d'une seule phase d'addition, d'une phase de stockage intermédiaire et d'une phase de décalage comme représenté dans la

figure 4-27. Cette méthode nécessite donc neuf registres à décalage, une mémoire de stockage intermédiaire et un additionneur parallèle à neuf entrées. La mémoire est composée de quatre lignes et la taille de chaque ligne est celle de la largeur de l'image. Le bloc de registres se compose de neuf registres à décalage câblés de telle sorte qu'ils aient en mémoire les pixels de la fenêtre de calcul courante. A chaque top de l'horloge pixel une nouvelle fenêtre est chargée dans le bloc et l'additionneur fournit un résultat image de la valeur moyenne de la fenêtre.

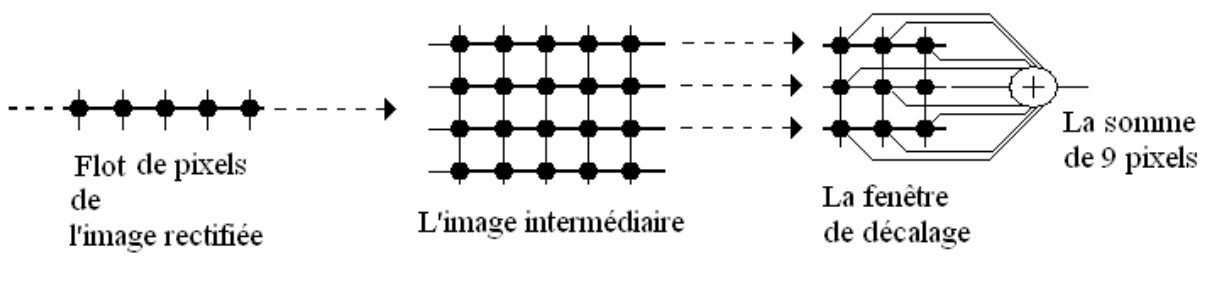


Figure 4-27 : Principe de la deuxième méthode.

Nous avons adopté pour l'architecture retenue, la première méthode qui consomme d'une part moins de ressources et présente d'autre part un temps de calcul inférieur.

Celle-ci se présente en général comme illustrée en figure 4-28. Le calcul est réalisé en trois étapes principales: deux étapes d'addition et une étape de stockage intermédiaire.

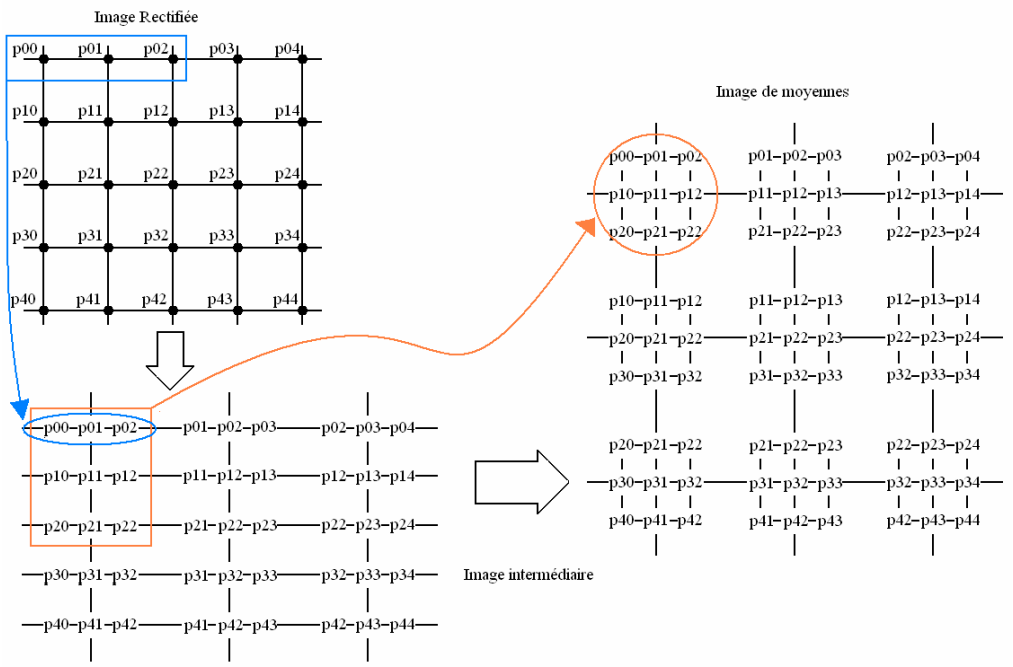


Figure 4-28 : Principe détaillé de la méthode adoptée.

A chaque top de l'horloge pixel (pclk), une chaîne de trois registres à décalage de profondeur 8 bits (pour une vidéo codée sur huit bits) mémorise l'amplitude de trois pixels successifs appartenant à la même ligne tel que représenté sur la figure ci-dessous.

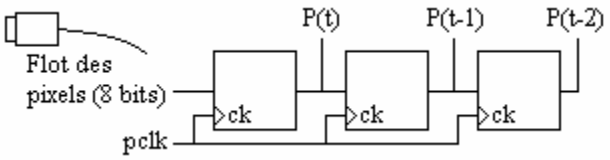


Figure 4-29 : stockage temporaire de trois pixels successifs.

Les trois valeurs successives sont simultanément additionnées pour fournir une image de la moyenne de ces trois amplitudes.

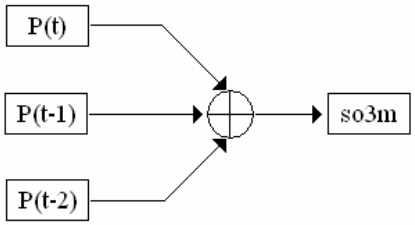


Figure 4-30 : Somme simultanée de trois pixels successifs

Ce résultat est stocké ensuite dans une mémoire intermédiaire qui se compose de trois lignes dont chacune a la taille de la largeur de l'image. L'étape de moyennage sur la fenêtre 3x3 comporte alors quatre phases de fonctionnement assurées par deux blocs de multiplexage placés respectivement à l'entrée et à la sortie de la mémoire.

Ces quatre phases assurent le rangement de toutes les trois lignes successives pour passer à l'étape suivante et sont présentées dans la figure qui suit.

Phase 1: la somme de trois pixels successifs est stockée dans la première ligne et simultanément les contenus des autres lignes sont dirigés vers les trois sorties.

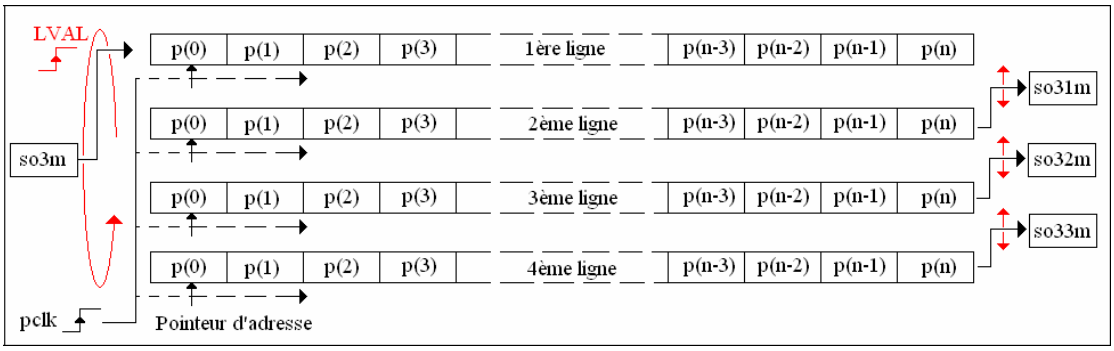


Figure 4-31 : système de stockage (première étape).

Phase 2: La somme de trois pixels est stockée dans la deuxième ligne et les contenus des autres lignes sont dirigés vers les trois sorties.

Phase 3: La somme de trois pixels est stockée dans la troisième ligne et les contenus des autres lignes sont dirigés vers les trois sorties.

Phase 4: La somme de trois pixels est stockée dans la quatrième ligne et les contenus des autres lignes sont dirigés vers les trois sorties.

A la sortie de l'étape de stockage intermédiaire on a trois flots de données représentant les trois lignes successives de la somme de trois pixels telle que calculée dans la première étape.

Ces trois sorties sont simultanément additionnées pour former la somme de neuf pixels correspondant à la fenêtre 3x3 considérée et au pixel associé dans l'image des moyennes.

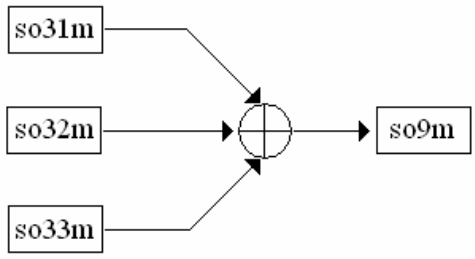


Figure 4-32 : Somme des neuf pixels

4.7.3 Le calcul de l'image Census

Cette étape s'appuie sur le flot de pixels issus de l'image moyenne pour générer en temps réel l'image Census et comporte les phases de préparation, rangement et calcul.

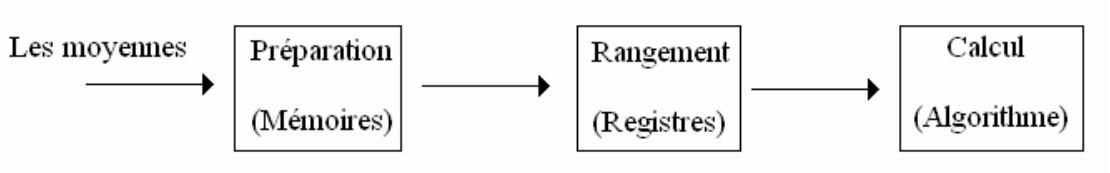


Figure 4-33 : Phases de la transformation census

La transformation Census est une suite de comparaisons entre pixels adjacents issus d'une même fenêtre et dont le choix de la taille détermine la qualité de la future image de disparités.

Dans l'architecture implémentée nous avons choisi une fenêtre de sept lignes et sept colonnes (7x7).

Pour réaliser cette transformation en temps réel, nous avons distingué trois phases principales : une phase de préparation, une phase de rangement et une phase de calcul.

La phase de préparation est une phase de stockage intermédiaire : les pixels correspondant aux fenêtres moyennées sont stockés au rythme de l'horloge pixel

dans une mémoire composée de huit lignes dont chaque ligne a la taille de la largeur de l'image traitée.

L'exploitation de cette mémoire est assurée par deux blocs de multiplexage : un multiplexeur 1 vers 8 en entrée pour sélectionner la ligne de la mémoire active en écriture et un multiplexeur 8 vers 7 en sortie pour sélectionner les sept lignes en lecture.

Les huit phases d'interconnexions sont décrites dans la figure qui suit.

Phase 1 : Les moyennes sont stockées dans la huitième ligne et les sept premières sont mises en lecture.

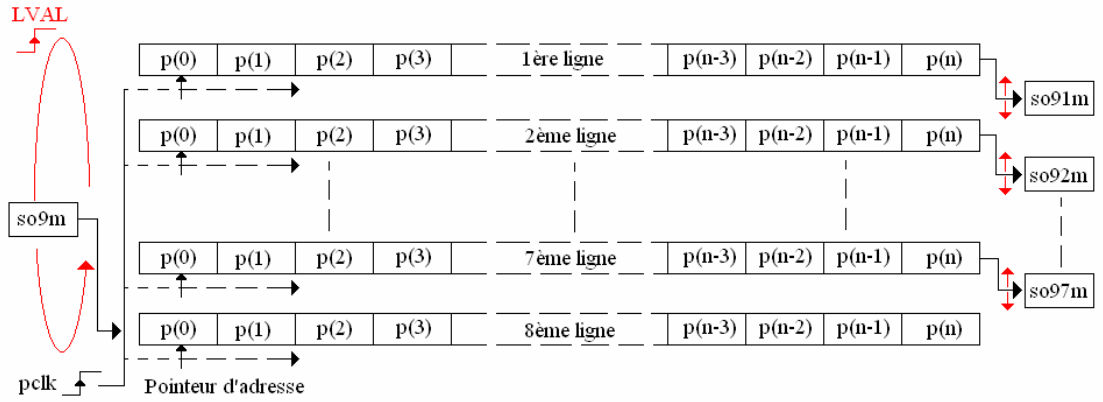


Figure 4-34 : système de stockage (deuxième étape).

Phase 2 : Les moyennes sont stockées dans la première ligne et les sept autres sont mises en lecture.

Phase 3 : Les moyennes sont stockées dans la première ligne et les sept autres sont mises en lecture.

Phase 4 : Les moyennes sont stockées dans la troisième ligne et les sept autres sont mises en lecture.

Phase 5 : Les moyennes sont stockées dans la quatrième ligne et les sept autres sont mises en lecture.

Phase 6 : Les moyennes sont stockées dans la cinquième ligne et les sept autres sont mises en lecture.

Phase 7 : Les moyennes sont stockées dans la sixième ligne et les sept autres sont mises en lecture.

Phase 8 : Les moyennes sont stockées dans la septième ligne et les sept autres sont mises en lecture.

Les pixels issus de la mémoire de stockage intermédiaire lors des différentes phases de lecture sont classés dans un bloc constitué de quarante neuf registres à décalage. Ce bloc, correspondant à une sous-image de taille 7x7, servira de base pour le calcul de la transformée de Census associée.

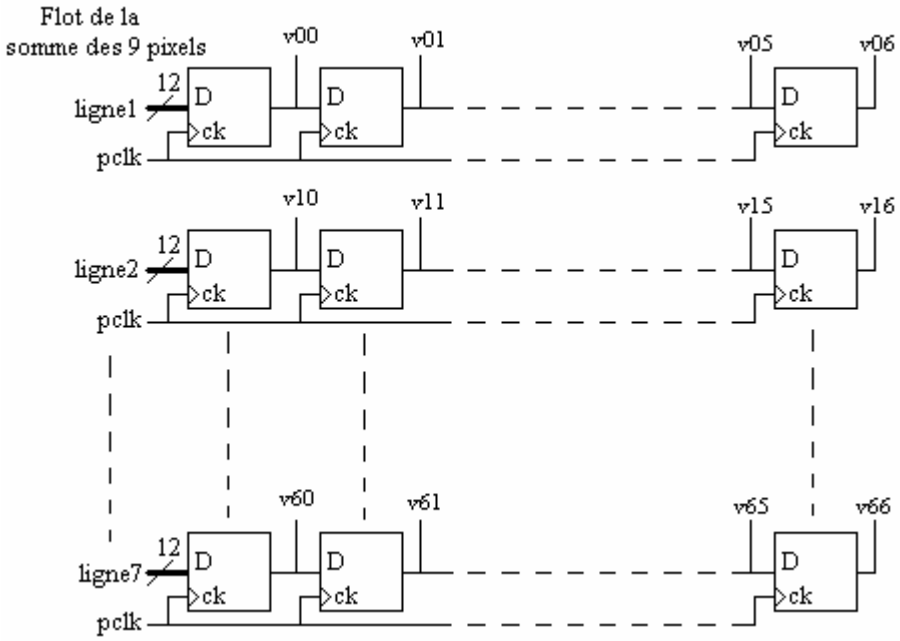


Figure 4-35 : Implémentation matérielle d'une fenêtre de Census 7 x 7.

La troisième étape correspond au calcul de la transformée de Census de la sous-image mémorisée dans l'étape précédente. Dans cette partie l'amplitude du pixel central de la fenêtre est comparée à celle de ses voisins (48 au total). Cette opération génère une chaine de 48 bits ayant les valeurs « 0 » ou « 1 » selon que l'amplitude du pixel central est respectivement inférieure ou supérieure à celle des voisins. Cette chaîne de 48 bits peut-être considérée comme une signature propre à

une région de l'image traitée. L'image Census peut donc à son tour être considérée comme un ensemble de signatures caractéristiques des différentes régions qui composent l'image d'origine.

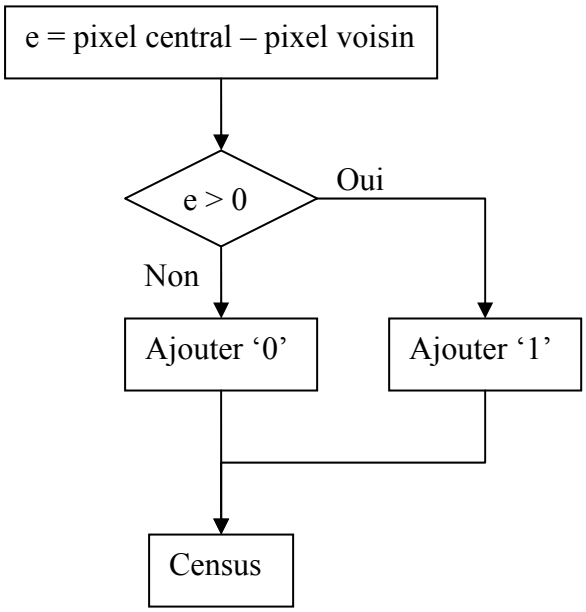


Figure 4-36 : Algorigramme de calcul du Census.

4.7.4 La corrélation Census

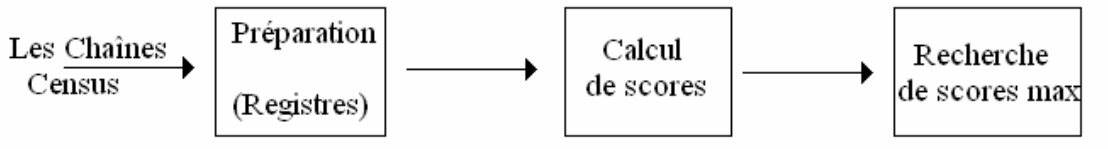


Figure 4-37 : Etapes de la corrélation census

La corrélation Census est le cœur de l'algorithme et le nœud le plus complexe dans l'architecture pour la stéréovision temps réel. L'objectif de cette étape est :

- Le calcul des scores associés aux comparaisons entre une région de référence dans l'image gauche et des régions candidates de l'image droite.
- La recherche, pour chaque point de l'image, du meilleur score correspondant au meilleur appariement.
- L'établissement de la carte des disparités à partir de laquelle pourra être reconstruite la scène tridimensionnelle.

Ce calcul comprend trois blocs : le premier effectue une étape de préparation, le deuxième, le calcul des scores et le troisième réalise la recherche du meilleur score.

Dans l'étape précédente nous avons choisi d'utiliser une fenêtre Census de taille 7x7, qui représente un bon compromis vu la texture des images traitées. Ce choix a pour conséquences de générer dans la phase de calcul de l'image Census des chaînes de quarante huit bits (7^2 -1) au rythme de l'horloge pixel. Ces chaînes sont mémorisées dans le bloc de préparation qui se compose de registres à décalage.

Le nombre de ces registres dépend de la disparité maximale, laquelle représente l'écart maximum entre les pixels correspondants des deux images gauche et droite de la scène.

Dans notre architecture nous avons pris en compte une disparité maximale de soixante quatre pixels qui correspondent au domaine de recherche des correspondances. Cela exige une chaîne de soixante cinq registres à décalage de 48 bits chacun dont le contenu est le Census des pixels candidats.

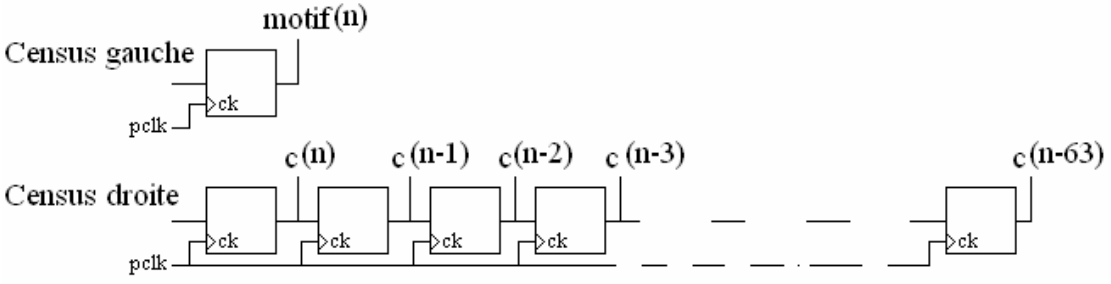


Figure 4-38 : Principe de stockage du pixel motif et des pixels candidats.

Le calcul des scores est en fait un calcul du poids de correspondance entre le census du pixel motif et le census des pixels candidats. Ce calcul se fait en appliquant d'abord la fonction logique XNOR entre les différents census (pixel motif et tous les pixels candidats) en même temps. Le degré d'appariement entre le motif et les candidats est représenté par le nombre de '1' présents dans chaque chaîne à la sortie de la fonction XNOR. Ce nombre de « 1 » est comptabilisé pour fournir un « score » ou « poids de correspondance » représentatif du niveau de ressemblance entre le motif d'origine et les motifs candidats.

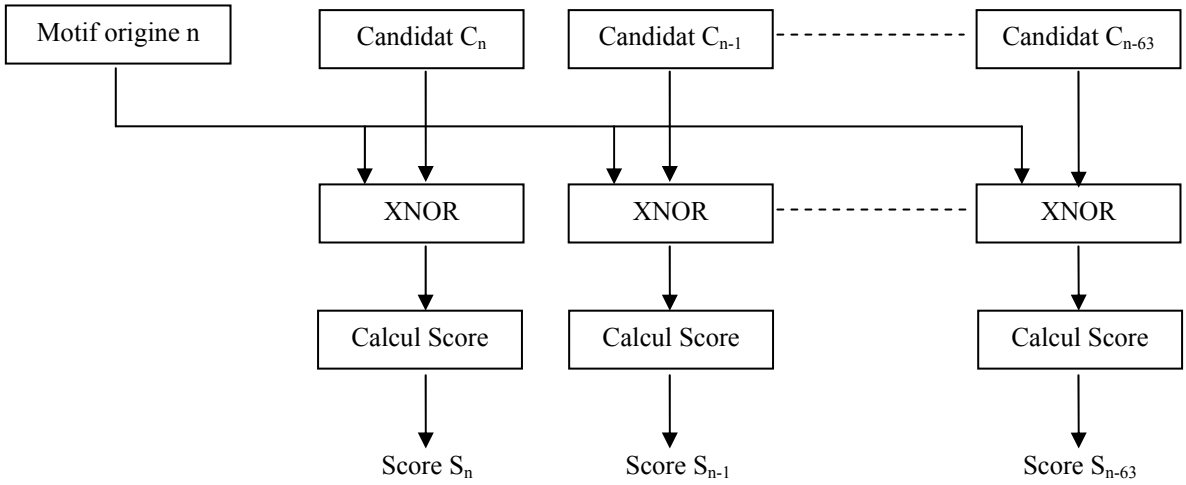


Figure 4-39 : le calcul des scores.

4.7.5 La création de l'image de disparité

Le dernier bloc est le bloc de la recherche, pour chaque pixel motif de l'image gauche, du meilleur score parmi les pixels candidats afin de dresser la carte des disparités. Pour effectuer cette recherche à la cadence de l'horloge pixel (25ns) nous avons effectué une recherche simultanée sur tous les pixels candidats (figure 4-40) et conçu une architecture pyramidale composée de plusieurs couches de comparateurs mis en cascade (figure 4-41).

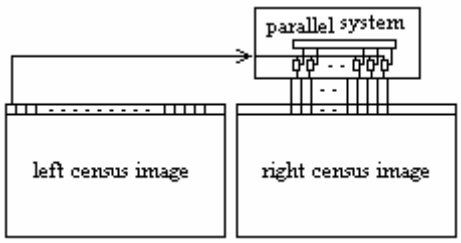


Figure 4-40 : principe de la recherche simultanée du meilleur score.

La fonction de chaque couche est de comparer un ensemble de scores deux à deux et de transmettre à la couche suivante le rang (la disparité) et le score des pixels les mieux-disant.

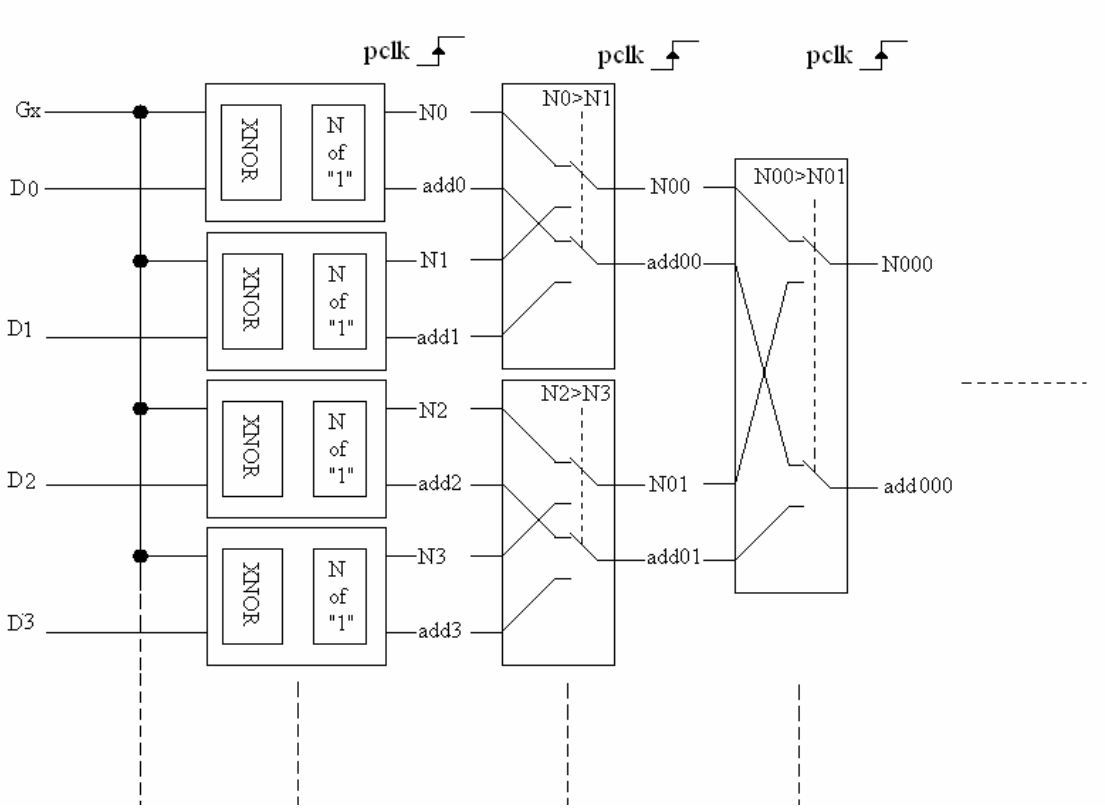


Figure 4-41 : Architecture pyramidale de recherche des meilleurs scores.

En conséquence chaque couche élimine la moitié des pixels candidats qu'elle reçoit. Pour une disparité maximale Dmax de soixante quatre pixels, nous aurons besoin d'utiliser six couches de comparaison (nombre de couches = $\log_2(D_{max})$ et le nombre de comparateurs associés à chacune est respectivement 32, 16, 8, 4, 2 et 1, ce qui fait un total de soixante trois unités de comparaison soit l'équivalent de la disparité maximale Dmax -1. A la sortie de la dernière couche nous obtenons le rang (la disparité) du pixel le mieux-disant et son score.

S'agissant des performances temps réel de cet étage, si l'on considère une implémentation purement combinatoire avec un temps moyen de propagation t_p par couche logique, le temps total T sera

$$T = tp \times \log_2(D_{max}).$$

Pour le circuit FPGA utilisé tp est de l'ordre de 10 ns, ce qui fait que pour une disparité Dmax de soixante quatre pixels cette architecture ne convient pas pour traiter des images à la fréquence pixel de 40 MHZ. Nous avons donc décomposé

l'étape de comparaison en six phases et retenu une architecture en pipe-line dans laquelle chaque couche est synchronisée avec l'horloge pixel. Un avantage est que le temps de propagation par couche est étendu à la période de l'horloge pixel, le rythme de calcul des disparités reste compatible avec la même horloge, par contre le résultat de la recherche n'est disponible qu'avec un retard pur de six tops d'horloge.

4.8 Estimation des ressources consommées

4.8.1 L'environnement d'évaluation

Pour des raisons de pré-existence d'expertise dans l'équipe de recherche, les différentes architectures de traitement ont été simulées et validées dans l'environnement Quartus II de la société ALTERA. Il s'agit d'un logiciel de conception pour circuits FPGA orienté SOPC (System On a Programmable Chip). Il inclut un ensemble d'outils correspondant à toutes les phases de conception d'architectures à base de FPGA [Quartus_03]:

- Description d'architectures hiérarchisées ou non en mode schémas, blocs ou texte.
- Synthèse VHDL et Verilog HDL.
- Outil d'aide à la génération d'architectures type « SOPC » (microcontrôleurs, DSP).
- Simulation matériel/logiciel.
- Outils de placement, routage, vérification et programmation.
- Outils d'optimisation des temps de propagation et ressources consommées.

L'architecture dédiée à la stéréovision temps réel a été décrite sous la forme de blocs fonctionnels auxquels sont associées pour partie des descriptions en langage VHDL et/ou des représentations graphiques lorsqu'il s'agit de fonctions précâblées dans le circuit FPGA (blocs mémoire par exemple). Pour cette application nous avons choisi pour composant FPGA cible le Stratix 1S40 qui a la particularité de disposer d'une granularité hétérogène, notamment de nombreux blocs mémoire. De la même

manière pour la phase d'implémentation et pour éviter les difficultés et les délais liés à la fabrication d'une carte imprimée prototype à base de ce composant, nous avons opté pour l'utilisation d'un kit de développement proposé par la société ALTERA.

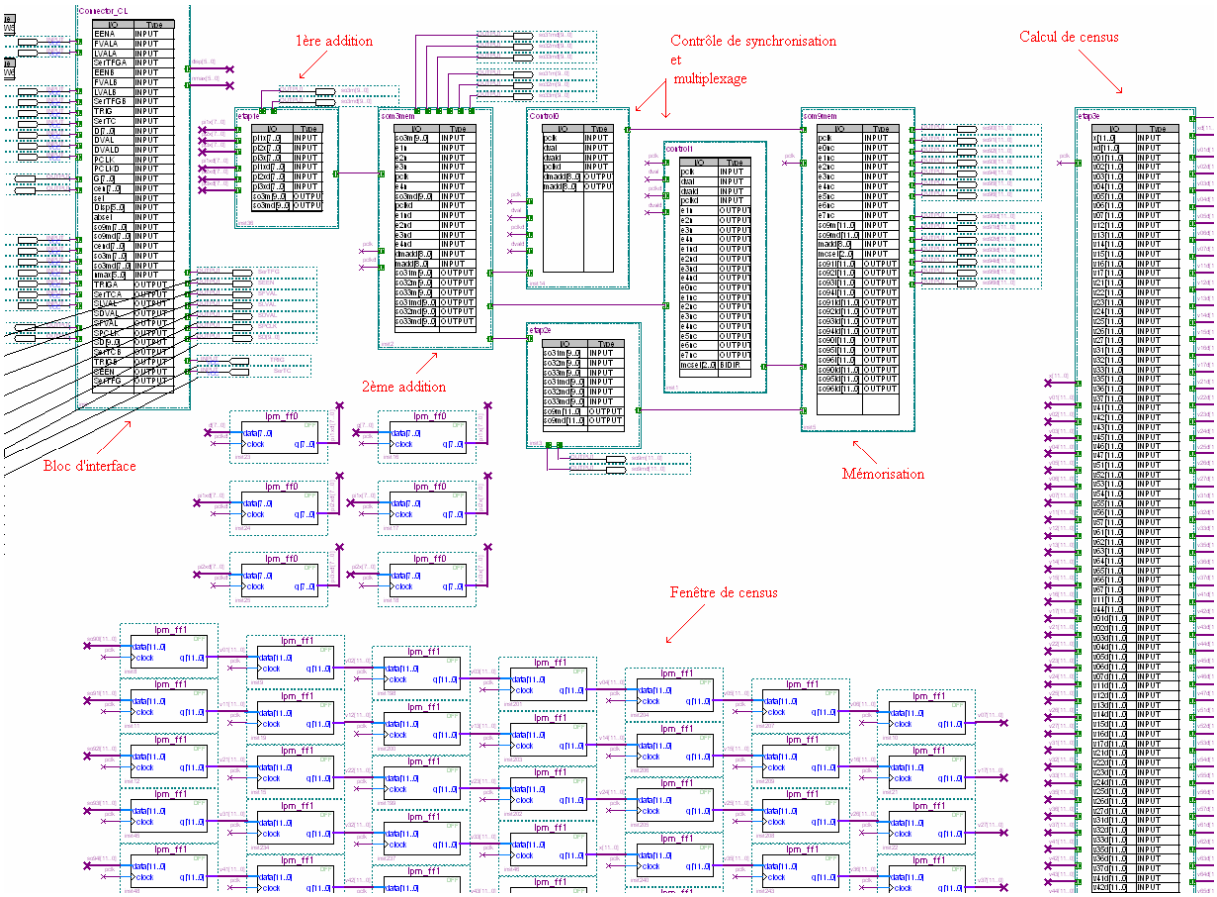


Figure 4-42 : Architecture globale de la stéréovision.

Pour évaluer les ressources en termes de mémoires et cellules logiques consommées par l'architecture de la stéréovision temps réel, nous avons synthétisé et simulé les différentes étapes de l'algorithme puis analysé les résultats de la compilation.

Les étapes prises en compte dans cette architecture sont la rectification, le moyennage, le calcul de l'image census, le calcul des scores et la recherche de la stéréo-correspondance.

4.8.2 La rectification

Cette étape fait appel à trois mémoires tampon (cf. figure 4-23). La première est réservée au stockage d'une partie de l'image rectifiée et sa taille dépend de l'amplitude de la distorsion maximale (Ad_{max}) et de la largeur de l'image. Les pixels étant codés sur huit bits, la taille de cette mémoire est :

$$m_1 \text{ (bits)} = Ad_{max} \times W \times Nb_{pix}$$

, où W la largeur de l'image traitée exprimée en nombre de colonnes et Nb_{pix} est le nombre de bits pour coder chaque pixel.

Les deuxième et troisième zones sont respectivement dédiées au stockage des coordonnées U et V associées aux pixels de l'image distordue. Ces deux zones sont initialisées avec les matrices de correction issues de la phase de calibration. La deuxième zone dépend de la largeur de l'image et sa capacité est :

$$m_2 \text{ (bits)} = W \times H \times Nb_w$$

, où W est la largeur de l'image traitée, H est son hauteur et Nb_w est le nombre de bits pour coder l'adresse u. La troisième zone dépend de la hauteur de l'image traitée et sa capacité est :

$$m_3 \text{ (bits)} = W \times H \times Nb_h,$$

, où Nb_h est le nombre de bits pour coder l'adresse v.

En combinant ces trois équations on trouve que la consommation de mémoire pour l'étape de la rectification (côtés gauche et droite), est :

$$M_0 = 2 (m_1 + m_2 + m_3)$$

$$= 2 W (Nb_{pix} \times Ad_{max} + H (Nb_w + Nb_h)) = 2 W (8 Ad_{max} + 0.75 W (E [\log_2 (W)] + 1 + E [\log_2 (H)] + 1)) = 2 W (8 Ad_{max} + 0.75 W (E [\log_2 (W)] + E [\log_2 (0.75 W)] + 2)).$$

, où E [] est la fonction de la valeur entière.

En considérant que la distorsion maximale est un quart de la hauteur de l'image traitée, les ressources de stockage consommées par la rectification sont :

$$M_0 = 2\,W\,(8 \times 0.25\,H + 0.75\,W\,(E\,[\log_2(W)] + E\,[\log_2(0.75\,W)] + 2)) = 2 \times W\,(8 \times 0.25 \times 0.75\,W + 0.75\,W\,(E\,[\log_2(W)] + E\,[\log_2(0.75\,W)] + 2)).$$

$$M_0 = W^2\,(3 + 1.5\,(E\,[\log_2(W)] + E\,[\log_2(0.75\,W)] + 2)).$$

4.8.3 Le moyennage 3 x 3

Le moyennage 3x3 nécessite une mémoire tampon permettant le stockage temporaire de trois lignes d'images successives (cf. figure 4-26). Celle-ci est constituée de trois blocs indépendants adressables simultanément. Le moyennage étant réalisé en deux étapes (moyennage ligne puis moyennage colonne), cette zone sert de stockage intermédiaire pour le moyennage ligne (stockage de la moyenne glissante sur trois pixels successifs).

La capacité exprimée en nombre de bits de la mémoire consommée dans cette étape est donnée par la relation:

$$M_1\,(bits) = 2\,(F_m + 1) \times W\,(Mod\,[F_m/2] + E\,[F_m/2] + Nb_{pix})$$

, où F_m la fenêtre de moyennage et Mod [] est la fonction qui donne le reste d'une division entière (E [5/2] = 2, Mod [5/2] = 1).

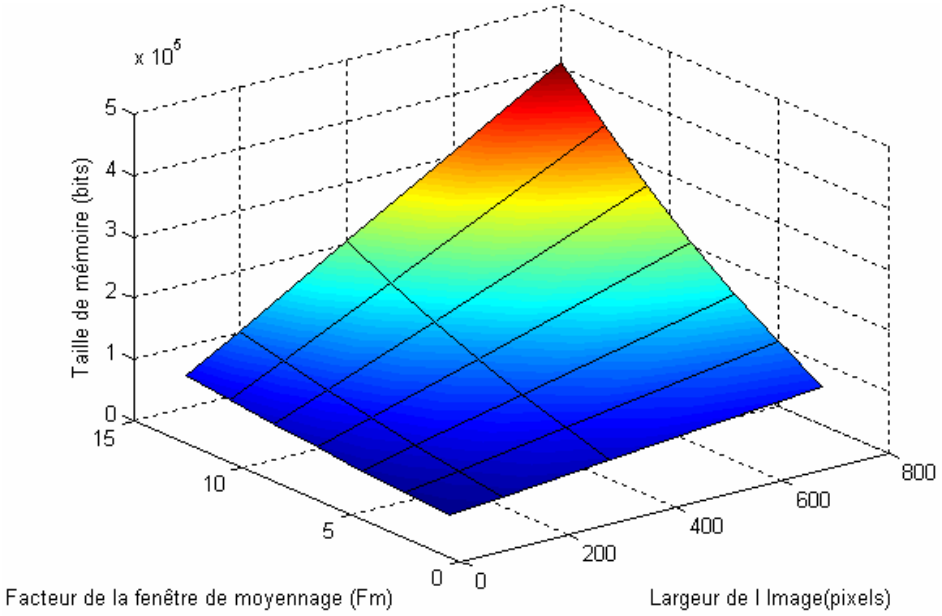


Figure 4-43 : Abaque de la mémoire consommée pour l'étape de moyennage.

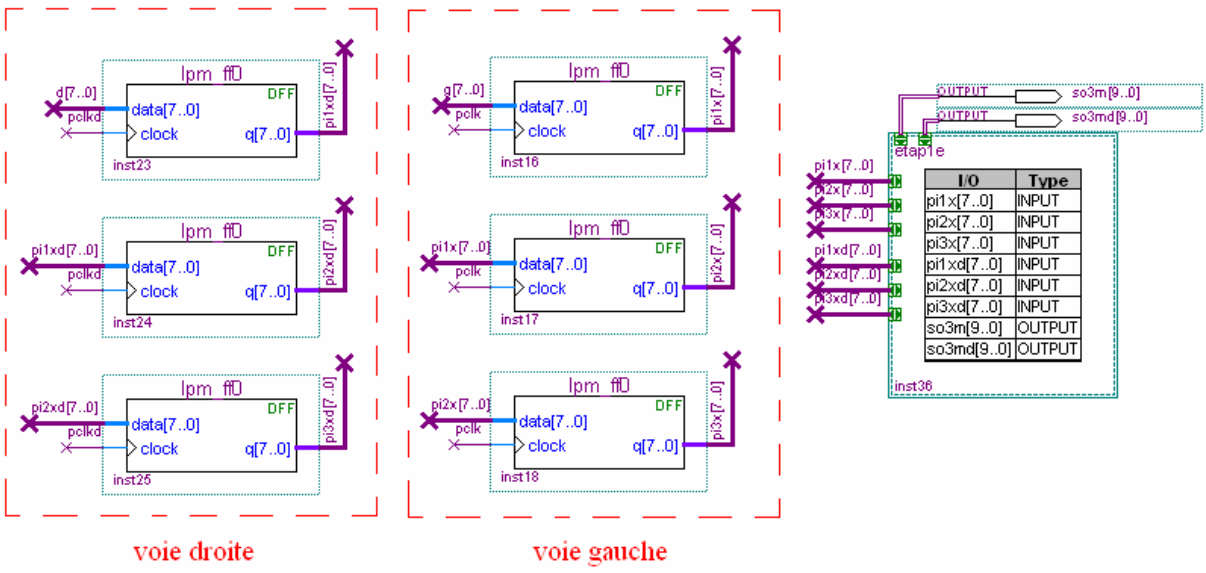


Figure 4-44 : l'architecture du moyennage (1ère étape).

En termes de ressources logiques, l'étape de moyennage nécessite :

- des registres à décalage intermédiaires pour le stockage du signal vidéo
- deux additionneurs parallèles (un pour les lignes et un pour les colonnes)
- des multiplexeurs en entrée et en sortie des blocs mémoire

Le nombre de registres à décalage dépend de la largeur de la fenêtre de moyennage. Pour une fenêtre 3 x 3, trois registres à décalage sont nécessaires. Le nombre d'éléments logiques consommés dans cette étape est :

$$n_1 = Nb_{pix} \; x \; F_m$$

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

LIBRARY altera_mf;
USE altera_mf.altera_mf_components.all;
--  Entity Declaration
ENTITY etape1e IS
    -- {{ALTERA_IO_BEGIN}} DO NOT REMOVE THIS LINE!
    PORT
    (
        pi1x : IN STD_LOGIC_VECTOR(7 downto 0);
        pi2x : IN STD_LOGIC_VECTOR(7 downto 0);
        pi3x : IN STD_LOGIC_VECTOR(7 downto 0);
        pi1xd : IN STD_LOGIC_VECTOR(7 downto 0);
        pi2xd : IN STD_LOGIC_VECTOR(7 downto 0);
        pi3xd : IN STD_LOGIC_VECTOR(7 downto 0);
        so3m : OUT STD_LOGIC_VECTOR(9 downto 0);
        so3md : OUT STD_LOGIC_VECTOR(9 downto 0)
    );
    -- {{ALTERA_IO_END}} DO NOT REMOVE THIS LINE!
END etape1e;
--  Architecture Body
ARCHITECTURE etape1e_architecture OF etape1e IS
COMPONENT parallel_add
    GENERIC (width : NATURAL; representation : STRING; size : NATURAL; msw_subtract : STRING;
    pipeline : NATURAL; result_alignment : STRING; widthr : NATURAL; shift : NATURAL);
    PORT (data : IN ALTERA_MF_LOGIC_2D (2 DOWNTO 0, 7 DOWNTO 0);
    result  : OUT STD_LOGIC_VECTOR (9 DOWNTO 0));
END COMPONENT;
SIGNAL sub_wire0,sub_wire1 : ALTERA_MF_LOGIC_2D (2 DOWNTO 0, 7 DOWNTO 0);
BEGIN
--so3m <= CONV_STD_LOGIC_VECTOR(pi1x + pi2x + pi3x, 10)when pclk = '1';
sub_wire0(0,0) <= pi1x(0);  sub_wire0(0,1) <= pi1x(1);  sub_wire0(0,2) <= pi1x(2);
sub_wire0(0,3) <= pi1x(3);  sub_wire0(0,4) <= pi1x(4);  sub_wire0(0,5) <= pi1x(5);
sub_wire0(0,6) <= pi1x(6);  sub_wire0(0,7) <= pi1x(7);
sub_wire0(1,0) <= pi2x(0);  sub_wire0(1,1) <= pi2x(1);  sub_wire0(1,2) <= pi2x(2);
sub_wire0(1,3) <= pi2x(3);  sub_wire0(1,4) <= pi2x(4);  sub_wire0(1,5) <= pi2x(5);
sub_wire0(1,6) <= pi2x(6);  sub_wire0(1,7) <= pi2x(7);
sub_wire0(2,0) <= pi3x(0);  sub_wire0(2,1) <= pi3x(1);  sub_wire0(2,2) <= pi3x(2);
sub_wire0(2,3) <= pi3x(3);  sub_wire0(2,4) <= pi3x(4);  sub_wire0(2,5) <= pi3x(5);
```

Figure 4-45 : Extrait du code VHDL du premier additionneur parallèle.

Les trois pixels successifs sont ensuite appliqués à un premier additionneur parallèle disposant de trois entrées et une sortie codées respectivement sur huit et dix bits. Le nombre de cellules logiques associées à cette étape est donnée par :

$$n_2 = Nb_{s1} + Nb_{e1} + 1.$$

, où N_e est le nombre d'entrées de l'additionneur parallèle, Nb_e le nombre de bits pour chaque entrée, Nb_s le nombre de bits de la sortie de l'additionneur.

$$Nb_{s1} = Mod\,[F_m/2] + E\,[F_m/2] + Nb_{pix},\ Nb_{e1} = Nb_{pix} \Rightarrow n_2 = Mod\,[F_m/2] + E\,[F_m/2] + 2\ Nb_{pix} + 1.$$

$$n_2 = Mod\,[F_m/2] + E\,[F_m/2] + 2 \times 8 + 1 = Mod\,[F_m/2] + E\,[F_m/2] + 17.$$

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
LIBRARY altera_mf;
USE altera_mf.altera_mf_components.all;
--  Entity Declaration
ENTITY etap2e IS
    -- {{ALTERA_IO_BEGIN}} DO NOT REMOVE THIS LINE!
    PORT
    (
        so31m : IN STD_LOGIC_VECTOR(9 downto 0);
        so32m : IN STD_LOGIC_VECTOR(9 downto 0);
        so33m : IN STD_LOGIC_VECTOR(9 downto 0);
        so31md : IN STD_LOGIC_VECTOR(9 downto 0);
        so32md : IN STD_LOGIC_VECTOR(9 downto 0);
        so33md : IN STD_LOGIC_VECTOR(9 downto 0);

        so9m : OUT STD_LOGIC_VECTOR(11 downto 0);
        so9md : OUT STD_LOGIC_VECTOR(11 downto 0)
    );
    -- {{ALTERA_IO_END}} DO NOT REMOVE THIS LINE!
END etap2e;
--  Architecture Body
ARCHITECTURE etap2e_architecture OF etap2e IS
COMPONENT parallel_add
GENERIC (width : NATURAL; representation : STRING; size : NATURAL; msw_subtract : STRING;
pipeline : NATURAL; result_alignment : STRING; widthr : NATURAL; shift : NATURAL);
PORT (data : IN ALTERA_MF_LOGIC_2D (2 DOWNTO 0, 9 DOWNTO 0);
result : OUT STD_LOGIC_VECTOR (11 DOWNTO 0));
END COMPONENT;
SIGNAL sub_wire1,sub_wire2,sub_wire3,sub_wire4: ALTERA_MF_LOGIC_2D (2 DOWNTO 0, 9 DOWNTO 0);
SIGNAL sub_wire1d,sub_wire2d,sub_wire3d,sub_wire4d: ALTERA_MF_LOGIC_2D (2 DOWNTO 0, 9 DOWNTO 0);
BEGIN
--so9m <= CONV_STD_LOGIC_VECTOR(so31m + so32m + so33m, 12);
sub_wire1(0,0) <= so31m(0); sub_wire1(0,1) <= so31m(1); sub_wire1(0,2) <= so31m(2);
sub_wire1(0,3) <= so31m(3); sub_wire1(0,4) <= so31m(4); sub_wire1(0,5) <= so31m(5);
sub_wire1(0,6) <= so31m(6); sub_wire1(0,7) <= so31m(7); sub_wire1(0,8) <= so31m(8);
sub_wire1(0,9) <= so31m(9);
sub_wire1(1,0) <= so32m(0); sub_wire1(1,1) <= so32m(1); sub_wire1(1,2) <= so32m(2);
sub_wire1(1,3) <= so32m(3); sub_wire1(1,4) <= so32m(4); sub_wire1(1,5) <= so32m(5);
sub_wire1(1,6) <= so32m(6); sub_wire1(1,7) <= so32m(7); sub_wire1(1,8) <= so32m(8);
sub_wire1(1,9) <= so32m(9);
sub_wire1(2,0) <= so33m(0); sub_wire1(2,1) <= so33m(1); sub_wire1(2,2) <= so33m(2);
sub_wire1(2,3) <= so33m(3); sub_wire1(2,4) <= so33m(4); sub_wire1(2,5) <= so33m(5);
sub_wire1(2,6) <= so33m(6); sub_wire1(2,7) <= so33m(7); sub_wire1(2,8) <= so33m(8);
sub_wire1(2,9) <= so33m(9);
parallel_add1 : parallel_add
GENERIC MAP(width => 10, representation => "UNSIGNED", size => 3, msw_subtract => "NO",
pipeline => 0, result_alignment => "LSB", widthr => 12, shift => 0)
PORT MAP (data => sub_wire1,result => so9m);

--so9md <= CONV_STD_LOGIC_VECTOR(so31md + so32md + so33md, 12) when e4n = '1';
sub_wire1d(0,0) <= so31md(0);   sub_wire1d(0,1) <= so31md(1);   sub_wire1d(0,2) <= so31md(2);
sub_wire1d(0,3) <= so31md(3);   sub_wire1d(0,4) <= so31md(4);   sub_wire1d(0,5) <= so31md(5);
sub_wire1d(0,6) <= so31md(6);   sub_wire1d(0,7) <= so31md(7);   sub_wire1d(0,8) <= so31md(8);
sub_wire1d(0,9) <= so31md(9);
```

Figure 4-46 : Extrait du code VHDL associé au deuxième additionneur parallèle.

Le deuxième additionneur a ses trois entrées et sa sortie codées respectivement sur dix et douze bits. Le nombre de cellules consommées pour cette phase est :

$$n_3 = 3\ (Mod\ [F_m/2] + E\ [F_m/2]) + 17.$$

Par ailleurs pour gérer la mémoire tampon nous avons intégré deux blocs de multiplexage : un à l'entrée et l'autre à la sortie. La figure 4-47 représente le code VHDL associé à cette phase.

```
ram3d : altsyncram
GENERIC MAP(intended_device_family => "Stratix", operation_mode => "DUAL_PORT",
        width_a => 10, widthad_a => 9, numwords_a => 500, width_b => 10, widthad_b => 9,
        numwords_b => 500, lpm_type => "altsyncram", width_byteena_a => 1,
        outdata_reg_b => "CLOCK1", indata_aclr_a => "NONE", wrcontrol_aclr_a => "NONE",
        address_aclr_a => "NONE", address_reg_b => "CLOCK1", address_aclr_b => "NONE",
        outdata_aclr_b => "NONE")
PORT MAP(wren_a => e3nd, clock0 => pclk, clock1 => pclk, address_a => dmadd,
        address_b => madd, data_a => so3md, q_b => so33xd);

ram4d : altsyncram
GENERIC MAP(intended_device_family => "Stratix", operation_mode => "DUAL_PORT",
        width_a => 10, widthad_a => 9, numwords_a => 500, width_b => 10, widthad_b => 9,
        numwords_b => 500, lpm_type => "altsyncram", width_byteena_a => 1,
        outdata_reg_b => "CLOCK1", indata_aclr_a => "NONE", wrcontrol_aclr_a => "NONE",
        address_aclr_a => "NONE", address_reg_b => "CLOCK1", address_aclr_b => "NONE",
        outdata_aclr_b => "NONE")
PORT MAP(wren_a => e4nd, clock0 => pclk, clock1 => pclk, address_a => dmadd,
        address_b => madd, data_a => so3md, q_b => so34xd);

so31m <= so34x when e1n = '1' else so31x;
so32m <= so34x when e2n = '1' else so32x;
so33m <= so34x when e3n = '1' else so33x;

so31md <= so34xd when e1n = '1' else so31xd;
so32md <= so34xd when e2n = '1' else so32xd;
so33md <= so34xd when e3n = '1' else so33xd;

END som3mem_architecture;
```

Figure 4-47 : Extrait du code VHDL (stockage et multiplexage de la sortie).

Le multiplexeur d'entrée contrôle les broches d'activation de la lecture/écriture dans la mémoire tampon constituée de quatre blocs et consomme un élément logique pour chaque broche. Soit au total:

$$n_4 = F_m + 1$$

De même le bloc de multiplexage en sortie de la mémoire tampon consomme :

$$n_5 = Nb_{s1} \; x \; F_m$$

$$n_5 = (Mod\,[F_m/2] + E\,[F_m/2] + Nb_{pix}) \; x \; F_m = (Mod\,[F_m/2] + E\,[F_m/2] + 8) \; x \; F_m.$$

En conséquence l'étape de moyennage consomme au total (voies gauche et droite) :

$$N_1 = 2\,(n_1 + n_2 + n_3 + n_4 + n_5).$$

$$N_1 = 2\,(8\,F_m + Mod\,[F_m/2] + E\,[F_m/2] + 17 + 3\,(Mod\,[F_m/2] + E\,[F_m/2]) + 17 + F_m + 1+ (Mod\,[F_m/2] + E\,[F_m/2] + 8) \; x \; F_m).$$

$$N_1 = 2\,(8\,F_m + (Mod\,[F_m/2] + E\,[F_m/2]) + 17 + 3\,(Mod\,[F_m/2] + E\,[F_m/2]) + 17 + F_m + 1+ (Mod\,[F_m/2] + E\,[F_m/2] + 8) \; x \; F_m) = 2\,(17\,F_m + (4 + F_m) \; x \; (Mod\,[F_m/2] + E\,[F_m/2]) + 35).$$

Soit:

$$N_1 = 2\ (17\ F_m + (4 + F_m)\ x\ (Mod\ [F_m/2] + E\ [F_m/2]) + 35).$$

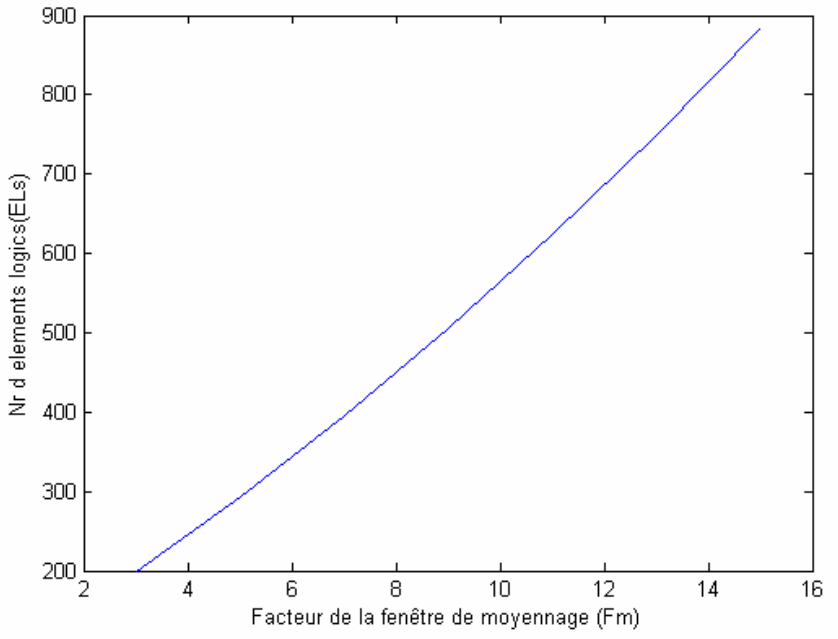


Figure 4-48 : Nombre d'éléments logiques consommés en fonction de la fenêtre de calcul.

On constate que l'étape de moyennage consomme 200ELs pour une fenêtre 3 x 3 et près de 900 ELs pour une fenêtre 15 x 15.

4.8.4 Le calcul de l'image census

L'étape précédente fournit un flot de données image de la moyenne d'une fenêtre glissante de taille 3x3. Ce flot, préalablement stocké dans une mémoire intermédiaire, est transmis ensuite à un opérateur assurant le calcul de la transformation Census. Nous avons implémenté cette transformation sur une fenêtre 7x7, laquelle est un bon compromis compte tenu des images traitées. Cette opération nécessite donc le stockage intermédiaire de huit lignes de l'image (sept accédées en lecture et une en écriture cf. figure 4-34).

```
Signal  so90x,so91x,so92x,so93x,so94x,so95x,so96x,so97x : STD_LOGIC_VECTOR (11 DOWNTO 0)
Signal  so90xd,so91xd,so92xd,so93xd,so94xd,so95xd,so96xd,so97xd : STD_LOGIC_VECTOR (11 D

BEGIN
ram0 : altsyncram
GENERIC MAP(intended_device_family => "Stratix", operation_mode => "DUAL_PORT",
        width_a => 12, widthad_a => 9, numwords_a => 500, width_b => 12, widthad_b => 9,
        numwords_b => 500, lpm_type => "altsyncram", width_byteena_a => 1,
        outdata_reg_b => "CLOCK1", indata_aclr_a => "NONE", wrcontrol_aclr_a => "NONE",
        address_aclr_a => "NONE", address_reg_b => "CLOCK1", address_aclr_b => "NONE",
        outdata_aclr_b => "NONE")
PORT MAP(wren_a => e0nc, clock0 => pclk, clock1 => pclk, address_a => madd,
        address_b => madd, data_a => so9m, q_b => so90x);

ram1 : altsyncram
GENERIC MAP(intended_device_family => "Stratix", operation_mode => "DUAL_PORT",
        width_a => 12, widthad_a => 9, numwords_a => 500, width_b => 12, widthad_b => 9,
        numwords_b => 500, lpm_type => "altsyncram", width_byteena_a => 1,
        outdata_reg_b => "CLOCK1", indata_aclr_a => "NONE", wrcontrol_aclr_a => "NONE",
        address_aclr_a => "NONE", address_reg_b => "CLOCK1", address_aclr_b => "NONE",
        outdata_aclr_b => "NONE")
PORT MAP(wren_a => e1nc, clock0 => pclk, clock1 => pclk, address_a => madd,
        address_b => madd, data_a => so9m, q_b => so91x);

ram2 : altsyncram
GENERIC MAP(intended_device_family => "Stratix", operation_mode => "DUAL_PORT",
        width_a => 12, widthad_a => 9, numwords_a => 500, width_b => 12, widthad_b => 9,
        numwords_b => 500, lpm_type => "altsyncram", width_byteena_a => 1,
        outdata_reg_b => "CLOCK1", indata_aclr_a => "NONE", wrcontrol_aclr_a => "NONE",
        address_aclr_a => "NONE", address_reg_b => "CLOCK1", address_aclr_b => "NONE",
        outdata_aclr_b => "NONE")
PORT MAP(wren_a => e2nc, clock0 => pclk, clock1 => pclk, address_a => madd,
        address_b => madd, data_a => so9m, q_b => so92x);

ram3 : altsyncram
GENERIC MAP(intended_device_family => "Stratix", operation_mode => "DUAL_PORT",
        width_a => 12, widthad_a => 9, numwords_a => 500, width_b => 12, widthad_b => 9,
        numwords_b => 500, lpm_type => "altsyncram", width_byteena_a => 1,
        outdata_reg_b => "CLOCK1", indata_aclr_a => "NONE", wrcontrol_aclr_a => "NONE",
        address_aclr_a => "NONE", address_reg_b => "CLOCK1", address_aclr_b => "NONE",
        outdata_aclr_b => "NONE")
PORT MAP(wren_a => e3nc, clock0 => pclk, clock1 => pclk, address_a => madd,
        address_b => madd, data_a => so9m, q_b => so93x);
```

Figure 4-49 : Extrait du code VHDL associé au stockage des moyennes.

La mémoire consommée par cette étape dépend de la largeur W de l'image traitée, du nombre de bits Nb_{s2} à stocker dans chaque case et de la taille F_c de la fenêtre Census. Sa capacité, exprimée en bits, est donc :

$$M_2 = 2\ (F_c + 1)\ x\ W\ x\ Nb_{s2}.$$

En considérant un moyennage 3 x 3 ($F_m = 3$; $Nb_{s2} = 12$), la taille de la mémoire consommée est :

$$M_2 = 24\ x\ (F_c + 1)\ x\ W$$

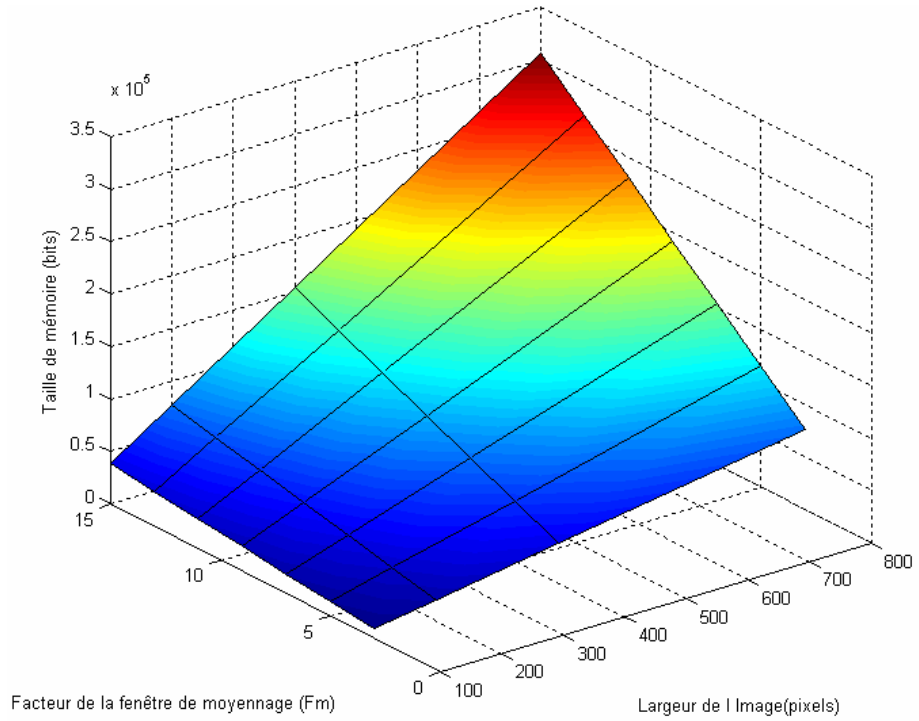


Figure 4-50 : Mémoire consommée par la transformée de Census.

Concernant les éléments logiques nécessaires, cette étape est constituée de quatre blocs principaux : deux multiplexeurs pour gérer les pointeurs d'entrées et de sorties de la mémoire intermédiaire, une fenêtre glissante à base de registres et un bloc pour le calcul de la transformée de Census.

Le multiplexeur d'entrée sélectionne une parmi les huit entrées possibles, chacune étant associée à une ligne de la zone de stockage. Il consomme pour chaque état un élément logique soit :

$$n_1 = F_c + 1$$

Le multiplexeur de sortie joue également un rôle d'aiguillage et connecte les sorties des huit mémoires vers les lignes de la fenêtre census en respectant l'ordre des lignes de l'image traitée. Il dépend du nombre de bits à aiguiller (Nb_{s2}) et du facteur de la fenêtre Census. Ce bloc consomme donc en considérant un moyennage 3x3 :

$$n_2 = (F_c + 1)\ Nb_{s2}\ x\ 2 = 24\ (F_c + 1)$$

La fenêtre Census est elle-même composée de quarante neuf registres à décalage (sept lignes et sept colonnes) de douze bit (Nb_{s2}) chacun soit douze éléments logiques par registre. La fenêtre Census nécessite donc en moyennage 3x3 :

$$n_3 = F_c^2 \times Nb_{s2} = 12\ F_c^2 \quad \textit{éléments logiques}$$

```
PROCESS
BEGIN
    WAIT UNTIL pclk = '1';
CASE mcsel IS
    WHEN "001" =>
so90l <= so91x; so91l <= so92x; so92l <= so93x;
so93l <= so94x; so94l <= so95x; so95l <= so96x; so96l <= so97x;

so90ld <= so91xd; so91ld <= so92xd; so92ld <= so93xd;
so93ld <= so94xd; so94ld <= so95xd; so95ld <= so96xd; so96ld <= so97xd;
    WHEN "010" =>
so90l <= so92x; so91l <= so93x; so92l <= so94x;
so93l <= so95x; so94l <= so96x; so95l <= so97x; so96l <= so90x;

so90ld <= so92xd; so91ld <= so93xd; so92ld <= so94xd;
so93ld <= so95xd; so94ld <= so96xd; so95ld <= so97xd; so96ld <= so90xd;
    WHEN "011" =>
so90l <= so93x; so91l <= so94x; so92l <= so95x;
so93l <= so96x; so94l <= so97x; so95l <= so90x; so96l <= so91x;

so90ld <= so93xd; so91ld <= so94xd; so92ld <= so95xd;
so93ld <= so96xd; so94ld <= so97xd; so95ld <= so90xd; so96ld <= so91xd;
    WHEN "100" =>
so90l <= so94x; so91l <= so95x; so92l <= so96x;
so93l <= so97x; so94l <= so90x; so95l <= so91x; so96l <= so92x;

so90ld <= so94xd; so91ld <= so95xd; so92ld <= so96xd;
so93ld <= so97xd; so94ld <= so90xd; so95ld <= so91xd; so96ld <= so92xd;
    WHEN "101" =>
so90l <= so95x; so91l <= so96x; so92l <= so97x;
so93l <= so90x; so94l <= so91x; so95l <= so92x; so96l <= so93x;

so90ld <= so95xd; so91ld <= so96xd; so92ld <= so97xd;
so93ld <= so90xd; so94ld <= so91xd; so95ld <= so92xd; so96ld <= so93xd;
    WHEN "110" =>
so90l <= so96x; so91l <= so97x; so92l <= so90x;
so93l <= so91x; so94l <= so92x; so95l <= so93x; so96l <= so94x;

so90ld <= so96xd; so91ld <= so97xd; so92ld <= so90xd;
so93ld <= so91xd; so94ld <= so92xd; so95ld <= so93xd; so96ld <= so94xd;
    WHEN "111" =>
so90l <= so97x; so91l <= so90x; so92l <= so91x;
so93l <= so92x; so94l <= so93x; so95l <= so94x; so96l <= so95x;

so90ld <= so97xd; so91ld <= so90xd; so92ld <= so91xd;
so93ld <= so92xd; so94ld <= so93xd; so95ld <= so94xd; so96ld <= so95xd;
    WHEN OTHERS =>
so90l <= so90x; so91l <= so91x; so92l <= so92x;
so93l <= so93x; so94l <= so94x; so95l <= so95x; so96l <= so96x;

so90ld <= so90xd; so91ld <= so91xd; so92ld <= so92xd;
so93ld <= so93xd; so94ld <= so94xd; so95ld <= so95xd; so96ld <= so96xd;
END CASE;
```

Figure 4-51 : Extrait du code VHDL associé au multiplexage des sorties de la mémoire tampon.

```
        v42d : IN STD_LOGIC_VECTOR(11 downto 0);    v43d : IN STD_LOGIC_VECTOR(11 downto 0);
        v44d : IN STD_LOGIC_VECTOR(11 downto 0);    v45d : IN STD_LOGIC_VECTOR(11 downto 0);
        v46d : IN STD_LOGIC_VECTOR(11 downto 0);    v47d : IN STD_LOGIC_VECTOR(11 downto 0);
        v51d : IN STD_LOGIC_VECTOR(11 downto 0);    v52d : IN STD_LOGIC_VECTOR(11 downto 0);
        v53d : IN STD_LOGIC_VECTOR(11 downto 0);    v54d : IN STD_LOGIC_VECTOR(11 downto 0);
        v55d : IN STD_LOGIC_VECTOR(11 downto 0);    v56d : IN STD_LOGIC_VECTOR(11 downto 0);
        v57d : IN STD_LOGIC_VECTOR(11 downto 0);    v61d : IN STD_LOGIC_VECTOR(11 downto 0);
        v62d : IN STD_LOGIC_VECTOR(11 downto 0);    v63d : IN STD_LOGIC_VECTOR(11 downto 0);
        v64d : IN STD_LOGIC_VECTOR(11 downto 0);    v65d : IN STD_LOGIC_VECTOR(11 downto 0);
        v66d : IN STD_LOGIC_VECTOR(11 downto 0);    v67d : IN STD_LOGIC_VECTOR(11 downto 0);
        cen : OUT STD_LOGIC_VECTOR(47 downto 0);    cend : OUT STD_LOGIC_VECTOR(47 downto 0)
    );
    -- ((ALTERA_IO_END)) DO NOT REMOVE THIS LINE!
END etap3e;
--  Architecture Body
ARCHITECTURE etap3e_architecture OF etap3e IS
COMPONENT lpm_ff
    GENERIC(lpm_width : NATURAL; lpm_type : STRING; lpm_fftype : STRING);
    PORT (clock : IN STD_LOGIC ; q : OUT STD_LOGIC_VECTOR (lpm_width-1 DOWNTO 0);
    data : IN STD_LOGIC_VECTOR (lpm_width-1 DOWNTO 0));
END COMPONENT;
signal cenx, cenxd : STD_LOGIC_VECTOR (47 DOWNTO 0);
BEGIN
cenx(47) <= '1' when x > v01 else '0'; cenx(46) <= '1' when x > v02 else '0'; cenx(45) <= '1' when x > v03 else '0';
cenx(44) <= '1' when x > v04 else '0'; cenx(43) <= '1' when x > v05 else '0'; cenx(42) <= '1' when x > v06 else '0';
cenx(41) <= '1' when x > v07 else '0';
cenx(40) <= '1' when x > v11 else '0'; cenx(39) <= '1' when x > v12 else '0'; cenx(38) <= '1' when x > v13 else '0';
cenx(37) <= '1' when x > v14 else '0'; cenx(36) <= '1' when x > v15 else '0'; cenx(35) <= '1' when x > v16 else '0';
cenx(34) <= '1' when x > v17 else '0';
cenx(33) <= '1' when x > v21 else '0'; cenx(32) <= '1' when x > v22 else '0'; cenx(31) <= '1' when x > v23 else '0';
cenx(30) <= '1' when x > v24 else '0'; cenx(29) <= '1' when x > v25 else '0'; cenx(28) <= '1' when x > v26 else '0';
cenx(27) <= '1' when x > v27 else '0';

cenx(26) <= '1' when x > v31 else '0'; cenx(25) <= '1' when x > v32 else '0'; cenx(24) <= '1' when x > v33 else '0';
cenx(23) <= '1' when x > v35 else '0'; cenx(22) <= '1' when x > v36 else '0'; cenx(21) <= '1' when x > v37 else '0';

cenx(20) <= '1' when x > v41 else '0'; cenx(19) <= '1' when x > v42 else '0'; cenx(18) <= '1' when x > v43 else '0';
cenx(17) <= '1' when x > v44 else '0'; cenx(16) <= '1' when x > v45 else '0'; cenx(15) <= '1' when x > v46 else '0';
cenx(14) <= '1' when x > v47 else '0';
cenx(13) <= '1' when x > v51 else '0'; cenx(12) <= '1' when x > v52 else '0'; cenx(11) <= '1' when x > v53 else '0';
cenx(10) <= '1' when x > v54 else '0'; cenx(9) <= '1' when x > v55 else '0'; cenx(8) <= '1' when x > v56 else '0';
cenx(7) <= '1' when x > v57 else '0';
cenx(6) <= '1' when x > v61 else '0'; cenx(5) <= '1' when x > v62 else '0'; cenx(4) <= '1' when x > v63 else '0';
cenx(3) <= '1' when x > v64 else '0'; cenx(2) <= '1' when x > v65 else '0'; cenx(1) <= '1' when x > v66 else '0';
```

Figure 4-52 : Extrait du code VHDL associé au calcul de la transformée de Census.

Le calcul de l'image Census consiste en une opération de comparaison entre le pixel central de la fenêtre et les pixels voisins. Chaque comparaison consomme un élément logique par bit et fournit en sortie un bit de la chaîne Census. Le nombre de pixels voisins dans une fenêtre étant ($F_c^2 - 1$), le bloc de calcul consommera :

$$n_4 = Nb_{s2} \; x \; (F_c^2 - 1) = 12 \; (F_c^2 - 1).$$

En conséquence, l'étape du calcul de l'image Census consommera au total pour les deux flots de pixels gauche et droite :

$$N_2 = 2 \; (n_1 + n_2 + n_3 + n_4)$$

$$N_2 = 2 \; ((F_c + 1) + (24 \; (F_c + 1)) + (12 \; F_c^2) + (12 \; (F_c^2 - 1)))$$

$$N_2 = 2(24 \; F_c^2 + 25 \; F_c + 13).$$

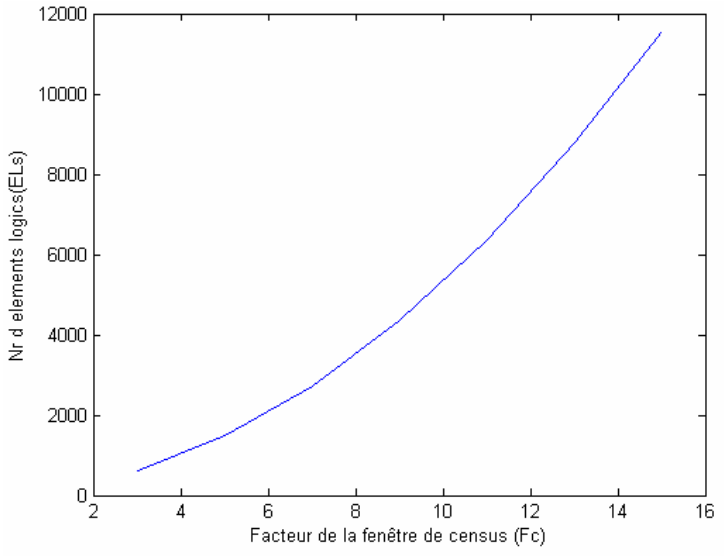


Figure 4-53 : Nombre d'éléments logiques consommés pour le calcul de l'image Census.

=> Le calcul de l'image Census nécessite 600 ELs pour une fenêtre Census 3x3 et 11576 ELs pour une fenêtre 15x15.

4.8.5 Le calcul des scores

Le calcul des scores résulte de la comptabilisation du nombre de comparaisons réussies entre un pixel motif et les pixels candidats (cf. paragraphe 4.7.4.). Cette opération nécessite :

- un jeu de registres de (F_c^2 -1) bits et dont le nombre est identique à la disparité maximale D_{max} + 1.
- D_{max} blocs à base de fonctions XNOR et possédant 2 x 48 entrées logiques.
- D_{max} blocs de calcul à 48 entrées pour évaluer chaque score.

Chaque registre consommant un élément logique par bit, cette première étape consomme :

$$n_1 = (F_c^2 - 1) \times (D_{max} + 1)$$

Le bloc de calcul des scores consomme deux éléments logiques pour chaque bit et chaque bloc 2 x Nb_{s2} éléments logiques. Cette étape nécessite donc en ELs:

$$n_2 = 2 \times (F_c^2 - 1) \times D_{max}$$

Le calcul des scores consomme au total :

$$N_3 = (F_c^2 - 1) \times (D_{max} + 1) + 2 (F_c^2 - 1) D_{max}$$

```
BEGIN
bufg : lpm_ff
GENERIC MAP (lpm_width => 48, lpm_type => "LPM_FF", lpm_fftype => "DFF")
PORT MAP (clock => pclk , q => mg, data => cen);

bufd0 : lpm_ff
GENERIC MAP (lpm_width => 48, lpm_type => "LPM_FF", lpm_fftype => "DFF")
PORT MAP (clock => pclk , q => m0d, data => cend);

bufd1 : lpm_ff
GENERIC MAP (lpm_width => 48, lpm_type => "LPM_FF", lpm_fftype => "DFF")
PORT MAP (clock => pclk , q => m1d, data => m0d);

bufd2 : lpm_ff
GENERIC MAP (lpm_width => 48, lpm_type => "LPM_FF", lpm_fftype => "DFF")
PORT MAP (clock => pclk , q => m2d, data => m1d);

bufd3 : lpm_ff
GENERIC MAP (lpm_width => 48, lpm_type => "LPM_FF", lpm_fftype => "DFF")
PORT MAP (clock => pclk , q => m3d, data => m2d);

bufd4 : lpm_ff
GENERIC MAP (lpm_width => 48, lpm_type => "LPM_FF", lpm_fftype => "DFF")
PORT MAP (clock => pclk , q => m4d, data => m3d);

bufd5 : lpm_ff
GENERIC MAP (lpm_width => 48, lpm_type => "LPM_FF", lpm_fftype => "DFF")
PORT MAP (clock => pclk , q => m5d, data => m4d);

bufd6 : lpm_ff
GENERIC MAP (lpm_width => 48, lpm_type => "LPM_FF", lpm_fftype => "DFF")
PORT MAP (clock => pclk , q => m6d, data => m5d);

bufd7 : lpm_ff
GENERIC MAP (lpm_width => 48, lpm_type => "LPM_FF", lpm_fftype => "DFF")
```

Figure 4-54 : Extrait du code VHDL associé au calcul des scores.

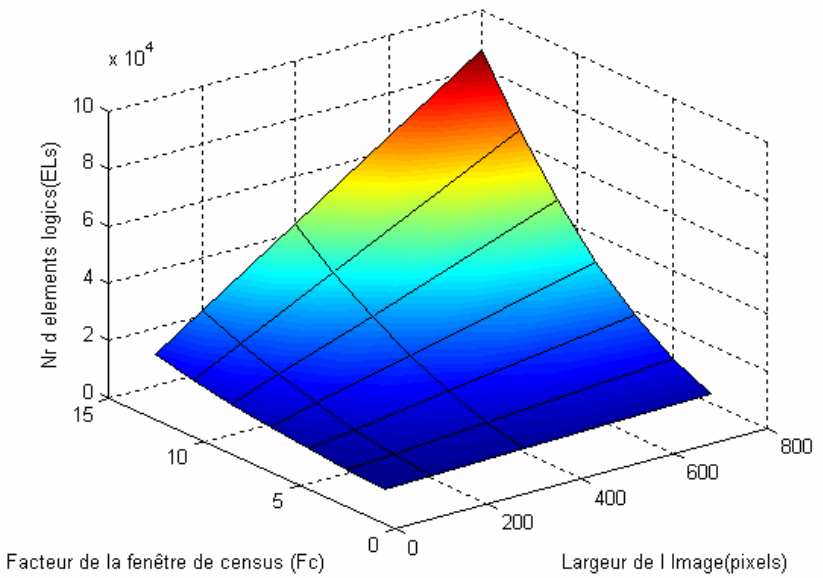


Figure 4-55 : Eléments logiques consommés par l'étape de calcul des scores.

4.8.6 La recherche des points stéréo-correspondants

L'étape de calcul des scores fournit autant de valeurs qu'il y a de pixels candidats, soit autant que la disparité maximale. Dans cette étape on se propose de rechercher, pour un pixel motif issu de l'image gauche, le rang du score le mieux-disant issu des pixels candidats de l'image droite.

Pour réaliser cette recherche en temps réel, nous avons conçu une architecture pyramidale (cf. figure 4-41) composée d'unités de comparaison pipelinées.

La fonction de l'unité de comparaison est de recevoir quatre entrées (deux valeurs de scores et deux adresses). Les valeurs de scores représentent le poids de correspondance des pixels candidats et les adresses indiquent les écarts correspondants entre ces deux pixels droits et le pixel motif gauche.

```
s49 : calscore GENERIC MAP (index => "110001")
            PORT MAP (clock => pclk, data => mg, vectr => m49d, add => a0d49, ns => n0s49);
s50 : calscore GENERIC MAP (index => "110010")
            PORT MAP (clock => pclk, data => mg, vectr => m50d, add => a0d50, ns => n0s50);
s51 : calscore GENERIC MAP (index => "110011")
            PORT MAP (clock => pclk, data => mg, vectr => m51d, add => a0d51, ns => n0s51);
s52 : calscore GENERIC MAP (index => "110100")
            PORT MAP (clock => pclk, data => mg, vectr => m52d, add => a0d52, ns => n0s52);
s53 : calscore GENERIC MAP (index => "110101")
            PORT MAP (clock => pclk, data => mg, vectr => m53d, add => a0d53, ns => n0s53);
s54 : calscore GENERIC MAP (index => "110110")
            PORT MAP (clock => pclk, data => mg, vectr => m54d, add => a0d54, ns => n0s54);
s55 : calscore GENERIC MAP (index => "110111")
            PORT MAP (clock => pclk, data => mg, vectr => m55d, add => a0d55, ns => n0s55);
s56 : calscore GENERIC MAP (index => "111000")
            PORT MAP (clock => pclk, data => mg, vectr => m56d, add => a0d56, ns => n0s56);
s57 : calscore GENERIC MAP (index => "111001")
            PORT MAP (clock => pclk, data => mg, vectr => m57d, add => a0d57, ns => n0s57);
s58 : calscore GENERIC MAP (index => "111010")
            PORT MAP (clock => pclk, data => mg, vectr => m58d, add => a0d58, ns => n0s58);
s59 : calscore GENERIC MAP (index => "111011")
            PORT MAP (clock => pclk, data => mg, vectr => m59d, add => a0d59, ns => n0s59);
s60 : calscore GENERIC MAP (index => "111100")
            PORT MAP (clock => pclk, data => mg, vectr => m60d, add => a0d60, ns => n0s60);
s61 : calscore GENERIC MAP (index => "111101")
            PORT MAP (clock => pclk, data => mg, vectr => m61d, add => a0d61, ns => n0s61);
s62 : calscore GENERIC MAP (index => "111110")
            PORT MAP (clock => pclk, data => mg, vectr => m62d, add => a0d62, ns => n0s62);
s63 : calscore GENERIC MAP (index => "111111")
            PORT MAP (clock => pclk, data => mg, vectr => m63d, add => a0d63, ns => n0s63);

t0 : calmax PORT MAP (clock => pclk, addr1 => a0d0, addr2 => a0d1, addx=> a1d0, scr1 => n0s0, scr2 => n0s1, scmax => n1s0);
t1 : calmax PORT MAP (clock => pclk, addr1 => a0d2, addr2 => a0d3, addx=> a1d1, scr1 => n0s2, scr2 => n0s3, scmax => n1s1);
t2 : calmax PORT MAP (clock => pclk, addr1 => a0d4, addr2 => a0d5, addx=> a1d2, scr1 => n0s4, scr2 => n0s5, scmax => n1s2);
t3 : calmax PORT MAP (clock => pclk, addr1 => a0d6, addr2 => a0d7, addx=> a1d3, scr1 => n0s6, scr2 => n0s7, scmax => n1s3);
t4 : calmax PORT MAP (clock => pclk, addr1 => a0d8, addr2 => a0d9, addx=> a1d4, scr1 => n0s8, scr2 => n0s9, scmax => n1s4);
t5 : calmax PORT MAP (clock => pclk, addr1 => a0d10, addr2 => a0d11, addx=> a1d5, scr1 => n0s10, scr2 => n0s11, scmax => n1
```

Figure 4-56 : une partie du code VHDL (le calcul du nombre de scores et les unités de comparaison).

L'unité de comparaison consomme deux éléments logiques pour chaque bit à basculer et comme la résolution de la valeur de disparité dépend de l'écart maximal, le nombre des éléments logiques consommés par l'unité de comparaison est,

$$n_1 = log_2 (D_{max}) + INT (log_2 (F_c^2 - 1)) + 1.$$

Le bloc de la recherche de correspondance, ayant (D_{max} – 1) unités de comparaison, consomme :

$$N_4 = (D_{max} - 1) \times (log_2 (D_{max}) + INT (log_2 (F_c^2 - 1)) + 1).$$

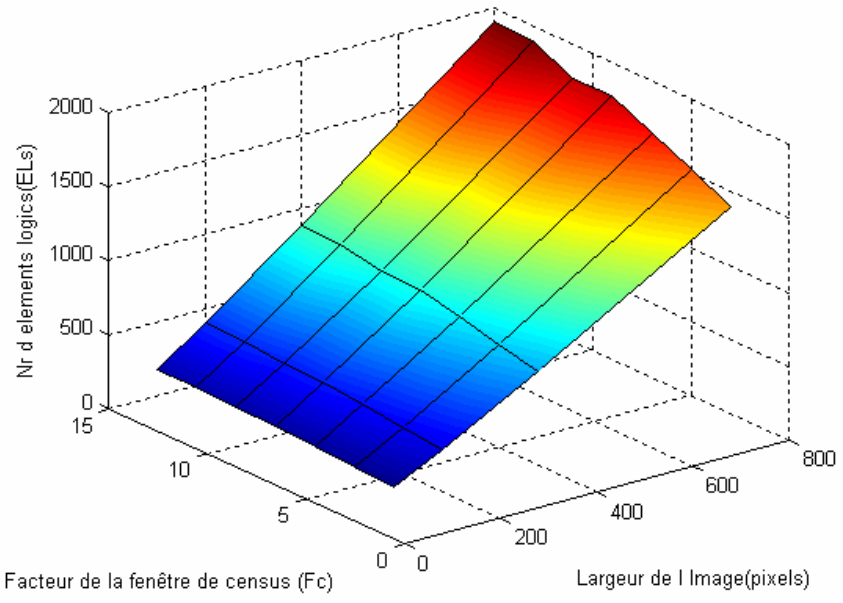


Figure 4-57 : l'abaque de ressources logique consommées par la recherche parallèle de correspondance.

4.8.7 L'architecture globale

En prenant en considération toutes les étapes nécessaires à la recherche des points stéréo-correspondants, l'architecture globale consomme :

$$N_{ELs} = N_0 + N_1 + N_2 + N_3 + N_4.$$

$$N_{ELs} = 0 + 2 (17 F_m + (4 + F_m) (MOD (F_m/2) + INT (F_m/2)) + 35) + 2 (24 F_c^2 + 25 F_c + 13) + F_c^2 (D_{max} + 1) + 2 (F_c^2 - 1) D_{max} + (D_{max} - 1) (log_2 (D_{max}) + INT (log_2 (F_c^2 - 1)) + 1).$$

$$N_{ELs} = 2 (17 F_m + (4 + F_m) (MOD (F_m/2) + INT (F_m/2))) + 70 + 48 F_c^2 + 50 F_c + 26 + F_c^2 (D_{max} + 1 + 2 D_{max}) - 2 D_{max} + (D_{max} - 1) (log_2 (D_{max}) + INT (log_2 (F_c^2 - 1)) + 1).$$

Soit :

$$N_{ELs} = 2 (17 F_m + (4 + F_m) (MOD (F_m/2) + INT (F_m/2))) + 96 + 50 F_c + F_c^2 (3 D_{max} + 49) - 2 D_{max} + (D_{max} - 1) (log_2 (D_{max}) + INT (log_2 (F_c^2 - 1)) + 1).$$

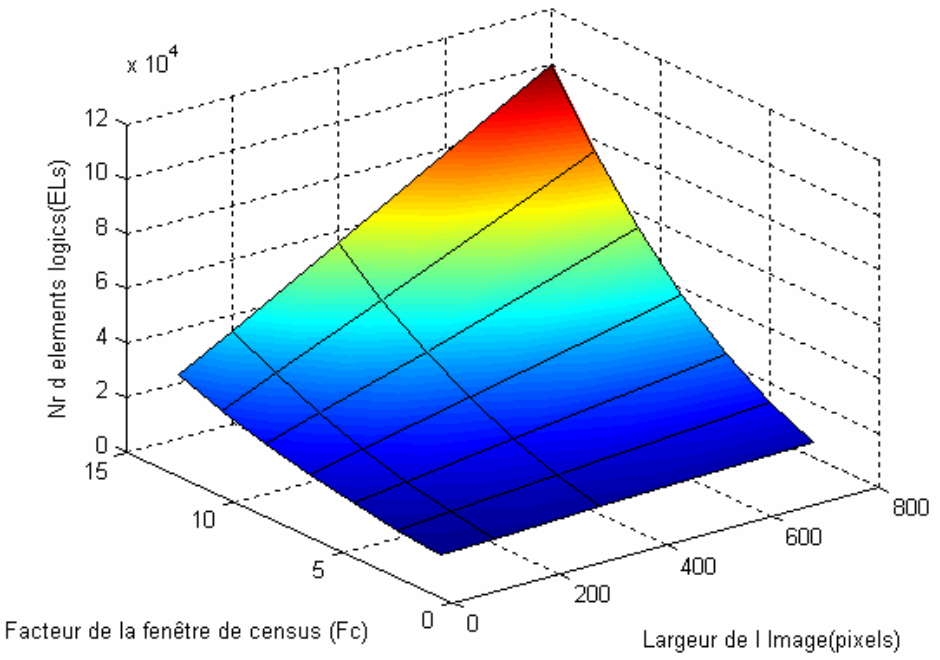


Figure 4-58 : la consommation logique de l'architecture.

Pour l'architecture globale avec fenêtre de moyennage 3 x 3, les ressource logiques consommées sont données par :

$$N_{ELs} = 2\,(51 + 14) + 96 + 50\,F_c + F_c^2\,(3\,D_{max} + 49) - 2\,D_{max} + (D_{max} - 1)\,(\log_2 (D_{max}) + INT\,(\log_2 (F_c^2 - 1)) + 1)$$

Soit :

$$N_{ELs} = 226 + 50\,F_c + F_c^2\,(3\,D_{max} + 49) - 2\,D_{max} + (D_{max} - 1)\,(\log_2 (D_{max}) + INT\,(\log_2 (F_c^2 - 1)) + 1)$$

, ou encore :

$$N_{ELs} = 226 + 50F_c + F_c^2\left(3D_{\max} + 49\right) - 2D_{\max} + \left(D_{\max} - 1\right)\left(\log_2 D_{\max} + E\left(\log_2\left[F_c^2 - 1\right]\right) + 1\right)$$

En ce qui concerne la mémoire consommée, celle-ci est donnée par :

$$M = M_0 + M_1 + M_2$$

Soit :

$$M = W^2\,(3 + 1.5\,(INT\,(\log_2 (W)) + INT\,(\log_2 (0.75\,W)) + 2)) + 2\,(F_m + 1)\,W\,x\,Nb_{s1} + 2\,(F_c + 1)\,W\,x\,Nb_{s2}$$

, ou encore:

$$M = W^2 (3 + 1.5 (INT (\log_2 (W)) + INT (\log_2 (0.75 W)) + 2)) + 2 (10 F_m + 12 F_c + 22) W.$$

Pour une fenêtre de moyennage de 3 x 3 cela devient :

$$M = W^2 (3 + 1.5 (INT (\log_2 (W)) + INT (\log_2 (0.75 W)) + 2)) + 2 (30 + 12 F_c + 22) W$$

, ou en simplifiant:

$$M = W^2\left(3 + 1.5\left\{E\left[\log_2 W\right] + E\left[\log_2\left(0.75W\right)\right] + 2\right\}\right) + W\left(104 + 12F_c\right)$$

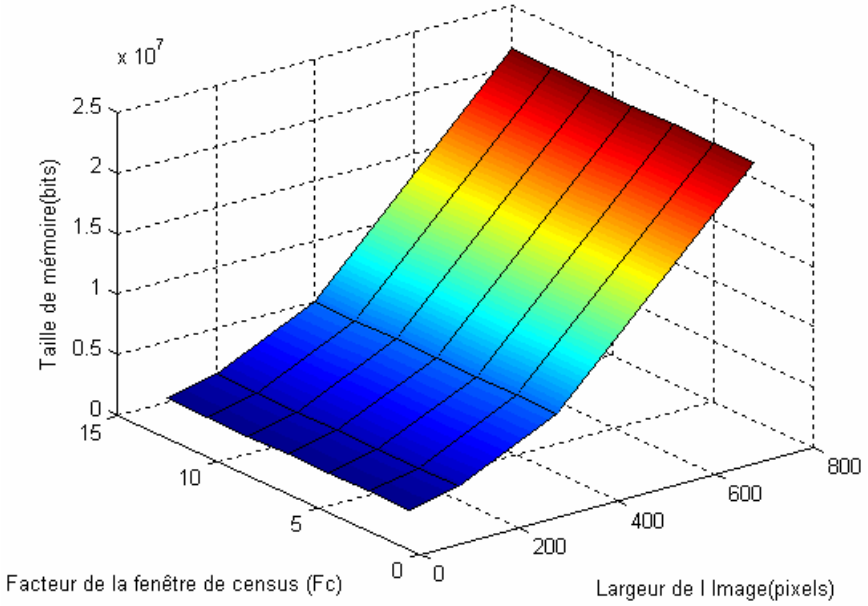


Figure 4-59 : l'abaque de la mémoire consommé par l'architecture de stéréovision temps réel.

4.9 Les performances temporelles obtenues

L'évaluation des performances a été effectuée sur l'architecture globale en prenant en compte les paramètres suivants :

- L'amplitude maximale de la distorsion est le quart de la hauteur de l'image (0.25 H).
- Fenêtre de moyennage de trois lignes et trois colonnes (3 x 3).
- Fenêtre census de sept lignes et sept colonnes (7 x 7).
- Traitement sur une image de taille 400 x 300 (H x W).
- Disparité maximale de 64 pixels.
- Fréquence de l'horloge pixel de 40 MHz.

- Signal vidéo codé sur 8 bits.

4.9.1 La rectification

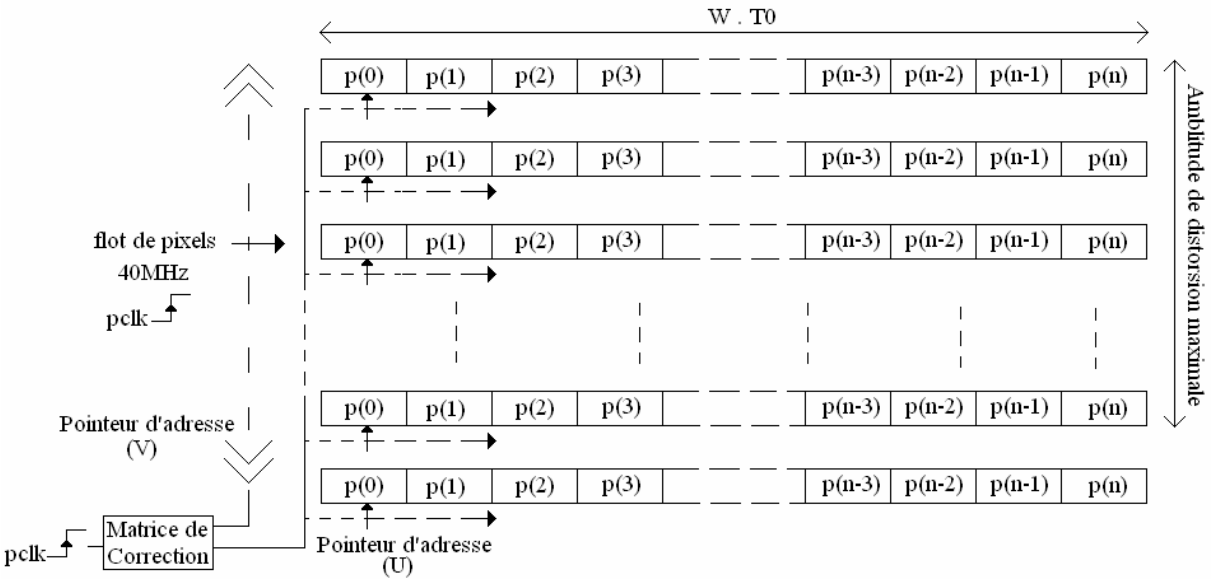


Figure 4-60 : le schéma analytique de la rectification.

Cette étape nécessite, avant d'effectuer la rectification proprement dite, un stockage intermédiaire d'autant de lignes de mémoire que d'amplitude de la distorsion maximale. Elle génère donc un temps de retard T_{rect} avant de démarrer le calcul de la stéréovision de :

$$T_{rect} = (T_0 \times W \times A_{dmax}) = T_0 \times W \times 0.25\ H$$

Soit : $25 \times 10^{-9} \times 400 \times 0.25 \times 300 = 750\mu s$ dans les conditions précitées.

La rectification, une fois amorcée, se fait au rythme de l'horloge pixel et ne consomme pas de temps calcul.

4.9.2 Le calcul de moyennage 3 x 3

Celui-ci exploite le principe décrit au paragraphe 4.7.2 à savoir un moyennage ligne (Figure 4-26) suivi d'un moyennage colonne. Le moyennage ligne entraine un retard pur de trois périodes d'horloge (figure 4-61).

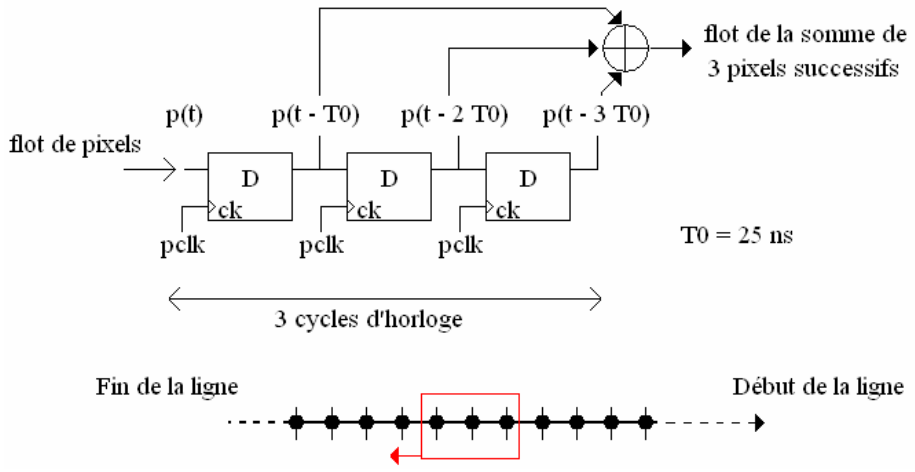


Figure 4-61 : Stockage intermédiaire et moyennage ligne

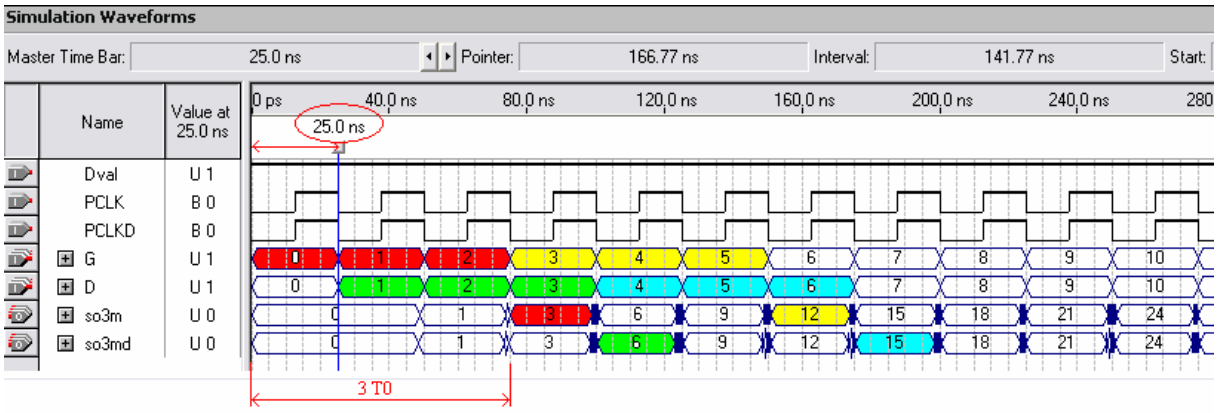


Figure 4-62 : Moyennage ligne : Retard pur de trois périodes d'horloge

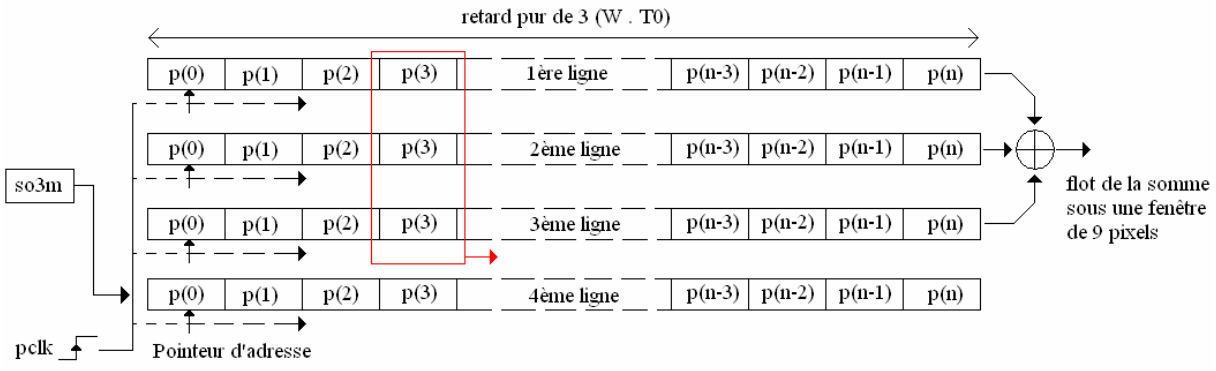


Figure 4-63 : Stockage intermédiaire et moyennage colonne

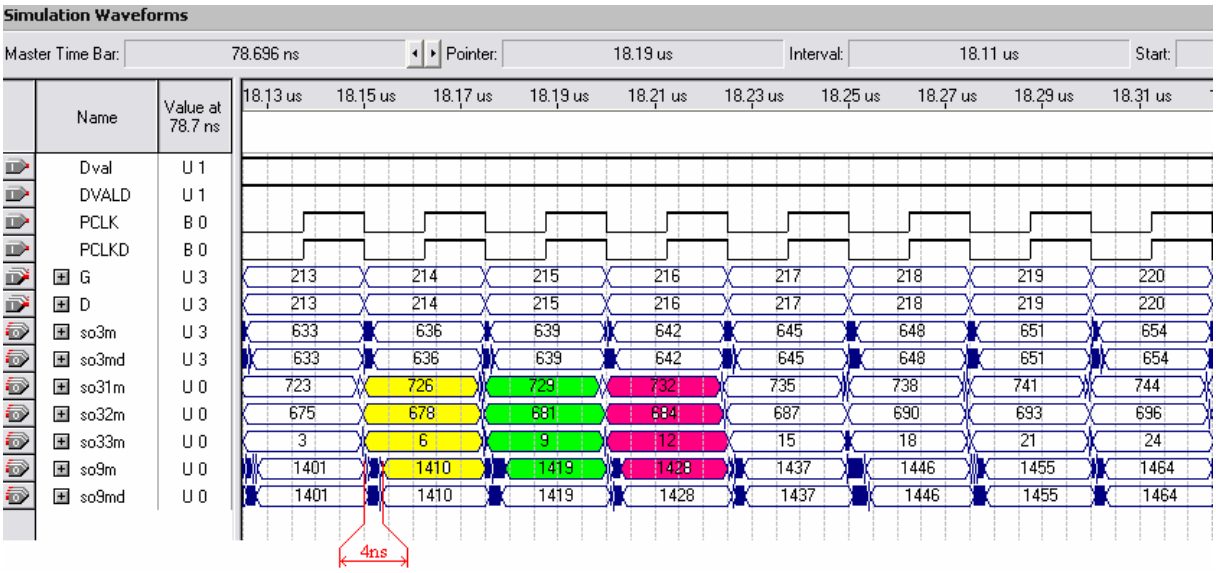


Figure 4-64 : Moyennage colonne : retard pur de 3 x (W x T_0)

Le moyennage colonne nécessite le stockage intermédiaire de trois lignes de mémoire ce qui a pour effet de générer un retard pur de 3x (W x T_0) où W est la largeur de l'image traitée. Le calcul de la somme, quant à lui, est obtenu en moins de 5 ns, ce qui est largement compatible avec la cadence de l'horloge pixel. Le retard T_{moy} introduit par le moyennage se résume à :

$$T_{moy} = 3\ T_0\ (1 + W)$$

Soit 30.075µs dans les conditions de la simulation.

4.9.3 Le calcul de census 7 x 7

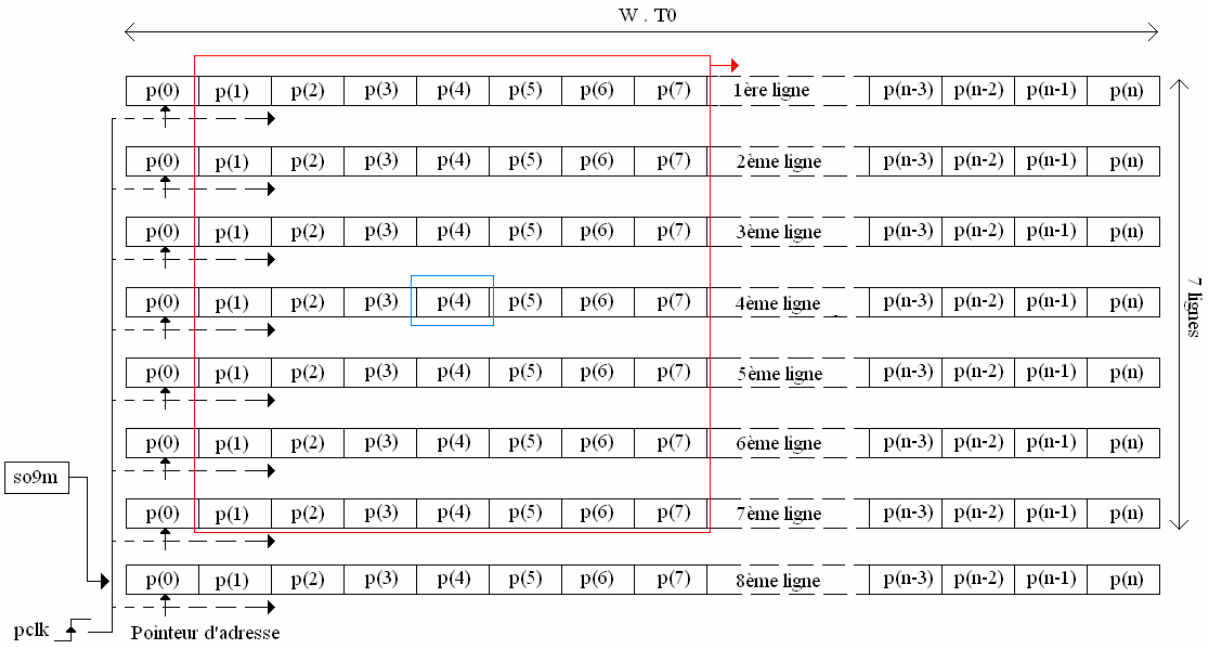


Figure 4-65 : Stockage intermédiaire de sept lignes de moyennes.

Cette étape nécessite le stockage de sept lignes intermédiaires avant de commencer à effectuer le premier calcul, ce qui représente pour chaque ligne un retard pur de (WxT_0). A ce temps il faut ajouter les sept périodes d'horloge nécessaires pour alimenter la fenêtre des registres impliquée dans le calcul du Census. Le temps T_{cen} global pour cette étape est donc :

$$T_{cen} = 7\ T_0\ (1 + W)$$

Soit 70.175µs.

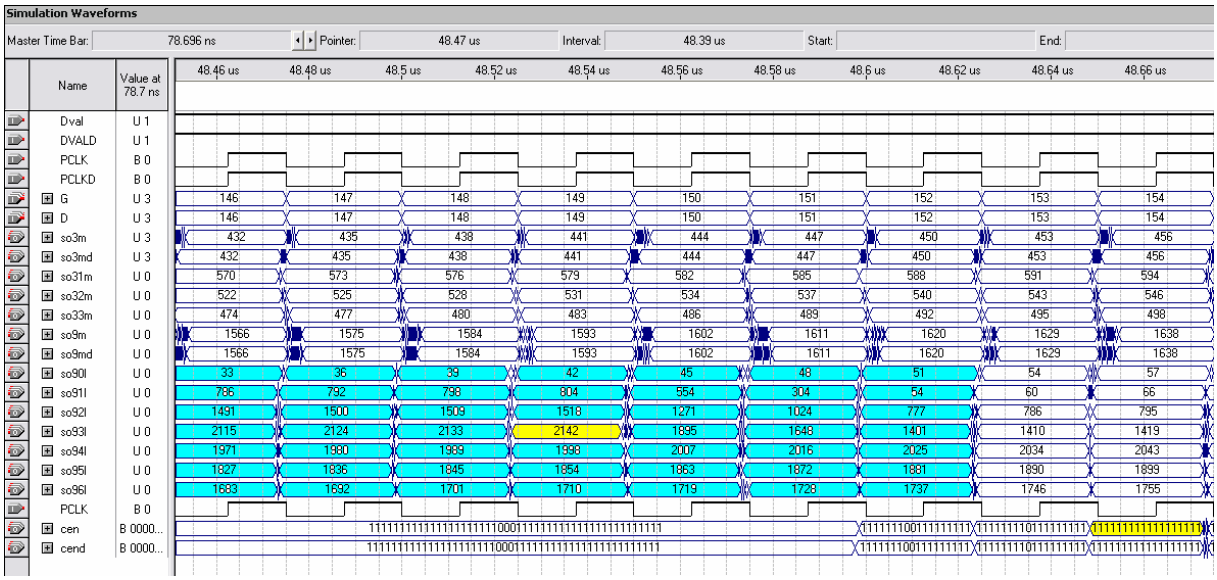


Figure 4-66 : Chronogrammes associés au calcul du Census.

4.9.4 La recherche des points stéréo-correspondants

Elle suit le principe de la figure 4-67 et introduit un retard pur de (64 T_0) correspondant au temps nécessaire pour remplir le buffer constitué de soixante quatre registres. Passé ce délai, à chaque front montant de l'horloge pixel, un pixel motif gauche et une chaîne de soixante quatre pixels droits candidats sont présentés au bloc de comparaison. Ce dernier est organisé sous la forme d'un pipe-line à six niveaux de comparaison dont chacun élimine la moitié des pixels candidats. Cette architecture, qui soutient le rythme de l'horloge pixel, introduit néanmoins un retard pur de sept périodes qui vient s'ajouter au temps précédent. La recherche des points stéréo-correspondants génère donc un retard pur T_{disp} de :

$$T_{disp} = 71\ T_0 \quad soit \quad 1.775\ \mu s$$

Le retard pur T_{arch} généré par l'architecture globale est :

$$T_{arch} = T_{rect} + T_{moy} + T_{cen} + T_{disp}.$$

Soit:

$$T_{arch} = T_0 \times W \times 0.25\ H + 3\ T_0\ (1 + W) + 7\ T_0\ (1+ W) + 71\ T_0 = T_0\ (3 + 7 + 71 + 10\ W + 0.25\ H \times W)$$

, ou encore:

$$T_{arch} = T_0 (81 + (10 + 0.25\ H)\ W) \qquad \textit{soit un retard total de 0.85ms.}$$

Ainsi l'architecture présentée peut soutenir un rythme de calcul de 333 images de disparités/sec en mode 400x300 avec un retard pur de 0.85ms et 130 images/sec avec un retard pur de 2.08ms en mode 640x480.

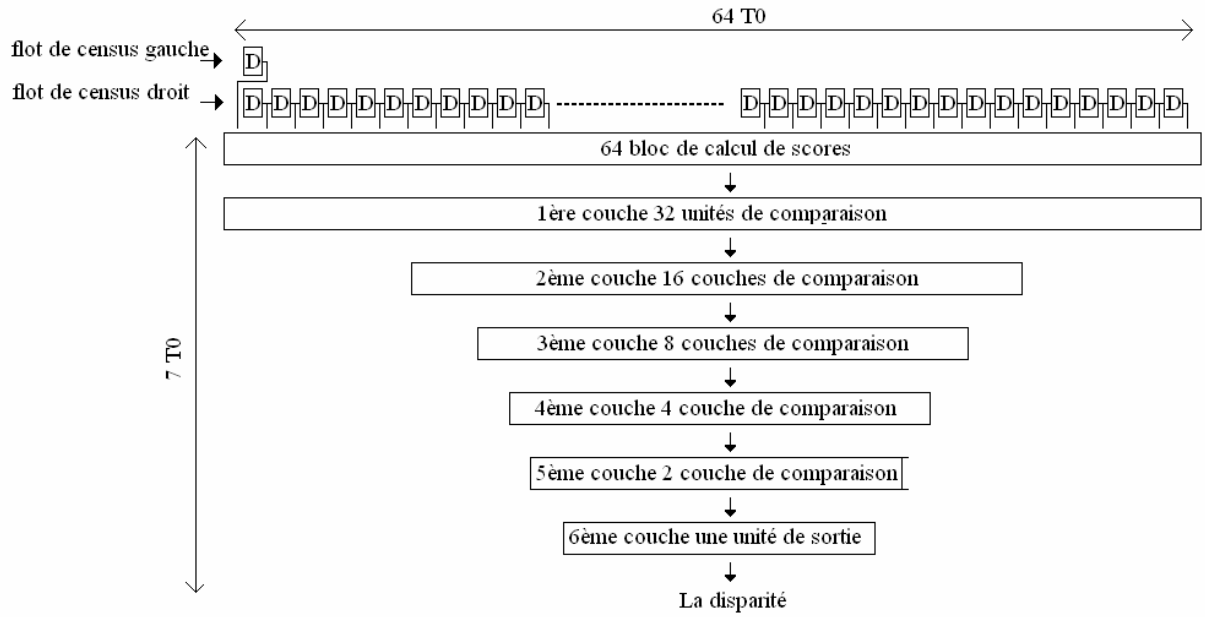


Figure 4-67 : Principe de la recherche des points stéréo-correspondants (calcul des disparités).

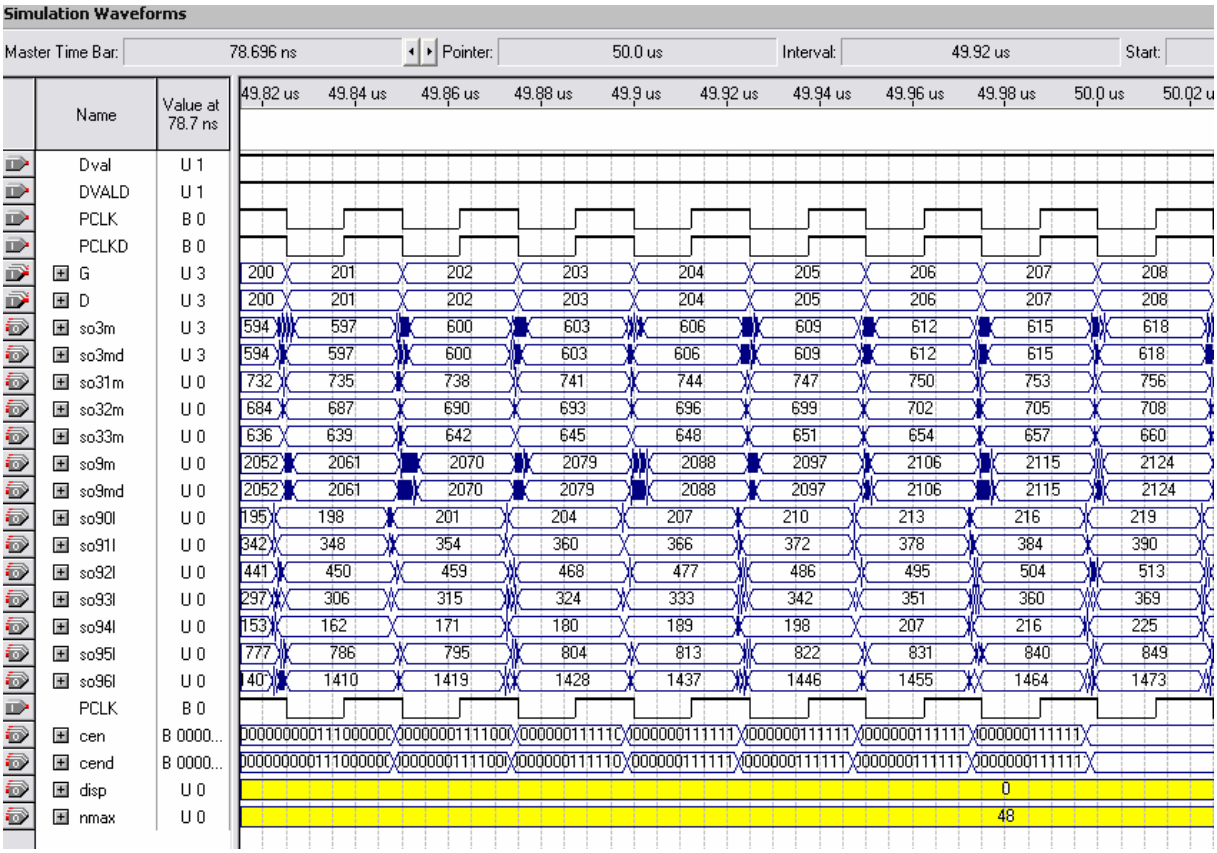


Figure 4-68 : Chronogramme de la recherche des points stéréo-correspondants.

4.10 La réalisation matérielle

4.10.1 La carte FPGA

Afin d'éviter la conception et la réalisation délicate d'une carte destinée à recevoir le circuit FPGA disponible uniquement en boîtier BGA, nous nous sommes orientés, dans un premier temps, vers l'utilisation d'un kit de développement proposé par la société ALTERA (NIOS-DEVKIT-1S40). La figure 4-69 et le tableau 4-2 donnent un aperçu des ressources disponibles sur ce kit et dans le circuit FPGA Stratix 1S40.

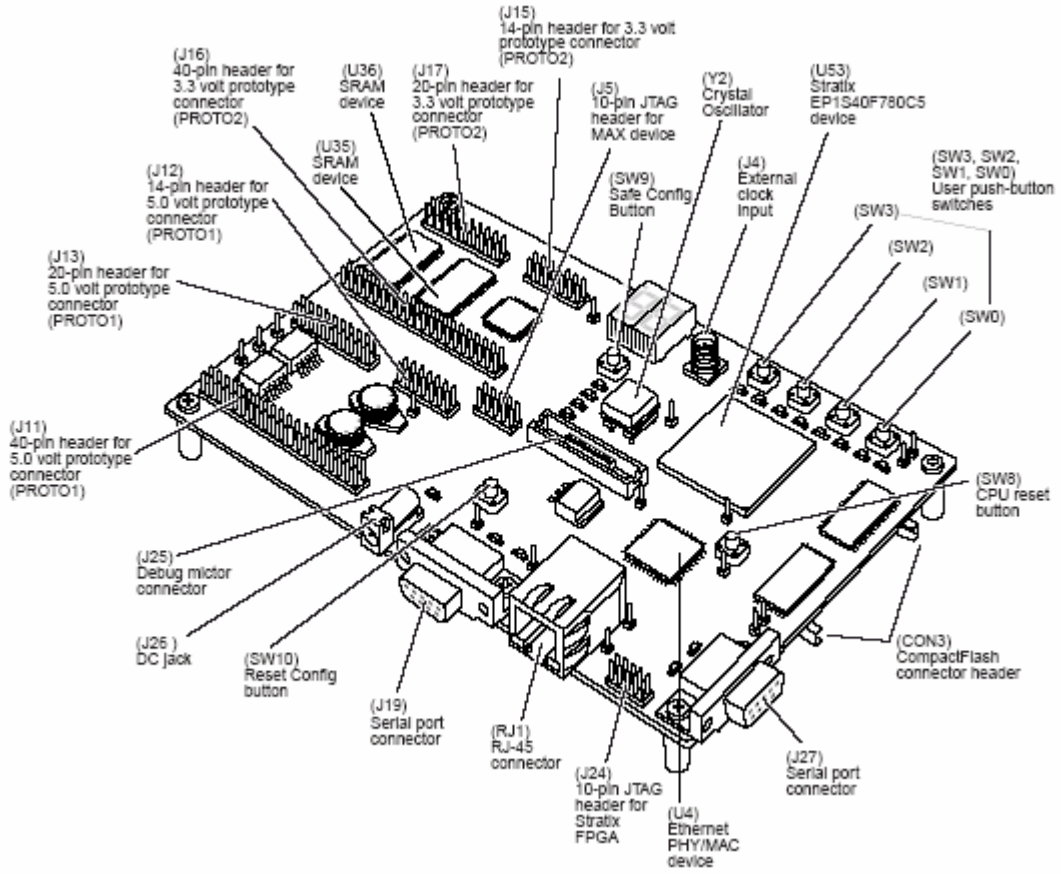


Figure 4-69 : NIOS-DEVKIT-1S40 : ressources disponibles

Eléments Logiques (Els)	41250
Blocs de Ram M512 (32x18 bits)	384
Blocs de Ram M4K (128x36 bits)	183
Blocs de Ram M512K (4k x144 bits)	4
Total Ram (bits)	3 423 744
Blocs DSP	14
Multiplieurs embarqués	112
PLL's	12
Entrées/sorties disponibles	822

Tab 4-2 : FPGA Stratix 1S40: ressources disponibles

4.10.2Les cartes interfaces

Pour rester compatible avec les caméras utilisées dans le banc d'évaluation et pour tenir le débit des données imposées par la fusion multi-longueur d'ondes du

projet PICASO, nous avons développé une carte interface à deux entrées vidéo au standard Camera-Link et une carte de sortie vidéo respectant ce même standard (figure 4-71). Celles-ci sont basées sur la mise en œuvre de composants spécialisés (figure 4-70) fabriqués par National Semiconductor, leader dans les liaisons de type LVDS.

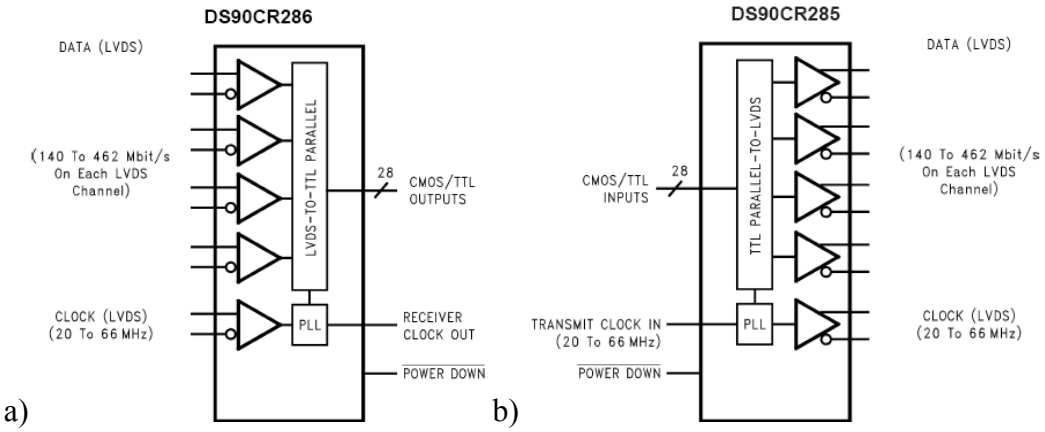


Figure 4-70: a) l'interface Camera-Link => TTL, b) l'interface TTL => Camera-Link.

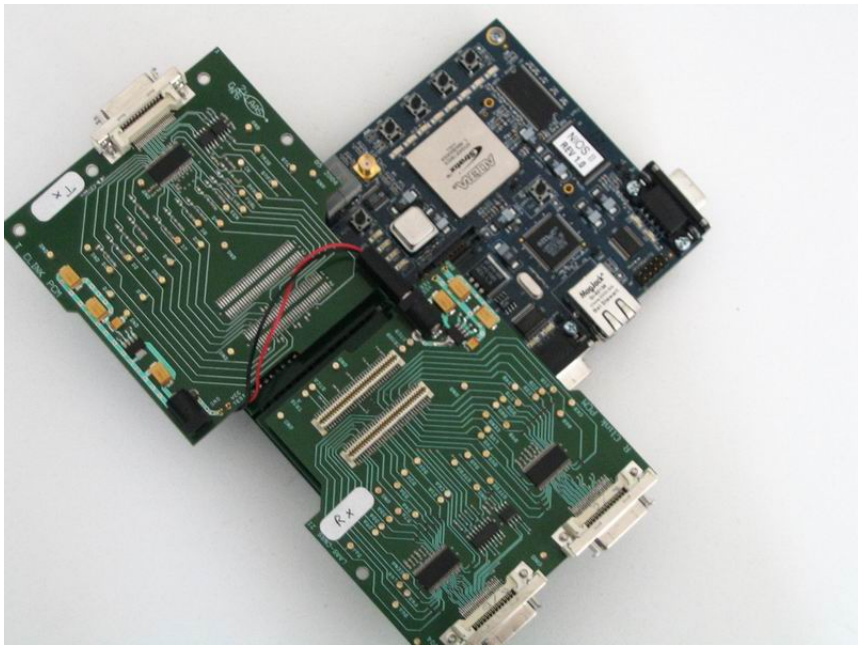


Figure 4-71 : Le kit ALTERA et les deux cartes au standard Camera-Link.

4.10.3 Test de l'interface Camera-Link

Celle-ci a été testée dans un premier temps avec l'analyseur logique pour l'aspect « timing » puis dans un deuxième temps avec le banc d'évaluation pour

l'aspect fonctionnel. Dans ce deuxième cas le kit FPGA était « transparent » pour les différents signaux. Les figures (4-72) et (4-73) représentent quelques chronogrammes issus de la carte interface.

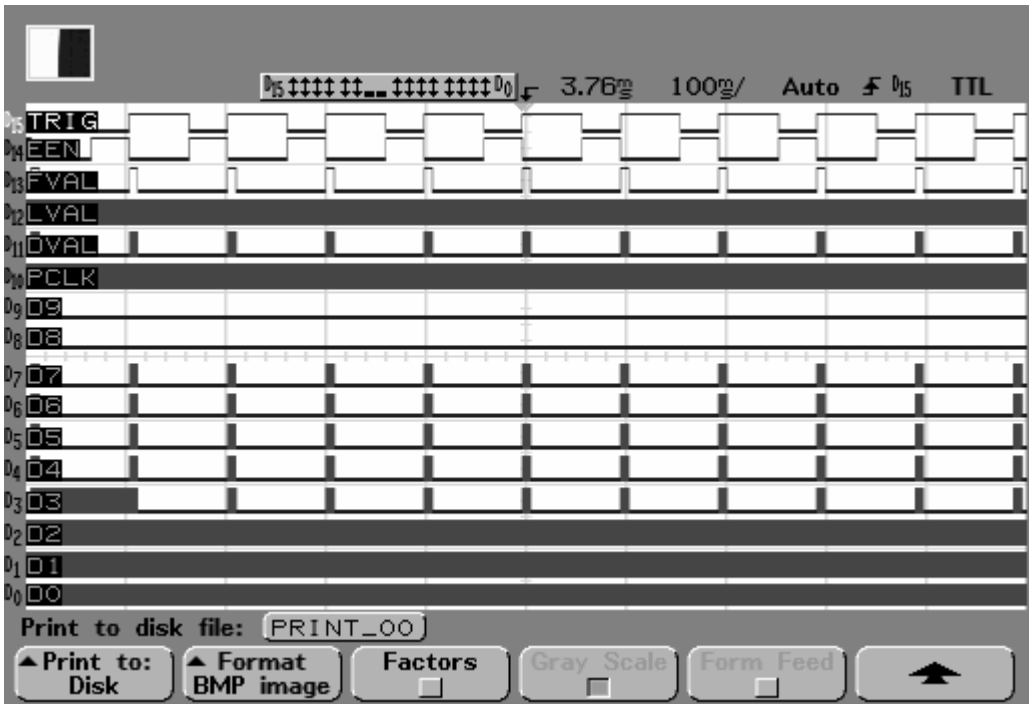


Figure 4-72 : Chronogramme signaux de synchronisation et données (100ms/carreau).

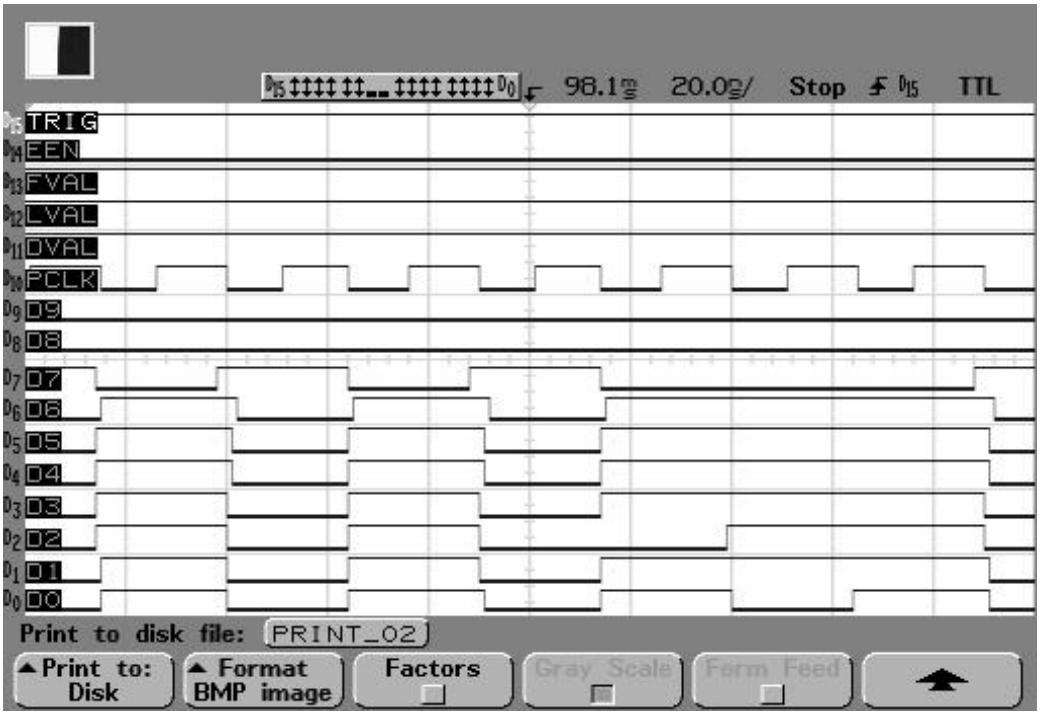


Figure 4-73 : Chronogramme signaux de synchronisation et données (20ns/carreau).

4.10.4 Test de l'architecture globale

Pour vérifier le bon fonctionnement de l'architecture globale, nous avons effectué deux types de test :

- un envoi de la même image sur les entrées gauche et droite.
- un envoi de deux images ayant des disparités connues.

Pour le premier test nous avons modifié le câblage de l'étage d'entrée à l'intérieur du circuit FPGA afin d'envoyer la même image sur les deux voies et avons pu vérifier l'obtention d'une image noire (disparité nulle pour tous les pixels).

Pour le deuxième test nous avons câblé dans le circuit FPGA un dispositif de retard appliqué à l'image et permettant ainsi de créer artificiellement des disparités connues. Nous avons pu vérifier l'adéquation entre l'image de disparité obtenue et les images d'origine.

Le troisième test est réalisé en vrai grandeur sur le circuit FPGA à partir de deux flots correspondant aux images droite et gauche configurées à une résolution de 640 x 480. Les différents calculs sont effectués à la fréquence pixel qui est de 40 MHz (horloge pixel des caméras) et nous permettent d'obtenir, à partir d'une fenêtre census 7 x 7, une performance de traitement de 130 images de disparités par seconde avec un retard pur total de 274 µs. Ce retard est lié aux étapes de rectification, moyennage et census et s'exprime par la relation ci-dessous :

$$T_{total} = T_0 \, [(71 + W\,Ad_{max}) + 10\,(1 + W)]$$

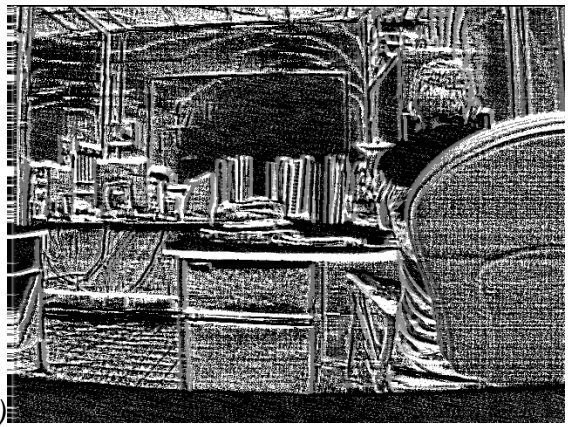
Où :

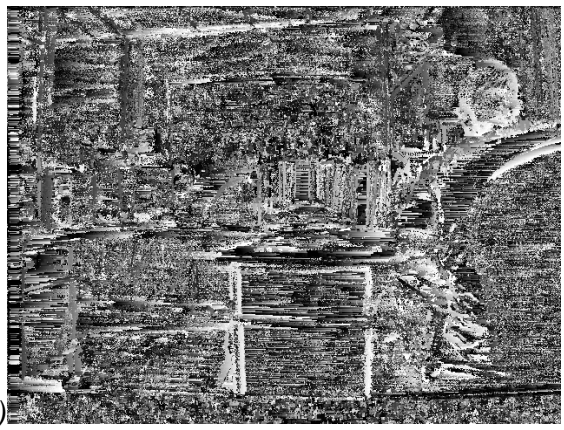
W est la largeur de l'image traitée,

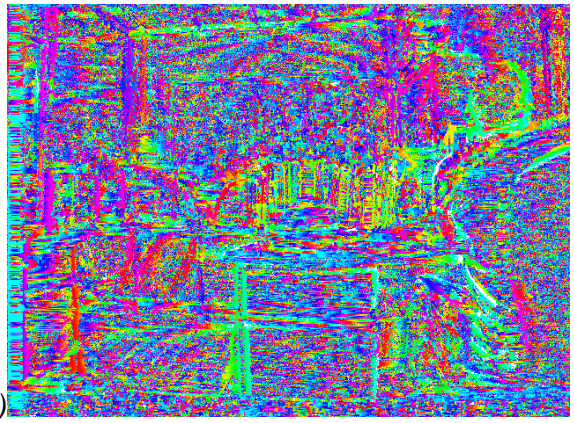
T_0 est la période de l'horloge pixel

Ad_{max} est l'amplitude maximale de la distorsion.


a)


b)


c)


d)


a)


b)

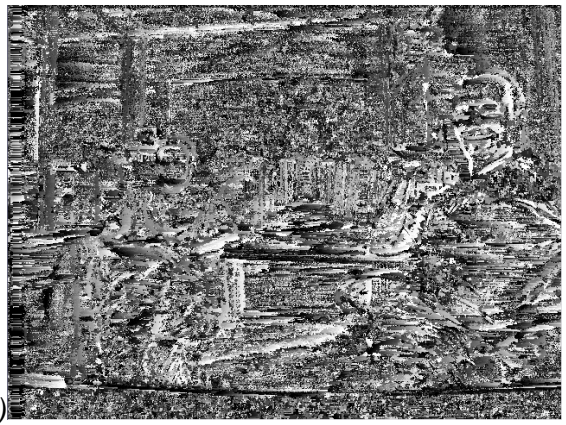
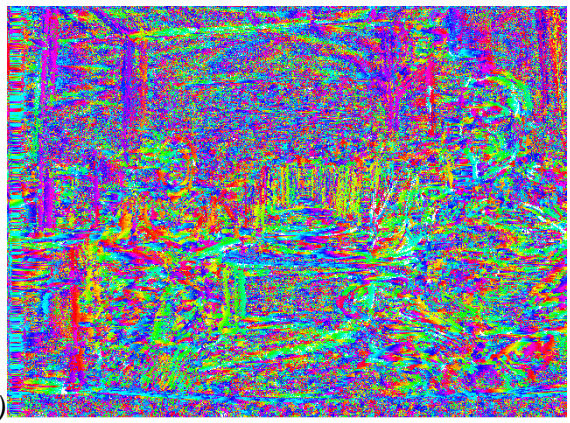
c)  d) 

Figure 4-74 : Les figures ci-dessus représentent deux exemples de scènes :

a) image d'origine gauche, b) image census droite, c) image de disparité (niveaux de gris), d) image de disparité (fausses couleurs).

4.11 Conclusions

Cette partie avait pour objectif de développer des architectures massivement parallèles capables, à partir de deux images monochromes droite et gauche, de générer des cartes de disparités en temps réel à la cadence de l'horloge pixel des caméras. Pour atteindre cet objectif nous avons évalué plusieurs solutions en tenant compte de l'aspect « temps de développement » et performances « temps réel ». Une analyse fine de toutes les opérations à réaliser avec leurs contraintes associées nous a fait éliminer d'emblée les solutions programmées à base de DSP. De la même manière les outils permettant de passer d'un code écrit en langage évolué vers des solutions câblées ne sont pas suffisamment performants à ce jour pour notre type d'application. En conséquence nous avons opté pour une implémentation de l'architecture dans un circuit FPGA à granularité hétérogène avec une description des architectures de traitement la plus proche possible du matériel lorsque les contraintes le nécessitaient et exploité les fonctions précâblées disponibles dans le circuit. Moyennant ces différents points l'architecture globale présentée dans ce chapitre nous a permis d'atteindre des cadences de 333 images de disparité par seconde en mode 400x300 et plus de 130 images par seconde en mode 640x480. Par ailleurs l'utilisation du langage VHDL comme outil de description et ses possibilités de généricité nous a permis d'obtenir un certain niveau de réutilisabilité des

différentes architectures limité toutefois par l'utilisation de fonctions spécifiques précâblées (blocs mémoire essentiellement).

4.12 Références bibliographiques

[Actel_00] Actel Corporation, Actel ProAsic FPGA family Data Sheet, Technical report, 2000.

[Altera_00] Altera Corporation, Apex CPLD family Data Sheet, Technical report, 2000.

[Areibi_01] S. Areibi et J. Zelek, 'A Smart Reconfigurable Visual System for the Blind', Tunisian-German Conference on: Smart Systems and Devices, pp: 628-633, Hamamat, Tunis, Mars 2001.

[Betz_99] V. Betz et J. Rose, 'FPGA Routing Architecture: Segmentation and Buffering to Optimize Speed and Density', International Symposium on Field Programmable Gate Arrays (FPGA), pages 59–68, 1999.

[Corke_99] P. Corke, P. Dunn et J. Banks, 'Frame-rate stereopsis using non-parametric transforms and programmable logic', IEEE International Conference on Robotics and Automation, 1928–1933, Detroit, USA, 1999.

[Darabiha_03] A. Darabiha, J. Rose, J.W. Maclean, 'Video-rate stereo depth measurement on programmable hardware', International conference on Computer Vision and Pattern Recognition, vol. 1, 203–210, Madison, Wisconsin, USA, 2003.

[DeHon_99] A. DeHon, W. Tsu, K. Macy, A. Joshi, R. Huang, N. Walker, T. Tung, O. Rowhani, G. Varghese et J. Wawrzynek, 'HSRA: High-Speed, Hierarchical Synchronous Reconfigurable Array', International Symposium on Field Programmable Gate Arrays (FPGA), pages 125–134, 1999.

[Di Stefano_02] L. Di Stefano et S. Mattoccia, 'Real-Time Stereo within the VIDET Project', Real Time Imaging, 8(5), 439–453, 2002.

[Di Stefano_04] L. Di Stefano, M. Marchionni et S. Mattoccia, 'A PC-based Real-Time Stereo Vision System', Machine GRAPHICS & VISION, vol. 13, no. 3,pp. 197-220, 2004.

[Estrin_63] G. Estrin, 'FIXED + VARIABLE computer', IEEE Transactions on Electronic Computers, 1963.

[Gluckman_99] J. Gluckman, S.K. Nayar et K.J. Thorek, 'Real-Time Omnidirectional and Panoramic Stereo', DARPA Image Understanding Workshop, 299–303, Monterey, USA, 1999.

[Hirschmuller_02] H. Hirschmuller, P. Innocent et J. Garibaldi, 'Real-Time Correlation-Based Stereo Vision with Reduced Border Errors', International Journal of Computer Vision, 47(1-3), 229–246, 2002.

[INRIA_93] O. Faugeras, B. Hotz, H. Mathieu, T. Viéville, Z. Zhang, P. Fua, E. Théron, L. Moll, G. Berry, J. Vuillemin, P. Bertin et C. Proy, 'Real time correlation-based stereo: algorithm, implementations and applications', Rapport de recherche n 2013 (INRIA), août 1993.

[Jia_03] Y. Jia, Y. Xu, W. Liu, C. Yang, Y. Zhu, X. Zhang et L. An, 'A Miniature Stereo Vision Machine for Real-Time Dense Depth Mapping', 3rd International Conference Computer Vision Systems, 268–277, Graz, Austria, 2003.

[Jia_04] Y. Jia, Y. Xu, W. Liu, C. Yang, Y. Zhu, X. Zhang et L. An, 'A Miniature Stereo Vision Machine (MSVM-III) for Dense Disparity Mapping', 17th International conference on Pattern Recognition, vol. 1, 728–731, Cambridge, UK, 23-26 août 2004.

[Kanade_96] T.Kanade, A.Yoshida, K.Oda, H.Kano et M.Tanaka, 'A Stereo Machine for Video-Rate Dense Depth Mapping and Its New Applications', Proceeding of IEEE International conference on Computer Vision and Pattern Recognition, pp.196-202, Juin 1996.

[Kimura_99] S. Kimura, T. Shinbo, H. Yamaguchi, E. Kawamura, K. Naka, 'A Convolver-Based Real-Time Stereo Machine (SAZAN)', Conference on Computer Vision and Pattern Recognition, 457–463, 1999.

[Konolige_97] K. Konolige, 'Small Vision Systems: Hardware and Implementation', 8th International Symposium on Robotics Research, 111–116, Hayama, Japan, 1997.

[Lu_99] G. Lu, H. Singh, M.H. Lee, N. Bagherzadeh, F.J. Kurdahi et E.M.C. Filho, 'The MorphoSys Parallel Reconfigurable System', European Conference on Parallel Processing, pages 727–734, 1999.

[Marshall_99] A. Marshall, T. Stansfield, I. Kostarnov, J. Vuillemin et B. Hutchings, 'A Reconfigurable Arithmetic Array for Multimedia Applications', 7th ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, pages 135–143, 1999.

[Muhlmann_02] K. Muhlmann, D. Maier, J. Hesser et R. Manner, 'Calculating Dense Disparity Maps from Color Stereo Images, an Efficient Implementation', International Journal of Computer Vision, 47(1-3), 79–88, 2002.

[Navarro_02] D. Navarro, G. Cathebras et G. Cambon, 'Rétine d'Acquisition du Flot Optique : Intégration de la Transformée de Census', 3ème Colloque du GDR CAO de Circuits et Systèmes Intégrés, pp. 93-96, 2002.

[Quartus_03] Altera corporation, 'Introduction to Quartus II', version 3.0, juin 2003.

[Tanahashi_01] H. Tanahashi, K. Yamamoto, CaihuaWang et Y. Niwa, 'Development of a stereo omnidirectional imaging system (SOS)', IEEE International conference on Industrial Electronics, Control and Instrumentation, 289–294, Nagoya, Japan, 2001.

[Van der Val_01] G. Van der Val, M. Hansen et M. Piacentino, 'The ACADIA Vision Processor', 5th International Workshop on Computer Architecture for Machine Perception, 31–40, Padova, Italy, 2001.

[Vhdl_98] R. Airiau, J.M. Bergé, V. Olive et J. Rouillard, 'VHDL langage, modélisation, synthèse', 2ème édition, 1998.

[Woodfill_97] J. Woodfill et B. Von Herzen, 'Real-Time Stereo Vision on the PARTS Reconfigurable Computer', 5th IEEE Symposium on FPGAs for Custom Computing Machines, pp. 201–210, Napa Valley, USA, avril 1997.

[Xilinx_94] Xilinx Corporation, XC4000 FPGA family Data Sheet, Technical report, 1994.

[Xilinx_98] Xilinx Corporation, Virtex FPGA family Data Sheet, Technical report, 1998.

[Yang_03] R. Yang et M. Pollefeys, 'Multi-Resolution Real-Time Stereo on Commodity Graphics Hardware', IEEE Computer Vision and Pattern Recognition 2003, pp. (I-211 - I-217), 18-20 June 2003.

Chapitre5

Application à la stéréo-endoscopie

5 APPLICATION A LA STEREO-ENDOSCOPIE

5.1 Problématique de la chirurgie endoscopique

La chirurgie mini-invasive vise le développement d'une chirurgie peu invasive. Comme le présente la figure 5-1, Elle consiste à réaliser des interventions chirurgicales à partir de petites incisions pratiquées dans l'abdomen du patient et d'instruments spécialisés.

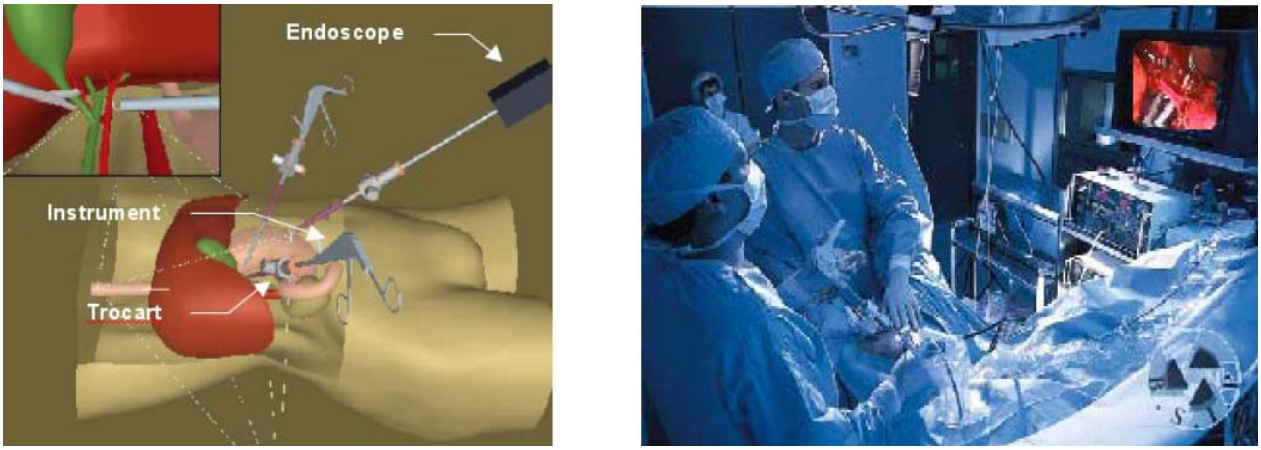


Figure 5-1 : la chirurgie endoscopique (instruments, vision).

A l'état normal, le contenu de l'abdomen est en contact étroit avec la paroi musculaire. Pour obtenir un espace qui permet d'introduire les instruments nécessaires à l'opération (caméra vidéo et outils chirurgicaux), il convient donc de réaliser une sorte de 'bulle' de travail en gonflant l'abdomen. Pour ce faire, l'intervention commence par l'injection de gaz CO2 dans l'abdomen. Cet espace de travail, que l'on appelle le pneumopéritoine, est créé à l'aide d'une aiguille protégée, qui est introduite à travers la paroi abdominale. Lorsque l'espace de travail a été créé, le chirurgien utilise des 'trocarts', qui sont des gaines munies de valves, qui permettent de contenir le gaz dans l'abdomen. C'est par l'intermédiaire de ces trocarts que sont introduits la caméra vidéo et de longs et fins instruments

chirurgicaux (5-12mm de diamètre, 20-30cm de long). Ces trocarts sont mis en place par des petites incisions cutanées réalisées sur la paroi abdominale. Par la suite, l'intervention se déroule 'à ventre fermé', le chirurgien manipulant les instruments depuis l'extérieur de l'abdomen, mais en suivant les différentes manipulations à l'intérieur du ventre sur un écran vidéo. Il est évident que les instruments chirurgicaux utilisés sont différents de ceux de la chirurgie conventionnelle, c'est-à-dire à 'ventre ouvert'. Certains de ces instruments, comme les agrafeuses automatiques, sont des mécanismes complexes, difficiles à stériliser et donc utilisés sur un seul patient. On qualifie ces instruments 'd'usage unique', ce qui signifie qu'ils sont jetés après l'opération.

Cette technique offre néanmoins des avantages importants car elle permet de réduire la taille des incisions et donc le traumatisme pour le patient. Cet aspect apporte une diminution du coût par une réduction du temps de séjour à l'hôpital. Cependant, en l'état des pratiques, les instruments de chirurgie mini-invasive sont longs, rigides et doivent être manipulés sans provoquer de traumatisme par le chirurgien autour d'un centre de rotation défini par le passage du trocart au travers de la peau. Cette pratique pose toutefois aux chirurgiens des problèmes de fatigue et de maniabilité des instruments, qui peuvent augmenter le risque de complications chirurgicales. Le travail et l'espace chirurgical sont plus complexes. Le chirurgien et son équipe sont, tout au long de l'intervention, penchés sur le patient et guidés par une simple image bidimensionnelle sur l'écran vidéo grâce à une caméra CCD située à l'extrémité de l'endoscope. Pour palier ces inconvénients et pour améliorer le confort de l'équipe chirurgicale, de nombreux chercheurs et quelques industriels aux USA [Sackier_94] se sont très tôt investis dans l'amélioration de la chirurgie coelioscopique en développant des systèmes robotisés avec l'objectif premier d'améliorer, par une instrumentation adaptée, l'efficacité de l'acte chirurgical, d'augmenter le confort et de limiter la fatigue du chirurgien.

[Payandeh_01] a présenté les résultats préliminaires des techniques d'imagerie pour la chirurgie laparoscopique donnant des informations spatiales et diagnostiques. En 2003 l'équipe de H. Tang a proposé un robot laparoscopique pour

les chirurgiens gynécologues utilisant des faisceaux laser [Tang_03]. S. Ko a présenté en 2004 une méthode pour l'intervention d'un robot chirurgien laparoscopique qui identifie les instruments chirurgicaux depuis l'image laparoscopique [Ko_04].

Sur le plan de l'amélioration de la vision on commence à trouver des stéréo-endoscopes mais ceux-ci sont fatiguants à l'usage et n'apportent souvent qu'une dimension qualitative du relief observé. En conséquence la conception de stéréo-endoscopes capables d'apporter en temps réel aux chirurgiens une information 3D favorise la précision de l'acte chirurgical et ouvre les portes de la chirurgie assistée par ordinateur.

5.2 La chirurgie laparoscopique 3D

5.2.1 Historique

C'est en 1904 que L. et H. Loewenstein déposent un brevet exposant le principe de leur stéréo-cystoscope. Malheureusement, celui-ci n'a pas pu être réalisé à cause du manque de performance de la technologie optique de cette époque. C'est seulement dans les années 1970 que R. Le loup invente un endoscope stéréo de bonne qualité mettant en œuvre un nouveau type de lentilles aux propriétés optiques fortement améliorées. Malgré l'intérêt suscité par ce nouvel équipement, son usage s'est peu répandu et la communauté a continué à travailler avec des endoscopes monoscopiques.

Avec l'extension de la chirurgie endoscopique à d'autres spécialités exigeant des manœuvres plus complexes, un intérêt croissant pour la vision 3D s'est manifesté. Cependant, pour couvrir le champ de la chirurgie endoscopique générale, la technologie n'est pas toujours suffisamment avancée pour fournir les systèmes qui peuvent rivaliser avec la qualité visuelle de bons endoscopes monoscopiques.

5.2.2 Etat de l'art

Le robot DaVinci (cf. figure 5-2) de la société Intuitive Surgical est le premier qui a été installé dans un bloc opératoire afin de réaliser des interventions

chirurgicales mini-invasives sur l'homme dans les domaines digestif et cardiovasculaire en 2000.

Figure 5-2 : Robot Da Vinci et son instrumentation articulée (EndoWrist) commandable manuellement a partir de deux manipulateurs.

Le système Da Vinci est composé dans sa version de base de trois structures porteuses permettant le placement des trois bras robotisés au dessus du patient (la dernière version du Da Vinci est composé de 4 bras : 3 instruments et un endoscope 3D [ISI]). L'ensemble des bras robotisés, fixé sur une même base, manipule l'endoscope et les instruments.

Figure 5-3 : Endoscope stéréoscopique du robot DA VINCI.

Le système de vision du Da Vinci est un des premiers systèmes commerciaux à offrir au chirurgien une vision tridimensionnelle de la zone opératoire [Burns_04] au moyen d'un endoscope à optique stéréoscopique et d'une projection séparée des images fournies par deux cameras vidéo (voir figure 5-3). Le système de commande du robot Da Vinci est basé sur une interface permettant au chirurgien d'orienter le

bras robotisé et son instrumentation comme s'il pratiquait l'intervention avec ses propres mains en chirurgie ouverte.

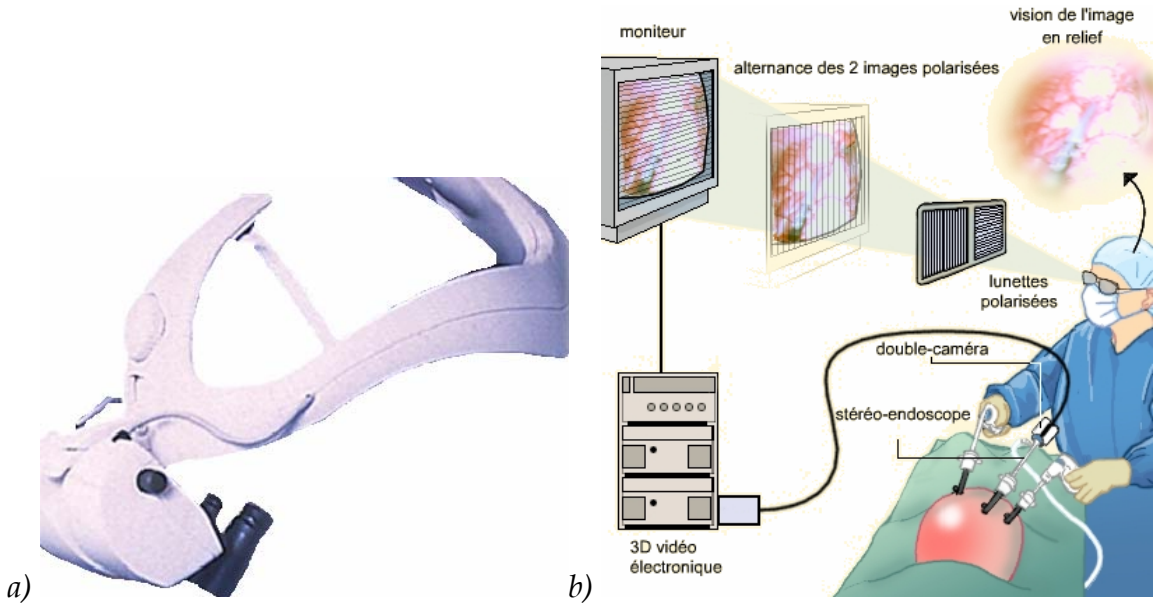


Figure 5-4 : a) casque posé sur la tête avec deux matrices optiques. b) séparation de projections par des lunettes spéciales.

Actuellement sur le marché, on trouve deux systèmes de stéréo-endoscopie. Le premier est basé sur le transport des deux projections gauche et droite de la scène devant les yeux du chirurgien par l'intermédiaire d'un casque posé sur la tête (figure 5-4.a). Des travaux, réalisés par l'Université de Caroline du Nord en collaboration avec la société InnerOptic Technology [ITO], ont permis de réduire l'offset entre les axes des projections et ceux des yeux du chirurgien (voir la figure 5-5) afin de se rapprocher de la réalité comme dans une opération à ventre ouvert [State_05]. Le second se présente comme un tableau de bord possédant deux ouvertures optiques par lesquelles le chirurgien peut suivre l'opération : c'est le cas du système de vision 3D installé sur le robot DaVinci (cf. figure 5-3).

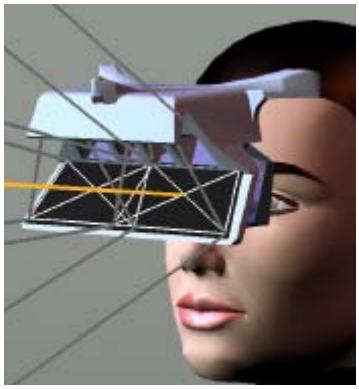


Figure 5-5: système de guidage pour suivre les axes optiques.

Le système 3D de KARL STORZ (cf. figure 5-4.b) est basé sur la technique qui consiste à visualiser deux images gauche et droite, préalablement polarisées verticalement et horizontalement, alternativement sur un même écran [Garcia_05], [Storz_03]. La perception du relief est obtenue par le port de lunettes polarisées.

Dans tous ces systèmes, la vision 3D est uniquement qualitative et ne donne que la sensation du relief puisque l'effet 3D est créé par le cerveau du chirurgien. Les progrès de la chirurgie et l'implication croissante des robots pour la réalisation d'actes chirurgicaux délicats et précis assistés par ordinateur, nécessitent la mise en œuvre de systèmes de vision 3D temps réel qualitatifs et quantitatifs. C'est dans ce contexte que nous avons évalué la faisabilité d'un stéréo-endoscope basé sur les techniques précédemment décrites dans ce mémoire et mettant en œuvre deux endoscopes du commerce.

5.3 Principe d'un endoscope

Qu'il soit à finalité médicale ou industrielle, l'endoscope a pour but d'obtenir des images de zones inaccessibles à l'œil nu. Pour assurer cette fonction on trouve sur le marché trois types d'endoscopes : l'endoscope rigide, l'endoscope souple et le vidéo-endoscope.

5.3.1 L'endoscope rigide

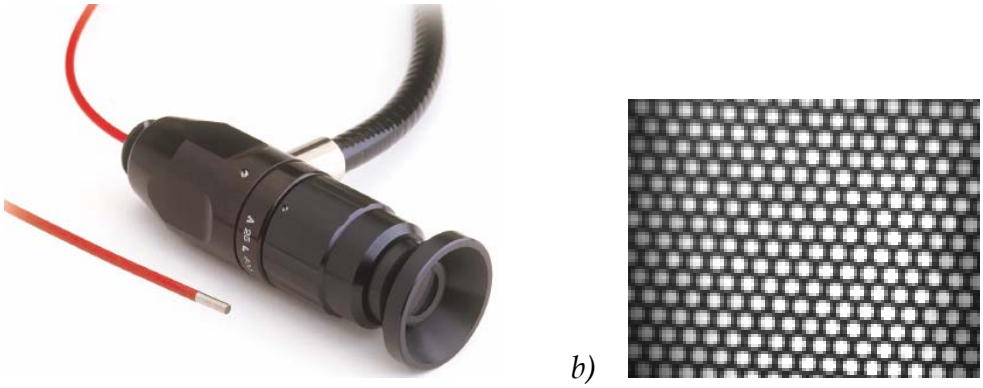
C'est l'endoscope le plus utilisé en chirurgie laparoscopique. Il fournit des images de très bonne qualité et représente une méthode simple pour visualiser des

scènes difficiles d'accès. Il est constitué d'un tube rigide et d'un ensemble de lentilles cylindriques qui renvoient l'image en bout du tube. Moyennant des bagues d'adaptation interchangeables, cette image peut-être exploitée directement à l'œil nu ou au travers d'un moniteur vidéo (voir figure 5-6). Malgré sa simplicité et son principe de fonctionnement, ce type d'endoscope ne peut pas être fléchi et par conséquent se révèle difficilement exploitable sur un porte-outil chirurgical.

Figure 5-6 : Endoscope rigide.

5.3.2 L'endoscope flexible

Le tube rigide est remplacé par une gaine souple renfermant une multitude de fibres optiques de quelques dizaines de microns de diamètre échantillonnant ainsi l'image et la ramenant vers un oculaire ou un capteur d'image type CCD. Il présente l'avantage de pouvoir être facilement utilisé pour des trajectoires complexes (voir figure 5-7.a) et peut parfaitement s'intégrer dans un porte-outil chirurgical. Toutefois il impose un rayon de courbure limite au-delà duquel un stress mécanique apparaît qui détériore les fibres optiques. La qualité de l'image obtenue est moins bonne qu'avec un endoscope rigide car il apparaît dans celle-ci un « maillage » lié à l'échantillonnage optique. Ce maillage s'avère pénalisant lors de la mise en œuvre des algorithmes de stéréovision passive dense (voir figure 5-7.b).



a) *b)*

Figure 5-7 :a) Endoscope flexible. b) Effet de « maillage » sur une image (après grossissement)

5.3.3 Le vidéo-endoscope

Dans ce dernier type d'endoscope, le capteur d'image (CCD, CMOS) se trouve en bout de l'endoscope au plus prés de la scène à observer et toute l'électronique de commande est déportée. Les interconnexions entre les deux parties sont réalisées par des fils électriques enveloppés dans une gaine en plastique souple. Avec la miniaturisation des capteurs d'image, cette solution tend à se répandre de plus en plus car elle offre la meilleure qualité d'image et la plus grande flexibilité. Elle permet donc de nous affranchir des deux problèmes essentiels posés par notre application.

a)  *b)* 

Figure 5-8 : a) le vidéo-endoscope. b) la matrice CCD du vidéo-endoscope.

5.4 Constitution du banc de stéréo-endoscopie

5.4.1 Choix des vidéo-endoscopes

Pour des raisons de coût nous avons utilisé des vidéo-endoscopes destinés à des applications industrielles de la société Fort Imaging (figure 5-8 a). Ils sont constitués :

- d'un capteur CCD Sony de 1/6ème de pouce de 440 000 pixels associé à une optique de 60° d'ouverture. Le diamètre extérieur de l'ensemble n'excède pas 5,5 mm.
- d'un câble de liaison de 4 mètres de long et de 2,8 mm de diamètre.
- d'un boîtier de commande qui a pour rôle de fournir les timings de fonctionnement du CCD, les alimentations et générer en sortie un signal vidéo aux standards RS170 et S-VHS.

5.4.2 Interfaçage avec le banc PICASO

A ce jour seule l'interface avec le banc PICASO a été réalisée. Elle est assurée par la mise en œuvre de deux cartes du commerce réf : NI-PCI 1405 acceptant les standards S-VHS et RS170.

5.4.3 Montage mécanique

Le banc d'essai est un montage mécanique simple des deux vidéo-endoscopes avec une base entre les centres optiques de 10 mm (figure 5-9.a). La figure 5-9.b représente le champ de vision utile ainsi que la profondeur de la scène observable.

Figure 5-9 : a) le banc d'essai. b) Géométrie du banc.

5.5 Calibration et rectification

Pour appliquer l'algorithme de corrélation census basé sur la recherche des points stéréo-correspondants, il est indispensable de procéder au calibrage des caméras. Cette opération permet d'obtenir deux tableaux de rectification qui corrigent les distorsions liées aux objectifs et l'alignement entre les deux images gauche et droite.

Pour calibrer le banc des µ caméras, nous avons pris une quinzaine d'images d'une mire (voire la figure 5-10) avec plusieurs positions (voir la figure 5-11 à gauche).

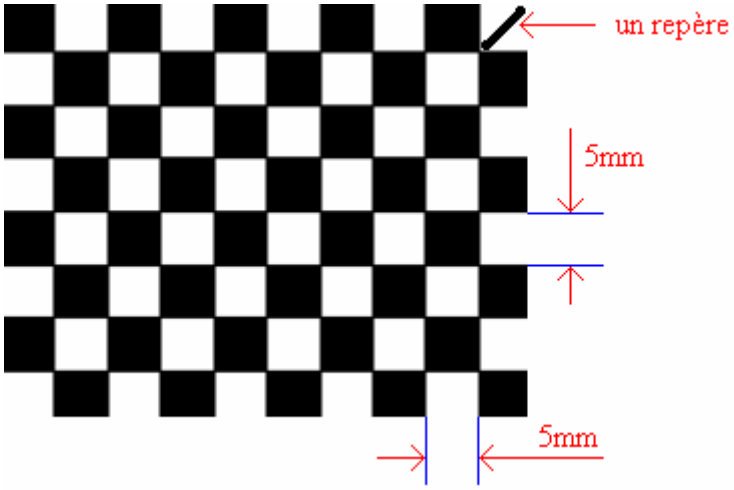


Figure 5-10 : Mire de calibrage du µbanc.

Pour obtenir les tableaux de rectification nous avons utilisé les boîtes à outils MatLab. Celles-ci nous permettent, à partir d'une série de prises de vue d'une mire, de calculer les deux matrices de correction de la distorsion, les deux matrices de rectification des images et tous les paramètres intrinsèques et extrinsèques des optiques. La figure 5-11 de droite représente les images après application des paramètres de rectification.

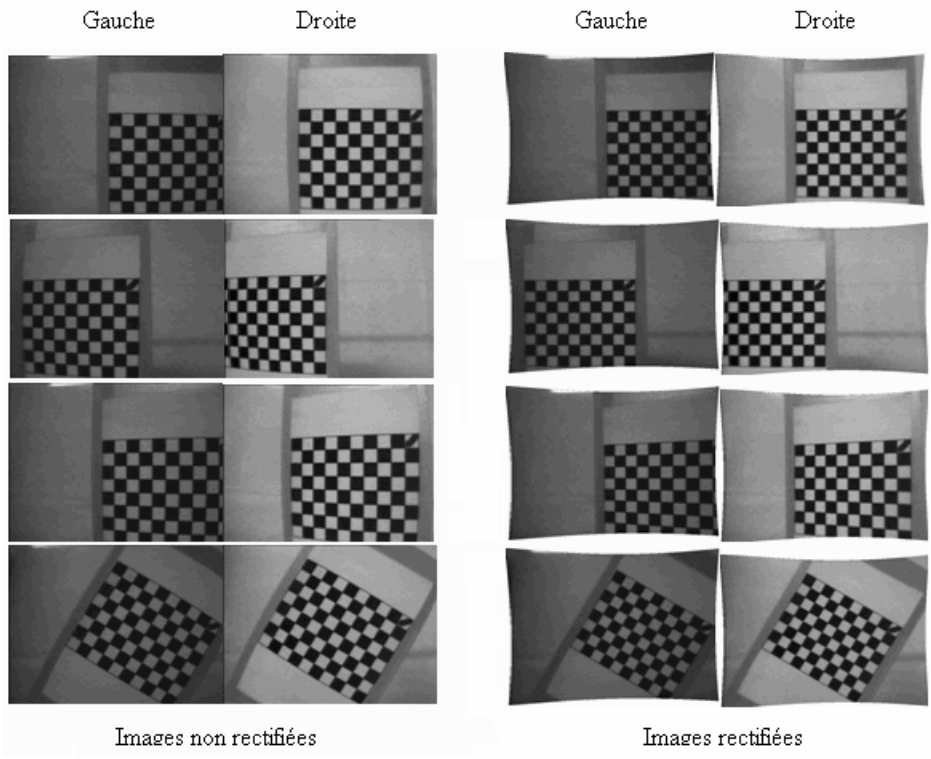


Figure 5-11 : La mire de calibrage et les images rectifiées.

5.6 Résultats obtenus

L'étape précédente nous a permis de déterminer les valeurs de α_v (un des paramètres intrinsèques) et de B (distance entre les centres optiques). A partir de ceux-ci et de la relation ci-dessous on peut donc déterminer la correspondance entre la disparité et la profondeur d'un point de la scène.

$$d = a_v B / Z$$

La figure 5-12 représente, pour α_v = - 857 (valeur donnée par la calibration) et B variant de 0.5 cm à 3 cm par pas de 0.5 cm, l'influence de la base (B) sur la disparité maximale en fonction de la profondeur Z à partir de laquelle la scène est observable. La visualisation de points 3D proches des centres optiques entraine une augmentation de la valeur de la disparité maximum (D_{max}) et par conséquent une consommation de ressources du circuit FPGA accrue. En revanche c'est dans cette zone qu'est obtenue la meilleure précision.

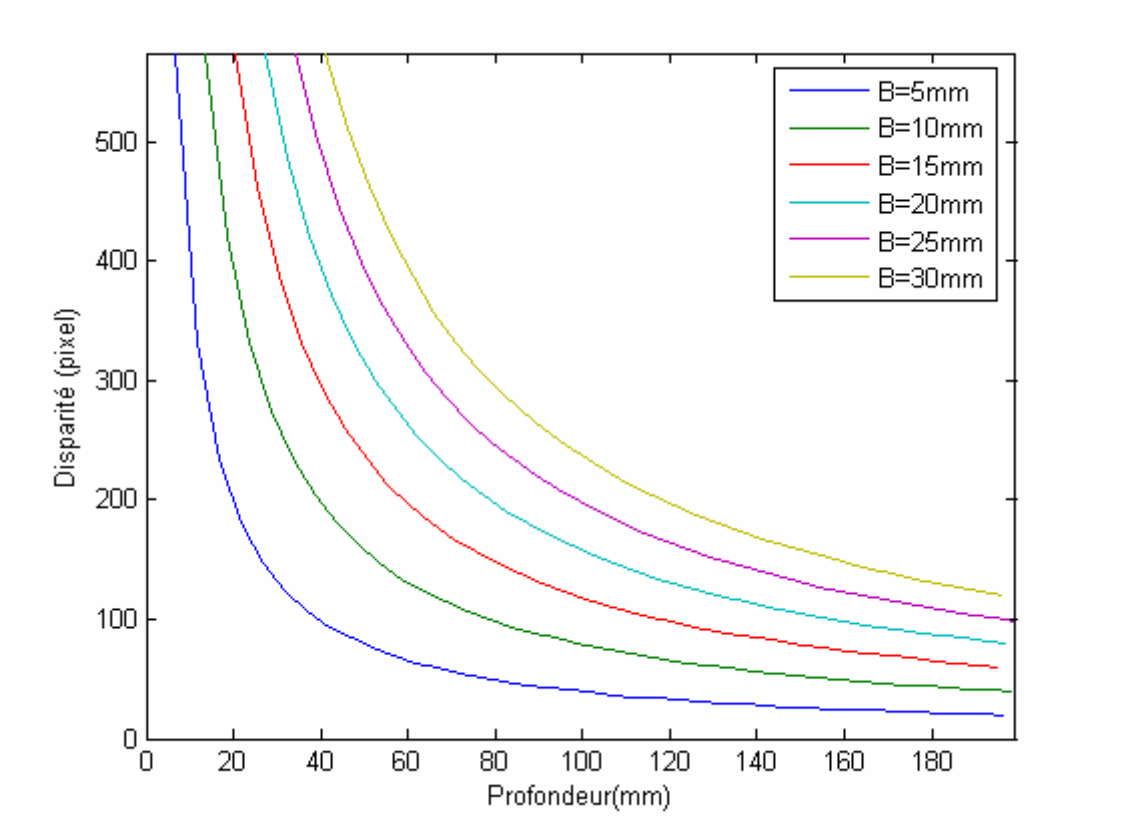


Figure 5-12 : Rapport entre disparité et profondeur en fonction de la base B (5 : 5 : 30) mm.

A titre d'exemple l'observation d'une zone définie par deux plans situés aux profondeurs Z_{max} = 100 mm et Z_{min} = 80 mm, entraine le balayage des pixels 85 à 107 de chaque ligne pour la recherche des meilleurs scores.

La figure 5-13 (bas) représente la carte de disparités d'un circuit électronique obtenue avec le µbanc à partir d'une fenêtre census de 7x7 et une valeur d_{max} de 100 pixels. Les niveaux noirs correspondent aux points les plus proches des centres optiques, les blancs représentent soit les points les plus éloignés, soit ceux pour lesquels aucune corrélation n'a été trouvée.

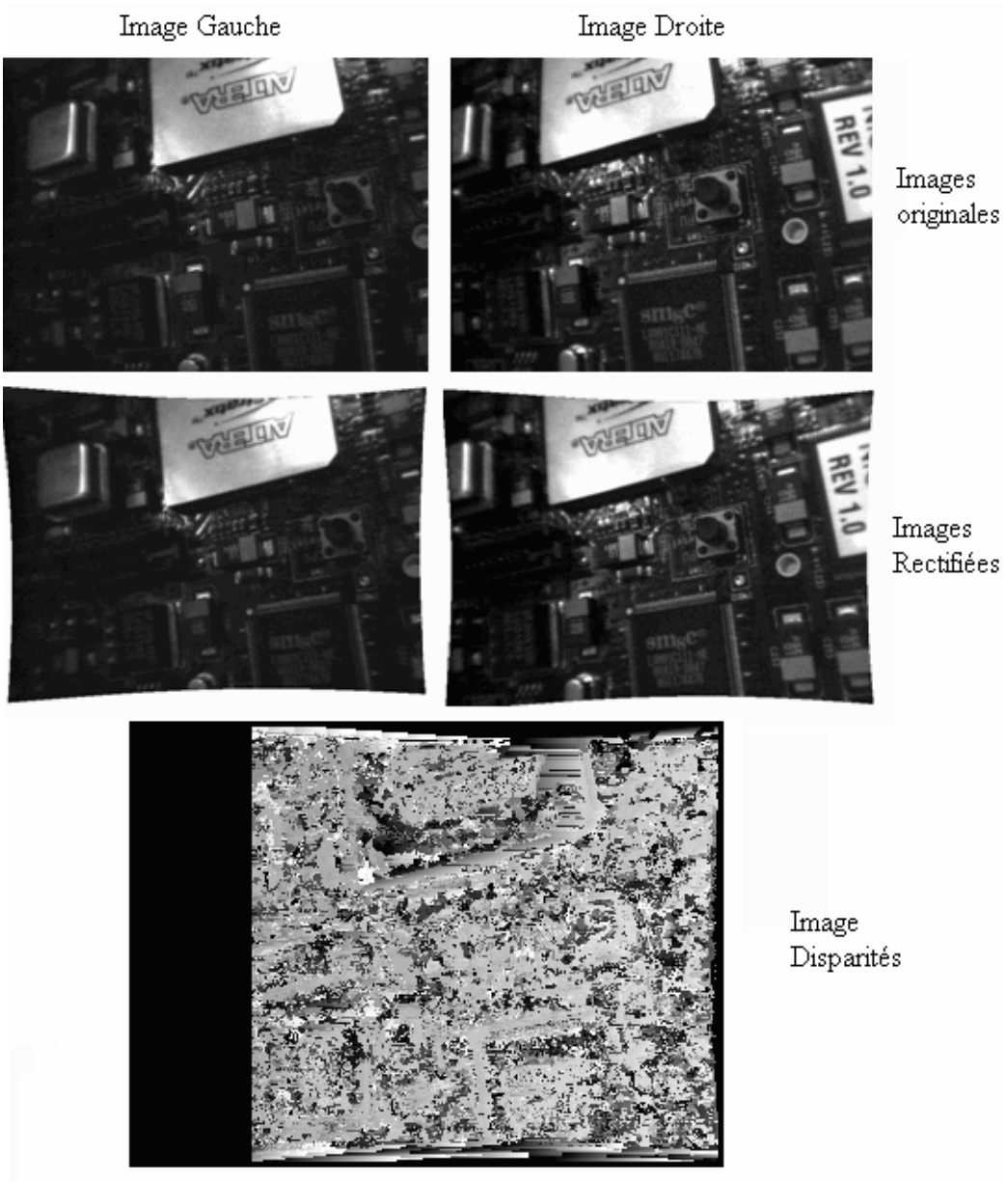


Figure 5-13 : Exemple d'image de disparités (circuit de composants électroniques).

5.7 Conclusion

Dans ce chapitre nous avons présenté une version du banc initial adapté à la problématique de la vision stéréo-endoscopique et avons montré que les exigences de miniaturisation de cette instrumentation pouvaient être remplies avec la technologie actuelle. Ainsi l'association de vidéo-endoscopes miniaturisés et d'une unité de traitement d'images embarquée dans un FPGA et basée sur les techniques de stéréovision dense permet de répondre aux besoins de la chirurgie laparoscopique moderne à savoir :

- embarcation sur un porte-outil chirurgical.
- obtention d'images 3D qualitatives et quantitatives.
- respect des contraintes temps réel pour la chirurgie robotisée.

Il reste cependant à mener :

- une phase d'évaluation sur des images médicales prises dans des conditions adéquates.
- Une phase de filtrage des images de disparité afin de réduire les artefacts.

5.8 Références Bibliographiques

[Burns_04] Burns C. R., Tierney M. S., Gere D., Stern S. D., Stereo imaging system and method for use in telerobotic systems. Brevet d'invention US6720988, Intuitive Surgical Inc., 13 avril 2004.

[Garcia_05] A. Garcia et D. Mutter, 'Vision 3D', http://www.websurg.com/ref/Vision_3D-ot02fr309.htm, 2005.

[ISI] Intuitive Surgical Inc., http://www.intuitivesurgical.com/news_room.

[ITO] Inner Technology Optic inc., http://www.inneroptic.com/index.htm.

[Ko_04] S.KO et D.Kwon, 'A Surgical Knowledge Based Interaction Method for a Laparoscopic Assistant Robot', IEEE International Workshop on Robot and Human Interactive Communication, Japan, Septembre 2004.

[Payandeh_01] Sh.Payandeh, X.Zhang et A.Li, 'Application of Imaging to the Laparoscopic Surgery', IEEE International Symposium on Computational Intelligence in Robotics and Automation, Canada, Juillet 2001.

[Sackier_94] J.M.Sackier et Y.Wang, 'Robotically assisted laparoscopic surgery: From concept to development', Dans Surgical Endoscopy vol. 8, 1994.

[State_05] A. State, K.P. Keller et H. Fuchs, 'Simulation-Based Design and Rapid Prototyping of a Parallax-Free, Orthoscopic Video See-Through Head-Mounted Display', 5th International Symposium on Management of Aquifer Recharge (ISMAR05), Berlin, Germany, 11-16 June 2005.

[Storz_03] Karl Storz inc., 'Karl Storz Tricam 3D Imaging System Unwrapped with the First Live 3D Telesurgery', Society of Laparoendoscopic Surgeons (SLS), USA, 2003.

[Tang_03] H.Tang, H.Van Brusse, D.Reynaerts, J.Vander Sloten et P.R.Koninckx, 'ALaparoscopic Robot with Initive Interface for Gynaecological Laser Laparoscopy', IEEE International conference on Robotics & Automation, Taiwan (Taipei), séptembre 2003.

Chapitre6

Conclusion et Perspectives

6 CONCLUSION GENERALE ET PERSPECTIVES

6.1 Conclusion générale

Le travail présenté dans ce mémoire s'est inscrit dans le cadre du projet interne PICASO dédié à la fusion d'images multi-longueurs d'ondes et visait deux objectifs :

- la conception d'architectures matérielles pour la stéréovision temps réel dans le contexte de la détection d'obstacles dans le domaine des transports.
- L'application de ces travaux au domaine de la chirurgie laparoscopique assistée par ordinateur notamment par la conception d'un stéréo-endoscope.

Après avoir brièvement passé en revue quelques méthodes adaptées à la vision 3D temps réel, nous avons présenté plus en détail la méthode de stéréovision passive dense, basée sur la vision stéréoscopique humaine, qui génère autant de points 3D que de pixels dans les zones perçues par deux caméras. Cette méthode nécessitant pour créer une image de disparité, l'extraction dans les deux images droite et gauche d'une même scène, de points stéréo-correspondants, nous avons évalué quelques critères de ressemblance. Dans ce contexte nous avons été amenés à concevoir un banc générique à base de caméras CMOS et interfaces Camera-Link fonctionnant sur PC dans l'environnement LabWindows CVI de National Instruments et facile d'emploi par la simplicité de son interface graphique. Ce banc nous a permis d'une part, moyennant le développement d'applicatifs en langage C ANSI, de procéder à la calibration des caméras et de calculer les matrices de rectification des images et d'autre part, d'effectuer un certain nombre d'essais prenant en considération les critères de ressemblance, les différents paramétrages ainsi que les temps calcul. Ces derniers critères nous ont fait retenir la méthode du Census comme méthode d'appariement, pour sa robustesse et la relative simplicité

de son algorithme. Bien qu'initialement dévolu à un partenaire du projet, nous avons également pris en considération la problématique de la calibration des caméras et la rectification des images en temps réel sans lesquelles il est impossible d'obtenir une carte de disparité convenable indispensable à toute reconstruction 3D.

Pour atteindre l'objectif d'obtention d'une carte de disparité à une cadence supérieure à 100 images par seconde en mode VGA, nous avons effectué une analyse fine de toutes les opérations à réaliser avec leurs contraintes associées et avons éliminé d'emblée les solutions programmées à base de DSP. De la même manière les outils permettant de passer d'un code écrit en langage évolué vers des solutions câblées ne nous ont pas apparues suffisamment performantes à ce jour pour notre type d'application. En conséquence nous avons opté pour une implémentation de l'architecture dans un circuit FPGA à granularité hétérogène de la société ALTERA avec une description des architectures de traitement la plus proche possible du matériel lorsque les contraintes le nécessitaient et exploité les fonctions précâblées disponibles dans le circuit. Moyennant ces différents points l'architecture globale présentée dans ce mémoire nous a permis d'atteindre, à partir de fenêtres census de taille 7x7, des cadences de 333 images de disparité par seconde en mode 400x300 et plus de 130 images par seconde en mode 640x480. Par ailleurs l'utilisation du langage VHDL comme outil de description et ses possibilités de généricité nous a permis d'obtenir un certain niveau de réutilisabilité et de paramétrage des différentes architectures limité toutefois par l'utilisation de fonctions spécifiques précâblées (blocs mémoire essentiellement).

S'agissant de l'application de la vision 3D temps réel à la stéréo-endoscopie, nous avons montré qu'il était possible d'exploiter des vidéo-endoscopes flexibles incluant exclusivement le capteur d'image et de déporter la partie commande du capteur ainsi que le traitement stéréo des images. La miniaturisation de ce dispositif et plus particulièrement des optiques qui présentent de fortes distorsions en champ proche impose cependant d'affiner l'étape de rectification. Enfin ce stéréo-endoscope a été testé dans l'environnement du banc d'évaluation car il ne nous a pas été

possible de le coupler, faute de temps, à la carte FPGA car nous ne disposions pas de l'interface vidéo adéquate.

6.2 Perspectives

Bien que nous nous soyons particulièrement attachés à concevoir des architectures rapides pour le portage de la stéréovision dense dans le domaine temps réel, de nombreuses pistes subsistent pour améliorer les performances que nous avons obtenues. Parmi celles-ci et plus particulièrement concernant le portage sur FPGA, on peut citer :

- la prise en compte de la sub-pixelisation dans les phases de rectification des images et de recherche des meilleurs scores.
- l'augmentation de la taille de la fenêtre census notamment lorsque les images traitées sont peu texturées. Cette opération peut entrainer des problèmes de routage dans le circuit FPGA et impliquer de repenser une partie de l'architecture.
- l'implantation d'un dispositif de filtrage d'image, lequel introduira inévitablement un retard supplémentaire dans l'apparition de l'image 3D.

En ce qui concerne le capteur d'image proprement dit, d'autres améliorations peuvent être apportées, notamment par le fort développement que connaissent aujourd'hui les capteurs CMOS-APS. On peut en effet envisager d'intégrer au capteur d'image une table de rectification téléchargeable et prenant en compte les défauts géométriques de montage et d'optique. La capacité des imageurs CMOS à pouvoir être lus de manière aléatoire facilite cette opération et permet ainsi de disposer d'images temps réel rectifiées sans introduction de temps de latence lié aux différents stockages intermédiaires.

S'agissant de la partie stéréo-endoscopie, il est également envisageable d'intégrer au capteur un convertisseur analogique numérique à sortie série-différentielle : celle-ci offre l'avantage de débits pouvant aller jusqu'à 800

Mégabits/sec et d'être directement interfaçable avec les circuits FPGA récents. Ces différents aspects permettront :

- D'améliorer la qualité de l'image par une numérisation au plus près de la grandeur analogique.
- De réduire le nombre et par conséquent la taille des interconnexions avec le boîtier de traitement car il est préférable pour des raisons d'extrême miniaturisation, de conserver une structure déportée.
- D'effectuer la reconstruction 3D et d'évaluer les modes de représentation dans le cadre de l'interfaçage visuel avec le chirurgien.

Enfin dans un contexte plus général une réflexion peut être menée pour voir comment un tel système de stéréo-endoscopie peut s'interconnecter avec un robot chirurgical ou participer à une technique en plein essor telle que celle de la « réalité augmentée ».