Titre: Title:	Algorithmes et architectures de calcul spécialisés pour un système optique autosynchronisé à précision accrue
Auteur: Author:	Hakim Khali
Date:	1999
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Khali, H. (1999). Algorithmes et architectures de calcul spécialisés pour un système optique autosynchronisé à précision accrue [Thèse de doctorat, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/8833/

UNIVERSITÉ DE MONTRÉAL

ALGORITHMES ET ARCHITECTURES DE CALCUL SPÉCIALISÉS POUR UN SYSTÈME OPTIQUE AUTOSYNCHRONISÉ À PRÉCISION ACCRUE

HAKIM KHALI

DÉPARTEMENT DE GÉNIE ÉLECTRIQUE ET DE GÉNIE INFORMATIQUE

ECOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION

DU DIPLÔME DE PHILOSOPHIAE DOCTOR (PH.D.)

(GÉNIE ÉLECTRIQUE)

SEPTEMBRE 1999

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée:

ALGORITHMES ET ARCHITECTURES DE CALCUL SPÉCIALISÉS POUR UN SYSTÈME OPTIQUE AUTOSYNCHRONISÉ À PRÉCISION ACCRUE

présentée par: Khali Hakim

en vue de l'obtention du diplôme de: Philosophiae Doctor

a été dûment acceptée par le jury d'examen constitué de:

M. BOIS Guy, Ph.D., président.

M. SAVARIA, Yvon, Ph.D., membre et directeur de recherche.

M. HOULE Jean-Louis, Ph.D., membre et codirecteur de recherche.

M. SAWAN Mohamad, Ph.D., membre.

M. TREMBLAY Marc, Ph.D., membre externe.

À mes parents qui ont tant attendu ce moment

À mes enfants pour le temps que je leurs ai pris

À mon épouse pour son soutien et sa patience

A mon frère et à ma soeur

Remerciements

Je souhaite exprimer toute ma reconnaissance et ma gratitude aux membres du jury:

Monsieur Guy Bois, professeur à l'Ecole Polytechnique de Montréal qui m'a fait l'honneur de présider autant le jury de mon examen de synthèse que le jury de ma soutenance de thèse.

Monsieur Yvon Savaria, professeur à l'Ecole Polytechnique de Montréal, qui a dirigé les travaux de cette thèse. Je le remercie pour les discussions fructueuses que nous avons eues et sans lesquelles ce travail n'aurait pu aboutir. Je tiens aussi à le remercier pour son soutien moral et financier, ce qui m'a permis d'amener ce travail à terme.

Monsieur Jean-Louis Houle, professeur à l'Ecole Polytechnique de Montréal, qui a codirigé les travaux de cette thèse. Je tiens à le remercier pour les discussions utiles que nous avons eues et sa disponibilité tout au long de ce travail. Finalement, j'aimerais aussi le remercier pour son soutien moral et financier.

Monsieur Mohamad Sawan, professeur à l'Ecole Polytechnique de Montréal, qui a accepté d'être membre autant du jury de mon examen de synthèse que du jury de ma soutenance de thèse.

Monsieur Marc Tremblay, professeur à l'université Laval à Québec, qui a accepté d'être l'examinateur externe de cette thèse. Je tiens aussi à le remercier d'avoir accepté de faire le déplacement pour participer à ce jury.

Je tiens aussi à remercier:

le centre d'excellence IRIS pour avoir financé les travaux de cette thèse et m'avoir donné l'opportunité de travailler sur l'imagerie 3D.

Messieurs Angelo Beraldin et Marc Rioux, chercheurs au CNRC à Ottawa, pour les discussions utiles que nous avons eues concernant l'aspect optique de cette thèse.

Finalement, je tiens à remercier Mme Ghyslaine E-Carrier, ainsi que les étudiants du groupe GRM de l'École Polytechnique de Montréal pour l'excellent climat de travail qui a régné tout au long de ces années.

Résumé

La reconstruction de la forme des objets constituent une tâche fondamentale en imagerie 3D basée sur des caméras optiques à illumination active. La fidélité et la précision de cette reconstruction va non seulement dépendre de la caméra optique utilisée, mais aussi des caractéristiques physiques et géométriques des objets considérés. Parmi les caractéristiques physiques qui altèrent la fidélité de cette reconstruction est la variation de la réflectance des objets. Cette dernière affecte directement la forme du signal laser incident sur le détecteur optique. Lorsque cette réflectance n'est pas uniforme, le signal laser incident sur le détecteur optique, originellement de distribution gaussienne, est déformé ce qui conduit à une erreur de mesure. Les méthodes de correction de cette erreur sont souvent complexes, ce qui empêche des corrections en temps réel. Elles sont parfois trop spécifiques, ce qui empêche leur utilisation de façon généralisée.

Cette thèse vise à proposer une nouvelle approche pour la correction de l'erreur introduite par une variation de réflectance et des architectures VLSI (Very Large Scale Integration ou en français ITGE pour intégration à très grande échelle) capables d'accélerer efficacement les algorithmes proposés pour des systèmes optiques de hautes performances (1 million de points par seconde et plus).

Concernant les méthodes de correction de l'erreur, une contribution scientifique a été

apportée sur la formalisation mathématique de la relation objet-détecteur optique en fonction des principaux paramètres associés à un système optique. Nous retrouvons:

- le développement de deux méthodes de correction de l'erreur engendrée par une variation de réflectance:

  - la méthode DIRECTE qui consiste à corriger l'erreur en se basant sur des données collectées lors d'une phase dite de calibration;

  - la méthode ITERATIVE qui consiste à effectuer des calculs itératifs afin de minimiser une fonction objectif. La position du minimum indique la valeur de la correction à apporter.

Concernant l'accélération des calculs associés aux méthodes proposées, une approche de conception matériel-logiciel est utilisée pour identifier les traitements qui seront respectivement effectués sur un processeur DSP (Digital Signal Processor) et sur du matériel dédié de type ASIC (Application-Specific Integrated Circuit) ou FPGA (Field Programmable Gate Array). De façon spécifique, nous retrouvons:

- un modèle mathématique basé sur la loi d'Amdhal capable de prédire l'accélération globale d'une application basée sur des tables LUT(Look Up Tables). Ce modèle met en évidence la pertinence d'utiliser les tables LUT afin d'accélérer les fonctions de calcul impliquées dans les traitements associés à des systèmes de mesure;
- un algorithme de partitionnement matériel-logiciel d'une application basée sur des fonctions pouvant être implémentées en logiciel ou en matériel sous forme de tables LUT;

- une proposition de deux architectures VLSI dédiées à l'accélération de la méthode ITERATIVE. Ces deux architectures sont comparées suivant une métrique *AT* (area/time) afin d'identifier les régions d'utilisation efficace associées à chaque architecture.

Les méthodes de correction proposées permettent une précision de correction au moins égale, et parfois supérieure, à la résolution du système optique. Une performance supérieure en temps de calcul est obtenue pour les approches d'accélération proposées par rapport à des processeurs DSP commerciaux tels que le TMS320C40 et le ADSP-21060.

Les résultats obtenus montrent la pertinence du domaine de recherche et constitue une base de départ nouvelle pour le développement de méthodes de correction plus élaborées visant des systèmes optiques de hautes performances et capables de bénéficier de l'évolution continuelle des technologies VLSI.

Abstract

The reconstruction of object shapes is a fundamental task in 3D imaging using optical cameras based on active illumination. The accuracy of this reconstruction depends not only on the chosen camera, but also on object physical and geometrical characteristics. Among the object physical characteristics that may corrupt this reconstruction is a varying reflectance which directly affects the reflected laser signal integrated by the optical detector. When the object reflectance is not uniform, the original gaussian laser spot, being integrated by the optical detector, is deformed (no longer gaussian) which leads to erroneous measures. Available correction algorithms suffer a high computational complexity which prevents real-time corrections. They are also too specific which prevents their use in all types of optical systems.

This thesis aims at proposing a new approach to correct errors caused by a varying reflectance and VLSI (Very Large Scale Integration) architectures to accelerate the proposed algorithms, targeting high-performance optical systems (1 million 3D points per second and more).

The scientific contribution consists in formalyzing the mathematical relation between the laser spot and its image on the detector including the main system parameters. It consists in:

- developping two methods to correct the error caused by a varying reflectance. These methods are:

  - the DIRECT method which corrects the errors based on data collected during a calibration phase;

  - the ITERATIVE method which tries to minimize an objective function. The position of the optimum indicates the corresponding correction.

Concerning the acceleration of the proposed methods, a hardware-software codesign approach is used to identify the computations that are respectively suitable to run on a DSP (Digital Signal Processor) and on dedicated hardware like an ASIC (Application-Specific Integrated Circuit) or a FPGA (Field Programmable Gate Array). More specifically, we have:

- a mathematical model based on Amdhal's law to predict the global acceleration of an application based on LUT (Look Up Tables). This model shows the importance of using LUTs to accelerate the computations involved in metrology-based systems;
- a hardware-software partitioning algorithm which targets applications based on functions that can be implemented in software or in hardware as LUTs;
- two VLSI architectures to accelerate the ITERATIVE correction algorithm. These two architectures are compared based on an AT (area/time) metric to identify the corresponding suitable regions of utilization.

The proposed correction methods show a resulting accuracy at least equal to system

resolution, and sometimes better. The proposed VLSI architectures and acceleration approaches show a better performance over commercial DSPs, like the TMS320C40 and ADSP-21060.

The obtained results show the pertinence of the work in this research area and constitutes a new basis for the development of more powerful correction methods which target high-performance optical systems. These new correction methods should benefit from the continuous evolution of VLSI technologies.

Table des matières

Liste des figures

Liste des tables

Liste des sigles et abréviations

ASIC: Application-specific integrated circuit.

CCD: Charged coupled device.

CLB: Configuration logic bloc.

CORDIC: Coordinate rotation digital computer.

COREGEN: Core generator.

CPI: Cycles per instruction.

CPU: Central processing unit.

DRAM: Dynamic random access memory.

DSP: Digital signal processor.

FFT: Fast Fourier transform.

FIFO: First in first out.

FPGA: Field programmable gate array.

HPRC: High-performance reconfigurable computing.

IFFT: Inverse fast Fourier transform.

ITGE: Intégration à très grande echelle.

LUT: Lookpup table.

MTF: Modulation transfer function.

PE: Processeur élémentaire.

RISC: Reduced instruction set computer.

RTR: Run-time reconfiguration.

SDRAM: Synchronous dynamic random access memory.

SHARC: Super Harvard architecture.

SRAM: Static random access memory.

SSRAM: Synchronous static random access memory.

VHDL: Very high-speed integrated circuit hardware description language.

VLSI: Very large scale integration.

Liste des principaux symboles

AT: produit des métriques de coût et performance.

$Cost(F_{x,y})$: opérateur de coût associé à la partition $F_{x,y}$.

D: distance séparant les centre de deux pixels adjacents.

Δp: variation de la position p du centre de masse du spot laser réflechi sur le détecteur optique.

Δz: variation de la profondeur z suite à une variation de position Δp.

F: ensemble des fonctions F_i à partitionner.

$F_{x,y}$: partition F composée de x modules mémoires de type 1 et de y modules mémoires de type 2.

G_s: accéleration globale au niveau système de l'application.

$I(k)$: intensité du pixel k.

$I_{ref}(k)$: intensité du pixel k pour une réflectance uniforme unitaire.

M_l: facteur de magnification latérale.

M_r: facteur de magnification longitudinale.

$\bar{p}^{(j)}$: image sur le pixel j du centre de masse $\bar{x}^{(j)}$.

$P(F_{x,y})$: opérateur de performance associé à la partition $F_{x,y}$.

Q_{mem} : quantité de mémoire requise par une application pour une implantation par tables LUT.

$R(x)$: réflectance associée au point x sur l'objet.

σ : ecart type du spot laser source à distribution gaussienne.

$size(f)$: opérateur de coût permettant d'estimer la quantité de mémoire requise par la fonction f lorsqu'elle est implantée sous forme de table LUT.

W : largeur d'un pixel.

$\bar{x}^{(j)}$: centre de masse sur l'objet des points ayant contribué à l'énergie du pixel j.

(x, y, z) : coordonnées cartésiennes d'un point dans l'espace.

Introduction

La représentation et visualisation des objets en trois dimensions est d'une grande importance dans de nombreuses applications, telles que les applications médicales, spatiales, militaires, etc. La représentation et visualisation en trois dimensions, aussi appelée couramment imagerie 3D, est essentiellement basée sur l'enveloppe 3D des objets à considérer (Rioux, 1994). Cette enveloppe 3D peut être reconstituée de façon synthétique en utilisant des algorithmes spécifiques d'infographie, ou en utilisant des systèmes spécialisés d'acquisition de données. Ces systèmes d'acquisition se divisent en général en deux grandes catégories: systèmes avec contact et systèmes sans contact. Les systèmes avec contact sont constitués d'une sonde qui est posée physiquement sur la surface de l'objet. Chaque position de la sonde sur l'objet permet l'acquisition d'un point 3D. L'enveloppe 3D sera obtenue lorsque la sonde aura entièrement parcouru la surface de l'objet. Ces systèmes sont caractérisés par leur grande précision mais souffrent d'une vitesse d'acquisition réduite. Les systèmes sans contact sont basés sur des caméras optiques et consistent à illuminer directement ou indirectement la surface de l'objet avec une source de lumière. Chaque point 3D de l'objet est reconstitué grâce à la connaissance de la quantité de lumière réfléchie et de la géométrie de la caméra. Ces systèmes optiques sont caractérisés par leur vitesse d'acquisition mais souffrent d'une perte de précision comparée à celle obtenue par les systèmes avec contact. Cette perte de précision est en grande partie due aux bruits parasites, à la géométrie de l'objet, à l'uniformité de sa surface et à son indice de réflectance. L'indice de réflectance, compris entre 0 et 1, permet de quantifier

la quantité de lumière réfléchie par l'objet suite à une illumination: si une quantité de lumière E est projetée sur une surface de réflectance R, alors la quantité de lumière réfléchie est évaluée à $E \cdot R$. En général, plus la valeur $E \cdot R$ est élevée, plus la mesure est précise. Lorsque la surface illuminée présente un indice de réflectance uniforme (constant pour l'illumination courante), la lumière incidente sur l'objet sera réfléchie de façon uniforme. Cependant, lorsque l'indice de réflectance n'est pas uniforme, par exemple en présence d'un saut de réflectance, la lumière incidente ne sera plus réfléchie de façon uniforme et présentera des distorsions qui seront la cause de l'erreur de mesure. Plusieurs travaux de recherche ont été réalisés afin de compenser les erreurs de mesure engendrées par un indice de réflectance variable. Mundy et Porter (1987) ont développé une méthode appelée "Difference Ratio Normalization". Cette méthode consiste à illuminer la scène avec deux "spots" à distribution gaussienne, séparés par une petite distance d. Pour chaque illumination, l'intensité du signal incident sur le détecteur optique est donnée par: $I_i(x) = P_i(x) \cdot R(x)$, où $P_i(x)$ est le signal optique incident sur l'objet et $R(x)$ la réflectance correspondante. La sortie $N(x)$ du détecteur optique est donnée par: $N(x) = \dfrac{I_1(x) - I_2(x)}{I_1(x) + I_2(x)} = \dfrac{P_1(x) - P_2(x)}{P_1(x) + P_2(x)}$. Cette méthode présente un ensemble de limitations, à savoir:

- La méthode suppose que les deux illuminations $P_1(x)$ et $P_2(x)$ ont vu le même signal de réflectance $R(x)$. Cette hypothèse, quoique plausible, n'est pas toujours vérifiée en pratique.
- Le point 3D associé à la mesure courante est donné par $N(x)$. Trouver le zéro de

$N(x)$ est une tâche difficile en raison de la taille finie des pixels et du bruit pouvant corrompre le signal optique. De plus, l'expression analytique de $N(x)$ est très spécifique et ne peut être utilisée avec des détecteurs conventionnels.

- Chaque point 3D de l'objet est obtenu par une double illumination de la surface. Cette technique limite le débit du système en raison d'une sous-utilisation de la source de lumière.

Levine (1992) a décrit une méthode qui consiste à ajuster le centre de masse d'une image en se basant sur la distribution de son intensité. Cet ajustement est effectué le long de plusieurs axes équidistants. Pour chaque axe, un ensemble de paramètres sont calculés, puis inclus dans une fonction globale de pondération. Cependant, selon Levine, en se basant sur une analyse statistique, cette méthode n'a pu améliorer de façon répétitive la correction de l'erreur introduite par une variation de réflectance. Soucy *et al* (1990) ont développé un modèle mathématique capable d'estimer l'erreur engendrée par une variation de réflectance. Cette erreur est proportionnelle à l'intégrale de la variance du signal optique reçu. Les auteurs rapportent qu'un tel modèle peut être utilisé comme un bon prédicteur de la position de la discontinuité de réflectance. Malgré cela, la complexité de la correction apportée rend difficile l'utilisation d'une telle méthode pour des corrections en temps réel dans des systèmes optiques de hautes performances. Finalement, la meilleure approche recommandée afin de compenser les erreurs introduites par une variation de réflectance est de produire des signaux optiques de très petites tailles. Le contrôle de la taille et de la forme du signal optique (Kim, 1988) exige une électronique de pointe et de haute technologie, ce qui conduirait inévitablement à une augmentation

considérable des coûts du système. Les algorithmes de correction sont souvent complexes, ce qui limite les performances des systèmes optiques. Ils sont parfois trop spécifiques, ce qui empêche leur utilisation de façon généralisée. Lorsque le débit des systèmes optiques est faible, la complexité des algorithmes proposés ne constitue pas un problème majeur. Cependant, pour des systèmes optiques à haut débit, les algorithmes proposés ne sont plus adéquats et ils constituent une limitation majeure. En raison des développements technologiques réalisés dans les domaines des capteurs optiques et des lasers, les systèmes optiques modernes sont de plus en plus performants et ciblent des débits supérieurs à 1 million de points 3D/seconde. Une telle caméra optique est en cours de développement au CNRC (Conseil national de la recherche du Canada) à Ottawa. Cette caméra cible un débit variant entre 1 et 10 millions de points 3D/seconde. Elle est appelée caméra optique à balayage autosynchronisé. Cette réalité nous a motivés à ré-examiner la correction de l'erreur associée à une variation de réflectance. Notre objectif est double: proposer une nouvelle approche pour la correction de l'erreur introduite par une variation de réflectance et des architectures VLSI dédiées capables de supporter efficacement les algorithmes proposés et les équations permettant de calculer les coordonnées cartésiennes du point 3D associé à la mesure courante. Dans cette thèse, le chapitre 1 présente les principales notions d'optiques ayant été utilisées dans la formalisation du problème à résoudre. Une attention particulière est accordée à la description de la caméra autosynchronisée. Le chapitre 2 présente les principaux facteurs affectant la qualité d'une mesure optique, et tout particulièrement la variation de l'indice de réflectance. Le chapitre 3 présente les deux méthodes de correction que nous avons développées pour

corriger l'erreur engendrée par une variation de l'indice de réflectance. Ces méthodes sont analysées suivant leurs coûts, précisions et complexités. Le chapitre 4 présente une analyse de la complexité des transformations géométriques permettant d'établir la relation entre la mesure courante et le point 3D correspondant. Des méthodes de calcul et des architectures VLSI dédiées sont proposées pour accélérer de telles transformations. Finalement, une conclusion présente une synthèse des travaux effectués, les résultats atteints et les axes de recherche à poursuivre.

Chapitre 1 : Les caméras optiques à illumination active

Dans ce chapitre, nous allons présenter les principales notions d'optiques nécessaires à la compréhension des travaux effectués durant cette recherche. Nous allons tout particulièrement décrire la caméra à balayage autosynchronisé qui servira de référence pour l'ensemble des travaux présentés.

1.1 Notions fondamentales

L'imagerie numérique 3D moderne consiste à illuminer un objet avec une source de lumière. Cette dernière est en général un faisceau laser pulsé mono ou polychromatique à distribution gaussienne. L'illumination explicite d'un objet par une source laser est appelée illumination active. La lumière laser incidente est focalisée sur l'objet grâce à une lentille de focalisation. La lumière incidente sur l'objet est réfléchie par ce dernier, collectée par une lentille de collection et focalisée sur un détecteur optique. Un détecteur optique est un dispositif qui intègre la lumière reçue et fournit à sa sortie des informations sur la position du signal intégré sur le détecteur optique (appelé aussi détecteur de position). La connaissance de cette position et la géométrie optique de la caméra utilisée permettront alors le calcul des coordonnées cartésiennes du point de l'objet associé à la mesure courante. La figure 1.1 illustre un exemple simplifié d'une caméra optique de base. La caméra illustrée à la figure 1.1 décrit un moyen simple pour mesurer la profondeur d'un objet par rapport à une profondeur de référence. Pour l'exemple illustré, la

profondeur R est donnée par:

$$R = R_0 - \frac{A\left(\frac{S}{B}\right)}{\sin(\theta) - \left(\frac{S}{B}\right)\cos\theta} \tag{1.1}$$

où R_0 est la profondeur nominale, A est la distance entre l'intersection des axes optiques et la lentille de collection, B est la distance entre le détecteur optique et la lentille de collection, S est la distance sur le détecteur optique séparant le spot laser reçu et le spot laser correspondant à la profondeur nominale, et θ est l'angle de parallaxe. Le rapport des distances $\frac{B}{A}$ est appelé facteur de magnification. Le facteur de magnification permet d'estimer la taille du spot laser reçu par rapport à la taille du spot laser émis. La taille du spot laser reçu est toujours plus petite ou égale à la taille du spot laser émis, donnant ainsi un facteur de magnification plus petit ou égal à 1.

Les systèmes optiques conventionnels basés sur l'illumination active utilisent une large ouverture angulaire entraînant ainsi des phénomènes d'ombrage (Rioux, Bechthold, Taylor, Duggan, 1987). Une solution simple et évidente serait de réduire l'ouverture angulaire, améliorant ainsi la profondeur de vision. Cependant, cela a pour effet négatif d'altérer l'intensité des spots et la résolution globale du système. Ce type de système optique doit constamment faire un compromis entre la profondeur de vision (liée à la résolution de la caméra et à la limite de diffraction) et l'intensité des spots (liée à l'ouverture de la lentille de collection). Les effets d'ombrage peuvent être considérablement atténués en synchronisant le faisceau laser avec le détecteur optique. Cette synchronisation permet un large champ de vision et des angles de triangulation faibles. La figure 1.2

illustre le principe de balayage synchronisé.

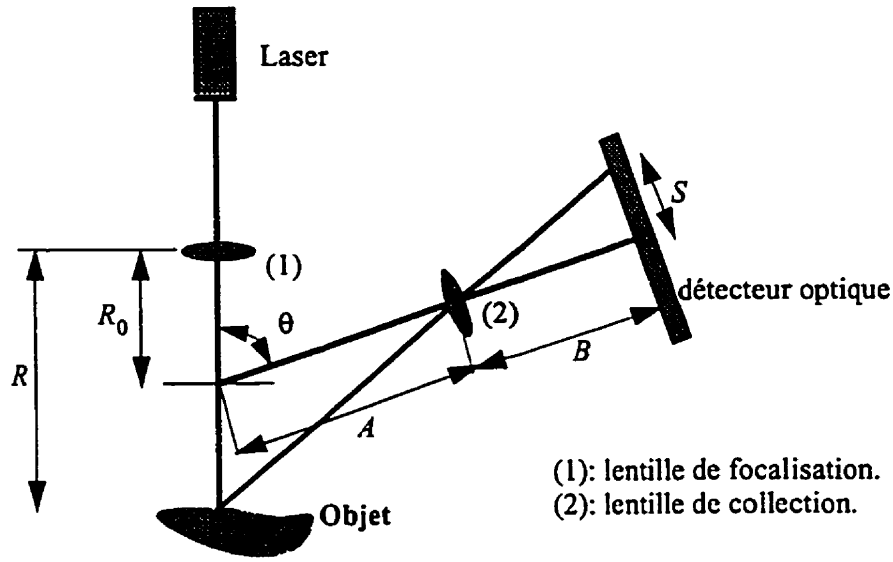

Figure 1.1: Caméra de base à illumination active.

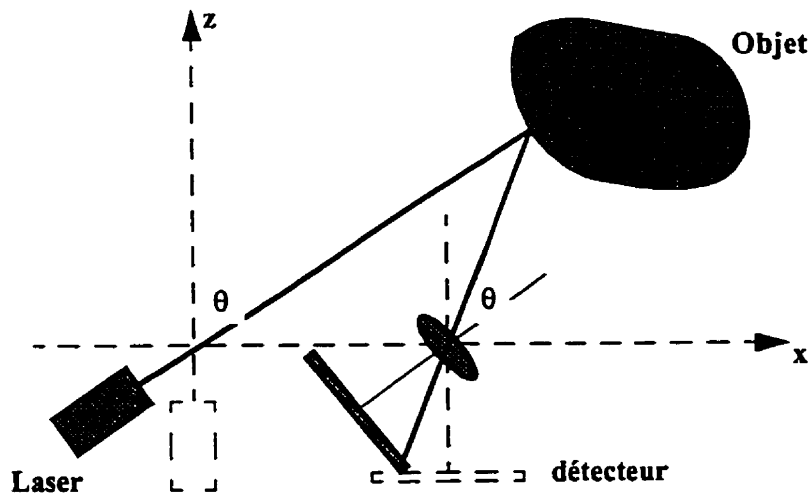

Figure 1.2: Balayage synchronisé.

Rioux (1990,1984) a introduit un concept innovateur appelé balayage autosynchronisé qui assure la synchronisation des faisceaux laser incidents et réfléchis par un miroir ayant 2 faces polies et réfléchissantes. La première face réfléchit le faisceau laser incident vers l'objet, tandis que la seconde face réfléchit le faisceau laser en provenance de l'objet vers la lentille de collection. Cette technique élimine totalement les problèmes de déphasage mécanique et augmente considérablement la précision du système. La figure 1.3 illustre le principe d'autosynchronisation. Afin d'obtenir une image 3D, un second axe a été ajouté dans la dimension y. Chaque point de l'objet est identifié de façon unique par ses coordonnées cartésiennes 3D. Ces dernières sont obtenues par des transformations géométriques associées à la caméra et la lecture de 3 données essentielles: la position courante de l'axe x, la position courante de l'axe y et la position du faisceau laser incident sur le détecteur optique.

Dans le reste de cette thèse, la caméra autosynchronisée sera utilisée comme la caméra de référence pour valider les solutions apportées aux problèmes soulevés.

1.2 La caméra autosynchronisée

1.2.1 La relation objet-caméra

Comme nous l'avons mentionné à la section précédente, chaque point de l'objet est identifié de façon unique par ses coordonnées cartésiennes, lesquelles sont données par les équations suivantes:

$$x = x(p, i) = \frac{X_g(i)}{p - P_\infty} - X_0(i) . \tag{1.2}$$

$$y = y(p, i, j) = \left(\frac{Z_g(i)}{p - P_\infty} - Z_0(i) \right) \sin\phi(j) + H_y(j) . \tag{1.3}$$

$$z = z(p, i, j) = \left(\frac{Z_g(i)}{p - P_\infty} - Z_0(i) \right) \cos\phi(j) + H_z(j) . \tag{1.4}$$

où

$$X_g(i) = \sum_{n=0}^{5} A_n i^n , \; X_0(i) = \sum_{n=0}^{5} B_n i^n , \; Z_g(i) = \sum_{n=0}^{5} C_n i^n , \; Z_0(i) = \sum_{n=0}^{5} D_n i^n .$$

$$H_y(j) = \sum_{n=0}^{5} E_n j^n , \; H_z(j) = \sum_{n=0}^{5} F_n j^n \text{ et } \phi(j) = \sum_{n=0}^{3} G_n j^n .$$

Les valeurs $A_n, B_n, C_n, D_n, E_n, F_n, G_n$ pour $n = 0 \ldots 5$, sont des paramètres représentant la géométrie de la caméra. Les paramètres (p, i, j) représentent respectivement la position du faisceau laser sur le détecteur optique et les positions discrètes des scanners associés aux axes x et y.

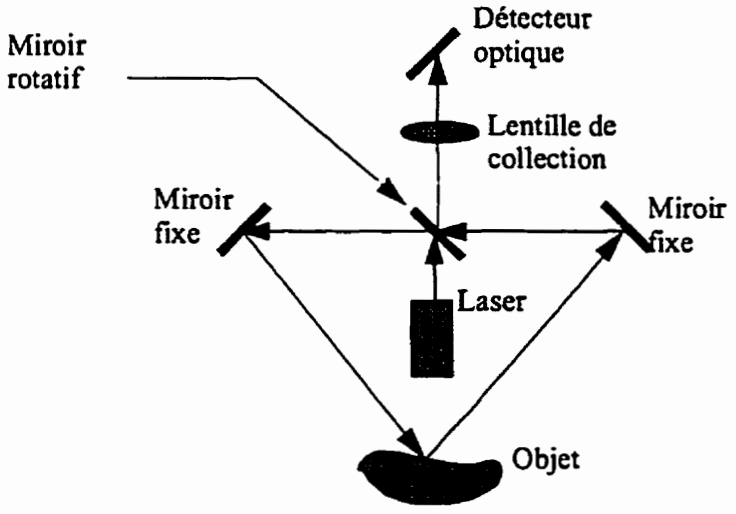

Figure 1.3: Balayage autosynchronisé.

1.2.2 Le détecteur optique

Le détecteur optique est une barrette de cellules photosensibles appelées pixels. Les pixels sont de largeur W et sont disposés avec un pas de répétition de D. La figure 1.4 illustre un détecteur optique de base. La région séparant 2 pixels adjacents est en général non-photosensible et est communément appelée zone noire.

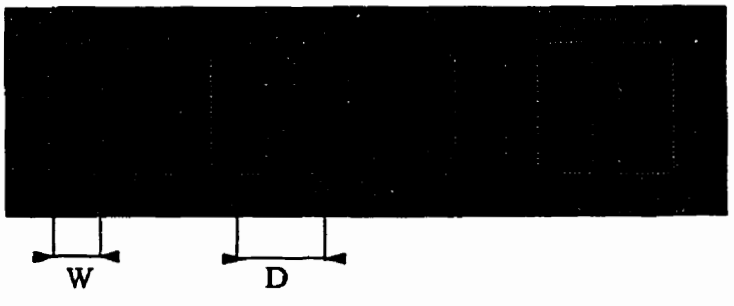

Figure 1.4: Détecteur optique à cellules photosensibles.

Les détecteurs optiques à cellules photosensibles sont très populaires et permettent d'obtenir des informations sur le profil topographique de la cible. Dans le cas des applications d'analyse d'images, des détecteurs 2D sont utilisés. D'autres détecteurs optiques sont aussi utilisés dans des applications optiques commerciales. On distingue principalement la photodiode à effet latéral qui fournit à sa sortie une lecture précise et rapide du centre de masse du faisceau optique incident sur sa surface. Cependant, cette caractéristique empêche son utilisation dans des applications de traitement d'images. Les détecteurs optiques de type photomultiplieur sont rapides et très sensibles et sont surtout utilisés dans les applications utilisant des techniques de mesure de type durée de vol (anglais: time of flight) où la distance de l'objet est calculée en mesurant l'intervalle de temps séparant l'émission du signal acoustique ou optique et sa réception.

1.2.3 Centre de masse du faisceau laser détecté

Le calcul des coordonnées (x, y, z) d'un point donné de l'objet est basé sur la connaissance des valeurs (p, i, j) obtenues par mesure. En raison de la fonction d'intégration du détecteur optique, ce dernier ne délivre à sa sortie que les intensités des pixels. Une intensité de pixel représente l'intégrale sur une bande de largeur W de la portion du faisceau laser réfléchi ayant sensibilisé le pixel en question. Cette caractéristique fondamentale du détecteur optique rend impossible la connaissance de la position du centre (centre du spot à distribution gaussienne) du faisceau laser réfléchi sur le détecteur optique. Afin de palier à ce problème, il est courant d'avoir recours au calcul de la position du **centre de masse** du faisceau laser réfléchi sur le détecteur optique. Ce centre de masse est géné-

ralement calculé de la façon suivante (Backes, Stevenson,1990):

$$c = \frac{\sum_{k=1}^{M} I(k) \cdot k}{\sum_{k=1}^{M} I(k)} \qquad (1.5)$$

où k est le numéro de pixel et $I(k)$ l'intensité correspondante.

Par conséquent, la valeur de p utilisée dans le calcul des coordonnées cartésiennes (x, y, z) représente en fait la position du centre de masse, et non le centre, du faisceau laser réfléchi sur le détecteur optique.

Chapitre 2 : Précision des systèmes optiques

Dans ce chapitre, nous allons présenter les principaux facteurs affectant la qualité d'une mesure optique. Nous allons surtout porter une attention particulière aux effets engendrés par une variation de réflectance, dont cette thèse fait l'objet.

2.1 Introduction

Au chapitre précédent, nous avons présenté les éléments de base constituant un système optique à illumination active. Ces éléments peuvent être regroupés en 4 catégories:

- éléments mécaniques: constitués par les scanners associés aux miroirs rotatifs;
- éléments électroniques: constitués principalement par le détecteur optique;
- éléments optiques: constitués par la source laser et les lentilles;
- la cible.

Une mesure optique peut être vue comme une chaîne constituée des 4 catégories précédemment énumérées. A chaque maillon de la chaîne, le faisceau laser est affecté, altérant ainsi la mesure optique en cours (Blais, Rioux, Beraldin,1988). Dans ce qui suit, nous

allons décrire la nature de l'altération associée à chacune des 4 catégories ainsi que les moyens disponibles pour en réduire l'effet.

2.2 Les scanners rotatifs

Les scanners rotatifs sont des éléments mécaniques utilisés pour assurer la rotation des miroirs. Vu que les positions angulaires des miroirs sont utilisées dans le calcul des coordonnées (x, y, z), il est important que les scanners utilisés soient suffisamment précis afin d'éviter les déphasages mécaniques qui pourraient mener à des calculs erronés de coordonnées cartésiennes.

2.3 Le détecteur optique

Le détecteur optique à cellules photosensibles intègre la lumière incidente sur chaque pixel sensibilisé. Tel que mentionné au chapitre précédent, chaque pixel a une largeur finie W et ils sont uniformément espacés selon un pas de D. La largeur finie des pixels implique un calcul de centre de masse (équation 1.5) avec des intensités de pixels discrètes (alors que le signal laser est une fonction continue) avec une perte d'énergie associée aux régions non-photosensibles séparant les pixels. Finalement, le calcul du centre de masse suppose, implicitement, que l'énergie d'un pixel donné est concentrée en son centre, ce qui biaise la valeur calculée du centre de masse du faisceau laser incident sur le détecteur. Ce dernier peut perturber le signal réfléchi par l'objet par différentes sources de bruit, principalement le bruit interne associé aux amplificateurs, le bruit externe et les

spécularités (en anglais: speckle). La spécularité est un phénomène d'interférence couramment rencontré dans les systèmes optiques utilisant un laser (Baribeau, Rioux,1991). Elle se présente sous la forme d'un spot granuleux et peut être visible à l'oeil nu lorsqu' une source laser illumine une surface diffuse. La taille de la granularité détermine l'amplitude de la spécularité. La taille du speckle est fonction de la longueur d'onde du laser, du diamètre de la lentille et la distance séparant le détecteur de la lentille. De nombreux travaux ont montré que l'effet du speckle est négligeable lorsque sa taille est plus petite que la taille d'un pixel. Il apparaît clairement que les caractéristiques géométriques du détecteur optique ont un impact très important sur la qualité et la précision des mesures. De plus, il est évidemment très recommandé d'avoir un rapport signal sur bruit élevé. Finalement, dans la majorité des applications optiques, les effets dus au calcul du centre de masse à partir d'intensités de pixels discrètes et la supposition implicite que l'énergie associée est concentrée au centre du pixel sont négligés.

2.4 La source laser et la lentille de collection

Au chapitre 1, nous avons présenté un exemple de correspondance entre le détecteur optique et la profondeur à mesurer (figure 1.1). Dans une situation idéale, la source laser illuminerait uniquement un point de l'objet. Malheureusement, la réalité est toute autre. En effet, vu que le signal laser possède une distribution gaussienne, plusieurs points de l'objet sont alors illuminés simultanément. L'ampleur de la zone éclairée va dépendre de la taille efficace (diamètre) du signal laser. Il est donc important d'avoir un diamètre

aussi petit que possible pour garantir une correspondance fidèle point à point entre l'objet et le détecteur. Cependant une réduction du diamètre du signal laser implique des coûts additionnels. Quant à la lentille de collection, celle-ci agit comme un filtre passe-bas et elle est sujette à l'effet d'aberration optique qui peut entraîner le déplacement sur le détecteur du faisceau laser reçu, donc le déplacement de son centre de masse. Le choix des caractéristiques de la lentille relève d'une grande importance afin de ne pas altérer le signal laser reçu.

2.5 La cible

Nous avons vu dans les sections précédentes que les choix de la taille du diamètre du faisceau laser, des caractéristiques optiques de la lentille, des caractéristiques géométriques du détecteur, et des caractéristiques mécaniques des scanners constituaient des éléments clés dans la réalisation de mesures précises. Une fois qu'un choix judicieux de ces composantes a pu être réalisé, la cible demeure la seule inconnue du système. Une cible donnée peut altérer la qualité de la mesure courante de plusieurs façons: par sa géométrie, pour la nature du (des) matériaux qui la compose (ent), par la rugosité de sa surface, etc (Svetkoff, 1991). Si on suppose que la surface de la cible est très peu rugeuse, deux facteurs retiennent l'attention: la variation de la profondeur et la variation de la réflectance.

2.5.1 La variation de profondeur

Nous avons mentionné à la section 2.4 que le spot laser avait une taille utile déterminée par son diamètre, ce qui avait pour conséquence d'illuminer plusieurs points de la scène simultanément. Supposons que les points illuminés se trouvent sur 2 profondeurs différentes, par exemple en présence d'une arête. Le spot laser initial est alors partitionné en 2 sous-spots. Chacun de ces sous-spots va alors suivre un chemin optique différent et va sensibiliser des pixels situés dans une zone éloignée de la zone sensibilisée par l'autre sous-spot. La figure 2.1 illustre le phénomène lié à la variation de profondeur.

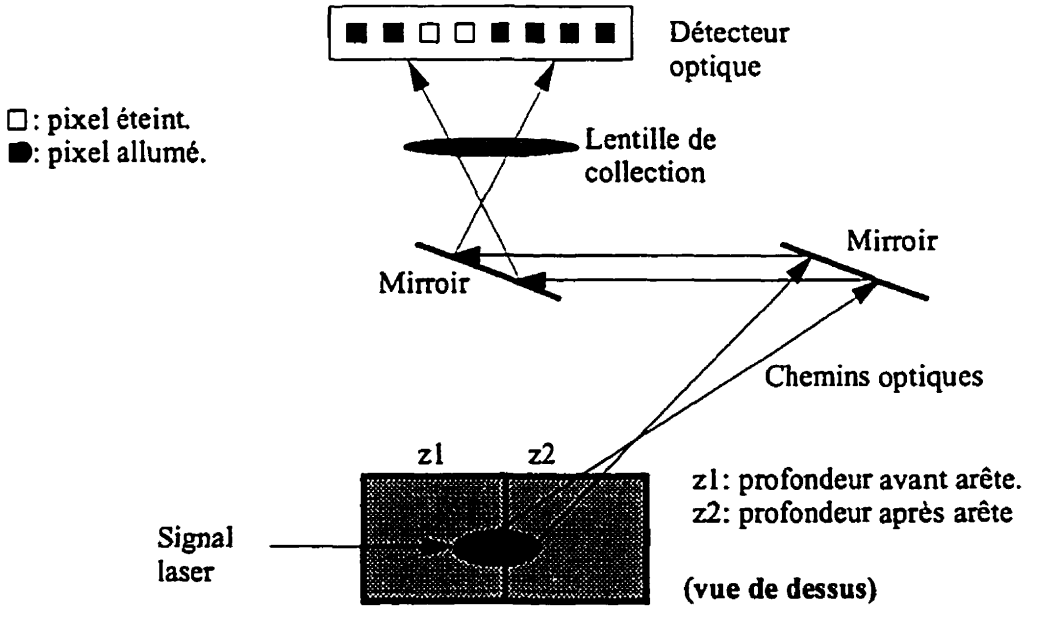

Figure 2.1: Effet d'un saut de profondeur sur le signal laser incident.

Le calcul du centre de masse de la combinaison des 2 sous-spots va automatiquement conduire à un calcul erroné. Il faut d'abord déterminer lequel des 2 sous-spots doit être

conservé avant de calculer le centre de masse correspondant.

2.5.2 La variation de réflectance

La réflectance caractérise la faculté d'une surface à réfléchir la lumière. La réflectance est identifiée par une valeur comprise entre 0 et 1 et est surtout fonction de la nature de la surface, sa couleur, etc. On dira qu'une surface a une réflectance de R, si pour une énergie E reçue, une énergie $E \cdot R$ est retournée. Une variation de réflectance peut être facilement illustrée par le passage d'une zone sombre à une zone éclairée et vice-versa. Les variations de réflectance les plus couramment rencontrées sont les sauts et les gradients de réflectance. Les figures 2.2 et 2.3 illustrent 2 exemples de variation de réflectance.

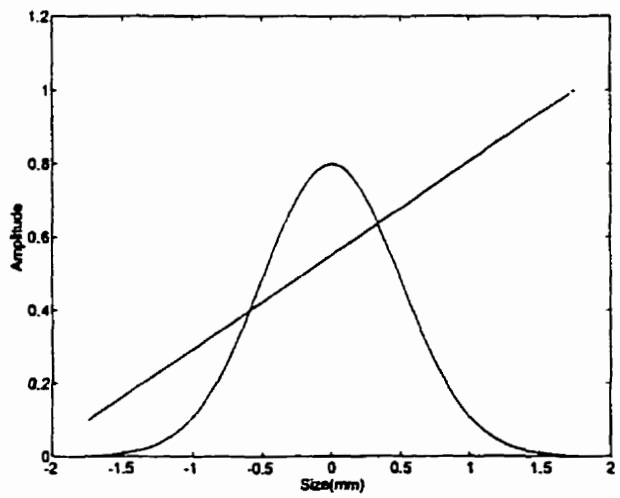

Figure 2.2: Exemple d'un spot gaussien et d'un gradient de réflectance.

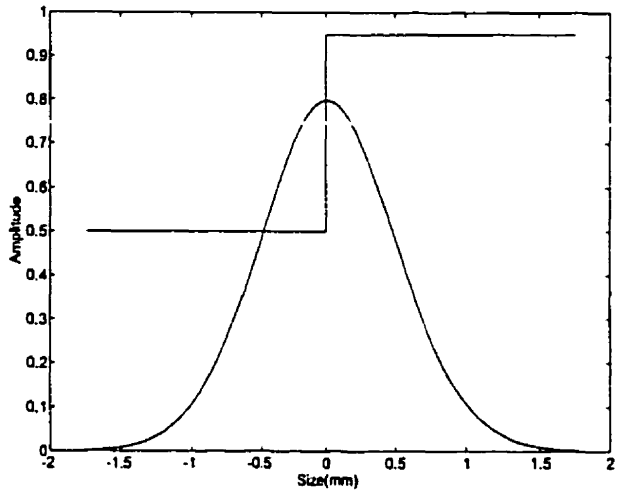

Figure 2.3: Exemple d'un spot gaussien et d'un saut de réflectance.

Supposons que la cible à analyser est située à une profondeur constante z_1, avec une surface présentant un saut de réflectance. Lorsque le signal laser gaussien illumine la surface en question, il est alors multiplié par le signal de réflectance, engendrant ainsi un signal résultat "pseudo-gaussien" déformé (Buzinski, Levine, Stevenson,1992). La figure 2.4 illustre le signal laser déformé. Ce signal "pseudo-gaussien" va alors suivre un chemin optique afin d'atteindre le détecteur. En raison de la déformation du signal incident sur le détecteur, le centre de masse du signal gaussien initial (position p_1) va se déplacer vers la portion déformée (position p_2). La position p_2 du nouveau centre de masse (au lieu de la position p_1), va alors être utilisée dans les équations de calcul des coordonnées (x, y, z). Les valeurs calculées sont malheureusement erronées car l'objet

à analyser sera localisé à la profondeur z_2, alorsque celui-ci est à la profondeur z_1. Cette "illusion optique" est appelée artefact. Il est donc nécessaire de corriger cette "illusion optique" afin d'améliorer la précision des mesures effectuées et de mener au calcul correct des coordonnées (x, y, z). L'erreur associée à une variation de réflectance peut être amplifiée si elle est combinée à une variation de profondeur conduisant ainsi au partitionnement du spot laser initial en 2 sous-spots déformés "pseudo-gaussien".

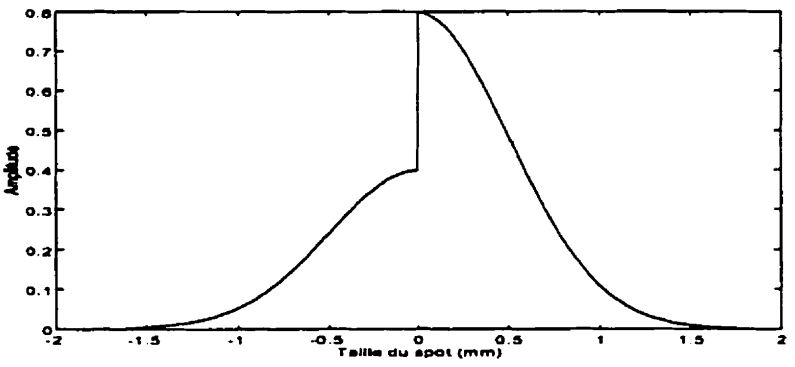

Figure 2.4: Signal résultant de la déformation du signal laser initial.

Dans le cadre d'une partie de cette thèse, nous allons nous intéresser à la correction des erreurs introduites par une variation de réflectance pour un objet situé à une profondeur constante (pas de saut de profondeur pour la mesure courante).

Chapitre 3 : Correction des artefacts associés à une variation de réflectance

Dans ce chapitre, nous allons décrire 2 méthodes permettant de corriger les artefacts engendrés par une variation de réflectance. La description de ces méthodes comprendra les algorithmes de correction, leurs coûts et l'impact sur la précision des mesures effectuées.

3.1 Introduction

Dans le chapitre précédent, nous avons introduit les problèmes de précision associés à une variation de réflectance. La perte de précision réside dans le déplacement du centre de masse du signal optique réfléchi en raison de sa déformation causée par la variation de réflectance. Le déplacement du centre de masse aura pour conséquence de localiser la mesure courante à une position autre que celle où elle devrait être. Le détecteur optique utilisé a une résolution de 1/64ième de pixel en abscence de variation de réflectance. Cette résolution peut tomber jusqu'à 1/4 de pixel en présence d'une variation de réflectance (Soucy, 1990). Afin de corriger cette erreur, nous avons développé 2 méthodes adaptées à la correction des erreurs dues aux sauts et gradients de réflectance. Ces 2 méthodes seront respectivement appelées méthode directe et méthode itérative.

3.2 Méthode directe

La correction effectuée par la méthode directe est basée sur des informations associées à la mesure courante et à des informations collectées lors d'une phase dite de calibration. La phase de calibration consiste à prendre des cibles de référence dont le profile de réflectance est connu et à effectuer des mesures. Ces cibles de référence sont placées à des profondeurs et à des positions angulaires prédéterminées. Les profils de réflectance utilisés sont les sauts, les gradients et la réflectance unitaire. Pour chaque mesure effectuée sur une cible de référence, l'algorithme de correction calcule des facteurs de correction dont les valeurs sont fonction du profil de réflectance. Ces facteurs de correction représentent la valeur à ajouter ou à retrancher au centre de masse associé à la mesure courante afin de corriger le déplacement erroné. Après avoir effectué les mesures sur l'ensemble des cibles localisées aux différentes profondeurs et positions angulaires prédéterminées, l'algorithme passe à la seconde phase dite phase de correction. La phase de correction consiste à prendre la cible inconnue et à effectuer des mesures aux positions angulaires choisies lors de la phase de calibration. Le processus de correction comporte les étapes suivantes:

1) Reconstituer un estimé du profil de réflectance en divisant le spot laser reçu par un spot de référence correspondant à une réflectance unitaire pour la position angulaire courante. Ce spot de référence a été obtenu lors de la phase de calibration.

2) Calculer le centre de masse du signal de réflectance reconstitué.

3) Lire le facteur de correction correspondant au profil de réflectance reconstitué. Si ce profil n'a pas été calibré, alors effectuer une interpolation entre les 2 profils calibrés les plus proches.

4) Appliquer la formule de correction.

La méthode directe est principalement caractérisée par sa faible complexité de calcul, ce qui permet d'obtenir de grands débits de données corrigées. Cependant, cette performance se fait au détriment du coût associé à la phase de calibration. Ce coût est essentiellement fonction du nombre de mesures à effectuer et de la qualité des profils de réflectance. Il faut néanmoins souligner que ce coût est non-récurrent et toutes les données collectées lors de la phase de calibration peuvent être utilisées pour des cibles inconnues situées dans le volume de mesure calibré. Ce volume étant délimité par les profondeurs et positions angulaires prédéterminées. Quant à la précision atteinte après correction, celle-ci est en général très proche de la résolution du système en raison des mesures effectuées lors de la phase de calibration.

Dans ce qui suit, nous allons présenter l'article décrivant la méthode directe. Cet article a été soumis pour publication à IEEE Transactions on Instrumentation and Measurements (Octobre 1997). L'article commence par une introduction expliquant les problèmes de précision associés à une réflectance variable. Cette introduction est suivie par la description de la caméra autosynchronisée et ses principales caractéristiques. Puis, une revue de la littérature des principales méthodes de correction est alors présentée. Ensuite, les principaux développements mathématiques sont alors introduits. Ces développements mathématiques caractérisent principalement la relation objet-détecteur en présence d'une variation de réflectance. Ces développements sont alors suivis de l'algorithme de la méthode directe et des résultats de simulations fonctionnelles obtenus en modélisant tout

le processus de calibration-correction pour une caméra autosynchronisée. L'article se termine par une conclusion où on résume les principales caractéristiques de la méthode directe, ses avantages, ses inconvénients, et des futurs éléments de recherche à développer.

3.3 Improvement of Sensor Accuracy in the Case of a Variable Surface Reflectance Using Active Laser Range Finder

H. Khali, Y. Savaria and J. L. Houle
Ecole Polytechnique de Montréal
Electrical and Computer Engineering Department
P.O. Box 6079, Station Down-Town, Montreal, Quebec, H3C 3A7

M. Rioux and J. A. Beraldin
National Research Council of Canada
Visual Information Technology
Montreal Road, Building M-50, Ottawa, Ontario K1A 0R6

D. Poussart
Laval University
Electrical Engineering Department
Quebec City, Quebec, G1K 7P4

Keywords: Laser range finders, artifact correction, reflectance modelling.

Abstract

In active laser range finders, the computation of the (x, y, z) coordinates of each point of a scene can be performed using the detected centroid p of the image spot on the sensor. When the reflectance of the scene under analysis is uniform, the intensity profile of the image spot is a gaussian and its centroid is correctly detected assuming an accurate peak position detector. However, when a change of reflectance occurs on the scene, the

intensity profile of the image spot is no longer gaussian. This change introduces a deviation Δp on the detected centroid p which will lead to erroneous (x, y, z) coordinates. This paper presents two heuristic models to improve the sensor accuracy for a variable surface reflectance. Two typical cases are analyzed in details: a gradient of reflectance and a step of reflectance. These models can be good candidates to develop a method for real-time correction because of their low complexities. Simulation results are presented to show the quality of the correction and the resulting accuracy.

3.4 Introduction

Active optical triangulation systems project light towards an object. The light is then reflected by the surface and collected by a detector. The collected energy can provide information about the object under analysis, for example its range. The data to be processed is a 3-D surface map [2], which is captured by scanning a laser beam onto the scene. Measurements are made on the illuminated points in the scene and are mapped onto (x, y, z) rectangular coordinates by applying a set of trigonometric transformations. When the reflectance of the scene under analysis is uniform, the intensity profile of the image spot is gaussian and its centroid is correctly detected assuming an accurate peak position detector. However, when a change of reflectance occurs on the scene, the intensity profile of the image spot is no longer gaussian. This change introduces a deviation Δp on the detected centroid p, which will lead to erroneous (x, y, z) coordinates. This paper presents two heuristic models well adapted to real-time correction of these errors. Our objective is to reduce the effects of variable surface reflectance for an optical

system based on active illumination. Two cases are analyzed: a gradient of reflectance and a step of reflectance. The object under analysis is assumed to have a constant range and a uniform texture. This paper is divided as follows. Section 3.5 describes an optical system and its major components. Section 3.6 presents the methods available in the literature to reduce the effects of a variable surface reflectance. Section 3.7 presents an analysis of the relationship between surface reflectance and image intensity distribution on the detector. This analysis shows how the reflectance for the actual measurement can be reconstructed and how the centroid deviation can be corrected. Section 3.8 presents simulation results based on an autosynchronized camera to show the accuracy of the correction on centroid deviation. Section 3.9 concludes the paper.

3.5 Description of an optical system based on active illumination

An optical system is basically composed of a light source (laser) which illuminates an object, a deflection mirror, collecting lens and a detector to collect the reflected light. Because of practical considerations, we will focus our attention, without loss of generality, on an autosynchronized system [1]. Figures 3.1 and 3.2 show respectively a general view of an autosynchronized system and its basic geometry. In this section, we will review the main parameters that will be taken into account in our analysis to develop our correction heuristic.

3.5.1 Laser scanner

The considered system is based on synchronized scanners for which several geometries can be found in [1]. The arrangement shown in Figure 3.1 is commonly used and is based on a double-sided coated mirror that synchronizes both the projection axis and the detection axis. This geometry reduces the amount of shadow effect, while maintaining a good resolution on a very large field. Figure 3.2 shows the basic geometry of an autosynchronized system.

3.5.2 Scheimpflug condition

In an optical system, the defocusing problem increases when the object under analysis is brought close to the camera, while maintaining a high resolution. This problem can be overcome with the Scheimpflug condition. This condition states that if the detector is tilted by an angle β, then any point along the projection axis will be in focus on the position sensor. The angle β is given by: $\beta = \tan^{-1}(f_0/d)$, where f_0 is the effective focal length, and d the distance between the collecting lens and the projection axis.

3.5.3 Lateral magnification

In an optical system, it is important to estimate the image size of the laser spot reflected by the scene. This estimation is easily done by computing the lateral magnification factor given by:

$M_l = f_0/(l\cos\beta)$ where l is the distance between the scene and the collecting lens. If we define S_o and S_d respectively as the laser spot size on the object and the image spot

size on the detector, then we have: $S_d = M_l S_o$. The importance of M_l in system design appears when using a subpixel resolution peak detector [4], where it is important to have an image of the beam that is much wider than a pixel on the detector. M_l is typically between 0.1 and 1.0.

3.5.4 Longitudinal (range) magnification

The position of the image spot on the detector is a function of the object range z. However, any change in the range by Δz will introduce a displacement of Δp of the image spot on the detector. Δz and Δp are related to each other by the range magnification M_r given by:

$M_r = \frac{\Delta p}{\Delta z} = M_l \sin\gamma$ where γ is the triangulation angle. This relationship strongly varies with distance.

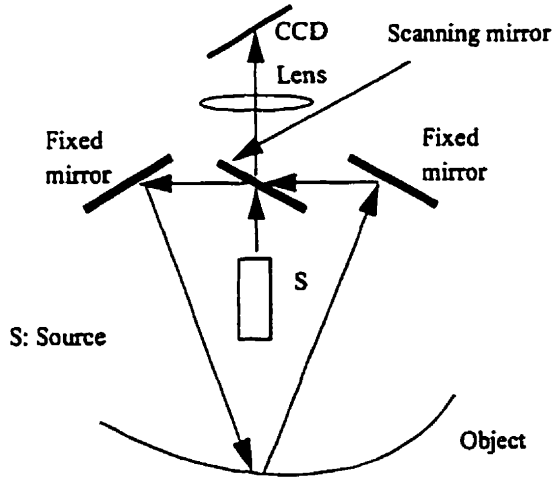

Figure 3.1: General view of an autosynchronized system.

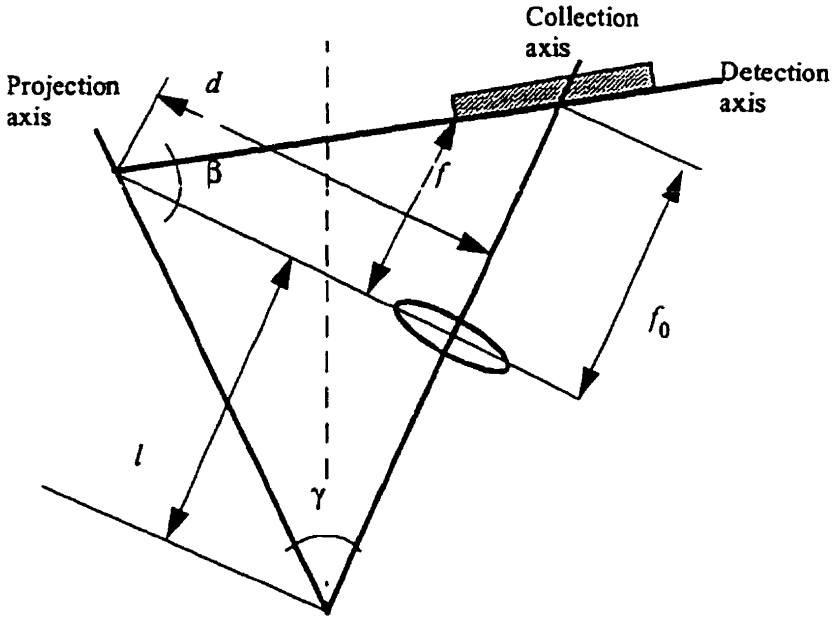

Figure 3.2: Basic geometry of an autosynchronized system.

3.5.5 Relationship between the scene and the sensor

Several different designs of autosynchronized cameras are possible depending on specific applications. In this paper, we will concentrate on the "space camera" designed for long-range measurements. Additional details concerning the geometry of the camera can be found in [5] and [6]. With this type of camera, the range of the object is given by:

$$z(p,\kappa) = Zinf(\kappa) - Z\delta inf(\kappa) + Pinf \cdot \frac{(Zo(\kappa) - Z\delta o(\kappa) - (Zinf(\kappa) - Z\delta inf(\kappa)))}{Pinf - p}$$

where p is the position on the sensor of the laser spot, κ is the optical angle, $Pinf$ is the vanishing point and $Zinf(\kappa), Z\delta inf(\kappa), Zo(\kappa), Z\delta o(\kappa)$ are trigonometric transformations that describe the geometry of the camera. We can see from this relationship that any error on p will introduce an error on the range z. Moreover, for a constant range

object, a variable surface reflectance will introduce a deviation on p that will be interpreted as a change in the range. Therefore, best accuracy is obtained by correcting p before computing z.

3.6 Previous work

Reducing the effects of variable surface reflectance is essential in order to improve system accuracy. Correction models can be found in the literature to achieve that goal. Mundy and Porter [7] presented a method called the Difference Ratio Normalization. With this method, the scene is illuminated in sequence by two gaussian beams separated by a small distance d. The image intensity of each incident beam is given by: $I_i(x) = R(x)P_i(x)$ where $R(x)$ and $P_i(x)$ are respectively the reflectance and the incident beam power spatial distribution. The sensor output $N(x)$ is defined as: $N(x) = \frac{I_1(x) - I_2(x)}{I_1(x) + I_2(x)} = \frac{P_1(x) - P_2(x)}{P_1(x) + P_2(x)}$. Note that $N(x)$ depends only on the incident beams, since the effects of reflectance have been cancelled. With this method, we need to find the zero of $N(x)$. This task can be very difficult, because of the finite width of a pixel and the noise that corrupts the true signal. Moreover, this method reduces the throughput of the optical system because of the time multiplexing of the beams. Another method presented in [8] adjusts the centroid of the actual image based on information provided by its intensity distribution. This adjustment is done along 18 equidistant axes through the center of the image. However based on a statistical analysis, this centroid adjustment method did not repeatably improve the calculated image spot centroid. Soucy

et al. [9] presented a mathematical model stating that the interpolation error, defined as the difference between the center of gravity of a 2-D gaussian on a scan line and the actual position of the center of gravity of the imaged spot, is proportional to the integral of the variance of the imaged spot. The authors reported that this model can be used as a good estimator of the position of the discontinuity. However, its implementation for real-time measurements can be very complex. Up to now, the best way to reduce the effects of variable surface reflectance is to produce small illuminating spot size. This can be achieved with the method proposed in [10], which provides means of controlling the size and the shape of laser beams.

3.7 Modeling of a variable surface reflectance

In this section, we consider an optical system based on an autosynchronised camera. The surface is illuminated with a monochromatic gaussian laser beam of size $2\Gamma\sigma$, where σ is the standard deviation of the laser spot and Γ a spot size parameter. The image intensity is collected by a 1-D discrete detector composed of N cells (pixels). All the pixels have a width of W and a distance of D between two adjacent pixel centers. The pixels have a sensitive region as illustrated in Figure 3.3. Thus, with this model, all the energy reflected by the object is collected with no loss. To simplify the correction models, we will assume that the whole pixel is fully sensitive, and we will not take into account the transition regions as illustrated in Figure 3.3. We will show in the simulation results that even with this assumption, the resulting accuracy is still satisfactory. Without loss of

generality, we will assume for simplicity that the image spot on the detector has a size of $2d+1$ non zero pixels, where $d = M_l \Gamma \sigma / D$. The same analysis can be applied in the case of an image spot covering $2d$ non-zero pixels, with minor changes to the mathematical expressions. In our analysis, M_l will be estimated using the non-zero pixels. As mentioned in the previous section, M_l is a function of the range z of the target. In all our analysis, perfect geometric imaging (pinhole model) is assumed to take place. The range of the object is supposed to be constant. Since we assume a 1-D detector, our analysis will be carried out for the image spot cross section. The effects of noise and speckle are not taken into account.

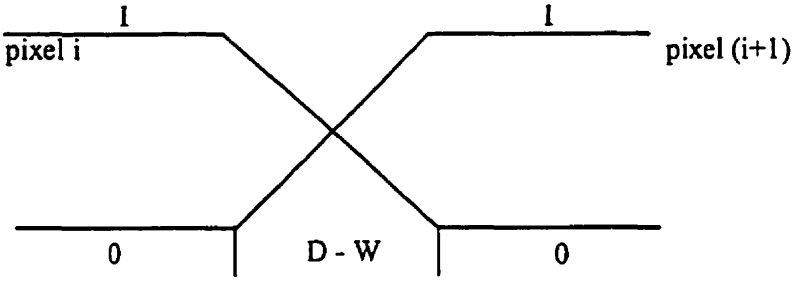

Figure 3.3: Sensitivity transition region between two pixels.

3.7.1 Impact of a reflectance gradient

In this subsection, we want to study the effect of an object reflectance gradient on the centroid deviation of the detector image spot. Figure 3.4 illustrates the gaussian laser

beam and the gradient. Because of the system magnification and the width of the pixels, several adjacent points on the object under analysis will contribute to the total energy received by a given pixel. In our model, these points will be processed as adjacent uniformly distributed samples. Let us suppose that k points of the object $x_i^{(j)}, i = 1...k$, will contribute to the energy of pixel j, $1 \leq j \leq N$.

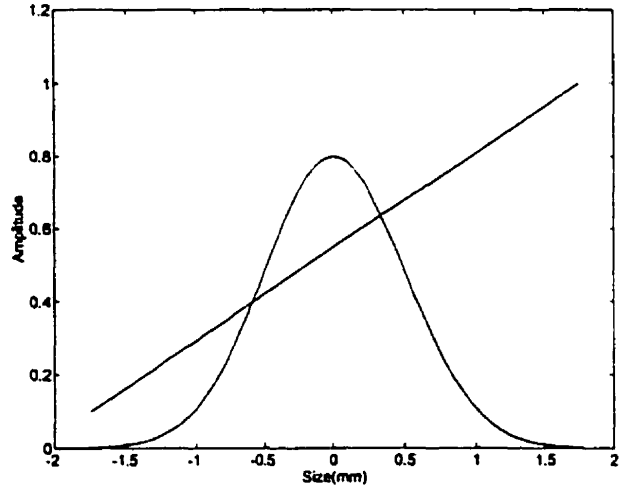

Figure 3.4: Gaussian object spot and its corresponding reflectance gradient.

Let $I_{ref}(j)$ be the intensity of pixel j when the reflectance is uniform (equal to 1). This parameter could be measured during a camera calibration phase with a target of uniform reflectance. $I_{ref}(j)$ can be approximated by

$I_{ref}(j) = \frac{1}{M_l}\sum_{i=1}^{k} f\left(x_i^{(j)}\right)\Delta x_i$, where $f\left(x_i^{(j)}\right)$ is the intensity on the object of the projected point $x_i^{(j)}$ and Δx_i is the interval between two successive projected points. The denominator M_l is used to conserve energy in the one-dimensional analysis. Recall that M_l is the lateral magnification factor defined in section 3.5.3.

With a gradient of reflectance, the intensity $I(j)$ of pixel j becomes:

$I(j) = \frac{1}{M_l}\sum_{i=1}^{k}\left(cx_i^{(j)} + e\right) f\left(x_i^{(j)}\right)\Delta x_i$ where c and e are respectively the slope and the base of the gradient. We define the reconstructed reflectance $R(j)$ as the ratio $\frac{I(j)}{I_{ref}(j)}$. Thus we have:

$$R(j) = \frac{I(j)}{I_{ref}(j)} = \frac{c\sum_{i=1}^{k} x_i^{(j)} f\left(x_i^{(j)}\right)\Delta x_i + e\sum_{i=1}^{k} f\left(x_i^{(j)}\right)\Delta x_i}{\sum_{i=1}^{k} f\left(x_i^{(j)}\right)\Delta x_i}.$$

After some simplifications, $R(j)$ becomes:

$$R(j) = \frac{c\sum_{i=1}^{k} x_i^{(j)} f\left(x_i^{(j)}\right)\Delta x_i}{\sum_{i=1}^{k} f\left(x_i^{(j)}\right)\Delta x_i} + e. \qquad (3.1)$$

Let us define $(\bar{x})^{(j)}$ as

$$(\bar{x})^{(j)} = \frac{\sum_{i=1}^{k} x_i^{(j)} f\left(x_i^{(j)}\right)\Delta x_i}{\sum_{i=1}^{k} f\left(x_i^{(j)}\right)\Delta x_i}.$$

Replacing $(\bar{x})^{(j)}$ in (3.1), we obtain

$$R(j) = c(\bar{x})^{(j)} + e, \tag{3.2}$$

where $(\bar{x})^{(j)}$ is the centroid on the object of the k points in the illuminating spot. If $c = 0$ (uniform reflectance), then $R(j) = e$. Equation 3.2 is given in the object coordinate system and can be rewritten in the sensor coordinate system as follows

$$R'(j) = \frac{c}{M_l}(\bar{p})^{(j)} + e', \tag{3.3}$$

where $(\bar{p})^{(j)}$ is the image on the sensor of $(\bar{x})^{(j)}$ and e' the base.

Equation (3.3) shows that a gradient on the scene is also a gradient on the detector with respect to our assumptions. In first approximation, the total power collected by each pixel can be placed at its center. Thus $(\bar{p})^{(j)}$ can be replaced by j when computing the centroid of the complete spot. In this case, $R'(j)$ can be rewritten as:

$$R'(j) = aj + b, \tag{3.4}$$

where a and b are respectively the slope and the base of $R'(j)$ in the new coordinate system.

The previous approximation is motivated by the fact that, at the sensor output, only the pixel total collected voltages are measurable. Since we do not have an easy mathematical expression that describes the centroid deviation of a gaussian multiplied by a gradient, our goal is to build a heuristic model that predicts the centroid deviation of the image spot given the centroid deviation of the gradient on the detector. In the following, we assume a reflectance gradient $R'(x)$ on the detector. This gradient has a slope a, a base b at its center, and a size of $2d+1$ pixels. To simplify the following equations, we will translate the original pixel indexes to a new one such that the spot on the detector occupies indexes $-d \le j \le d$. The centroid $\bar{x}$ of $R'(x)$ is given by:

$\bar{x} = \dfrac{\int x\,(ax+b)\,dx}{\int (ax+b)\,dx} = \dfrac{ad^2}{3b}$, where b is the value of $R'(x)$ at its center. $\bar{x}$ is a function of the three variables a, d, b.

To develop our correction heuristic, we first developed a simulation model of an auto-synchronised camera that scans a constant range target. Our model takes a gradient of reflectance as an input and gives the image intensity on the detector as an output. For each calibrated slope a_i, $i = 1 \ldots L$, where L is the number of measurements, we extract from the model the variables a_i, d_i, b_i.

We define G as the exact value of the centroid of the image spot corresponding to a given position θ of the scanning mirror, g_i the centroid given by the algorithm that computes the centroid position, and $\bar{x}_i$ the centroid of the gradient. Our goal is to predict an accurate value of G given a_i, $\bar{x}_i$ and g_i. When $a_i = 0$ (uniform reflectance), $\bar{x}_i = 0$, which means that $G = g_i$. However when $a_i \neq 0$ (nonuniform reflectance),

simulations showed that $g_i - x_i < G$. Thus, we define $B_i = g_i - \bar{x}_i$ and $f(a_i) = (G - B_i)/a_i$. G can now be expressed as:

$G = B_i + f(a_i)\, a_i = g_i - \bar{x}_i + f(a_i)\, a_i$. The value $f(a_i)$ represent a correction coefficient that can be obtained during camera calibration using targets with known reflectance gradients. In practice, when the measured value a_j is not tabulated, we must find i that satisfies $a_i < a_j < a_{i+1}$. Then $f(a_j)$ can be calculated by interpolation between $f(a_i)$ and $f(a_{i+1})$. For each calibrated base, we have a table of $f(a_i)$. This correction model is essentially based on the measured value a_i. This makes it suitable for real-time correction because of its low complexity. The correction and calibration algorithms for a given range z can be summarized as follows.

Algorithm 1-1 (Target calibrations in the case of a reflectance gradient)

Begin

Step 1: Use targets with uniform reflectance (=1) and known reflectance gradients.

Step 2: For each position θ of the scanning mirror, perform measurements on the different targets and extract from the sensor the following parameters:

* I: image spot for a non uniform reflectance target.
* I_{ref}: image spot for a uniform reflectance target.
* (a, b) : the slope and the base of the gradient on the sensor by dividing I and I_{ref}.
* g: the centroid of the measured image spot.

Step 3: Compute $x, f(a)$: the centroid of the reconstructed reflectance and the corresponding

correction factor.

END.

Algorithm 1-2 (Correction for the effects of a reflectance gradient)

Begin

Step 1: Read the incident spot $I(i)$ from the detector.

Step 2: Read the position θ of the scanning mirror.

Step 3: Estimate d using the non zero pixels.

Step 4: Compute the reflectance $R'(j) = \frac{I(j)}{I_{ref}(j)}$, where $I_{ref}(j)$ is the reference spot corresponding to θ and $-d \le j \le d$.

Step 5: Compute the centroid g of the incident spot.

Step 6: Extract the slope a and the base b from $R'(j)$ using (3.4):

$a = R'(1) - R'(0)$,

$b = R'(0)$.

Step 7: Compute the centroid $\bar{x}$ of $R'(j)$.

Step 8: **IF** a is not tabulated **THEN**

- Find a_i, a_{i+1} such that $a_i \le a \le a_{i+1}$.

- Compute $f(a)$ by interpolation between $f(a_i)$ and $f(a_{i+1})$.

END

Step 9: Compute the estimated value of the exact centroid position given by $G = g - \bar{x} + f(a)\,a$.

END.

3.7.2 Impact of a reflectance step

In this subsection, we want to study the effect of a step of reflectance, which occurs on the object, on the centroid deviation of the image spot on the detector. Figure 3.5 illustrates the situation considered in our analysis.

Let us assume that the reflectance step is mapped on the detector to a pixel with index n, $1 \le n \le N$. Since the image spot on the detector occupies $(2d+1)$ non zero pixels, we translate the original pixel index n to a new pixel index j, such that the image spot on the detector occupies pixel indexes $-d \le j \le d$. With a reflectance step δ, the intensity $I(j)$ of pixel j becomes:

$I(j) = \sum_{i=1}^{l} \alpha f\left(x_i^{(j)}\right)\Delta x_i + \sum_{i=l+1}^{k} (\alpha+\delta) f\left(x_i^{(j)}\right)\Delta x_i$ where l is the number of projected points among k on pixel j having a reflectance of α, which is the base of the step.

The reconstructed reflectance $R(j)$, as defined in section 3.7.1, becomes

$$R(j) = \frac{I(j)}{I_{ref}(j)} = \frac{\alpha \sum_{i=1}^{l} f\left(x_i^{(j)}\right)\Delta x_i}{\sum_{i=1}^{k} f\left(x_i^{(j)}\right)\Delta x_i} + \frac{(\alpha+\delta) \sum_{i=l+1}^{k} f\left(x_i^{(j)}\right)\Delta x_i}{\sum_{i=1}^{k} f\left(x_i^{(j)}\right)\Delta x_i} \qquad (3.5)$$

Recall that $I_{ref}(j)$ is the reference intensity of pixel j measured with a uniform reflectance target (equal to 1) during camera calibration.

Defining $x(l)$ as

$$x(l) = \frac{\sum_{i=1}^{l} f\left(x_i^{(j)}\right)\Delta x_i}{\sum_{i=1}^{k} f\left(x_i^{(j)}\right)\Delta x_i}, \text{ we obtain } \frac{\sum_{i=l+1}^{k} f\left(x_i^{(j)}\right)\Delta x_i}{\sum_{i=1}^{k} f\left(x_i^{(j)}\right)\Delta x_i} = 1 - x(l) .$$

Replacing $x(l)$ in (3.5), we have:

$$R(j) = \alpha x(l) + (\alpha+\delta)\,(1 - x(l)) .$$

$$\text{Thus } R(j) = \alpha + \delta(1 - x(l)) . \qquad (3.6)$$

Thus $R(j)$ is a function of the three variables $\alpha, \delta, x(l)$.

Equation (3.6) shows that the image on the detector of a laser spot illuminating a given step of reflectance on the object varies only with $x(l)$. Therefore we have:

$$\lim_{x(l)\to 0} R(j) = \alpha+\delta, \quad \lim_{x(l)\to 1/2} R(j) = \alpha+\delta/2, \quad \lim_{x(l)\to 1} R(j) = \alpha .$$

$x(l)$ is directly related to the position p_x on pixel j of the reflectance step. We have:

$x(l) \rightarrow 0 : p_x$ is at the beginning of pixel j.

$x(l) \rightarrow 1/2 : p_x$ is at the middle of pixel j.

$x(l) \rightarrow 1 : p_x$ is at the end of pixel j.

All the pixels i ($i < j$ or $i > j$) are assumed to have respectively a uniform reflectance of α and $(\alpha + \delta)$.

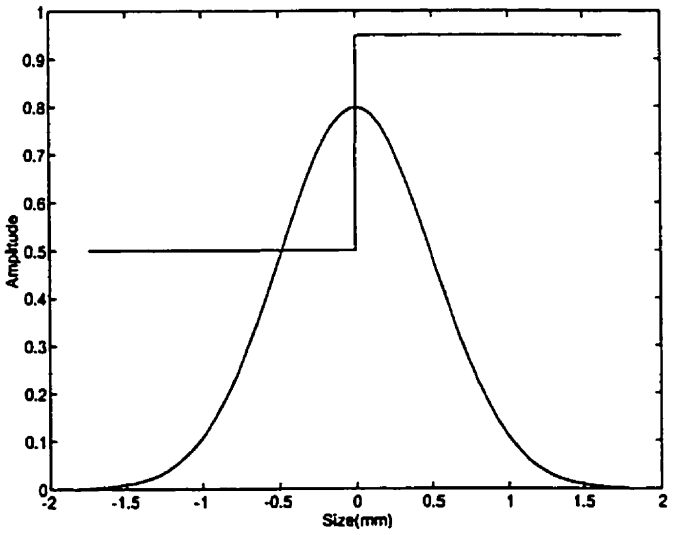

Figure 3.5: Gaussian object spot and its corresponding reflectance step.

To simplify the coming equations, let us define the variable h such as $j = -d + h$. Recall that j is the index of the pixel on which the reflectance step is mapped after suitable translation and j ranges from $-d$ to d without loss of generality. The centroid x of the image on the detector in the case of a reflectance step on the object is given by

$$\bar{x} = \frac{\sum_{i=-d}^{d} iI(i)}{\sum_{i=-d}^{d} I(i)} . \qquad (3.7)$$

With some algebraic manipulations, we can expand the denominator and the numerator of $\bar{x}$ as

$$\sum_{i=-d}^{d} I(i) = h\alpha + I(j) + (2d-h)(\alpha+\delta) .$$

$$\sum_{i=-d}^{d} iI(i) = \alpha\left(-hd + \frac{h(h-1)}{2}\right) + jI(j) + (\alpha+\delta) \left((2d-h)j + \frac{(2d-h+1)(2d-h)}{2}\right) .$$

From (3.7), we then have the following results:

$$\lim_{h \to 0, I(j) \to \alpha+\delta} \bar{x} = 0 , \quad \lim_{h \to 2d, I(j) \to \alpha} \bar{x} = 0 .$$

Simulations results showed that the closer we are from the center of the spot, the larger is the centroid deviation. Moreover, for a given pixel j, the higher is the step δ, the larger is the centroid deviation. Based on these remarks, we decided to use the following method to develop our heuristic model.

During the camera calibration phase, for each translated pixel index j on the detector, $-d \le j \le d$ on which the reflectance step may be mapped, we select a set of targets cha-

racterized by a known calibrated step of value δ, $\delta_1 \leq \delta \leq \delta_m$ where δ_1 and δ_m are known calibrated steps. For each (j, δ), we compute the coefficient of correction $f(j, \delta)$. Thus, we have $m(2d+1)$ coefficients corresponding to $m(2d+1)$ measurements performed during the calibration phase.

We will use a very similar approach to compute the correction factor $f(j, \delta)$ as in the case of the reflectance gradient (section 3.7.1). We obtain:

$f(j, \delta) = \frac{G - B_i}{(d - |j|)\delta}$ for $|j| \neq d$. G is the exact position of the centroid of the image spot corresponding to a given position θ of the scanning mirror, and $B_i = g_i - \bar{x}_i$. We can now express G as: $G = g_i - \bar{x}_i + f(j, \delta)\,((d - |j|)\delta)$. Recall that g_i is the detected centroid on the detector of the image spot, and xc_i the centroid on the detector of the image of the reflectance step. In practice, when the step that has been measured is not tabulated, we can interpolate it with the two nearest tabulated values. For each calibrated value α, we have a table of coefficients $f(j, \delta)$. This correction model assumes that the effects due to j and δ are combined. More specifically we have:

- $(\delta = 0) \Rightarrow G = g_i$;
- $(|j| = d) \Rightarrow \forall\delta, G = g_i$. The proposed correction model may introduce an error on the correction because the information concerning the position of the step in the pixel is not available at the sensor output. We will show later how to overcome this problem in practice. The correction and calibration algorithms can be summarized as follows.

Algorithm 2-1 (Target calibrations in the case of a reflectance step)

Begin

Step 1: Use targets with uniform reflectance (=1) and known reflectance steps.

Step 2: For each position θ of the scanning mirror, perform measurements on the different targets and extract from the sensor the following parameters:

* I: image spot for a non uniform reflectance target.

* I_{ref}: image spot for a uniform reflectance target.

* (j, δ) : index of the pixel on which the step of reflectance occurred and its value.

* g: the centroid of the image spot.

Step 3: Compute $x, f(j, \delta)$: the centroid of the reconstructed reflectance and the corresponding correction factor.

END.

Algorithm 2-2 (Correction of the effects of a reflectance step)

Begin

Step 1: Read the incident spot $I(i)$ from the detector.

Step 2: Read the position θ of the scanning mirror.

Step 3: Estimate d using the non zero pixels.

Step 4: Compute the reflectance $R(k) = \frac{I(k)}{I_{ref}(k)}$, where $I_{ref}(k)$ is the reference spot

corresponding to θ and $-d \le k \le d$.

Step 5: Compute the centroid g of the incident spot.

Step 6: Extract from $R(k)$ the step base α, the pixel index j and the step δ given by:

$\alpha = R(j-1)$;

$\delta = R(j+1)$.

Step 7: Compute the centroid $\bar{x}$ of $R(k)$.

Step 8: **IF** δ is not tabulated **THEN**

- Find δ_i, δ_{i+1} such that $\delta_i \le \delta \le \delta_{i+1}$.

- Compute $f(j, \delta)$ by interpolation between $f(j, \delta_i)$ and $f(j, \delta_{i+1})$.

END

Step 9: Compute the estimated value of the exact centroid position given by

$G = g - \bar{x} + f(j, \delta)\ (d - |j|)\ \delta$.

END.

3.7.3 Comments concerning the proposed correction models

In this section, we will make some comments concerning the two models presented in the previous two sections. These comments can be summarized as follows:

- The accuracy of the models of correction depends essentially on the accuracy of the extracted values a, I, d, j, δ, α during the camera calibration phases.

- If the number of entries of the correction tables is large enough, a first order interpolation is sufficient. However interpolations of higher orders could be used for a better

accuracy.

- During the camera calibration phases, multiple targets of known reflectance must be used to generate the calibration tables.

- For both correction models, the magnification factor cannot be exactly known because it is not available at the sensor output. Moreover, two image spots can have the same number of nonzero pixels d with different magnification factors, assuming two different ranges z_1 and z_2. This inaccuracy can be reduced by using a detector with small-area pixels. Thus, dividing the image spot located at an unknown range z_2 with a calibrated reference spot at a known range z_1 will still be valid.

3.8 Simulation results

In this section, we present results to show the efficiency of the models. To make these results more realistic, we have inserted the effect of the collecting lens in our simulator, to take into account the diffraction effect of the system. The MTF (modulation transfer function) [11], [12] of the collecting lens is shown in Figure 3.6 and it is modeled given by:

$MTF(u) = (2\text{acos}((u\lambda f) - \sin(2\text{acos}(u\lambda f)))) / \pi$ where λ is the wavelength of an incoherent light, f is the circular lens-aperture and u the spatial frequency. A block diagram of the assumed optical system is shown in Figure 3.7.

We will consider a target located at a constant range $z = 1m$ and a laser beam with a standard deviation $\sigma = 500\mu m$ and $\alpha = 3.5$. For all the simulations, we chose the

parameters listed in Table 3.1.

Table 3.1: Simulation parameters

Detector	Collecting lens
pixel size: $W = 46\mu m$	wavelength:$820nm$
interpixel distance: $D = 50\mu m$	diameter: $25mm$
number of pixels: $N = 256$	fnumber:4.35

3.8.1 Simulation of a reflectance gradient

We define a gradient of reflectance on the target with a constant base equal to 0.1. We vary the slope of the gradient for a reflectance ranging from 0.1 to 1.0 in steps of 0.1 for a given position θ of the scanning mirror. Figure 3.8 shows the different gradients that have been simulated. Since the range of the object is constant, the spot size on the detector is also constant and is equal to 13 pixels ($d = 6$). This value of d can be obtained mathematically or by simulations. Results of the camera calibration phase are shown in Table 3.2, where a is the estimated value of the slope, b is the value of $R'(j)$ at the center ($R'(0)$), xc the centroid of $R(j)$, and G the exact position of the centroid at position θ. All these parameters have been extracted from $R(j)$, as indicated in algorithm 1. Figure 3.9 shows the error on the centroid after correction for all values of a. For all the corrections where a is different from the calibrated value, a first order interpolation was used to estimate $f(a)$. Results of correction showed that the error, in all cases, was less than 0.011 pixel which is satisfactory for our applications.

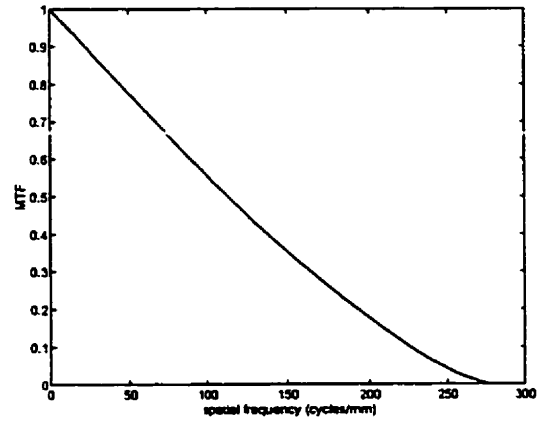

Figure 3.6: MTF of the collecting lens.

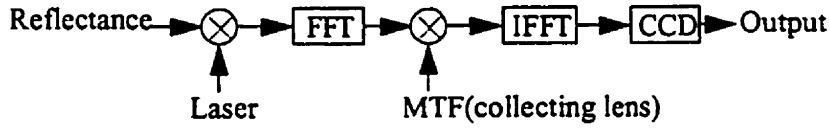

Figure 3.7: Model of the optical system under study.

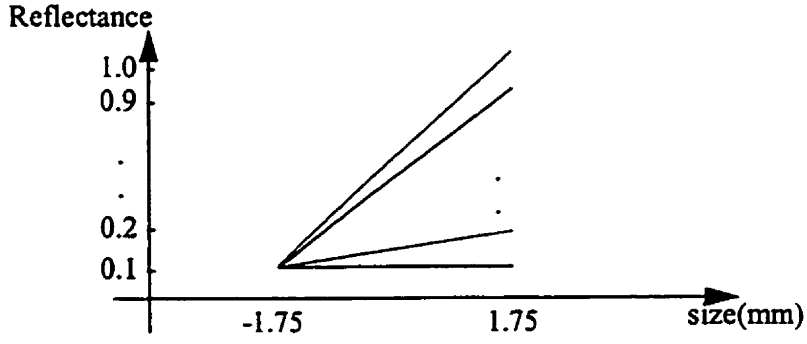

Figure 3.8: Simulated reflectance gradients during camera calibration.

Table 3.2: Extracted parameters during the camera calibration phase.

Centroid positions g	a	b	$\bar{x}$	$A = g - \bar{x}$	$G - A$	$f(a) = \frac{G-A}{a}$
133.1026	0.0000	0.1000	0.0000	133.1026	0.0000	60.7894
133.2629	0.0082	0.1492	0.6560	132.6069	0.4958	60.7894
133.3430	0.0163	0.1984	0.9867	132.3563	0.7464	45.7576
133.3910	0.0245	0.2476	1.1860	132.2050	0.8976	36.6859
133.4230	0.0326	0.2968	1.3192	132.1039	0.9988	30.6160
133.4459	0.0408	0.3460	1.4145	132.0314	1.0713	26.2696
133.4631	0.0489	0.3951	1.4861	131.9770	1.1257	23.0038
133.4764	0.0571	0.4443	1.5418	131.9346	1.1681	20.4602
133.4871	0.0652	0.4935	1.5865	131.9006	1.2020	18.4231
133.4958	0.0734	0.5427	1.6230	131.8728	1.2299	16.7549

In Table 3.2, the first value of G (133.1026) is obtained when $a = 0$ (uniform reflectance). However the corresponding correction factor $f(0)$ is undefined (zero denominator). To overcome this problem, we set $f(0)$ to a default value that allows the use of interpolation for noncalibrated slopes.

3.8.2 Simulation of a reflectance step

We define a reflectance step on the target with a constant base equal to 0.5. Two parameters are taken into account:

- the position of the step of reflectance;

- the value of the step of reflectance.

Since the position of the step can be anywhere in the gaussian spot, we decided to simulate the effects of three particular positions:

- step located at the first quarter of the gaussian spot (before the center);

- step located at the second quarter of the gaussian spot (at the center);

- step located at the third quarter of the gaussian spot (after the center);

For each position, the value of the step varies from 0.5 to 1.0 in steps of 0.1 for a given position θ of the scanning mirror. Figure 3.10 shows the different steps that have been simulated. The results obtained during camera calibration for each position are in Tables 3.3, 3.4, and 3.5. Figure 3.11, 3.12, and 3.13 show the error on the centroid after correction for the 3 positions for different noncalibrated values.

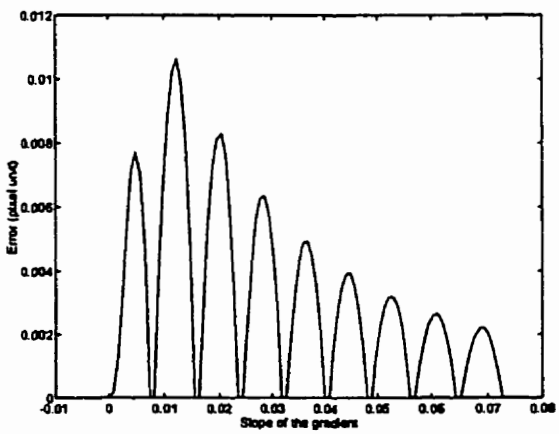

Figure 3.9: Error on centroid after correction.

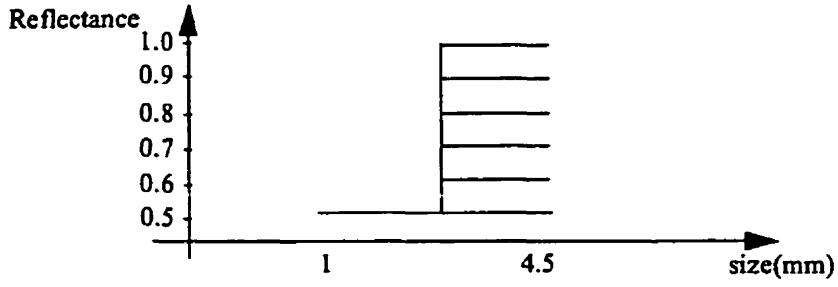

Figure 3.10: Simulated reflectance steps during camera calibration.

Table 3.3: Step in the first quarter of the spot (j=-3)

Centroid positions g	$\bar{x}$	δ	$f(j, \delta)$
133.1026	0.0000	0.0000	0.7100
133.1262	0.2107	0.1000	0.6237
133.1433	0.3733	0.2000	0.5545
133.1562	0.5026	0.3000	0.4990
133.1663	0.6079	0.4000	0.4535
133.1744	0.6952	0.5000	0.4156

Table 3.4: Step in the second quarter of the spot (j=0)

Centroid positions g	$\bar{x}$	δ	$f(j, \delta)$
133.1026	0.0000	0.0000	0.2973
133.2238	0.2848	0.1000	0.2727
133.3247	0.5225	0.2000	0.2504
133.4100	0.7240	0.3000	0.2315
133.4831	0.8969	0.4000	0.2152
133.5465	1.0470	0.5000	0.2011

Table 3.5: Step in the third quarter of the spot (j=3)

Centroid positions g	$\bar{x}$	δ	$f(j, \delta)$
133.1026	0.0000	0.0000	0.6882
133.1311	0.2240	0.1000	0.6520
133.1591	0.4278	0.2000	0.6189
133.1866	0.6139	0.3000	0.5888
133.2138	0.7845	0.4000	0.5611
133.2405	0.9415	0.5000	0.5358

As in the case of the reflectance gradient, $f(j, 0)$ was set to a default value. Tables 3.3, 3.4, and 3.5 show that the deviation on centroid position increases when the reflectance step occurs at the center of the spot and decreases as it moves away (for example, for $\delta = 0.3$, the centroid positions are 133.1562, 133.4100 and 133.1866 for $j = -3$, $j = 0$ and $j = 3$ respectively). Based on this remark, we may reduce the number of required calibrated Tables $f(j, \delta)$. Instead of calibrating all the pixels j, $-d \leq j \leq d$, we only calibrate a small number of pixels on both sides of the center. Any step occurring outside these pixels will be neglected. The number of pixels to be calibrated depends on the size of the image spot on the sensor.

3.8.3 Qualitative analysis of the correction models

In this section, we give some comments concerning aspects that may reduce the

effectiveness of the two models presented in the previous two sections. These comments can be summarized as follows:

- During the calibration process, the extraction of j is very important. The question is where to locate the reflectance step on a given pixel? Since each pixel has a finite size, one measurement may not be sufficient to ensure a good accuracy when dealing with non calibrated steps that could happen anywhere in the pixel. This problem can be overcome by having several calibrated measurements for each pixel. All the measurements are obtained from laser spots separated on the object by a fraction of pixel. In practice for a given measurement on a given pixel, we have to find among all the calibrated measurements of this pixel the one that is very close to the actual measurement. This means that we need to take into account not only the pixel on which the step occured, but also its corresponding energy. However this method could make the signal processing process more complicated.

- Besides the precision of the extracted value of j, the choice of an accurate algorithm to detect the peak position for subpixel resolution has a strong impact on the accuracy of the heuristics. In our simulations, we chose the centroid algorithm. However, better results could be obtained with derivative filters. An analysis of different methods for centroid position determination can be found in [13].

- In practice, the extracted parameters may be corrupted by different sources of noise. One of these sources is the speckle noise. The effects of speckle depend on its size which is function of the wavelength of the laser light, the diameter of the imaging lens and the distance between the imaging lens and the detector. The effects due to speckle can be

neglected if the size of each pixel is larger than the speckle size. Additional details about speckle effects can be found in [14] and [15].

3.9 Conclusion

In this paper, we presented two heuristic models for a fast correction of the deviation caused by a change of reflectance on a constant range target. The results showed that these models are satisfactory for the targeted applications. However, they can be improved by using higher interpolation orders. The effectiveness of these models relies on the accuracy of the parameters extracted during camera calibration. The inaccuracy caused by the estimation of the magnification factor can be neglected for the benefit of a quick correction. The effects due to speckle can be neglected if the size of each pixel is larger than the speckle size. These models could be easily implemented for real-time correction because of their low complexities. They also give information about the object reflectance profile which can be used by other image processing algorithms. The proposed models offer a good trade-off between accuracy, speed and complexity of implementation for applications performing 3-D treatments on targets of small volumes. Even if it takes time to generate the calibration tables based on targets with known reflectances and ranges, all these tables can be used for objects of equal 3-D volumes. Finally the use of these models can be extended to other cameras based on active triangulation.

As future work, we intend to study and develop new correction models that avoid reference spot measurements.

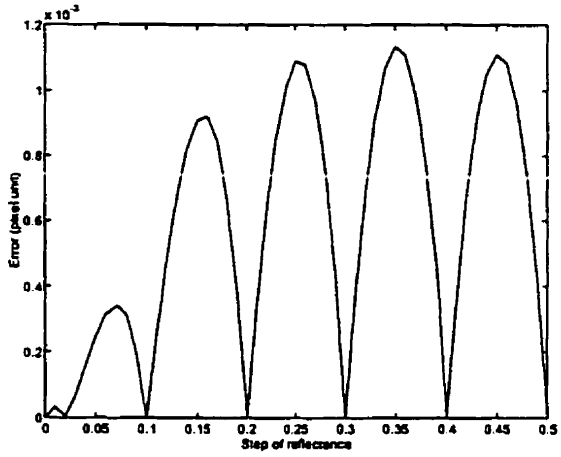

Figure 3.11: Error on centroid after correction (j=-3).

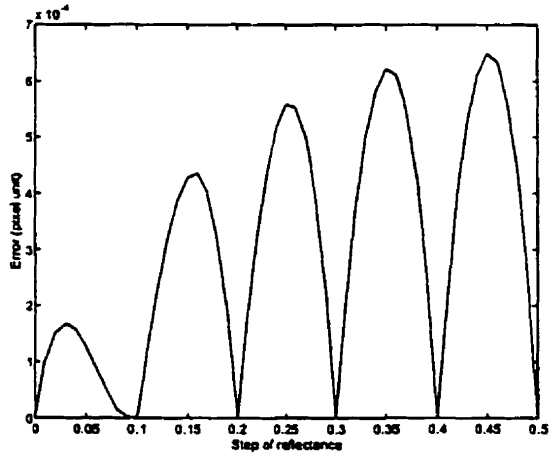

Figure 3.12: Error on centroid after correction (j=0).

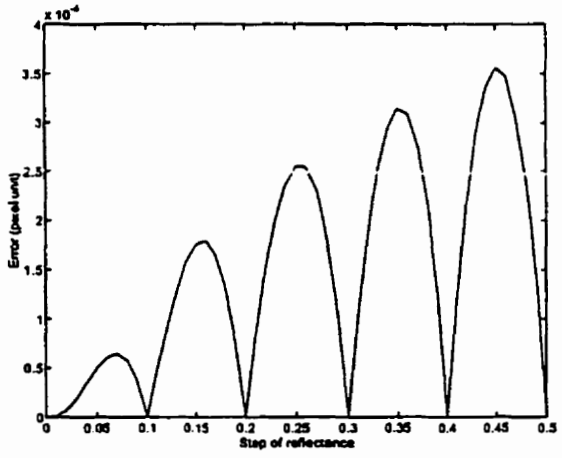

Figure 3.13: Error on centroid after correction (j=3).

3.10 References

[1] M. Rioux, "Laser Range Finder Based on Synchronized Scanners", Appl. Opt., 23, 1984, pp. 3837-3844.

[2] M. Rioux, "Applications of Digital 3-D Imaging", Canadian Conf. on Electr. and Comp. Eng., Ottawa, Sept. 1990, pp. 37.3.1-37.3.10.

[3] M. Rioux, G. Bechthold, D. Taylor and M. Duggan, "Design of a Large Depth of View Three-Dimensional Camera for Robot Vision", Opt. Eng., Vol. 26, No. 2, Dec. 1987, pp. 1245-1250.

[4] F. Blais and M. Rioux, "Real-Time Numerical Peak Detector", Signal Process., **11(2)**, 1986, pp. 145-155.

[5] F. Blais, M. Rioux, and J.-A. Beraldin, "Practical Considerations for a Design of a High-Precision 3-D Laser Scanner System", Proc. Soc. Photo. Opt. Instrum. Eng., 1988, pp. 225-246.

[6] J.-A. Beraldin, S. F. El-Hakim, and L. Cournoyer, "Practical Range Camera Calibration", SPIE Proceedings, Videometrics, **2067**, 1993, pp. 21-31.

[7] J. L. Mundy and G. B. Porter, "Part 3: A Three-Dimensional Sensor Based on Structured Light", Published in ***Three-Dimensional Machine Vision***, edited by Takeo Kanade, Kluwar Academic Publishers, 1987, pp. 3-61.

[8] A. R. Levine, ***High-Performance Optical Triangulation Ranging***, Ph.D Thesis, Purdue University, 1992.

[9] M. Soucy, D. Laurendeau, D. Poussart and F. Auclair, "Behavior of the Center of Gravity of a Reflected Gaussian Laser Spot near a Surface Reflectance Discontinuity", Elsevier, Industrial Metrology, 1, 1990, pp. 261-274.

[10] R. C. Kim, ***Design and Fabrication of Custom Beam Forming Optics for Optical***

Metrology and Signal Processing, Master Thesis, University of Minnesota, 1988.

[11] ***Handbook of Optics***, Vol. 1&2, McGraw-Hill Inc, 1995.

[12] E. P. Baltsavias, H. A. Beyer, D. Fritsch, and R. K. Lenz, ***Tutorial: Fundamentals of Real-Time Photogrammetry***, ETH Zurich, Switzerland, 1990.

[13] P. G. Backes and W. H. Stevenson, "A Comparison of Methods for Accurate Image Centroid Position Determination with Matrix Sensor", Journal of Laser Applications, Vol. 2, No. 2, Spring 1990, pp. 33-39.

[14] R. Baribeau and M. Rioux, "Centroid Fluctuations of Speckled Targets", Appl. Opt., **30**, 30, 1991, pp. 3752-3755.

[15] R. Baribeau and M. Rioux, "Influence of Speckle on Laser Range Finders", Applied Optics, Vol. 30, No. 21, 1991, pp. 2873-2878.

3.11 La méthode itérative

Dans la section précédente, nous avons présenté la méthode directe qui permet de corriger les artefacts associés à une variation de réflectance. Cette méthode se distingue principalement par sa rapidité de correction, obtenue grâce à des informations collectées lors de la phase de calibration. Cependant cette phase de calibration peut devenir problématique pour certains types d'applications, par exemple les applications spatiales. En effet, dans ce type d'environnement, il est extrêmement difficile et coûteux d'effectuer des mesures sur des cibles de référence. Afin de palier à ce problème, nous avons développé une seconde méthode dite itérative, qui s'offre comme une alternative à la méthode directe. La méthode itérative ne comporte que la phase de correction et elle est principalement adaptée à la correction des artefacts causés par des gradients de réflectance. La phase de calibration est remplacée par un ensemble de calculs itératifs basés sur le calcul de minimums et de maximums d'opérateurs mathématiques, afin d'inférer un estimé de la position du spot laser réfléchi sur le détecteur optique. Avant de présenter l'article correspondant à la méthode itérative, nous allons décrire les principaux traitements effectués ainsi que l'impact qu'a le facteur de magnification sur la correction finale. Supposons, et cela sans perte de généralité, que le spot optique réfléchi a une taille de 7 pixels dont les indices varient de $j-3$ à $j+3$, où j est l'indice du pixel central. En raison de la symétrie du spot gaussien et de la largeur finie d'un pixel (W), le centre du spot réfléchi se trouve sur le pixel j. Notons qu'à priori, nous ne connaissons pas la position exacte du centre du spot sur le pixel j. De plus, l'abscence d'information sur la

valeur du facteur de magnification ne nous permet pas de connaître avec précision l'étalement du spot réfléchi sur les pixels de queue. La connaissance du facteur de magnification est très importante lors de la reconstitution d'un estimé du gradient de réflectance. De la même façon que pour la méthode directe, la méthode itérative reconstruit un estimé du gradient de réflectance en divisant le spot laser reçu par un spot de référence correspondant à une réflectance unitaire. Cependant, vu qu'il n'y'a pas de phase de calibration, le spot de référence ne peut être physiquement mesuré. Cette incapacité de mesurer le spot de référence peut être contournée en inférant ce spot. Inférer un spot de référence consiste à accumuler (par calcul) un nombre prédéterminé d'échantillons d'un signal gaussien sur une bande de largeur finie afin de calculer les intensités de pixels. Le nombre d'échantillons à accumuler par pixel dépend du facteur de magnification. Après avoir reconstruit un estimé du signal de réflectance pour la mesure courante, l'algorithme calcule les différences finies d'ordre 1 et 2 du signal de réflectance. Cette étape vise à évaluer la qualité du gradient reconstitué. Si le signal reconstitué est un vrai gradient, alors les différences finies d'ordre 2 devraient converger vers 0. L'algorithme calcule ensuite le maximum des différences finies d'ordre 2, ce qui termine les calculs associés à la position courante du spot de référence sur le pixel central. La prochaine position à tester est donnée par la position courante incrémentée d'un pas. A chaque nouvelle position, on effectue l'ensemble des opérations décrites précédemment. Après avoir exploré l'ensemble des positions possibles sur le pixel central (pour le pas correspondant), l'algorithme calcule le minimum de tous les maximums précédemment calculés. Le but de cette étape est de localiser la position du signal de réflectance le plus proche d'un gradient. Tous ces

calculs ont été effectués pour une valeur donnée du facteur de magnification. En raison de l'abscence d'information sur l'étalement du spot laser réfléchi sur les pixels de queue, il est nécessaire de varier le facteur de magnification courant pour permettre une recherche fine de la position exacte du spot réfléchi. Pour cette raison, l'algorithme modifie la valeur courante du facteur de magnification avec un pas donné et réitère l'ensemble des calculs précédemment décrits. Une fois que l'algorithme a testé l'ensemble des valeurs possibles associées au facteur de magnification (pour le pas donné), il calcule alors le minimum de tous les minimums déjà calculés. La position de ce minimum final correspond alors à la position estimée du spot laser réfléchi. Par opposition à la méthode directe, la méthode itérative est caractérisée par une complexité de calcul plus importante, en raison du processus de recherche de la meilleure position possible du spot de référence. La complexité de calcul pour une recherche linéaire peut être bornée par $O((W/R)^2)$ où W est la largeur d'un pixel et R la résolution du système. Quant à la précision, celle-ci se compare à la précision atteinte par la méthode directe et peut être meilleure dans le cas de cibles non calibrées. Dans ce qui suit, nous allons présenter l'article correspondant à la méthode itérative. Cet article a été soumis pour publication à IEEE Transactions on Instrumentation and Measurements (Septembre 1998). L'article commence par une introduction qui situe le problème lié à une variation de réflectance et explique la difficulté à utiliser la méthode directe pour certains types d'applications. Cette introduction est suivie par la présentation des fondements mathématiques utilisés par la méthode itérative. L'algorithme est alors décrit en détail avec les différentes étapes de calcul. Des résultats de simulations fonctionnelles sont alors présentées dans le cas de

l'utilisation d'une caméra autosynchronisée. Finalement, l'article se termine par une conclusion qui résume les principales caractéristiques de la méthode itérative.

3.12 Iterative Model for Accurate Centroid Correction Applied to Laser Range Finders in the Case of a Reflectance Gradient

H. Khali, Y. Savaria and J. L. Houle
Ecole Polytechnique de Montréal
Electrical and Computer Engineering Department
P.O. Box 6079, Centre-Ville, Montreal, Quebec, H3C 3A7

M. Rioux and J. A Beraldin
National Research Council of Canada
Visual Information Technology
Montreal Road, Building M-50, Ottawa, Ontario K1A 0R6

D. Poussart
Laval University
Electrical Engineering Department
Quebec City, Quebec, G1K 7P4

Keywords: Laser range finders, artifact correction, reflectance modelling.

Abstract

In active laser range finders, the computation of the (x, y, z) coordinates of each point of a scene can be performed using the detected centroid p of the image spot on a sensor. When a change of reflectance occurs on the scene, a deviation Δp on the detected centroid p is introduced. This deviation leads to erroneous (x, y, z) coordinates. To correct this error, we previously proposed a correction heuristic that uses a spot of reference measured during a camera calibration phase with a target of uniform reflectance.

However, depending on the application, the reference spot is not always easily measurable. This paper presents how to avoid the measurements of spots of reference and how to infer iteratively their positions on the sensor. Simulation results are presented to show

the quality of the correction and the resulting accuracy in the case of a gradient of reflectance.

3.13 Introduction

In active laser range finders, the computation of the (x, y, z) coordinates of each point of a scene is performed using the detected centroid p of the image spot on a sensor. When the reflectance of the scene under analysis is uniform, the intensity profile of the image spot is a gaussian and its centroid is correctly computed. However, when a change of reflectance occurs on the scene, the intensity profile of the image spot is no longer a gaussian. This change introduces a deviation Δp on the detected centroid p which will lead to erroneous (x, y, z) coordinates. To correct this deviation, we developed a correction heuristic [1] that predicts an accurate spot centroid given by $G = g - \bar{x} + f(a)\,a$ where G is the corrected spot centroid, g the centroid of the incident spot on the sensor, x the centroid of the reconstructed reflectance gradient on the sensor, a the slope of the reconstructed reflectance gradient and $f(a)$ a correction factor. The estimated reconstructed reflectance associated to each pixel of the sensor is given by $R(j) = \frac{I(j)}{I_{ref}(j)}$, where $R(j)$ is the estimated reconstructed reflectance associated to pixel j, $I(j)$ is the energy collected by pixel j and $I_{ref}(j)$ is the energy associated to pixel j for a uniform reflectance (equal to 1) during camera calibration. This model produces good results, as long as the spot of reference remains measurable for the targeted application. However, depending on the application (space applications for example), the spot of reference is not easily measurable. To overcome this problem, and to increase the scope of applica-

tion of our correction model, we developed a new iterative model that avoids the measurements of spots of reference. The proposed method iteratively infers the position of the incident spot on the sensor. This paper is divided as follows. Section 3.14 presents the mathematical relation between the spot on the scene and its image on the sensor. Section 3.15 presents a new iterative correction model that infers the position of the incident spot on the sensor. Section 3.16 presents simulation results to show the resulting accuracy. Section 3.17 concludes the paper.

3.14 Mathematical relation between the laser spot and the image spot

In this section, we will describe the framework on which our analysis is based.

We consider an optical system based on an autosynchronised camera [2]. The image intensity is collected by a 1-D discrete detector composed of N cells (pixels). All the pixels have a width of W and an interpixel distance of D between the centers of two successive pixels. Since we assume a 1-D detector, our analysis will be carried out for the image spot cross section. The effects of noise and speckle are not taken into account. Additional details concerning the description of an autosynchronized system and the assumptions made in our analysis can be found in [1], [3] , [4] and [5]. In an autosynchronized system, the range of the object is mathematically given by:

$$z(p,\kappa) = Zinf(\kappa) - Z\delta inf(\kappa) + Pinf \cdot \frac{(Zo(\kappa) - Z\delta o(\kappa) - (Zinf(\kappa) - Z\delta inf(\kappa)))}{Pinf - p}$$

where p is the position on the sensor of the image spot, κ is the optical angle, $Pinf$ is the vanishing point and $Zinf(\kappa), Z\delta inf(\kappa), Zo(\kappa), Z\delta o(\kappa)$ are trigonometric trans-

formations that describe the geometry of the camera. For simplicity and without loss of generality, the following analysis will be carried out in the 1-D domain.

Let $g(x)$ be a 1-D gaussian laser beam illuminating the scene. $g(x)$ is given by

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\bar{x}}{\sigma}\right)^2} \tag{3.8}$$

where x and σ are respectively the center and the standard deviation of the illuminating beam. In an optical system, the object spot size and the sensor spot size are related to each other by the lateral magnification factor. Let us assume that any distance x on the scene is imaged on the sensor as a distance p given by $p = x \cdot M_l$, where M_l is the lateral magnification factor at range z. Using the previous relation, (3.8) can be rewritten as

$$g(x) = g\left(\frac{p}{M_l}\right) = \frac{1}{\frac{\sigma'}{M_l}\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\frac{p}{M_l}-\frac{\bar{p}}{M_l}}{\frac{\sigma'}{M_l}}\right)^2} \tag{3.9}$$

After suitable transformations on (3.9), we obtain

$$g(x) = M_l \cdot \frac{1}{\sigma'\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{p-\bar{p}}{\sigma'}\right)^2} = M_l \cdot I(p) \tag{3.10}$$

where $I(p)$ is the image spot on the sensor and p and σ' are respectively the center and

the standard deviation of $I(p)$. From (3.10), we get

$$I(p) = \frac{g(x)}{M_l} \tag{3.11}$$

In (3.11), the denominator M_l allows to conserve energy in the one-dimensional analysis. With respect to our assumptions, (3.11) shows that a gaussian on the scene is imaged as a gaussian on the sensor with a standard deviation $\sigma' = M_l \cdot \sigma$. The total energy under $g(x)$ is equal to the total energy under $I(p)$. This result will be used to infer the reference spot as shown later.

3.15 Description of the iterative correction model

In this section, we will describe a new iterative correction model. This model avoids spot measurements during camera calibration and predicts iteratively the image spot position on the sensor.

In the case of a reflectance gradient occurring on the scene, the estimated reflectance intensity associated to each pixel of the sensor is given by [1]

$$R(j) = \frac{I(j)}{I_{ref}(j)} = aj + b \tag{3.12}$$

where $R(j)$ is the estimated reflectance intensity associated to pixel j, a, b are respectively the slope and the base of the reflectance gradient on the sensor, $I(j)$ is the energy of pixel j, and $I_{ref}(j)$ is the reference energy collected by pixel j during camera calibration for a uniform reflectance (equal to 1). Equation (3.12) is valid only when the spots $I(j)$ and $I_{ref}(j)$ are at the same position $\bar{p}$ and have the same size. To recons-

truct $R(j)$, we need $I_{ref}(j)$, which is a gaussian spot imaged on the sensor when the target has uniform reflectance, that spot being integrated over each pixel area. Reference pixel energies are determined by the position p of $I_{ref}(j)$ on the sensor. No explicit information concerning p is available at the sensor output, except the pixel energies. This lack of information can be compensated by using mathematical properties of a gaussian spot. One of its properties is its symmetry around p. This property will be used as follows. Let us assume, as an example, that the incident spot covers seven non-zero pixels indexed from $j-3$ to $j+3$. Because of the symmetry property, we can claim that p is located on pixel j. Our goal is to find p on pixel j with enough accuracy. We first start by assuming an initial position p at the center of pixel j. We compute at this position the estimated reflectance R and some other parameters described later. We then move at the next position p with a search step. This process is iterated a given number of times. Then, we select among all the visited positions p the one that satisfies a selection criterion. Two cases will be analysed: the spot covers an odd number of non-zero pixels and an even number of non-zero pixels. The number of non-zero pixels depends on the object range. We will start with the first case. To simplify the following mathematical relations, we will assume that the image spot covers seven pixels identified from $j-3$ to $j+3$. Of course, all the results can be directly extended to any number of odd non-zero pixels.

Let us assume that the center of the image spot (p) is exactly at the center of pixel j. Since the spot of reference I_{ref} is a symmetric function, the current assumptions lead to

$$I_{ref}(j+3) = I_{ref}(j-3) \tag{3.13}$$

$$I_{ref}(j+2) = I_{ref}(j-2) \tag{3.14}$$

$$I_{ref}(j+1) = I_{ref}(j-1) \tag{3.15}$$

Using (3.12), we can write:

$$I_{ref}(j) = \frac{I(j)}{aj+b} \tag{3.16}$$

Using (3.12) and (3.13), we obtain

$$\frac{I_{ref}(j)}{I_{ref}(j+3)} = \frac{I_{ref}(j)}{I_{ref}(j-3)} \rightarrow \frac{I(j)}{aj+b} \cdot \frac{a(j+3)+b}{I(j+3)} = \frac{I(j)}{aj+b} \cdot \frac{a(j-3)+b}{I(j-3)} \tag{3.17}$$

After suitable transformations on (3.17), we obtain

$$\frac{I(j)}{I(j+3)} \cdot \left(\frac{aj+b}{aj+b} + \frac{3a}{aj+b}\right) = \frac{I(j)}{I(j-3)} \cdot \left(\frac{aj+b}{aj+b} - \frac{3a}{aj+b}\right) \tag{3.18}$$

Equation (3.18) can be simplified as

$$\frac{(1+3x_1)}{I(j+3)} = \frac{(1-3x_1)}{I(j-3)} \tag{3.19}$$

where $x_1 = \frac{a}{aj+b}$. Using (3.19), x_1 can be rewritten as

$$x_1 = \frac{1}{3}\left(\frac{I(j+3)-I(j-3)}{I(j+3)+I(j-3)}\right) \tag{3.20}$$

The same transformations can be applied to (3.14) and (3.15). We obtain

$$x_2 = \frac{1}{2}\left(\frac{I(j+2)-I(j-2)}{I(j+2)+I(j-2)}\right) = \frac{a}{aj+b} \tag{3.21}$$

$$x_3 = \left(\frac{I(j+1)-I(j-1)}{I(j+1)+I(j-1)}\right) = \frac{a}{aj+b} \tag{3.22}$$

Equations (3.20), (3.21) and (3.22) show that when p is exactly at the center of pixel j, we have:

$$x_1 = x_2 = x_3 \tag{3.23}$$

However, when p is not at the center of pixel j, (3.23) is no longer valid. Depending on the position of p on pixel j, intensive simulations showed that $x_1 < x_2 < x_3$ (when p is on the left of the exact spot position), or $x_1 > x_2 > x_3$ (when p is on the right of the exact spot position). These relations will be used to guide the search toward the exact position of the image spot. It is important to mention that the values of (x_1, x_2, x_3) are directly computed from the pixel energies available at the sensor output. These values are constant during the search process and are independant from the inferred reference spots generated by the search algorithm.

3.16 Effect of the lateral magnification factor

Dividing the image spot by a reference spot requires that two conditions be satisfied:

a) both spots have the same magnification factor (same size).

b) both spots are at the same position.

During spot measurement, the reading of the output sensor can only provide the pixel energies, with no information about the spatial distribution of the image spot. This lack of information introduces an error in the estimation of the magnification factor. As a result, (3.12) is no longer exact. To overcome this problem, the proposed method finds the image spot position for a given value of the magnification factor. Then, it modifies

the current magnification factor value and computes the new image spot position. This process is repeated until a termination condition is met.

Let us define M_l^i as the current value of the magnification factor. Based on (3.12), we have

$$\lim_{M_l^i \to M_l} R^i(j) = aj + b \tag{3.24}$$

where $R^i(j)$ is the estimated reconstructed reflectance associated to pixel j according to the magnification factor M_l^i. Equation (3.24) shows that when M_l^i converges to M_l, the estimated reconstructed reflectance R converges at a gradient. Thus, the proposed method achieves this convergence by repeating the correction algorithm with several values of M_l^i until no more improvement is possible.

3.17 Algorithm of the iterative correction method

In this section, we will describe the main steps involved in the proposed correction model, then the algorithm itself is provided. Note that explanations of each step of the algorithm follow in section 3.18. We assume that the incident spot covers $(2d+1)$ non-zero pixels. The pixel index ranges from $(j-d)$ to $(j+d)$. We first give a description of the variables used in the algorithm.

Description of the main variables

$I(i)$: Intensity of pixel i, $i = (j-d) \ldots (j+d)$.

$I_{ref}(i)$: Intensity of pixel i for a uniform reflectance (equal to 1).

$R(i) = \dfrac{I(i)}{I_{ref}(i)}$: Estimated reflectance corresponding to pixel i.

p : Current search position.

start_position, end_position : Initial and final search positions for a given value of M_l. These positions are respectively the center and the end-side of pixel j.

δ : Search step value.

M_l : Current value of the magnification factor.

M_{l_start} : Initial value of the magnification factor. This value corresponds to an initial reference spot whose end-sides cover a small fraction of the tail-pixels.

ΔM_l : value of the variation on the magnification factor.

Algorithm

BEGIN

Input: - I : incident image spot on the sensor.

- p : center of pixel j.

compute $x_1 = \dfrac{1}{3}\left(\dfrac{I(j+3) - I(j-3)}{I(j+3) + I(j-3)}\right)$.

compute $x_2 = \dfrac{1}{2}\left(\dfrac{I(j+2) - I(j-2)}{I(j+2) + I(j-2)}\right)$.

IF $x_1 < x_2$ **THEN** $\Delta p = \delta$ **ELSE** $\Delta p = -\delta$ **ENDIF.**

$M_l = M_{l\_start}$.

current_min = ∞ (default value).

last_min = ∞ (default value).

WHILE NOT FINISH

FOR p = start_position **TO** end_position **STEP** Δp

FOR $k = j - d$ **TO** $j + d$

step 1: compute $I_{ref}(k)$ by integrating (3.11) over pixel k.

step 2: $R(k) = \frac{I(k)}{I_{ref}(k)}$.

ENDFOR

FOR k = 2 TO $2d + 1$

step 3a: $slope1(k-1) = R(k) - R(k-1)$.

ENDFOR

FOR k = 2 TO $2d$

step 3b: $slope2(k-1) = slope1(k) - slope1(k-1)$.

ENDFOR

step 4: **IF** $max(slope2) <$ current_min **THEN**

current_min = $max(slope2)$.

$index1 = p$

ENDIF

ENDFOR

step 5: **IF** current_min ≤ last_min **THEN**

last_min = current_min .

current_solution = index1 .

$M_l = M_l + \Delta M_l$.

ELSE

FINISH = TRUE

ENDIF

ENDWHILE

final_solution = current_solution .

END.

3.18 Comments concerning the proposed correction model

In this section, we explain each step of the previous algorithm.

Step 1: for each pixel, we compute the inferred reference energy $I_{ref}(i)$ using (3.11). The reflectance is uniform (equal to 1).

Step 2: for each pixel, we compute the corresponding reflectance value $R(i)$.

Steps 3a and 3b: we compute the first and second order finite difference of $R(i)$. When $R(i)$ is the true gradient, then $slope1$ contains the slope of $R(i)$, and $slope2$ should converge towards 0. This step requires that the incident spot on the sensor covers several pixels to assure that the finite-difference operator is applicable.

Step 4: we compute the maximum of $slope2$. This value is compared to the minimum of the previous maximums already computed. If the current maximum is less than the previous minimum, it means that the current position of the reference spot becomes the new best estimate of the exact position for the given value of M_l. This step shows how far or how close we are from the exact position. The closer $max(slope2)$ is from 0, the closer we are from the exact position.

Step 5: in this step, the algorithm decides if a new iteration with a new value of M_l is

required. Recall that the closer $max(slope2)$ is from 0, the closer we are from the exact reflectance gradient. As a result, the algorithm starts a new iteration around a new value of M_1 as long as the function $max(slope2)$ decreases. The final solution is reached when no more improvement is possible.

In practice, pixel intensities may be corrupted by different sources of noise which may affect the parameters of the search algorithm like (x_1, x_2). One of these sources is the speckle noise. The effects of speckle depend on its size which is function of the wavelength of the laser light, the diameter of the imaging lens and the distance between the imaging lens and the detector. The effects due to speckle can be neglected if the size of each pixel is larger than the speckle size. Additional details about speckle effects can be found in [8] and [9].

All the previous analysis can be applied to an even number of pixels. The same transformations and results hold. For a $2d$-pixel spot size indexed from $j-d$ to $j+(d-1)$, we obtain

$$x_1 = \frac{1}{2}\left(\frac{I(j+1) - I(j-2)}{I(j+1) + I(j-2)}\right)$$
$$x_2 = \frac{1}{3}\left(\frac{I(j+2) - I(j-3)}{I(j+2) + I(j-3)}\right)$$

All the steps in the search algorithm are applicable. The initial value of $\bar{p}$ will be at the mid-distance between pixels $(j-1)$ and j.

3.19 Estimation of ΔM_l

The purpose of this section is to develop an equation that estimates the value of ΔM_l which is the variation around the magnification factor M_l. Estimating ΔM_l will lead to the estimation of M_l which is essential to compute $R(j)$ given by (3.12). As mentioned in section 3.15, equation (3.12) is exact only when the spots $I(j)$ and $I_{ref}(j)$ are at the same position $\bar{p}$ and have the same size. Recall that M_l has a direct impact on the image spot size on the detector and depends directly on the distance between the object and the sensor. Let S_x be the estimated spot size on the scene, and f_p be the smallest pixel fraction that can be represented for a given optical system. The value of f_p is determined by the resolution of the targeted optical system. If l_p is the system resolution on the spot position on the sensor, then $l_p = f_p \cdot W$ where W is the pixel size. The estimated image spot size on the sensor S_y for the corresponding magnification factor M_l^i is given by

$$S_y = S_x \cdot M_l^i \tag{3.25}$$

Let us take ΔM_l on M_l^i to be the change of the magnification factor such that the image spot size increases by l_p on both sides. Thus, the new image spot size S'_y is given by

$$S'_y = S_y + 2l_p = S_x \cdot \left(M_l^i + \Delta M_l\right) \tag{3.26}$$

Substracting (3.25) from (3.26) and rearranging give

$$\Delta M_l = \frac{2l_p}{S_x} \tag{3.27}$$

The maximum number of iterations (worst case) on M_l can be represented by the situation in which the true image spot and the current reference spot differ in overall size by

two pixels (one on each side). In this case, the maximum number of iterations $MaxIterations$ on M_l is given by

$$MaxIterations = \frac{W}{T_p} \tag{3.28}$$

Equation (3.28) corresponds to a linear search. One may choose a dichotomic search which will lead to a reduced number of iterations. However the impact of such a search method was not characterized and is beyond the scope of this paper.

3.20 Simulation results

In this section, we present simulation results to show the accuracy of the search algorithm. A block diagram of the model of the optical system is presented in figure 3.14. The simulation consists of scanning a constant range object and computing the residual error between the solution given by the search algorithm and the exact position p of the incident spot given by

$$p = Pinf - \frac{Pinf \cdot (Zo(\kappa) - Z\delta o(\kappa) - (Zinf(\kappa) - Z\delta inf(\kappa)))}{z(p,\kappa) - Zinf(\kappa) + Z\delta inf(\kappa)}$$ (see section 3.14).

We will consider a target located at a constant range $z = 1m$ and a laser beam with a standard deviation $\sigma = 500\mu m$. For all the simulations, we chose the parameters listed in Table 3.6. To get more realistic results, we take into account in the model the effect of the collecting lens. The MTF(modulation transfer function) [6] and [7] of the collecting lens is given by

$MTF(u) = (2\text{acos}((u\lambda f) - \sin(2\text{acos}(u\lambda f))))/\pi$, where λ is the wavelength of

an incoherent light, f is the circular lens-aperture and u the spatial frequency. Two sets of simulations have been carried out. The first set describes the situation where the exact value of M_l is known. Figures 3.15 and 3.16 show the corresponding residual error for two different values of the search step δ. For both simulations, the residual errors is better than the actual system resolution limit which is close to $0.78\mu m$. The second set of simulations describes the situation where the exact value of M_l is unknown. Only an estimated value is available. Figure 3.17 shows the number of iterations required to obtain an accurate estimation of the image spot position. Figure 3.17 shows that the model tried 3 iterations on M_l before getting a residual error less than $1\mu m$ for the given simulation setup.

.

Table 3.6: Simulation parameters

Detector	Collecting lens
pixel size: $W = 46\mu m$	wavelength: $820nm$
interpixel distance: $D = 50\mu m$	diameter: $25mm$
number of pixels: $N = 256$	fnumber: 4.35

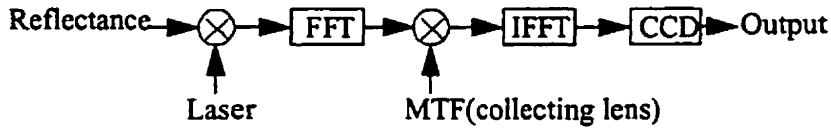

Figure 3.14: Model of the optical system under study.

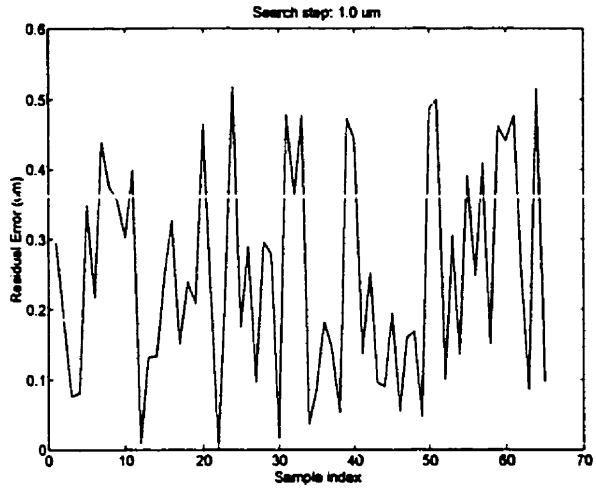

Figure 3.15: Residual error for $\delta = 1\mu m$.

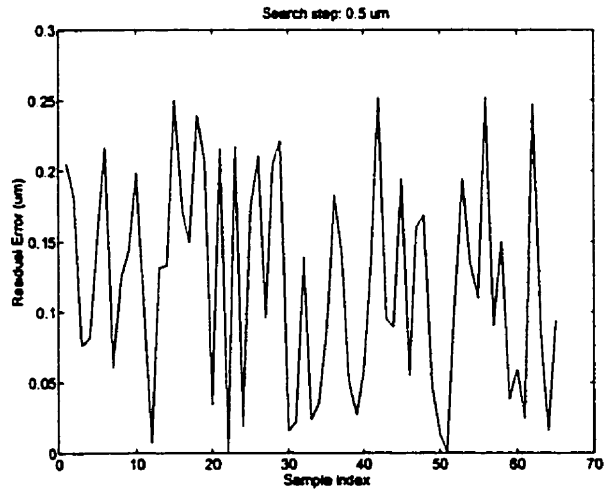

Figure 3.16: Residual error for $\delta = 0,5\mu m$.

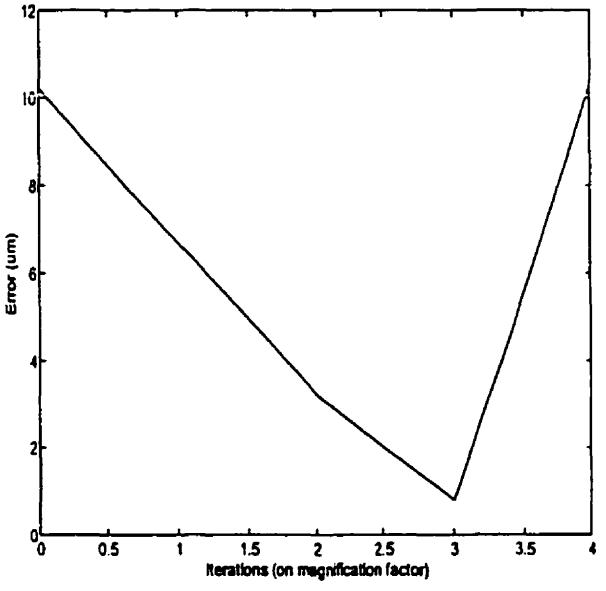

Figure 3.17: Residual error with a varying magnification factor.

3.21 Conclusion

In this paper, we presented a new iterative model for active laser range finders to correct the image spot deviation caused by a reflectance gradient. Compared to the correction algorithm presented in [1], this new model avoids spot reference measurements during camera calibration, which will lead to a reduced cost. Simulation results showed that the residual error was less than the actual system accuracy limit. Moreover, this residual error was less than the search step value. A better accuracy can be obtained by reducing the search step value to meet the requirements of high-precision applications. However,

this will increase the processing time of the search algorithm. Finally, this model can be extended to other cameras based on active triangulation.

3.22 References

[1] Khali, H., Savaria, Y., Houle, J. L., Rioux, M., Beraldin, J. A., and Poussart, D., "Improvement of Sensor accuracy in the Case of a Variable Surface Reflectance Using Active Laser Range Finders", submitted to IEEE Tansactions on Instrumentation and Measurements.

[2] M. Rioux, "Laser Range Finder Based on Synchronized Scanners", Appl. Opt., 23, 1984, pp. 3837-3844.

[3] F. Blais, M. Rioux, and J.-A. Beraldin, "Practical Considerations for a Design of a High-Precision 3-D Laser Scanner System", Proc. Soc. Photo. Opt. Instrum. Eng., 1988, pp. 225-246.

[4] J.-A. Beraldin, S. F. El-Hakim and L. Cournoyer, "Practical Range Camera Calibration", SPIE Proceedings, Videometrics, **2067**, 1993, pp. 21-31.

[5] J.-A. Beraldin, M. Rioux, R.A. Couvillon, and S. G. MacLean, "Development of a Real-Time Tracking Laser Range Scanner for Space Applications", Workshop on Com-

puter Vision for Space Applications, Antibes, France, Sept. 1993, pp. 161-171.

[6] ***Handbook of Optics***, Vol. 1&2, McGraw-Hill Inc, 1995.

[7] E. P. Baltsavias, H. A. Beyer, D. Fritsch, and R. K. Lenz, ***Tutorial: Fundamentals of Real-Time Photogrammetry***, ETH Zurich, Switzerland, 1990.

[8] R. Baribeau and M. Rioux, "Centroid Fluctuations of Speckled Targets", Appl. Opt., Vol. 30, No. 30, 1991, pp. 3752-3755.

[9] R. Baribeau and M. Rioux, "Influence of Speckle on Laser Range Finders", Applied Optics, Vol. 30, No. 21, 1991, pp. 2873-2878.

Chapitre 4 : Accélération du calcul des transformations géométriques

4.1 Introduction

Dans un système optique à illumination active, la reconstitution d'une image 3D se fait à partir de transformations géométriques définissant la relation caméra-objet (Khali, 1995). Ces transformations géométriques sont basées sur des informations collectées lors des mesures effectuées. Ces informations sont essentiellement les positions des miroirs et la position du spot réfléchi sur le détecteur optique. Lorsqu'un signal laser illumine une cible, il ne peut commencer une nouvelle mesure qu'après avoir attendu l'écoulement d'un délai Δt, qui représente le délai minimal requis par les cellules photosensibles pour intégrer la lumière réfléchie. Cette caractéristique du détecteur optique constitue l'un des facteurs essentiels dans l'obtention de caméras optiques à hauts débits. En effet, plus Δt est petit, plus le débit de la caméra est grand et plus Δt est grand, plus le débit est faible. Dans la première version de la caméra optique autosynchronisée, le débit atteint était de 20000 points/s, laissant ainsi un temps de $50\mu s$ entre 2 mesures pour calculer les coordonnées (x, y, z) . Un tel débit était peut-être convenable lorsque l'imagerie 3D était à ses balbutiements. Malheureusement, ceci n'est pas le cas pour les applications 3D modernes qui nécessitent des débits de données de plus en plus élevés. Pour cela, le CNRC a entamé la conception de détecteurs optiques rapides capables de délivrer 1 million de points/s, ce qui laisse un temps de $1\mu s$ entre 2 mesures pour le calcul en temps réel des coordonnées (x, y, z) . Une telle performance ne peut être atteinte

avec une approche de calcul traditionnelle, basée en général sur une version programmée des transformations géométriques. Une telle approche, même si elle demeure viable sur des processeurs de très hautes performances, telle que ceux la série ALPHA (Smith, Weiss, 1994) capable de fonctionner avec des horloges de l'ordre du GHz, entraînerait des coûts d'implantation considérables. Il devient alors nécessaire d'analyser ces transformations géométriques, d'en isoler les portions critiques lourdes en calcul, et de les accélérer adéquatement.

4.2 Analyse des équations des transformations géométriques

Comme nous l'avons mentionné à la section précédente, le calcul des coordonnées (x, y, z) est basé sur la connaissance de la position des miroirs et de la position du spot laser réfléchi sur le détecteur optique. Ces informations sont fournies à des équations qui définissent la géométrie de la caméra et de sa relation avec l'objet. D'une façon générale, les opérations impliquées dans le calcul des coordonnées (x, y, z) peuvent être divisées en 2 catégories (Khali, Savaria, Houle, Beraldin, Blais, Rioux, 1995):

1) opérations arithmétiques traditionnelles (addition, soustraction, multiplication et division),

2) opérations trigonométriques (sinus et cosinus).

Dans la majorité des processeurs modernes, les opérations de la catégorie 1 font partie du jeu d'instructions, permettant ainsi au programmeur de ne pas trop se préoccuper de leur implantation. Ces opérations sont en général optimisées en fonction de l'architecture du

processeur et peuvent présenter, dépendemment de l'instruction, des temps d'exécution de l'ordre du cycle d'horloge. Cette dernière caractéristique, n'est cependant pas valide pour l'opération de division qui est considérée comme une opération complexe et très peu utilisée (comparée à l'addition et à la multiplication). Il est très courant de doter les processeurs d'une opération de division avec une faible résolution (8 bits) capable de fonctionner à la vitesse du processeur et de laisser la liberté au programmeur de construire, à partir de la primitive de base, une opération de division de résolution supérieure (16, 32 et 64 bits). Quant aux opérations de la catégorie 2, celles-ci sont en général disponibles sous forme de bibliothèques mathématiques que le programmeur peut appeler dans son programme. Mis à part les délais associés à l'appel de ces bibliothèques, la performance associée à ces fonctions va grandement dépendre de l'algorithme utilisé pour les implémenter. Plusieurs algorithmes sont disponibles pour implémenter les fonctions trigonométriques (Fowkes, 1993; Koren, Zinaty, 1990; Rodrigues, Zurawski, Gosling, 1981; Walther, 1971). Cependant, les implémentations les plus couramment rencontrées pour les fonctions sinus et cosinus, sont basées sur 3 principaux algorithmes:

1) séries de Taylor,

2) Cordic,

3) tables LUT (lookup tables).

Les développements en série de Taylor sont considérés comme l'approche traditionnelle pour calculer une fonction sinus ou cosinus (Gradshteyn, Ryzhik, 1992). Le nombre de termes de la série dépend essentiellement de la précision désirée sur le résultat final. Les séries de Taylor sont basées sur l'opération de type "multiplier-accumuler" qui est une

opération de base dans les processeurs modernes. Quant à l'algorithme Cordic, celui-ci est utilisé pour évaluer des fonctions linéaires, circulaires ou hyperboliques (Dupras, Muller, 1993; Ercegovac, Lang, 1987; Haviland, Tuszynski, 1980, Kishore, Cavallaro, 1993, Takagi, Asada, Yajima, 1991). C'est un algorithme itératif dont le nombre d'itérations va dépendre de la précision sur le résultat final. Finalement, l'approche par tables LUT consiste à pré-calculer les résultats correspondant à toutes les valeurs possibles du paramètre d'entrée et à les stocker dans une mémoire. La largeur et la profondeur de cette mémoire va dépendre du nombre de bits voulus sur le résultat et du nombre de bits sur le paramètre d'entrée. L'algorithme LUT se différencie des 2 précédents (Cordic et Taylor) par le fait que ces 2 derniers calculent le résultat, alors que la version LUT ne le fait pas. Cette différence fondamentale nous amène à distinguer 2 modèles de calcul illustrés aux figures 4.1 et 4.2.

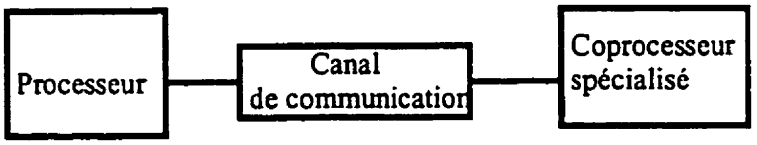

Figure 4.1: Modèle 1: calcul des résultats.

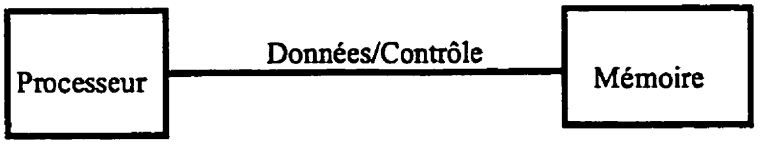

Figure 4.2: Modèle 2: tabulation des résultats.

4.2.1 Modèle 1: calcul des résultats

Dans ce modèle de calcul, on adopte une approche de type matériel-logiciel qui consiste à exécuter les opérations de la catégorie 1 (addition, soustraction, multiplication et division) sur le processeur et les opérations de la catégorie 2 (fonctions sinus et cosinus) sur le coprocesseur spécialisé (Khali, 1995). Ce dernier vise à accélérer les calculs basés sur les algorithmes de Taylor et Cordic. Il peut être implémenté sous la forme d'un circuit dédié (ASIC) ou d'un circuit reprogrammable de type FPGA, et cela afin d'exploiter le maximum de parallélisme disponible. La communication entre le processeur et son coprocesseur se fait grâce à un canal de communication qui peut être une mémoire à double ports, un bus, un port sériel/parallèle, etc. L'estimation du facteur d'accélération associé à ce modèle de calcul peut être obtenue en utilisant la loi d'Amdhal (Hennessy, Patterson, 1990; Hwang, 1993). L'accélération globale efficace SU_{eff} est donnée par (Savaria, Bois, Popovic, Wayne, 1988):

$$SU_{eff} = \frac{1}{\beta} \times \frac{1}{1 - f + (f/SU)} \tag{4.1}$$

où f est la fraction accélérée, β est le rapport du débit d'instructions du processeur et l'horloge du coprocesseur, et SU est le nombre soutenu d'instructions équivalentes du processeur exécutées par cycle d'horloge du coprocesseur. La performance d'un tel modèle de calcul va essentiellement dépendre de la vitesse du processeur et de celle du coprocesseur, de la quantité de parallélisme disponible, et de la faculté du processeur à calculer tout en communiquant avec le coprocesseur, et cela afin de réduire l'impact des délais des communications sur l'accélération globale. Ce modèle de calcul est adéquat sur des systèmes reconfigurables basés sur un DSP et un FPGA, où les portions de code

lourdes en calcul seraient implémentées en matériel et accélérées par le FPGA, alors que les autres portions de code seraient maintenues sur le DSP sous forme logicielle.

4.2.2 Modèle 2: tabulation des résultats

Dans le cas du modèle 1, les fonctions trigonométriques seraient accélérées grâce à l'utilisation d'un coprocesseur spécialisé. Ainsi, le processeur pourrait accéder aux résultats demandés au bout d'un nombre réduit de cycles (comparé à une implantation logicielle). Cependant, la nature itérative des algorithmes de calcul et les délais des communications constituent des facteurs qui limitent la performance globale du système. Afin de dépasser les limites de performance du modèle 1 avec des calculs du type envisagé, il est nécessaire et plus approprié de changer d'algorithme pour le calcul des fonctions critiques. Pour que ce changement d'algorithme soit efficace, il est important de tenir compte des caractéristiques générales d'un système optique à illumination active. En effet, si on examine les résolutions des positions angulaires et du spot réfléchi sur le détecteur optique, on constate que celles-ci sont finies et ne dépassent pas 16 bits pour les caméras commerciales. L'idéal dans un tel système serait de ne pas à avoir à calculer les fonctions trigonométriques, mais plutôt à les pré-calculer pour toutes les valeurs possibles des paramètres d'entrée. Cette approche est possible en adoptant une implantation par tables LUT. Une telle implantation est viable aussi longtemps que les tables n'ont pas une taille excessive et que le coût de la mémoire est réduit. Une analyse du marché des modules mémoires (Katayama, 1997) montre que non seulement les modules mémoires sont rapides et denses, mais aussi peu coûteux, que ce soit pour les mémoires conventionnelles de

type SRAM, DRAM ou avancées de type SSRAM, SDRAM, RAMBUS, etc. Tous ces facteurs nous amènent à analyser un second modèle de calcul, qui consiste à pré-calculer toutes les valeurs possibles des fonctions trigonométriques et à les stocker en mémoire. Chaque fois que le processeur a besoin d'un résultat, il lui suffit d'accéder à la mémoire contenant la table LUT correspondante. L'avantage majeur d'une telle approche est que l'accès mémoire peut se faire à la vitesse du processeur central, ce qui conduit à des facteurs d'accélération élevés.

Afin de mieux cerner les facteurs qui limitent l'accélération basée sur les tables, nous allons analyser à la section suivante l'accélération par table et nous allons présenter un modèle mathématique qui permet d'estimer cette accélération. Un tel estimateur permettra au concepteur d'apprécier l'impact d'utiliser les modèles de calcul 1 et 2.

4.3 Accélération des calculs par tables LUT (Khali, Savaria, Houle, 1997)

L'accélération basée sur des LUT consiste à remplacer l'exécution de N instructions par un accès mémoire. La valeur de cette accélération dépend d'un ensemble de paramètres liés tant à l'architecture qu'à l'application. La figure 4.3 illustre le modèle de calcul détaillé par LUT qui sera utilisé dans notre analyse.

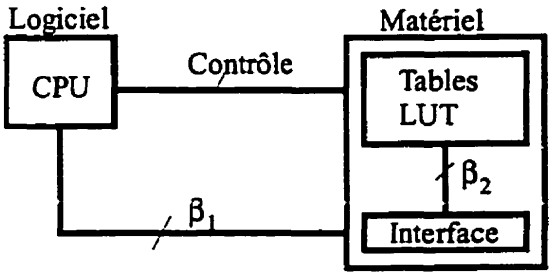

Figure 4.3: Modèle de calcul par tables LUT.

Afin de modéliser adéquatement l'accélération liée aux LUT, un ensemble de paramètres doivent être pris en compte, à savoir:

CPI: nombre moyen de cycles machine par instruction.

β_1 : taille en bits du bus de données du processeur.

β_2 : taille en bits des résultats délivrés par LUT.

k: temps d'accès mémoire moyen en cycles du processeur.

Ω: nombre moyen de résultats, du point de vue de l'application, que l'on peut lire simultanément par accès mémoire (par exemple, des fonctions partageant un argument commun telle qu'une position angulaire).

f: fraction du code de l'application qui peut être accélérée par LUT.

En se basant sur la loi d'Amdhal, un estimé de l'accélération moyenne G_s basée sur des LUT, est donné par:

$$G_s = \frac{1}{1 - f + \frac{f \cdot k}{N \cdot CPI \cdot min\left(\frac{\beta_1}{\beta_2}, \Omega\right)}} \tag{4.2}$$

L'expression analytique G_s montre que, pour une structure de système donné ((β_1, k)

fixes), la valeur de l'accélération va surtout dépendre de la granularité N des fonctions à accélérer. Plus une fonction est lente en logiciel, plus c'est payant de l'accélérer par une LUT. Néanmoins, la taille des résultats à transférer a aussi un impact important sur la performance globale. En effet, dépendamment du processeur utilisé, on distingue 3 principaux cas:

$$1)\ \beta_1 = \beta_2 \rightarrow \forall\Omega,\ min\left(\frac{\beta_1}{\beta_2}, \Omega\right) = 1 \rightarrow G_s = \frac{1}{1 - f + \frac{f \cdot k}{N \cdot CPI}}.$$

$$2)\beta_1 < \beta_2 \rightarrow \forall\Omega,\ min\left(\frac{\beta_1}{\beta_2}, \Omega\right) = \frac{\beta_1}{\beta_2} \rightarrow G_s = \frac{1}{1 - f + \frac{f \cdot k \cdot \beta_2}{N \cdot CPI \cdot \beta_1}}$$

$$3)\ \beta_1 > \beta_2 \rightarrow G_s = \frac{1}{1 - f + \frac{f \cdot k}{N \cdot CPI \cdot min\left(\frac{\beta_1}{\beta_2}, \Omega\right)}}.$$

Le cas 1 décrit la situation où la taille du résultat est égale à la taille du bus de données. Ceci limite le système au transfert d'une donnée utile par accès mémoire. Dans le cas 2, la taille du bus de données est plus petite que la taille du résultat dont la lecture nécessitera plus d'un accès mémoire. Quant au cas 3, celui-ci décrit la situation où le bus de données du processeur est plus grand que la taille d'un résultat. Dépendamment de la valeur de Ω, plusieurs résultats pourront alors être transférés simultanément pendant chaque accès mémoire, ce qui entraînera une accélération de calcul élevée. Le modèle d'accélération par LUT peut être étendu à une hiérarchie mémoire de n niveaux (Cache, Sram, Dram, etc). Chaque niveau i est caractérisé par les paramètres (N_i, β_{2i}, k_i) qui

représentent respectivement le nombre moyen d'instructions en logiciel des fonctions implémentées en LUT sur le niveau i, la taille en bits des résultats délivrés par LUT du niveau i, et le temps d'accès mémoire du niveau i. L'accélération G_s est alors donnée par:

$$G_s = \frac{1}{1 - f + \sum_{i=1}^{n} \frac{f_i}{L_{si}}} \tag{4.3}$$

où $f = \sum_{i=1}^{n} f_i$ et $L_{si} = \frac{N_i \cdot CPI \cdot min\left(\frac{\beta_1}{\beta_{2i}}, \Omega\right)}{k_i}$.

4.3.1 Résultats

Dans le but de vérifier la qualité de la prédiction d'accélération donnée par G_s, les transformations géométriques de la caméra autosynchronisée pour le calcul des coordonnées (x, y, z) ont été codées en langage C sur les simulateurs de deux processeurs DSP le TMS320C40 (C40) et le ADSP-21060 (SHARC). A titre de rappel, nous redonnons les équations de calcul des coordonnées (x, y, z) (section 1.2):

$$x = x(p, i) = \frac{X_g(i)}{p - P_\infty} - X_0(i) . \tag{4.4}$$

$$y = y(p, i, j) = \left(\frac{Z_g(i)}{p - P_\infty} - Z_0(i)\right) \sin\phi(j) + H_y(j) . \tag{4.5}$$

$$z = z(p, i, j) = \left(\frac{Z_g(i)}{p - P_\infty} - Z_0(i) \right) \cos\phi(j) + H_z(j) . \qquad (4.6)$$

où

$$X_g(i) = \sum_{n=0}^{5} A_n i^n, \; X_0(i) = \sum_{n=0}^{5} B_n i^n, \; Z_g(i) = \sum_{n=0}^{5} C_n i^n, \; Z_0(i) = \sum_{n=0}^{5} D_n i^n .$$

$$H_y(j) = \sum_{n=0}^{5} E_n j^n, \; H_z(j) = \sum_{n=0}^{5} F_n j^n \text{ et } \phi(j) = \sum_{n=0}^{3} G_n j^n .$$

Les simulations supposent un C40 opérant à 50 MHz avec une mémoire externe SRAM fonctionnant à 0 WS (wait-state). Le SHARC opère à 40 MHz avec 512Ko de mémoire SRAM interne. Toutes les opérations ont été effectuées en virgule flottante pour des raisons de simplicité. Les résultats obtenus sont présentés à la table 4.1.

Table 4.1: Résultats de performances.

Processeur	Implantation logicielle		Implantation par LUT	
	Nombre de cycles/point	Nombre de points/s	Nombre de cycles/point	Nombre de points/s
DSP C40	416	120192	68	735294
DSP SHARC	336	119048	56	714286

Soient G_{C40} et G_{SHARC} les accélérations respectives du C40 et du SHARC. Elles sont données par:

$$G_{C40} = \frac{416}{68} = 6.11, \; G_{SHARC} = \frac{336}{56} = 6 .$$

Afin de comparer ces valeurs avec les valeurs prédites par le modèle théorique, nous devons au préalable évaluer les paramètres du modèle. Nous avons:

$\beta_1 = \beta_2 = 32bits, min\left(\frac{\beta_1}{\beta_2}, \Omega\right) = 1, k = 1, f = 0.94, N = 11.4, CPI = 1$ (f et N sont des valeurs estimées). L'accélération globale G_s est alors égale à:

$$G_s = \frac{1}{1 - 0.94 + \frac{0.94}{11.4}} = 7.04.$$

L'accélération prédite par le modèle ($G_s = 7.04$) est légèrement différente de l'accélération atteinte ($G_{C40-SHARC} = 6$) car le modèle ne tient pas compte des délais associés à la gestion des pipelines (conflits), aux délais de branchement et aux différentes latences, ce qui n'est pas le cas du simulateur. Cependant, on constate que l'estimateur peut être efficacement utilisé au niveau système pour fournir au concepteur une première estimation de l'accélération associée à l'utilisation des tables pour une même précision de calcul. Afin de mesurer la limite d'accélération associée à l'utilisation des LUT par le langage C, nous avons codé les équations des transformations géométriques en langage assembleur sur le SHARC. Le simulateur a estimé le temps de traitement par point (x, y, z) à 28 cycles, donnant ainsi un débit estimé de 1428571 points/s, ce qui permet de soutenir la performance des nouveaux détecteurs optiques visés.

4.4 Partitionnement matériel-logiciel basé sur des LUT

L'accélération d'une fonction logicielle par LUT est une approche très attrayante. Cependant, la quantité de mémoire disponible dans un système de traitement de données est limitée. De plus, lors de la conception d'un tel système, le coût associé ne doit pas

dépasser une limite préétablie. Il est donc possible que l'on ne puisse pas implémenter toutes les fonctions logicielles d'une application donnée sous forme de LUT, et cela faute de ressources mémoires. Il faut alors choisir, parmi toutes les fonctions disponibles, lesquelles seront implémentées sous forme de LUT. Pour atteindre cet objectif, nous avons développé un algorithme de partitionnement pour des applications basées sur des LUT. Cet algorithme de partitionnement, de nature heuristique, permet, à partir d'une contrainte de coût globale, de déterminer quelles fonctions devraient être implémentées sous forme de LUT et la nature des modules mémoires à allouer afin de minimiser, si possible, le temps de traitement global. A la fin du partitionnement, deux ensembles seront alors disponibles: l'ensemble des fonctions implémentées en logiciel et l'ensemble des fonctions implémentées sous forme LUT. Dans ce qui suit, nous allons présenter les principaux aspects liés à l'algorithme de partitionnement proposé. Des détails supplémentaires sont disponibles dans l'article correspondant.

4.4.1 Modèle de l'architecture cible

L'architecture cible capable d'implémenter le partitionnement d'une application entre une partition logicielle et une partition matérielle basée sur des LUT est illustrée à la figure 4.4.

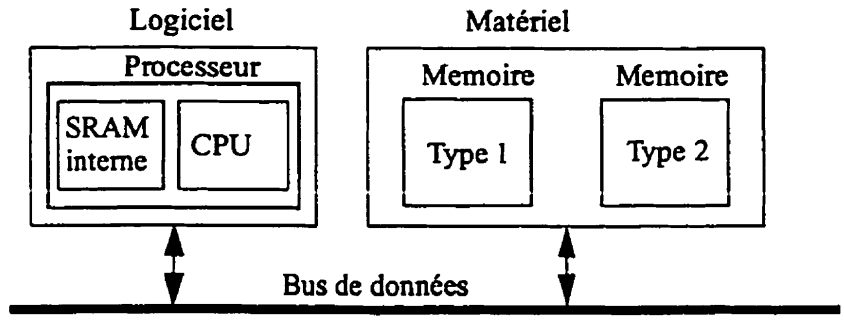

Figure 4.4: Architecture cible.

L'architecture cible illustrée à la figure 4.4 est basée sur un processeur central, tel qu'un processeur RISC ou DSP, qui implémente la partition logicielle. Quant aux LUT, celles-ci sont implémentées avec la mémoire interne du processeur et des modules mémoires externes de 2 types, tels que des modules mémoires SRAM ou DRAM. Chaque type de module mémoire est caractérisé par sa capacité, son coût et son temps d'accès. L'avantage majeur du modèle proposé est que le temps d'accès mémoire est déterministe, ce qui permet une évaluation précise de la performance de l'application partitionnée.

4.4.2 Spécification du problème

Soit une application définie par un ensemble de fonction $F = \{F_1, F_2, ..., F_n\}$. A chaque fonction F_i sont associés deux algorithmes: un algorithme décrivant F_i lorsqu'elle est implémentée en logiciel et un autre algorithme décrivant F_i lorsqu'elle est implémentée en matériel sous forme de table. Les résultats générés par ces fonctions sont combinés entre eux par des opérateurs arithmétiques ou logiques afin de générer le résultat final. Le problème à résoudre consiste à minimiser le temps de calcul $T(F)$ (ou

à maximiser le débit) en respectant une contrainte de coût $Cost(F) \leq C$. Afin d'atteindre cet objectif, on se propose de partitionner l'application visée (ensemble F) en deux partitions matériel-logiciel telles que la partition logicielle représente les fonctions F_l implémentées avec leurs algorithmes logiciels et la partition matérielle représentent les fonctions F_m implémentées avec leurs algorithmes matériels. Deux cas seront abordés: 1) tables de même taille et 2) tables de tailles différentes.

4.4.3 Stratégie de partitionnement

L'algorithme de partitionnement est de nature heuristique et il opère de manière "Greedy" (gourmande). Il essaie toujours de déplacer vers le matériel sous forme de LUT les fonctions logicielles les plus lentes. Pour atteindre cet objectif, il construit 2 vecteurs triés: V_time, qui représente le vecteur des temps d'exécution des fonctions en logiciel, et V_size qui représente le vecteur des tailles mémoires des LUT associées aux fonctions.

A) Tables LUT de tailles égales

Lorsque les LUT occupent la même taille mémoire (tous les éléments du vecteur V_size sont égaux), toutes les fonctions sont alors équivalentes du point de vue du matériel. L'algorithme commence par une partition entièrement logicielle et alloue x modules mémoires de type 1, et y modules de type 2. Puis, il effectue le remplissage, dans l'ordre, de la mémoire interne du processeur, de la mémoire externe de type 1 et de la mémoire externe de type 2 (du plus rapide au plus lent). Le remplissage d'une mémoire

se fait par le déplacement atomique d'un groupe de fonctions. Pour le couple (x, y) de modules mémoires, l'algorithme calcule un score de performance (une métrique) pour la partition courante. Ce score représente une estimation du temps de traitement global de l'application si elle était partitionnée de façon identique à la partition courante. Puis, l'algorithme alloue un nouveau couple (x, y) de modules mémoires et réitère les opérations précédemment énoncées. La solution finale est optimale (le temps de traitement global de l'application est minimal et la contrainte de coût est respectée) et est obtenue en calculant le minimum de tous les scores obtenus.

B) Tables LUT de tailles différentes

Lorsque les LUT occupent des tailles mémoires différentes (les éléments du vecteur V_size ne sont plus égaux), les fonctions ne sont plus équivalentes du point de vue du matériel. L'algorithme commence toujours par une partition entièrement logicielle. Cependant, au lieu de déplacer la fonction logicielle F_i la plus lente de V_time vers le matériel sous forme de LUT, il cherche à former un ensemble Ω de fonctions logicielles, ensemble qui préserve le coût matériel de F_i et maximise le gain en temps. Cet ensemble Ω est formé en utilisant le principe de "data-clustering". Si cet ensemble existe, alors il est déplacé vers le matériel sous forme de LUT et F_i est provisoirement maintenue en logiciel. Concernant l'allocation des modules mémoires, l'algorithme suit le même principe que celui associé aux tables LUT de tailles égales. La solution finale est obtenue en calculant le minimum de tous les scores obtenus. Cependant, l'optimalité de la solution dépend étroitement de la méthode de "data-clustering" choisie.

4.4.4 Résultats

L'algorithme de partitionnement a été appliqué aux transformations géométriques de la caméra autosynchronisée. Les résultats obtenus montrent un facteur d'accélération de 5 en utilisant un DSP TMS320C40 et des tables LUT, comparés à une version entièrement logicielle. Les deux types de mémoire utilisés fonctionnent respectivement à 0WS et 2 WS, ce qui a causé une légère perte de performance comparée aux résultats présentés à la section 4.3.1. Des performances supérieures pourraient être obtenues sur des processeurs disposant d'une mémoire interne plus grande, tel que le ADSP-21160 appelé SHARC-2.

Dans ce qui suit, nous allons présenter l'article dont a fait l'objet l'algorithme de partitionnement proposé. Cet article a été soumis pour publication à Elsevier Journal of Electrical and Computer Engineering (Mai 1999). L'algorithme commence par expliquer les motivations de développer un tel algorithme. Puis, il présente les principales hypothèses sur lesquelles l'algorithme est basé (architecture cible, spécification d'entrée, estimateurs, etc). Les stratégies de partitionnement sont alors présentées avec les algorithmes correspondants. Ces derniers sont suivis des résultats obtenus en appliquant l'algorithme de partitionnement sur les transformations géométriques de la caméra autosynchronisée. Finalement, l'article se termine par une conclusion qui résume les principales caractéristiques de la méthode de partitionnement proposée.

4.5 A System Level Implementation Strategy and Partitioning Algorithm for Applications Based on Lookup Tables

H. Khali, Y. Savaria and J. L. Houle
Ecole Polytechnique de Montréal
Electrical and Computer Engineering Department
P.O. Box 6079, Station Centre-Ville, Montreal, Quebec, H3C 3A7

Keywords: System partitioning, lookup tables, memory modules.

Abstract

Partitioning an application among interacting hardware and software components is an important part of system design. In this paper, we introduce a new partitioning algorithm, well adapted to applications that are composed of functions that can either be implemented in software or in hardware as lookup tables. Simulation results are presented to show the speedups that the method can produce.

4.6 Introduction

Partitioning an application among interacting hardware and software components is an important part of system design. This partitioning seeks to accelerate an application by extracting its critical portions for hardware implementation. Pushing parts of an application in hardware usually increases hardware costs and the method considered in this paper assumes a limited total hardware cost. Partitioning an application can be done at the operation level (e.g. Arithmetic operations), or at the algorithmic level (e.g. Processes), and can be divided in two categories: homogenous and heterogenous [1]. Homogenous partitioning splits a given hardware description into a minimal number of hardware components, which are smaller than a given size constraint. This partitioning is done

with respect to area, pinout and latency constraints. Heterogenous partitioning splits a given application into hardware and software components. The software components implement model functionality as an instruction-driven computation (RISC processors, DSP processors, microcontrollers,...). The hardware components operate as data-driven reactive components (ASIC, FPGA,...). Several automatic partitioning algorithms have been developed to investigate a large scope of solutions. Commonly used classes of algorithms include clustering, iterative-improvement, genetic and custom algorithms [2]-[11]. Some algorithms are fast, others such as genetic algorithms are slower. Not surprisingly, slower algorithms often find better solutions. They all start from a functional description of the system expressed in VHDL, Hardware C, graphs, etc. After checking the correctness of the description, the functionality is decomposed into functional blocks of some granularity. Each block is mapped into hardware or software. This mapping is guided by automatic estimators that evaluate cost functions. The output is a set of hardware and software blocks. Simulations of the designed hardware and compiled software can then be performed to observe the partitioning effects.

General-purpose partitioning algorithms target a wide variety of applications. However, this leads to some limitations:

- when some functions can be implemented with several algorithms (such as trigonometric functions), the designer usually rewrites the initial functional description of these functions and iterates again through the partitioning process. This makes the design process long and expensive, even if the partitioning task is automated. Indeed, when the designer wants to improve a given solution, he often uses slow and computationally

complex algorithms to find the optimal solution, which requires a large amount of cpu time. Also, depending on the type of application and system architecture, an accurate estimation of the communication time between the software and hardware partitions can be very difficult, which could lead to an inaccurate evaluation of the performance of a given hardware-software partition.

The previous limitations motivate the development of a custom partitioning algorithm well adapted to a specific type of applications. Among these applications, we focus on the ones composed of a set of functions that can either be implemented in software or in hardware as lookup tables (LUT). This type of application has the following characteristics:

- the whole software code of a given function can be replaced by a memory access. In this case, the communication time between software and hardware is the access time to the table, which is often deterministic. This characteristic can lead to exact knowledge of the communication time between software and hardware partitions.
- the cost of hardware is easily computed, and depends mainly on the number of memory modules chosen by the designer. This characteristic will lead to exact computation of hardware costs.

For these reasons, we propose a new algorithm, well adapted to the partitioning of applications based on lookup tables. Moreover, this work is motivated by the increasing number of applications that make use of lookup tables [13]-[16] and the availability on the

market of faster, wider and cheaper memory modules [18], which represent an easy and efficient way of accelerating critical software functions. The proposed partitioning algorithm is based on the knowledge of exact communication times between the software and hardware partitions and the knowledge of exact hardware costs. This paper is organized as follows. Section 4.7 to 4.11 respectively give a definition of the hardware-software partitioning problem, explain the partitioning strategy, present the partitioning algorithms, present some simulation results and conclude the paper.

4.7 Problem Definition

4.7.1 Functionality description

The initial description of the system functionality is a set of functions $F = \{F_1, F_2, ..., F_n\}$, each of which can be implemented in software or in hardware as lookup tables. This means that each function is described by two algorithms: one for software implementation and the other for hardware implementation. The targeted input functions are those which compute a result based on one or more parameters. The results generated by these functions are combined with logical or arithmetic operators to give the final result. Figure 4.5 shows a simple example of the initial description of the functions that belong to the target class. When a function is implemented as a LUT, its corresponding hardware size will increase exponentially with the number of address bits. Thus, it is necessary to limit the number of address bits for a given LUT to avoid a considerable cost increase. However, a function can have multiple input parameters if the total number of input bits is acceptable, which may lead to multidimensional LUTs. Moreover, if an input function is referenced several times in the initial description at different control

steps, there should be only one copy of the LUT accessed as many times as needed. In this case, the LUT cost will be taken into account only once.

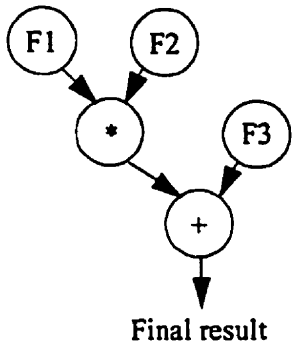

Figure 4.5: Example of an initial description of the functions which implementation needs to be optimized.

4.7.2 System architecture

A system architecture that can effectively implement partitions of an application between software and lookup tables (LUT) is shown in Figure 4.6. LUTs can be seen as a special kind of hardware implementation. Performance estimates and implementation costs are expressed in terms of that architecture in the rest of the paper. This architecture comprises a processor (RISC, DSP, etc.) that implements the software part of the application. Tables are implemented with a set of memory modules of two common types (such as SRAM, DRAM). Each type is characterized by three parameters: capacity, cost, and access time. Each time the processor needs data from the lookup table, it issues a read instruction from memory and waits for the memory access time before getting the data.

This computation time is deterministic, which helps the designer to get an accurate estimate of the partition performance. Depending on the LUT organization, memory can be interleaved to increase system bandwidth for pipelined access of contiguous memory locations. Several memory words can then be accessed per time unit. This technique is very useful to close the speed gap between a CPU and its main memory. Based on the architecture shown in Figure 4.6, a given function can be implemented on one of 4 possible modules:

- the processor core;
- an on-chip SRAM (this paper does not consider cache memory);
- an off-chip SRAM;
- an off-chip DRAM.

The task of the partitioning algorithm is to determine for each function a destination according to time/cost constraints.

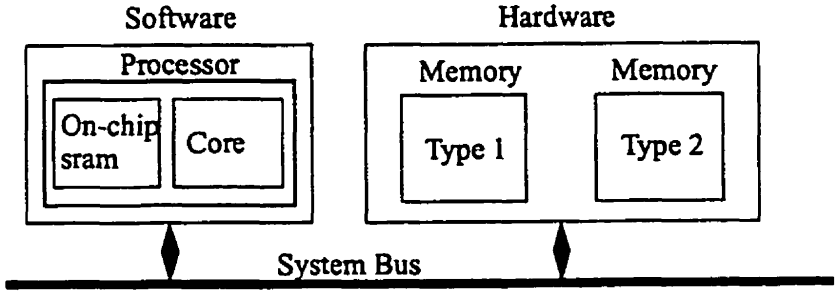

Figure 4.6: System architecture.

4.7.3 Partitioning problem

Partitioning an application among hardware and software modules requires cost and performance estimators. For applications based on lookup tables, the performance of a given partition depends mainly on the software structure, the speed of the processor, and the memory system components. The corresponding cost is a function of processor and memory modules costs. In what follows, we present the main parameters and estimators that allow to evaluate the cost/performance of a given partition.

We define T_{cycle} as the cycle time of the processor, and k_i, Q_i, C_i, as the access time, the capacity, and the cost of a memory module of type i, with $i = 1, 2$. Note that k_i is expressed in cycles, Q_i in memory words, and C_i in a monetary unit (\$). We define S, H_1, H_2, H_3 as the sets of functions respectively implemented in software, as lookup tables implemented with type 1 memories, as lookup tables implemented with type 2 memories, and as lookup tables implemented with on-chip memory. We have:

$F = S \cup H_1 \cup H_2 \cup H_3$,

$H_1 \cap H_2 = \emptyset, H_1 \cap H_3 = \emptyset, H_2 \cap H_3 = \emptyset, S \cap (H_1 \cup H_2 \cup H_3) = \emptyset$.

We define two operators $T(f)$ and $Size(f)$ to estimate the processing time of function f and its corresponding memory size when implemented as a lookup table. We have:

- $f \in H_1 \Rightarrow T(f) = k_1 \times T_{cycle}$.
- $f \in H_2 \Rightarrow T(f) = k_2 \times T_{cycle}$.
- $f \in H_3 \Rightarrow T(f) = T_{cycle}$
- $f \in S \Rightarrow T(f) =$ {depends on the software implementation}. In this case, $T(f)$ is

usually provided by a software profiler for the targeted processor. The value of $T(f)$ not only depends on the software code corresponding to function f, but also depends from which memory type this software code is executed. The software code may reside in internal or external memory. In this paper, we assume that the whole function software code resides in internal memory and can be executed at full processor speed. The value $Size(f)$ of a given function is invariant during the partitioning process and is given with the initial description of the targeted application.

We also define two operators $P(F_{x,y})$ and $Cost(F_{x,y})$ to estimate respectively the performance and the cost of partition $F_{x,y}$. We have:

$$P(F_{x,y}) = \sum_{f \in S}^{Card(S)} T(f) + \sum_{f \in H_1}^{Card(H_1)} T(f) + \sum_{f \in H_2}^{Card(H_2)} T(f) + \sum_{f \in H_3}^{Card(H_3)} T(f) + \tau \qquad (4.7)$$

where τ is the time needed by the processor to combine all the intermediate results of the functions to generate the final result, and (x, y) are respectively the number of memory modules of type 1 and 2 allocated by the partition. The first four terms of $P(F_{x,y})$ are summations over the subsets S, H_1, H_2 and H_3 respectively, and the number of terms in each of these summations is the cardinality of these subsets. The corresponding cost of $P(F_{x,y})$ is given by:

$Cost(F_{x,y}) = C_{cpu} + x \times C_1 + y \times C_2$, where C_{cpu} is the processor cost.

The value returned by $P(F_{x,y})$ can be seen as a worst case, and it does not take into account any possible parallelism between memory access and computations in various

modules. That corresponds to the simple case where the processor has one address bus and does not compute when accessing external memories. Taking into account such possible parallelism is very much processor specific, application specific and complex.

Given all the previous estimators, the partitioning problem can be defined as:

$$minimize\ P(F_{x,y})$$
$$Cost(F_{x,y}) \leq C_T$$

where C_T is the cost constraint. C_T does not include the memory cost to hold the application code. It only considers the cost limit to accelerate the critical software functions with LUTs. Two cases will be analyzed in the next section: fixed-size lookup tables and variable-size lookup tables.

4.8 Partitioning strategy

As mentioned in section 4.7.3, the memory size, $Size(f)$, of function f is invariant during the partitioning process. This size is known at "compile time". Thus, the cost of a partition, $Cost(F_{x,y})$, depends mainly on the number of memory modules required to implement this partition. The number of memory modules is directly affected by C_T. For this reason, we compute (x_{max}, y_{max}) as:

$x_{max} = \left\lfloor \frac{C_T - C_{cpu}}{C_1} \right\rfloor, (y = 0)$, and $y_{max} = \left\lfloor \frac{C_T - C_{cpu}}{C_2} \right\rfloor, (x = 0)$, where (x_{max}, y_{max}) are respectively the maximum number of memory modules of type 1 and 2 that are compatible with the global cost constraint. For each pair of (x, y) memory modules, we compute $P(F_{x,y})$. The final solution will be given by $min(P(F_{x,y}))$.

Before presenting our partitioning strategy, we introduce two sorted vectors: V_time and V_size, which contain respectively the function software times and the required memory size for each function implemented as a table. If a function is called several times, V_time will contain its corresponding total CPU time. Figure 4.7 shows an example of V_time and V_size. Recall that V_time is expressed in T_{cycle} and V_size in memory words. Of course, V_time and V_size are sorted independently of each others, which means that they can contain functions in different orders.

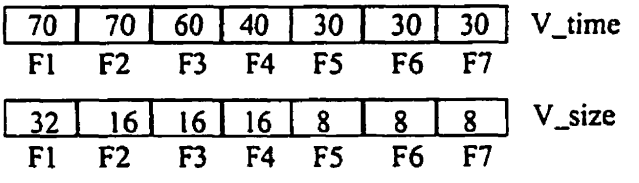

Figure 4.7: An example of V_time and V_size.

4.8.1 Fixed-size lookup tables

In this case, all software functions occupy the same amount of memory when implemented in hardware as lookup tables. All the elements of V_size are equal, which means that from the hardware point of view, all functions are equivalent. From the system point of view, the performance depends on the memory type where the lookup tables are implemented.

The proposed partitioning starts from an all software partition. The algorithm allocates x type 1 and y type 2 memory modules and calls *SplitAlg*1, which performs the partitioning. *SplitAlg*1 starts by computing the total memory capacities for the current parti-

tion. Let us respectively define $Sizeproc$, $Size_x$ and $Size_y$ as the on-chip memory capacity, the total memory capacity available in x type 1 modules and the total memory capacity available in y type 2 modules. Of course, $Sizeproc$ is a constant during the partitioning process. Then, the algorithm tries to fill the on-chip memory with a number of LUT functions given by $n_0 = \lfloor Sizeproc/Size(f) \rfloor$. If $n_0 \geq 1$, then the first n_0 functions in V_time are moved in one-step to the on-chip memory. If there are less than n_0 functions in V_time, then all the available functions are assigned to the internal processor memory. However, if V_time is not empty, the algorithm tries to fill the type 1 memory modules with a number of LUT functions given by $n_1 = \lfloor Size_x/Size(f) \rfloor$. If $n_1 \geq 1$, then up to the next remaining n_1 functions in V_time are moved in one step to the type 1 memory modules. Finally, if V_time is still not empty, the algorithm tries to fill the type 2 memory modules with a number of LUT functions given by $n_2 = \lfloor Size_y/Size(f) \rfloor$. If $n_2 \geq 1$, then up to the next remaining n_2 functions in V_time are moved in one step to the type 2 memory modules. The remaining functions in V_time, if any, are implemented in software. Then, the algorithm computes $P(F_{x,y})$ and iterates the previous partitioning process for a new pair of (x, y) modules. When the number of allocated memory modules, given by (x, y) is greater than what is required by the partition, empty memory modules may be created. Avoiding these empty memory modules decreases the partition cost. This cost reduction can be achieved by computing (x_{eff}, y_{eff}) which represents the effective number of memory modules consumed by the partition. For each pair (x, y), it exists a unique (x_{eff}, y_{eff}), such that $x_{eff} \leq x$ and $y_{eff} \leq y$. The partition cost is then given by $P\left(F_{x_{eff}, y_{eff}}\right)$ instead of

$P(F_{x,y})$. The final solution is obtained by computing $min\left(P\left(F_{x_{eff}, y_{eff}} \right) \right), x = 0 \ldots x_{max}, y = 0 \ldots y_{max}$. To reduce the number of processing steps, the algorithm starts to iterate from x_{max} or y_{max}, depending on which one is the smallest.

4.8.2 Variable-size lookup tables

In this case, from the hardware point of view, the functions are no more equivalent. Thus, before moving the current slowest software function f, we need to find the sets Ω_j of software functions, such that:

$$Size(\Omega_j) \leq Size(f)$$
$$D_j = (T(\Omega_j) - T(f)) > 0$$

where Ω_j, if they exist, are the sets that preserve the cost corresponding to f and produce cumulative software times larger then the one corresponding to f. To achieve that goal, one may use a data-clustering algorithm available in the literature, for example the bin-packing algorithm [12] to form sets Ω_j. Thus, instead of moving the current slowest software function f to hardware, the algorithm moves the set Ω, if it exists, such that:

$$Size(\Omega) \leq Size(f)$$
$$T(\Omega) - T(f) = max(T(\Omega_j) - T(f)), \forall j$$

After choosing a particular data-clustering method, the algorithm calls $SplitAlg2$, which performs the partitioning. $SplitAlg2$ follows a partitioning strategy similar to the case of fixed-size lookup tables. Let us stress that $SplitAlg2$ does not necessarily move

the current slowest function to hardware, but it rather moves the set Ω that fits and provides the largest reduction in processing time.

4.9 Algorithms

In this section, we present the partitioning algorithms corresponding to fixed-size and variable-size lookup tables. Let us assume, for simplicity and without loss of generality, that type 1 memory is faster and more expensive that type 2 memory. Thus, we have $k_1 < k_2$, $C_1 > C_2$, and $Q_2 > Q_1$. As defined in section 3, (x_{max}, y_{max}) are given by: $x_{max} = \left\lfloor \frac{C_T - C_{cpu}}{C_1} \right\rfloor, (y = 0)$ and $y_{max} = \left\lfloor \frac{C_T - C_{cpu}}{C_2} \right\rfloor, (x = 0)$. Note that since $C_1 > C_2$, then $x_{max} \leq y_{max}$.

4.9.1 Fixed-size lookup tables

Algorithm Split_fixed_size
Begin
Input: function software times and sizes: V_time, V_size.
system functionality set: F.

Output: software partition: S.
hardware partitions: H_1, H_2 and H_3.

For $x = 0$ **To** x_{max}
$S = F, H_1 = \emptyset, H_2 = \emptyset, H_3 = \emptyset$.
Allocate y type 2 memory modules such that $y = \left\lfloor \frac{(C_T - C_{cpu}) - (x \times C_1)}{C_2} \right\rfloor$.
*SplitAlg*1 $(x, y,$ V_time, V_size, $S, H_1, H_2, H_3, x_{eff}, y_{eff})$.

Compute $P\left(F_{x_{eff}, y_{eff}}\right)$ using (4.7).

End

Compute $min\left(P\left(F_{x_{eff}, y_{eff}}\right)\right)$.

End.

Procedure *SplitAlg*1

Begin

Input: (x, y) : respective number of allocated memory modules of type 1 and 2.
sorted vectors V_time and V_size.

Output: software partition: S.
hardware partition formed by sets (H_1, H_2, H_3) .
(x_{eff}, y_{eff}) : the effective number of utilized memory modules of type 1 and 2.

Compute $Size_x = x \times Q_1, Size_y = y \times Q_2$.

Compute $n_0 = \lfloor Sizeproc/\text{V\_size}(1) \rfloor$.

If $n_0 \geq 1$ **Then**

move to H_3 the n_0 first functions of V_time.

update V_time

End

If V_time **Not** Empty **Then**

Compute $n_1 = \lfloor Size_x/\text{V\_size}(1) \rfloor$.

If $n_1 \geq 1$ **Then**

move to H_1 the n_1 remaining first functions of V_time.

update V_time

compute x_{eff}.

End

End

If V_time **Not** Empty **Then**

Compute $n_2 = \lfloor Size_y/\text{V\_size}(1) \rfloor$.

If $n_2 \geq 1$ **Then**
move to H_2 the n_2 remaining first functions of V_time.
update V_time
compute y_{eff}.
End
End

Update S
End.

For each (x, y) memory module configuration, the algorithm partitions the initial software set S using procedure $SplitAlg1$, which moves to hardware the slowest software functions. When the partitioning is done, we compute its performance using the estimator $P\left(F_{x_{eff}, y_{eff}}\right)$. This process is repeated for all considered values of (x, y). The final solution is given by computing $min\left(P\left(F_{x_{eff}, y_{eff}}\right)\right)$.

4.9.2 Variable-size lookup tables

Algorithm Split_variable_size
Begin
Input: function software times and sizes: V_time, V_size.
system functionality set: F.

Output: software partition: S.
hardware partitions: H_1, H_2 and H_3.

For $x = 0$ **To** x_{max}
$S = F, H_1 = \emptyset, H_2 = \emptyset, H_3 = \emptyset$.
Allocate y type 2 memory modules such that $y = \left\lfloor \frac{(C_T - C_{cpu}) - (x \times C_1)}{C_2} \right\rfloor$.
$SplitAlg2$ (x, y, S, V_time, V_size, H_1, H_2, H_3, x_{eff}, y_{eff}) .

Compute $P\left(F_{x_{eff}, y_{eff}} \right)$ using (4.7).

End

Compute $min\left(P\left(F_{x_{eff}, y_{eff}} \right) \right)$.

End.

Procedure *SplitAlg2*

Begin

Input: (x, y) : respective number of memory modules of type 1 and 2.
sorted vectors V_time and V_size.

Output: software partition: S.
hardware partition formed by sets (H_1, H_2, H_3).
(x_{eff}, y_{eff}) : the effective number of utilized memory modules of type 1 and 2.

Compute $Size_x = x \times Q_1$, $Size_y = y \times Q_2$.

While V_time **Not Empty**

Take current function f.

Build sets Ω_j such that $Size(\Omega_j) \leq Size(f)$.

$$D_j = (T(\Omega_j) - T(f)) > 0$$

Find set Ω such that $Size(\Omega) \leq Size(f)$.

$$T(\Omega) - T(f) = max(T(\Omega_j) - T(f)), \forall j$$

If set Ω exists **Then**

If $Size(\Omega) \leq Sizeproc$ **Then**

Move Ω to H_3.

$Sizeproc = Sizeproc - Size(\Omega)$.

Else

If $Size(\Omega) \leq Size_x$ **Then**

Move Ω to H_1.

$Size_x = Size_x - Size(\Omega)$.

Else

If $Size(\Omega) \leq Size_y$ **Then**
Move Ω to H_2.
$Size_y = Size_y - Size(\Omega)$.

Else Keep f in software in set S.
End
End
Else Keep f in software in set S.
End
Update (V_time)
compute x_{eff}.
compute y_{eff}.
End
End.

This algorithm is similar to the previous one except for procedure $SplitAlg2$ which uses a suitable data-clustering method for moving to hardware the slowest software functions.

4.9.3 Comments concerning the proposed algorithms

The partitioning algorithms presented in the previous section are driven by suitable knowledge of which function to move to hardware at each iteration. This fundamental information is provided by the sorted vector V_time. Once a function is moved to hardware, it is never reconsidered. Thus, the proposed partitioning algorithms can be classified as greedy algorithms and do not use backtracks. It is well known that greedy algorithms are usually not optimal and can be easily trapped in local minima. In the case of fixed-size lookup tables, the proposed algorithm is optimal because all the software functions have the same hardware cost. Thus, the performance of the partition depends only on the function software times. In the case of variable-size lookup tables, the pro-

posed algorithm is usually not optimal (the optimality depends on the data clustering method). However, at each move, the total software time is considerably decreased. Finally, the complexity of the partitioning algorithm is constant and is independent of the number of functions in the case of fixed-size lookup tables. In the case of variable-size lookup tables, the complexity of the partitioning algorithm depends on the complexity of the data-clustering algorithm. Finding the optimal set Ω that satisfies cost/performance constraints may lead to an exponential complexity.The total complexity can be bounded by $O(n^2)$ assuming a linear data clustering algorithm and a state of the art sorting algorithm. This complexity of $O(n^2)$ describes a worst case, in which, for each software function F_j, the partitioning algorithm examines all software functions $F_i, i \neq j$ in V_time to build the corresponding sets Ω_j.

4.10 Results

In this section, we will describe the application chosen to evaluate our partitioning algorithm. This application computes a 3-D image from data collected by an optical system based on an autosynchronized triangulation system [17]. This system is based on a laser illuminating a target, mirrors to reflect the light, and collecting lens to focus the reflected light on a sensor. The (x, y) dimensions are controlled by two scanners and the z dimension is related to the position of the reflected light on the sensor. The re-mapping equations from the camera coordinate system to the target coordinate system are given by:

$$x = x(p, i) = \frac{X_g(i)}{p - P_\infty} - X_0(i) .$$

$$y = y(p, i, j) = \left(\frac{Z_g(i)}{p - P_\infty} - Z_0(i) \right) \sin\phi(j) + H_y(j) .$$

$$z = z(p, i, j) = \left(\frac{Z_g(i)}{p - P_\infty} - Z_0(i) \right) \cos\phi(j) + H_z(j) .$$

where

$$X_g(i) = \sum_{n=0}^{5} A_n i^n , \; X_0(i) = \sum_{n=0}^{5} B_n i^n , \; Z_g(i) = \sum_{n=0}^{5} C_n i^n , \; Z_0(i) = \sum_{n=0}^{5} D_n i^n .$$

$$H_y(j) = \sum_{n=0}^{5} E_n j^n , \; H_z(j) = \sum_{n=0}^{5} F_n j^n \text{ et } \phi(j) = \sum_{n=0}^{3} G_n j^n .$$

The values $A_n, B_n, C_n, D_n, E_n, F_n, G_n$ for $n = 0 \ldots 5$, are parameters describing the camera, (p, i, j) represent respectively the position of the collected light on the sensor and the discrete positions of the x-y scanners. (p, i, j) are respectively given on 16, 10 and 10 bit resolutions. All computations are assumed to be performed by a TMS320C40 DSP processor (C40) from Texas Instrument [19]. The functions involved in the remapping equations are implemented using Horner's rule and Taylor's series. The corresponding execution times are estimated using the C40 data book and are presented in Table 4.2. Table 4.3 presents the characteristics of the memory modules.

Table 4.2: Characteristics of the software functions

Functions	Estimated software times (cycles)	Memory required (Kwords)
X_g	12	1
X_0	12	1
Z_g	12	1
Z_0	12	1
H_y	12	1
H_z	12	1
$\sin(\phi(j))$	20 (*)	1
$\cos(\phi(j))$	20 (*)	1
$\frac{1}{p - P_\infty}$	12	64

(*): Based on Taylor's series.

Table 4.3: Characteristics of memory modules

Memory types	Access times (cycles)	Capacities (Kwords)
1	1	3
2	3	16

The computations of the (x, y, z) coordinates describe the case of variable-size LUT.

Function $SplitAlg2$ has been implemented using a modified version of the bin-packing

algorithm. The results of the partitioning are presented in Tables 4.4 and 4.5.

Table 4.4: Contents of hardware and software sets after partitioning

Type 1 memory	Type 2 memory	Cpu
X_g	$\frac{1}{p - P_\infty}$	$\varnothing$
H_y	X_0	
H_z	Z_g	
$\sin(\phi(j))$		
$\cos(\phi(j))$		
Z_0		

Table 4.5: Characteristics of the final solution

Characteristics
Initial software time: 132 cycles.
New processing time: 25 cycles.
Speed-up: 132/25 = 5.28
Number of allocated type 1 modules: 2
Number of allocated type 2 modules: 5
Total memory required: 72 Kwords.

Table 4.4 shows that the partitioning algorithm moved the largest LUT to memory type 2 modules and kept in memory type 1 modules as many LUTs as it was possible because

the time gains are greater. Table 4.5 shows that with the given configuration, we can run 5 times faster using LUT. This speedup can be increased by using other DSPs with a large amount of on-chip memory, like the SHARC ADSP-21060 [20].

Figure 4.8 shows the intermediate results of the estimated processing times for the different combinations of type 1 and type 2 memory modules allocated during the partitioning process. From figure 4.8, we can see that the partitioning algorithm, for the given configuration, finds the best partition by combining type 1 and type 2 modules to reach a balance between cost and speed. In fact, when using LUTs, the designer has to find the best way to combine slow-large-capacity memory modules with fast-small-capacity memory modules. Memory modules are getting faster, wider and less expensive. Their access times vary from 10 ns (asynchronous RAM) down to 5 ns (synchronous RAM), which allow a high acceleration factor for a function over its corresponding software version. These fast memories should be coupled to state-of the-art processors to maximize system performances. However, from system point of view, the choice of suitable processors and memory modules is based on cost-performance constraints. This aspect can be analyzed using a metric that produces a score that reflects the cost-performance implementation of a function in software or in hardware as a LUT. Let us define $\gamma(f)$ as a metric given by:

$\gamma(f) = A(f) \cdot T(f)$, where $A(f)$ is the cost required to execute function f using a processor or a LUT, and $T(f)$ its corresponding execution time. The characteristics of processors and memory modules on which our analysis is based are listed in Tables 4.6 and

4.7.

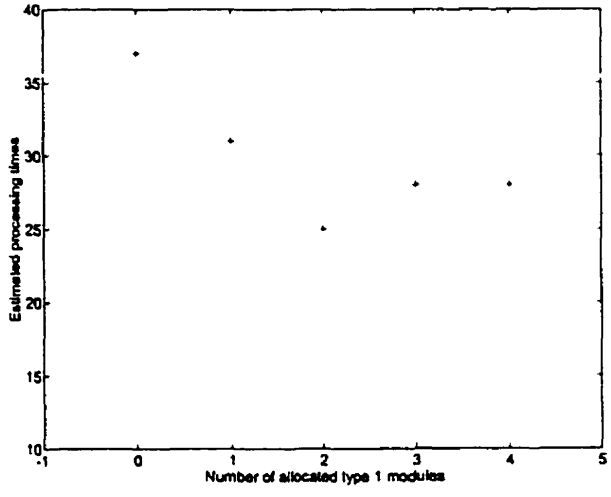

Figure 4.8: Estimated processing times for each number of allocated type 1 & 2 memory modules.

Table 4.6: Processor characteristics

Processors	Bus clock frequencies (MHz)	Internal memory sizes (32-bit words	Cost ($)
TMS320C44	30	2K	185
TMS320C67	167	16K	395

Table 4.7: Memory module characteristics

Memory modules	Access times (ns)	Sizes	Costs ($)
SRAM	12	256Kx16	124
SDRAM	8	64 MB (module)	128

The cost of a LUT-based function depends on the amount of memory required to implement the LUT. To model the impact of this characteristic, we take function $\sin(\phi(j))$ as a target function (see Table 4.2) where we vary its corresponding address size. Three situations have been analyzed:

a) $\sin(\phi(j))$ is implemented in software;

b) $\sin(\phi(j))$ is implemented as a LUT using on-chip processor memory and SRAM memory modules;

c) $\sin(\phi(j))$ is implemented as a LUT using on-chip processor memory and SDRAM memory modules.

In all simulations, we compute $\gamma(\sin(\phi(j)))$ for each value of its address size. Figures 4.9 and 4.10 show the 3 situations described above. These figures show that the γ metric of C67 is better than the one corresponding to the C44 in the majority of situations. This is a result of the technology push which allows the design of very fast processors with a small increase in cost. The C67 cost is twice the cost of a C44, but its bus clock frequency is 5.5 times faster. Moreover, simulations show the advantage of coupling high-speed processors with SDRAM over conventional SRAM [21]. The high density and fast access time make SDRAM memory modules very attractive in system design. However,

large LUTs may require so much memory (functions with several input parameters), that its corresponding software implementation is more cost effective (see Figure 4.9). Thus, in practice, the proposed algorithm can be used with several types of memory modules and processors combinations to identify the best trade-off between cost and performance.

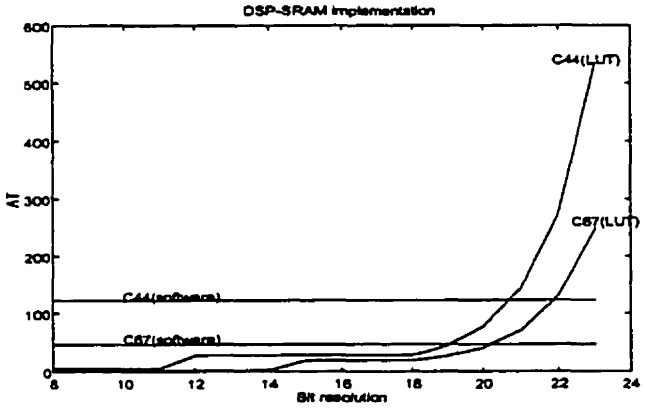

Figure 4.9: γ metric for SRAM memory.

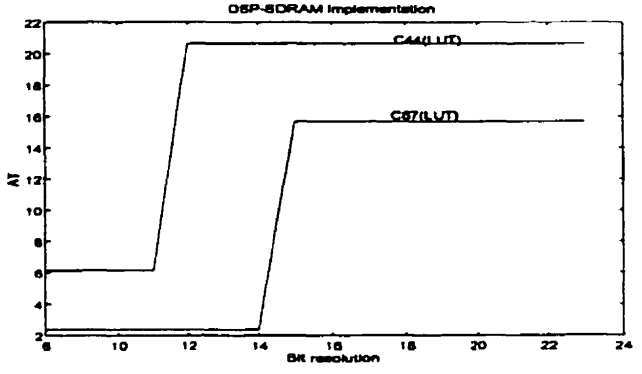

Figure 4.10: γ metric for SDRAM memory.

4.11 Conclusion

In this paper, we have presented a new system-level partitioning algorithm for applications based on functions that can be implemented either in software or in hardware as lookup tables. This algorithm can be classified in the category of greedy algorithms and does not use backtracks. Two cases were analyzed: fixed-size and variable-size lookup tables. The complexity of the algorithm can be bounded by $O(n^2)$, assuming a linear data clustering algorithm, which leads to a reasonable computation time.

Our partitioning algorithm targets applications that can make use of a three-level

memory hierarchy and can be easily extended to systems with any number of memory levels. Our partitioning algorithm can be easily modified to target HPRC [22] (high-performance reconfigurable computing) systems in which each library function is identified by a software implementation on a DSP and a hardware implementation as a core on a FPGA, which could be viewed as another kind of LUT based accelerators. Moreover, if the targeted HPRC system supports RTR [23] (run time reconfiguration), then the performance of the hardware-software partitioning can be improved by moving to hardware, at an appropriate time, the next software function. Finally, the proposed partitioning algorithm could be added to a high-level design tool to widen the scope of targeted applications.

4.12 References

[1] G. De Micheli and R. K. Gupta, "System Synthesis via Hardware-Software Co-Design", Technical Report No. CSL-TR-92-548, Oct. 92.

[2] F. Vahid, "A Survey of Behavioral-Level Partitioning Systems", Technical Report #91-71, University of California, Irvine, Oct. 91.

[3] R. Gupta and G. De Micheli, " System-Level Synthesis using Re-Programmable Components", Proceedings of EDAC, pp. 2-7, 92.

[4] R. Ernst and J. Henkel, " Hardware-Software Codesign of Embedded Controllers based on Hardware Extraction", Proceedings of International Workshop on Hardware-

Software Codesign, 92.

[5] Z. Peng and K. Kuchcinski, " An Algorithm for Partitioning of Application Specific Systems", Proceedings of EDAC, pp. 316-321, 93.

[6] R. Ernst, J. Henkel, and Th. Benner, " Hardware-Software Cosynthesis for Micro-Controllers", IEEE Design&Test of Computers, Vol. 10, No. 4, pp. 64-75, Dec. 93.

[7] R. K. Gupta and G. De Micheli, " Hardware-Software Cosynthesis for Digital Systems", Vol. 10, No. 3, pp. 29-41, Sept. 93.

[8] E. Dirkes Laganese and D. E. Thomas, " Architectural Partitioning of System Level Synthesis of Integrated Circuits", IEEE Transactions on CAD/ICAS, Vol. 10, No. 7, pp. 847-860, July 91.

[9] T. Ben Ismail, K. O'Brien, and A. Jerraya, " Interactive System-Level Partitioning With PARTIF", Proceedings of EDAC, IEEE Computer Society Press, 94.

[10] D. D. Gajski and F. Vahid, " Specification and Design of Embedded Hardware-Software Systems", IEEE Design&Test of Computers, pp. 53-67, Spring 95.

[11] K. H. Kernighan and S. Lin, " An Efficient Heuristic Procedure for Partitioning Graph", Bell System Technical Journal, Vol. 49, No. 2, pp. 291-307, Feb. 70.

[12] ***Near-Optimal Bin-Packing Algorithm***, D. S. Johnson, PhD Thesis, MIT, 1974.

[13] K. L. Huang and H. Chen, " Color Reproduction Apparatus with Scalable Variable-Point", Proceedings of IS&T's 11th International Congress on Advances in Non-Impact Printing Technologies, pp. 463-466, 95.

[14] B. A. Thomas, " Need for Constraints in Component-Separable Color Image Processing", Proceedings of SPIE, Vol. 2421, pp. 35-42, 95.

[15] R. Richter, " Spatially Adaptive Fast Atmospheric Correction Algorithm", International Journal of Remote Sensing, Vol. 17, No. 6, pp. 1201-1214, Apr. 96.

[16] D. W. Jacobs, " Space Requirements of Indexing under Perspective Projections", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 18, No. 3, pp. 330-333, 96.

[17] J-. A. Beraldin, S. F. El-Hakim, and L. Cournoyer, " Practical Range Camera Calibration", Proceedings of SPIE, Vol. 2067 Videometrics II, pp. 21-31, 93.

[18] J. Child, " Memory Modules Get Faster, Wider, Cheaper", Computer Design, pp. 115-125, May 96.

[19] ***TMS320C4x, User's Guide,*** Texas Instruments, 1991.

[20] ***ADSP-21060/62 SHARC,*** Preliminary Data Sheet, Analog Devices, Nov. 94.

[21] K. Castille, "TMS320C6x EMIF to External SDRAM/SGRAM Interface", Application report: SPRA433, Texas Instrument, April 1998.

[22] Y. Savaria, G. Bois, P. Popovic, and A. Waye, "Computational Acceleration Methodologies: Advantages of Reconfigurable Acceleration Subsystems", SPIE, Vol. 2914, pp. 195-205, 1996.

[23] M. J. Wirthlin and B. L. Hutchings, "Sequencing Run-Time Reconfigured Hardware with Software", ACM/SIGDA Int. Symp. Field Programmable Gate Arrays, Monterey, CA, Feb 1996, pp. 122-128.

Chapitre 5 : Accélération de la correction d'erreur associée à une variation de réflectance

Comme cela a été mentionné au début du chapitre 4, le calcul des coordonnées (x, y, z) associées à une mesure est basé sur la connaissance des paramètres (p, i, j) qui sont respectivement la position du spot réfléchi sur le détecteur optique, la position du miroir associée à la dimension x et la position du miroir associée à la position y. Nous avons montré que les LUT constituaient un moyen simple et efficace pour accélérer le calcul des coordonnées (x, y, z). Cependant, nous avons vu au chapitre 2 qu'une variation de réflectance introduit une erreur sur la position p, ce qui conduit à des coordonnées (x, y, z) erronées. Cette erreur doit alors être corrigée avec les méthodes de correction proposées et cela avant d'appliquer les transformations géométriques conduisant au calcul des coordonnées (x, y, z). Dans des systèmes optiques à faibles débits, on pourrait se contenter d'une implantation programmée des méthodes de correction. Cependant, cette approche deviendrait très vite problématique et limitative pour des systèmes optiques à débits élevés. Il devient alors pertinent d'analyser la complexité de calcul des méthodes de correction et de proposer des modèles d'accélération afin d'accélérer dans sa globalité, le calcul des coordonnées (x, y, z).

5.1 Analyse de la complexité de calcul des méthodes de correction

Nous avons présenté au chapitre 3 deux méthodes de correction de l'erreur associée à une variation de réflectance, à savoir la méthode directe et la méthode itérative. Dans le

cas de la méthode directe, la correction apportée est donnée par:

- un gradient de réflectance: $G = g - \bar{x} + a \cdot f(a)$;

- un saut de réflectance: $G = g - \bar{x} + \delta \cdot (d - |j|) \cdot f(j, \delta)$.

On constate que les équations de correction présentent une complexité de calcul faible et peuvent être aisément supportées par un processeur DSP moderne. Quant à la division du spot laser reçu par un spot calibré pour la reconstruction de la réflectance, cette opération serait encore plus efficace sur un processeur DSP ou RISC, en raison de la disponibilité de la division de façon câblée et ne nécessitant qu'une dizaine de cycles CPU (Wenheng, Prasanna, 1998). L'implantation de la même opération sur du matériel dédié (ASIC ou FPGA) conduirait à une performance inférieure et à un coût matériel plus élevé. Quant à la méthode itérative, celle-ci présente une complexité de calcul élevée engendrée par les facteurs suivants:

a) la boucle sur les différents facteurs de magnification à tester;

b) pour chaque facteur de magnification, la boucle sur les différentes positions du spot de référence à tester;

c) pour chaque position du spot de référence, la boucle sur le nombre de pixels du spot;

d) pour chaque pixel, le nombre d'échantillons à accumuler pour calculer l'intensité du pixel correspondant.

Afin d'avoir un ordre de grandeur de cette complexité, supposons que nous disposons des paramètres suivants:

- largeur d'un pixel $W = 50\mu m$,

- résolution du système $R = 0.78\mu m$,

- taille moyenne d'un spot: 10 pixels,

- nombre moyen d'échantillons à accumuler par pixel: 64.

Le nombre total d'opérations à effectuer pour l'algorithme associé à la méthode itérative (voir section 3) est alors donné par:

$$N_{oper} = \frac{50}{0.78} \cdot \frac{1}{2} \cdot \frac{50}{0.78} \cdot 10 \cdot ((64+1) + (10-1) + (10-2) + (10-2) + 1)$$

$$N_{oper} \approx 1389000$$

On constate que la valeur de N_{oper} peut devenir problématique pour des systèmes optiques à débits élevés. En effet, supposons que nous disposons d'un processeur dont la fréquence d'horloge est de 100 MHz. Si chaque opération s'exécute en un cycle CPU, le processeur ne pourra calculer que 72 corrections par seconde. Afin d'accélérer la vitesse de correction (augmentant ainsi le débit), il est nécessaire d'identifier le parallélisme disponible dans l'algorithme de correction, ce qui permettra l'élaboration d'une architecture matérielle dédiée plus appropriée. Comme la complexité de calcul de la méthode itérative est de loin plus grande que celle de la méthode directe, on se propose d'analyser à la section suivante différentes approches pour améliorer la vitesse de correction de l'erreur associée à une variation de réflectance.

5.2 Accélération de la méthode itérative

Tel que mentionné à la section précédente, la complexité de calcul de la méthode itérative est principalement fonction du nombre de facteurs de magnification, du nombre de positions à analyser sur le pixel central, de la taille du spot reçu et du nombre d'échan-

tillons à accumuler pour calculer l'intensité d'un pixel donné. Cependant deux aspects importants, en plus de ceux énumérés précédemment, doivent être pris en compte pour une implantation efficace de la méthode itérative:

a) toutes les itérations des boucles sur les facteurs de magnification et des positions du spot de référence sur le pixel central sont indépendantes les unes des autres,

b) les convertisseurs analogique-numérique fonctionnent sur un nombre de bits fini (en général <= 16 bits), ce qui laisserait le choix de calculer ou de tabuler l'énergie associée à un pixel de référence.

La caractéristique a) indique que plus on dispose de processeurs, plus on pourra leurs assigner des itérations indépendantes conduisant ainsi à un degré de parallélisme élevé. L'exploitation de ce parallélisme permettra d'augmenter le facteur d'accélération global. Quant à la caractéristique b), celle-ci ajoute un second niveau d'optimisation, en plus de a), puisqu'elle permet de remplacer le calcul des énergies des pixels de référence par de simples accès mémoires à des LUT. Comme nous l'avons montré dans ce chapitre, plus le nombre d'instructions logicielles associées à un calcul est grand, plus le gain associé à l'utilisation d'une LUT est grand. Ces deux caractéristiques a) et b) de la méthode itérative nous amènent à considérer deux approches pour accélérer le traitement:

- approche 1: on calcule des itérations en parallèle et on calcule les énergies des pixels,
- approche 2: on calcule des itérations en parallèle et on tabule les énergies des pixels en utilisant des LUT.

Avant de développer les architectures associées aux deux approches, on redonne l'algorithme de correction de la méthode itérative qui nous servira de base de travail. Cet algo-

rithme sera présenté sous forme d'actions à accomplir. Pour les détails, le lecteur peut se référer à la version originale de l'algorithme présentée au chapitre 3.

Algorithme: méthode itérative

Début

Boucle 1: Pour tous les facteurs de magnification

Boucle 2: Pour toutes les positions du spot de référence

1- calculer les énergies des pixels du spot de référence courant.

2- calculer la réflectance associée à chaque pixel.

3- calculer les différences finies d'ordre 1 du signal de réflectance.

4- calculer les différences finies d'ordre 2 du signal de réflectance.

5- calculer le maximum des différences finies d'ordre 2.

Fin Boucle 2.

Fin Boucle 1.

Calculer le minimum de tous les maximums calculés.

Fin

5.3 Approche 1: calcul des itérations en parallèle et calcul des énergies de pixels

L'algorithme de correction associé à la méthode itérative est constitué de deux boucles principales identifiées par Boucle 1 et Boucle 2. A l'intérieur de ces deux boucles, le traitement suivant est effectué:

1- calculer les énergies des pixels du spot de référence courant.

2- calculer la réflectance associée à chaque pixel.

3- calculer les différences finies d'ordre 1 du signal de réflectance.

4- calculer les différences finies d'ordre 2 du signal de réflectance.

5- calculer le maximum des différences finies d'ordre 2.

Ces cinq traitements peuvent être encapsulés dans une même unité fonctionnelle qui réalise les fonctions de base d'un processeur dédié. La figure 5.1 illustre une architecture possible du processeur élémentaire correspondant.

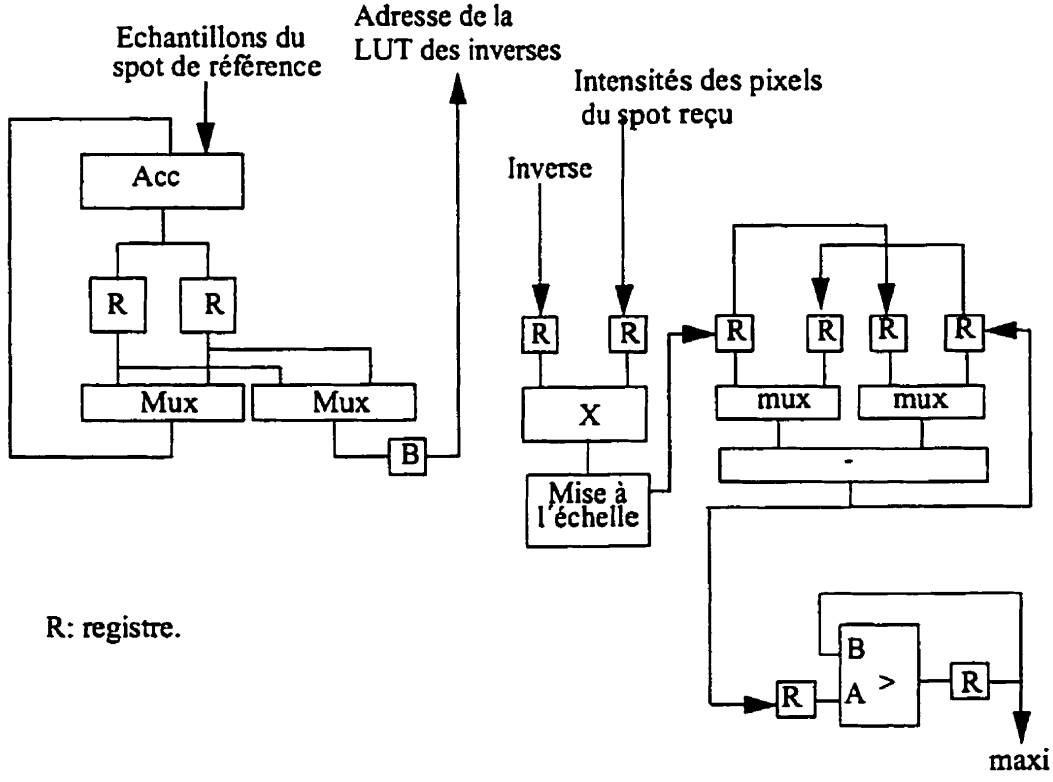

Figure 5.1: Processeur élémentaire PE1 correspondant à l'approche 1.

En raison du calcul de l'énergie associée à chaque pixel de référence, qui est en fait un processus d'accumulations répétitives, les unités opératives calculant la réflectance

(multiplieur), les différences finies (soustracteur) et le maximum (comparateur) sont inoccupés pendant une certaine partie du traitement. Cette perte de performance peut être compensée en augmentant le nombre de processeurs élémentaires calculant en parallèle en raison de l'indépendance des itérations associées aux boucles 1 et 2. La figure 5.2 illustre une architecture possible utilisant plusieurs processeurs élémentaires.

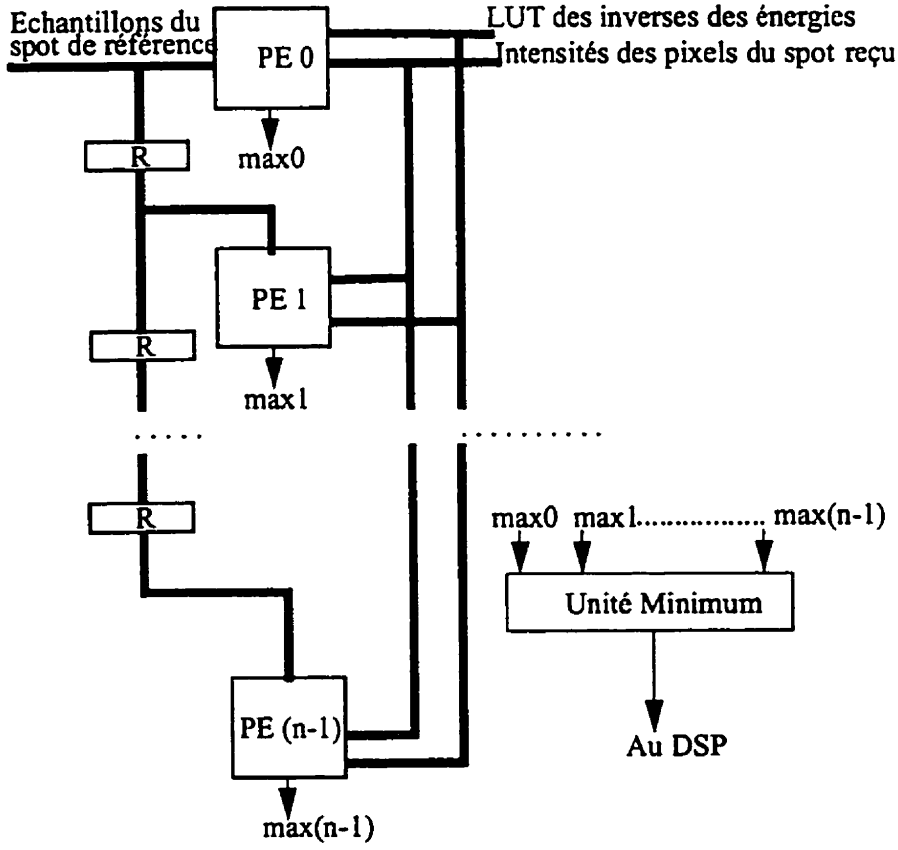

Figure 5.2: Architecture multiprocesseurs pour un seul facteur de magnification.

A la figure 5.2, chaque processeur élémentaire calcule une itération de la boucle 2, ce qui

revient à analyser en parallèle plusieurs positions du spot de référence pour un même facteur de magnification. L'inverse des énergies calculées est obtenu par LUT en raison du coût élevé d'implanter une telle opération en matériel. Dans le cas où l'on désire calculer plusieurs itérations de la boucle 2 associées à plusieurs itérations de la boucle 1, l'architecture présentée à la figure 5.2 peut être dupliquée autant de fois que nécessaire. La figure 5.3 illustre une possible architecture pour le calcul en parallèle de plusieurs facteurs de magnification. Telqu'illustré à la figure 5.3, chaque colonne de processeurs traite un facteur de magnification donné. Afin d'éviter les goulots d'étranglement associés au calcul des inverses, les LUT correspondantes ont été dupliquées et distribuées sur les différentes colonnes de processeurs. Cette architecture offre un parallélisme massif permettant ainsi une accélération substantielle (par rapport à une implantation logicielle) qui va essentiellement dépendre du nombre de processeurs que l'on peut intégrer en matériel. Dans le cas de l'utilisation des FPGAs de la famille VIRTEX de Xilinx (Xilinx, 1999), le concepteur dispose d'un nombre impressionnant de ressources matérielles pouvant contenir un nombre avoisinant les 100 processeurs élémentaires. Cependant, cette architecture présente un contrôle complexe et nécessite un FPGA avec un nombre de broches très élevé afin de permettre l'accès en parallèle à plusieurs LUT.

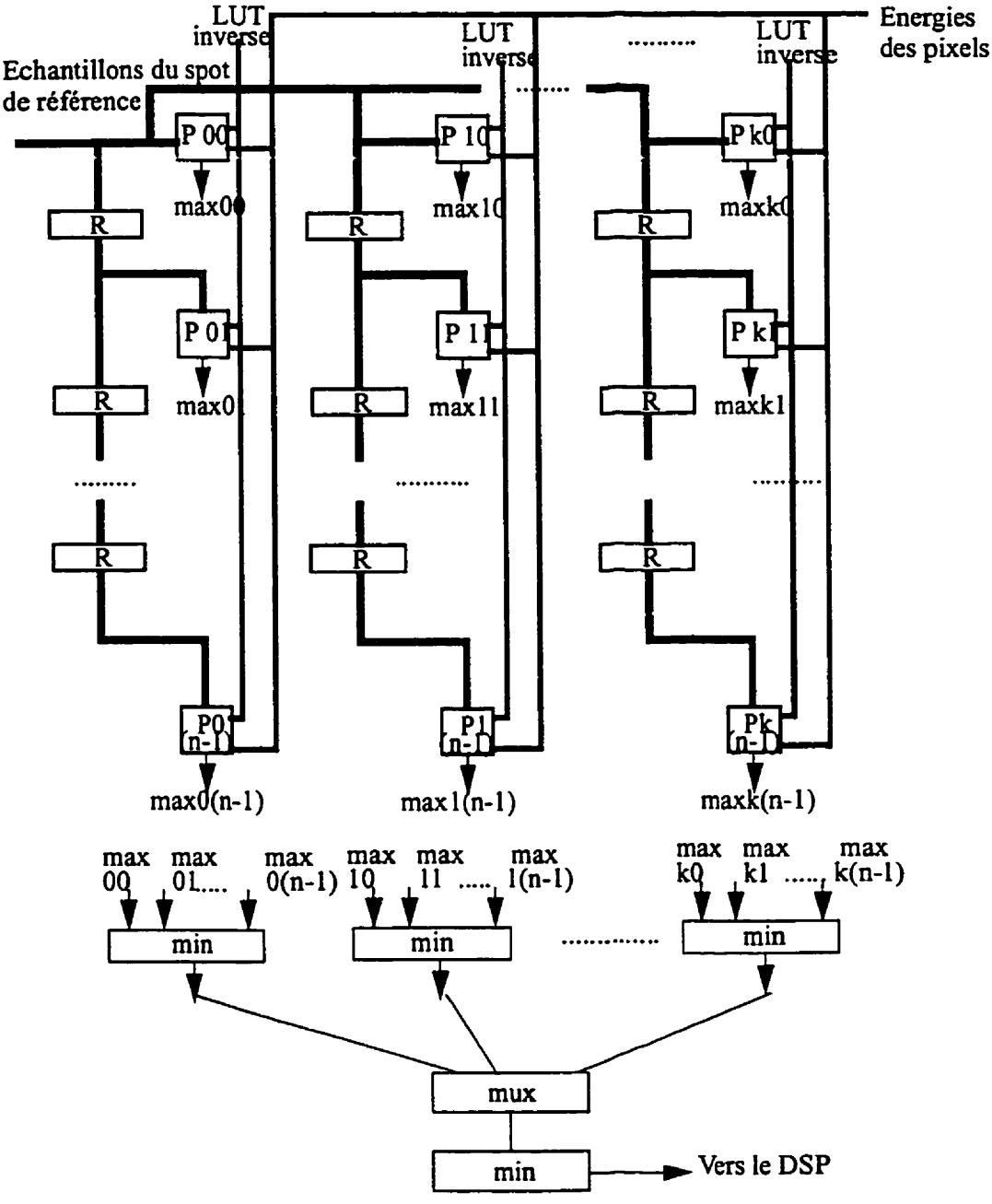

Figure 5.3: Architecture multiprocesseurs pour plusieurs facteurs de magnification.

5.4 Approche 2: calcul des itérations en parallèle et tabulation des énergies des pixels des spots de référence

L'un des inconvénients associés à l'approche 1 est le temps d'inactivité de certaines unités opératives pendant le calcul des énergies des pixels des spots de référence. Cette inactivité entraîne une perte de performance malgré l'utilisation de plusieurs processeurs élémentaires fonctionnant en parallèle. La situation idéale serait qu'à chaque cycle d'horloge, une donnée utile soit produite ou consommée. Cette situation idéale pourrait être atteinte ou approchée en modifiant l'approche de calcul. Au lieu de calculer les énergies associées aux pixels, il est possible de tabuler ces énergies, donc de les pré-calculer, et de les stocker dans une LUT. Cette approche est rendue possible en raison du caractère prédictif de la méthode itérative. En effet, la séquence des facteurs de magnification à tester et les positions des spots de référence à "visiter" sont connues à l'avance, ce qui rend possible le stockage dans une LUT de tous les calculs intermédiaires. En fait, ce sont les inverses de ces énergies qu'il faudrait pré-calculer pour éviter d'effectuer le calcul de l'inverse par matériel. Cependant, une telle approche de calcul va nécessiter une quantité de mémoire qui est fonction du nombre de facteurs de magnification, du nombre de positions des spots de référence, des tailles des spots de référence et de la taille des résultats de la table. Soient les variables suivantes:

- N_{mag}: le nombre de facteurs de magnification à considérer,
- N_{pos}: le nombre de positions du spot de référence à analyser,
- S_{min}: la taille minimale en nombre de pixels du spot de référence,
- S_{max}: la taille maximale en nombre de pixels du spot de référence,

- n: le nombre de spots de référence à considérer,

- k: la taille en octets d'un résultat de la table LUT.

La quantité de mémoire Q_{mem} nécessaire pour stocker l'ensemble des résultats intermédiaires dans une LUT est donnée par:

$$Q_{mem} = \sum_{i=S_{min}}^{S_{max}} N_{mag} \cdot N_{pos} \cdot k \cdot i = N_{mag} \cdot N_{pos} \cdot k \cdot \sum_{i=S_{min}}^{S_{max}} i \qquad (5.1)$$

L'équation (5.1) peut être réécrite de la manière suivante:

$$Q_{mem} = N_{mag} \cdot N_{pos} \cdot k \cdot \left(n \cdot S_{min} + \frac{n \cdot (n-1)}{2} \right) \qquad (5.2)$$

Afin d'avoir une idée de l'ordre de grandeur de Q_{mem}, appliquons (5.2) à un système optique basé sur la caméra autosynchronisée, où la largeur de pixel $W = 50\mu m$, et la résolution du système $R = 0.78\mu m$. En supposant que $S_{min} = 5$, $n = 50$ et $k = 1$, nous avons:

- $N_{mag} = \frac{W}{R} = 64.10$,

- $N_{pos} = \frac{W}{2R} = 32.05$ (pour le calcul de N_{mag} et N_{pos}, voir l'article sur la méthode itérative). La quantité de mémoire Q_{mem} requise par le traitement est alors égale à:

$Q_{mem} = 64.10 \cdot 32.05 \cdot 1475 \approx 3030247 \approx 3Mo$. Dans le cas où on dispose d'un détecteur optique avec une largeur de pixel plus grande, par exemple $W = 70\mu m$, et une résolution plus fine, par exemple $R = 0.5\mu m$, La quantité de mémoire Q_{mem} requise par le traitement est alors égale à:

$Q_{mem} = 140 \cdot 70 \cdot 1475 = 14455000 \approx 15Mo$. Les figures 5.4 et 5.5 illustrent la variation de Q_{mem} en fonction du rapport $\frac{W}{R}$ respectivement pour $k = 1$ et $k = 2$. Dans les deux cas de figures, $S_{min} = 5$ et $n = 50$.

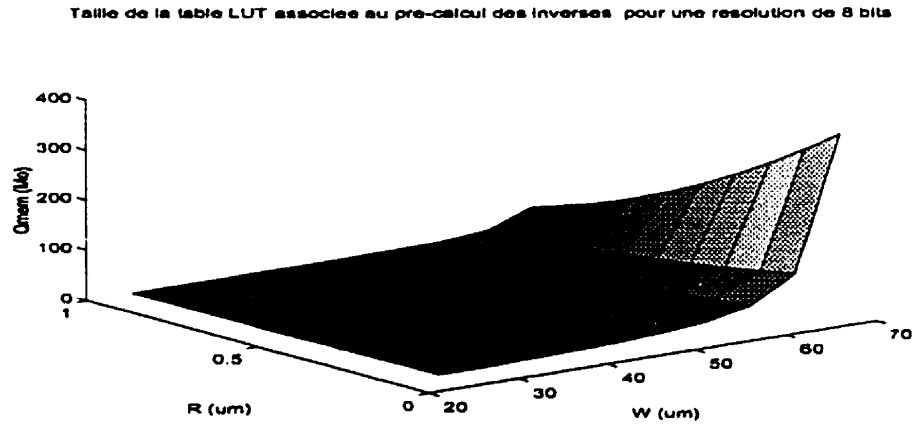

Figure 5.4: Quantité de mémoire requise Q_{mem} pour $k = 1$.

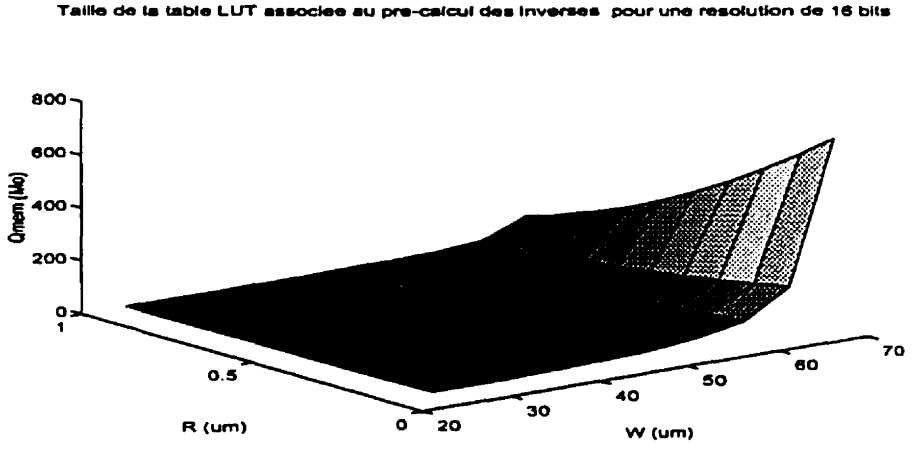

Figure 5.5: Quantité de mémoire requise Q_{mem} pour $k = 2$.

La croissance quadratique de Q_{mem} en fonction du rapport $\frac{W}{R}$ peut devenir problématique pour des applications nécessitant des systèmes optiques à résolutions fines. Cependant, cela va dépendre des contraintes coût/performance requises pour résoudre le problème de correction de l'erreur associée à une variation de réflectance. Quant au processeur élémentaire, celui-ci possède une architecture plus simple (comparée à celle de l'approche 1) en raison du remplacement de l'unité d'intégration par un simple accès

mémoire. La nouvelle architecture du processeur élémentaire est présentée à la figure 5.6.

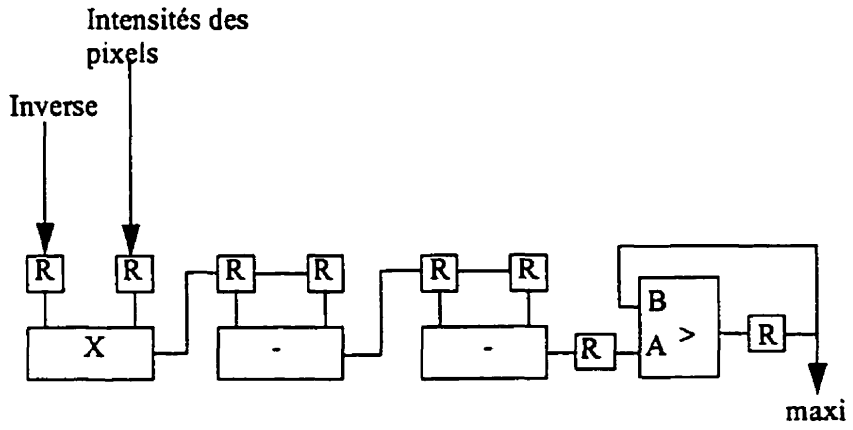

Figure 5.6: Processeur élémentaire PE2 correspondant à l'approche 2.

Tel qu'illustré à la figure 5.6, le nouveau processeur élémentaire est un pipeline capable de délivrer un résultat utile par cycle d'horloge. En supposant que la mémoire est capable de délivrer une donnée utile par cycle d'horloge, toutes les unités opératives sont efficacement utilisées, ce qui permet d'augmenter la performance nominale du processeur élémentaire. De la même façon que dans l'approche 1, l'approche 2 peut bénéficier des deux sous-variantes de calcul, à savoir une architecture uni-colonne où plusieurs processeurs élémentaires calculent pour le même facteur de magnification, ou bien une architecture multi-colonnes où plusieurs processeurs élémentaires calculent pour plusieurs facteurs de magnification. Cependant, quelque soit l'approche de calcul utilisée et la sous-variante associée, une même architecture de système peut être élaborée. Cette

architecture de système est basée sur un concept matériel-logiciel où un processeur principal (DSP, RISC,etc) fournit à un coprocesseur matériel (ASIC, FPGA) les pixels associés au spot laser courant. Le coprocesseur implémente l'algorithme de correction et renvoie les résultats au processeur. La figure 5.7 illustre le modèle d'architecture de système proposé.

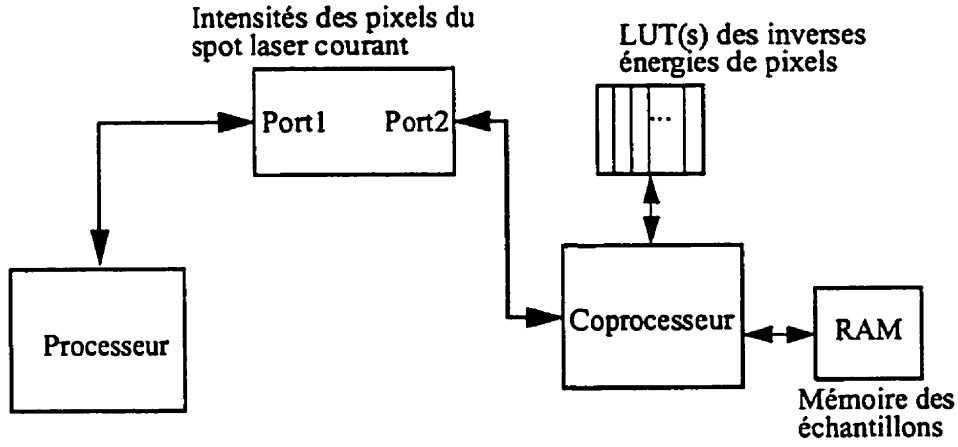

Figure 5.7: Architecture de système pour les approches de calcul 1 & 2.

Tel qu'illustré à la figure 5.7, le processeur communique avec son coprocesseur grâce à une mémoire à double ports ou une mémoire de type FIFO bidirectionnel. Les données échangées sont les intensités des pixels du spot laser courant, des paramètres de contrôle et les résultats de calcul. Quant au coprocesseur, celui-ci lit les données nécessaires aux calculs et effectue son traitement. Dépendemment de l'approche de calcul retenue, certaines mémoires sont optionnelles. En effet, si l'approche 1 est choisie, alors le coprocesseur a besoin de la LUT des inverses des énergies des pixels et des échantillons du spot

de référence. Par contre, si l'approche 2 est choisie, alors le coprocesseur n'a besoin que de la mémoire des LUTs des inverses des énergies des pixels pour tous les facteurs de magnification et pour toutes les positions du spot de référence. Une caractéristique importante de l'architecture système proposée est la faculté que possèdent le processeur et son coprocesseur de communiquer simultanément ce qui permet de traiter un nouveau spot laser reçu pendant que l'on reçoit les résultats associés au spot laser précédent. Cela permet de réduire l'impact des délais des communications sur la performance globale du système.

5.5 Coûts et performances des approches de calcul proposées

Comme nous l'avons mentionné à la section précédente, la performance du système va essentiellement dépendre de l'approche de calcul retenue et du nombre de processeurs élémentaires disponibles. Cependant cette performance a un coût qui est fonction des coûts associés au processeur, à la mémoire de communication, au coprocesseur et aux mémoires servant à implémenter les LUT. Afin d'estimer les paramètres de coûts et de performance, nous allons d'abord définir les principales variables à utiliser. Soient les variables suivantes:

- N_{mag}: nombre de facteurs de magnification,
- α: nombre moyen d'échantillons à accumuler pour calculer l'énergie d'un pixel,
- T: taille moyenne (en pixels) d'un spot,
- a: nombre de cycles CPU par accès mémoire,

- τ : temps d'un cycle CPU,

- C_{proc} : coût du processeur,

- C_{coproc} : coût du coprocesseur,

- C_{com} : coût de la mémoire de communication,

- C_{mem} : coût par unité mémoire,

- N_{echan} : nombre d'échantillons associés à un spot laser,

- b_{pix} : nombre de bits pour représenter l'énergie d'un pixel,

- b_{inv} : nombre de bits pour représenter l'inverse de l'énergie d'un pixel de référence,

- b_{echan} : nombre de bits pour représenter un échantillons du spot laser.

Les coûts C_{appl} et C_{app2} associés aux approches de calcul 1 et 2 sont respectivement donnés par:

$$C_{appl} = C_{proc} + C_{com} + C_{coproc} + \left(N_{echan} \cdot b_{echan} + 2^{b_{pix}} \cdot b_{inv} \right) \cdot C_{mem} \qquad (5.3)$$

$$C_{app2} = C_{proc} + C_{com} + C_{coproc} + Q_{mem} \cdot C_{mem} \qquad (5.4)$$

où Q_{mem} est la quantité de mémoire requise pour stocker tous les résultats d'intégrales (voir section 5.4). Si on suppose que les deux approches de calcul sont basées sur des processeurs, coprocesseurs et mémoires de communication identiques, la différence de coût est directement reliée à la quantité de mémoire requise pour les calculs. Afin de mieux quantifier cette différence de coût, nous allons calculer C_{appl} et C_{app2} en nous basant sur des coûts unitaires couramment rencontrés sur le marché des circuits intégrés. Nous avons:

- C_{proc} = 200$: DSP TMS320C44,

- C_{com} = 80$: FIFO bidirectionnel,

- C_{coproc} = 400$: FPGA XC4036-2,

- C_{mem} = 8$: Mémoire SRAM 64Kx16.

En se basant sur ces coûts, deux configurations seront analysées:

Configuration 1:

- $N_{echan} = 1024, b_{pix} = b_{inv} = b_{echan} = 16$,

-$Q_{mem}(W = 50\mu m, R = 0.7\mu m) = 7.4$Mo.

Les coûts C_{app1} et C_{app2} sont alors respectivement égaux à:

$C_{app1} = 200 + 80 + 400 + 2 \cdot 8 = 696$\$.

$C_{app2} = 200 + 80 + 400 + 60 \cdot 8 = 1160$\$.

$\frac{W}{R} = 71.4, \frac{C_{app2}}{C_{app1}} = 1.67$

Configuration 2:

- $N_{echan} = 1024, b_{pix} = b_{inv} = b_{echan} = 16$,

-$Q_{mem}(W = 70\mu m, R = 0.1\mu m) = 713$Mo.

Les coûts C_{app1} et C_{app2} sont alors respectivement égaux à:

$C_{app1} = 200 + 80 + 400 + 2 \cdot 8 = 696$\$.

$C_{app2} = 200 + 80 + 400 + 5704 \cdot 8 = 46312$\$.

$\frac{W}{R} = 700, \frac{C_{app2}}{C_{app1}} = 66.54$.

En comparant les coûts monétaires des deux approches de calcul pour les configurations 1 et 2, on constate que le coût de la mémoire requise pour l'approche 2 augmente rapidemment avec le rapport $\left(\frac{W}{R}\right)$. En effet, dans le cas de la configuration 1, le rapport $\frac{C_{app2}}{C_{app1}} = 1.67$ pour un rapport $\frac{W}{R} = 71.4$. Tandis que dans le cas de la configuration 2, le rapport $\frac{C_{app2}}{C_{app1}} = 66.54$ pour un rapport $\frac{W}{R} = 700$. Une augmentation du rapport $\frac{W}{R}$ par un facteur de 10 a provoqué une augmentation du rapport $\frac{C_{app2}}{C_{app1}}$ par un facteur de 40. Dans le cas de la configuration 2, la quantité de mémoire requise est tellement grande que le choix d'un autre type de mémoire s'impose. En effet, les progrès considérables effectués dans le domaine de la technologie VLSI ont permis d'augmenter de façon considérable la densité des mémoires dynamiques, d'améliorer leur vitesse et d'en réduire les coûts. Il est courant de disposer de boîtiers de mémoire dynamique synchrone avec une densité de 64Mbits et un coût de l'ordre de 20$. En se basant sur ce type de mémoire, une troisième configuration peut être analysée:

Configuration 3:

- $N_{echan} = 1024, b_{pix} = b_{inv} = b_{echan} = 16,$

- $Q_{mem}(W = 70\mu m, R = 0.1\mu m) = 713\text{Mo}.$

Les coûts C_{app1} et C_{app2} sont alors respectivement égaux à:

$C_{app1} = 200 + 80 + 400 + 1 \cdot 20 = 700\$.$

$C_{app2} = 200 + 80 + 400 + 90 \cdot 20 = 2480\$.$

$\frac{W}{R} = 700, \frac{C_{app2}}{C_{app1}} = 3.54.$

Il est clair que le coût supplémentaire associé à une approche par LUT basée sur de la

SDRAM est de loin plus acceptable que celui associé à la configuration 2. Cela montre l'importance du choix du type de mémoire pour l'application visé.

Néanmoins à ce coût est associée une performance. Cette performance est déterminée par les temps de traitement respectifs des deux approches de calcul. Soient $T1_{soft}$ et $T2_{soft}$, respectivement les temps de traitement en logiciel associés aux approches de calcul 1 et 2. En se basant sur l'algorithme de la méthode itérative, ces temps de traitement peuvent être estimés de la manière suivante:

$$T1_{soft} = N_{mag} \cdot \frac{N_{mag}}{2} \cdot (T(\alpha+2) + (T-1) + (T-2) + (T-2))\tau$$

$$T1_{soft} = \frac{N_{mag}^2}{2} \cdot (T(\alpha+5) - 5)\tau$$

$$T2_{soft} = N_{mag} \cdot \frac{N_{mag}}{2} \cdot (T(a+2) + (T-1) + (T-2) + (T-2))\tau$$

$$T2_{soft} = \frac{N_{mag}^2}{2} \cdot (T(a+5) - 5)\tau$$

Le gain logiciel en vitesse G_{soft} est alors donné par:

$$G_{soft} = \frac{T1_{soft}}{T2_{soft}} = \frac{\frac{N_{mag}^2}{2} \cdot (T(\alpha+5) - 5)\tau}{\frac{N_{mag}^2}{2} \cdot (T(a+5) - 5)\tau} = \frac{T(\alpha+5) - 5}{T(a+5) - 5}$$

Si on néglige la constante (-5) au numérateur et au dénominateur, le gain G_{soft} devient:

$G_{soft} \approx \frac{\alpha+5}{a+5}$.

L'expression de G_{soft} montre que le gain de vitesse en logiciel va principalement dépen-

dre de α, le nombre d'échantillons du spot laser à accumuler pour calculer l'énergie d'un pixel, et de a qui représente le temps d'accès à la LUT qui elle même contient toutes les intégrales précalculées. Le gain G_{soft} peut être estimé d'une autre manière en utilisant l'équation (4.2) pour la prédiction du gain en vitesse basé sur des LUT. En effet, les paramètres de (4.2) sont donnés par:

$$- f = \frac{T \cdot \alpha}{T(\alpha+2) + (T-1) + (T-2) + (T-2)} = \frac{T \cdot \alpha}{T(\alpha+5) - 5} \approx \frac{\alpha}{\alpha+5},$$

$$- N = \alpha,\ k = a,\ CPI = 1,\ min\left(\frac{\beta_1}{\beta_2}, \Omega\right) = 1.$$

Le gain en vitesse G_s de (4.2) est alors égal à:

$$G_s = \frac{1}{1 - \left(\frac{\alpha}{\alpha+5}\right) + \frac{\left(\frac{\alpha}{\alpha+5}\right) \cdot a}{\alpha \cdot 1 \cdot 1}} = \frac{1}{\frac{5}{\alpha+5} + \frac{a}{\alpha+5}} = \frac{\alpha+5}{a+5}.$$

On constate que le gain G_{soft} est égal au gain G_s.

Dans un contexte matériel-logiciel, la performance du système va dépendre du nombre de processeurs élémentaires disponibles sur le coprocesseur. Dans le cas où le coprocesseur est un circuit FPGA, Ce nombre de processeurs élémentaires va essentiellement dépendre de la complexité du FPGA et des complexités respectives des processeurs élémentaires. La complexité d'un FPGA est en général identifiée par le nombre de blocs logiques (CLB: configurable logic bloc) disponibles (Villasenor, Hutchings, 1998). La structure d'un bloc logique varie d'un FPGA à un autre et d'un fabriquant à un autre. Soient N_{CLB}, C_{PE1}, C_{PE2} respectivement le nombre de CLB disponibles sur le FPGA et les complexités respectives des processeurs élémentaires PE1 et PE2. Les nombres de

processeurs élémentaires N_{PE1}, N_{PE2} que l'on intègre sur le FPGA sont donnés par:

$N_{PE1} = \frac{N_{CLB}}{C_{PE1}}$, $N_{PE2} = \frac{N_{CLB}}{C_{PE2}}$. (Le nombre réellement disponible de processeurs élémentaires sera en réalité plus petit en raison de la complexité associée au contrôle. Cependant, pour des fins de discussion et d'analyse, l'aspect contrôle ne sera pas pris en compte). En se basant sur les architectures respectives des processeurs élémentaires, on constate que chacun peut effectuer, durant chaque cycle d'horloge, plusieurs opérations équivalentes N_{eq} du processeur principal. Ces opérations équivalentes sont la multiplication, l'addition, la soustraction, la comparaison, les accès mémoire et les transferts registres à registres. Pour les architectures proposées, N_{eq} peut être approximé à 10. Les temps de traitement $T1_{FPGA}$ et $T2_{FPGA}$ associés respectivement aux approches de calcul 1 et 2 sont donnés par:

$T1_{FPGA} = \frac{T1_{soft}}{(N_{PE1} \cdot N_{eq})}$, $T2_{FPGA} = \frac{T2_{soft}}{(N_{PE2} \cdot N_{eq})}$ (ces temps de traitement ne tiennent pas compte des délais de communication. Cependant, en raison d'une utilisation d'une mémoire à double ports, des transferts simultanés peuvent avoir lieu ce qui réduit considérablement l'impact des délais de communication). Les expressions analytiques des coûts, C_{app1} et C_{app2}, et des temps de traitement, $T1_{FPGA}$ et $T2_{FPGA}$, montrent que nous sommes en présence de deux architectures, l'une limitant les coûts (approche 1), et l'autre maximisant la performance (approche 2). La combinaison de ces deux paramètres permettra de choisir l'approche de calcul qui satisfait le mieux des contraintes de coût-performance. Soient AT_1 et AT_2 les métriques de coût-performance associés res-

pectivement aux approches de calcul 1 et 2. Ces métriques sont données par:

$AT_1 = T1_{FPGA} \cdot C_{app1}$, $AT_2 = T2_{FPGA} \cdot C_{app2}$

Dans but d'analyser la relation entre AT_1 et AT_2, nous avons simulé leurs variations en fonction du rapport $\frac{W}{R}$. Les coûts C_{proc}, C_{com}, C_{coproc}, C_{mem} sont ceux retenus lors du calcul de C_{app1} et C_{app2}. En utilisant l'outil COREGEN de Xilinx (Xilinx, 1998), les valeurs de C_{PE1} et C_{PE2} sont respectivement égales à 200 et 170 CLB sur un coprocesseur de type FPGA XC4036. Les valeurs de N_{PE1} et N_{PE2} sont respectivement égales à 6 et 7. La figure 5.8 illustre la variation de AT_1 et AT_2 en fonction de la résolution du système R pour une largeur de pixel donnée.

La figure 5.8 montre qu'il existe un point d'intersection entre AT_1 et AT_2 où les deux métriques sont équivalentes. Cependant, avant ce point, l'approche 1 est préférable, alorsqu'après ce point, l'approche 2 est privilégiée. Il est donc possible de choisir l'approche de calcul la plus adéquate pour une métrique coût-performance en fonction des coûts associés au processeur, coprocesseur, mémoire de communication et mémoire de calcul. Une analyse du marché du semiconducteur montre que les coûts associés aux mémoires sont relativement faibles, ce qui rend l'approche 2 accessible. Cependant, le rapport $\frac{W}{R}$ peut rendre cette approche difficilement utilisable en raison de la très quantité de mémoire requise, ce qui favoriserait l'approche 1. Cette dernière, même si elle est pénalisée par un temps de traitement plus grand que celui de l'approche 2, bénéficie de la complexité grandissante des circuits FPGA, comme ceux de la famille VIRTEX de Xilinx. L'utilisation d'une telle famille de FPGA permettrait l'intégration de plusieurs dizaines de processeurs élémentaires, conduisant ainsi à un facteur d'accélération consi-

dérable. Concernant les coûts monétaires associés à de tels FPGA, ils sont relativement bas (par rapport à la performance offerte, 100MHz et plus), ce qui les rend très accessibles pour des systèmes offrant un bon rapport coût-performance.

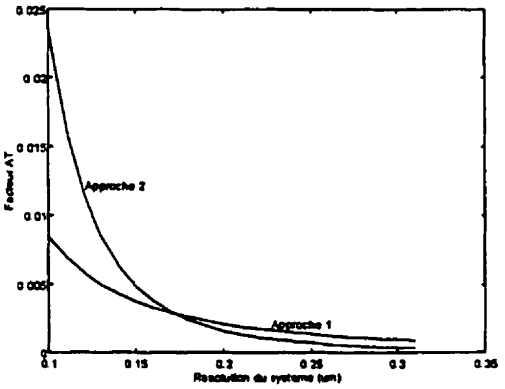

Figure 5.8: Variation des fonctions AT_1 et AT_2 en fonction de la résolution R.

5.6 Intégration des solutions proposées dans un système d'acquisition optique

Les travaux précédemment présentés peuvent être intégrés dans un système d'acquisition optique afin d'en améliorer la précision et d'en accélérer le traitement des transformations géométriques pour le calcul des coordonnées (x, y, z). La figure 5.9 présente le

diagramme bloc d'un système optique intégrant le module de correction d'erreurs et le module d'accélération des transformations géométriques.

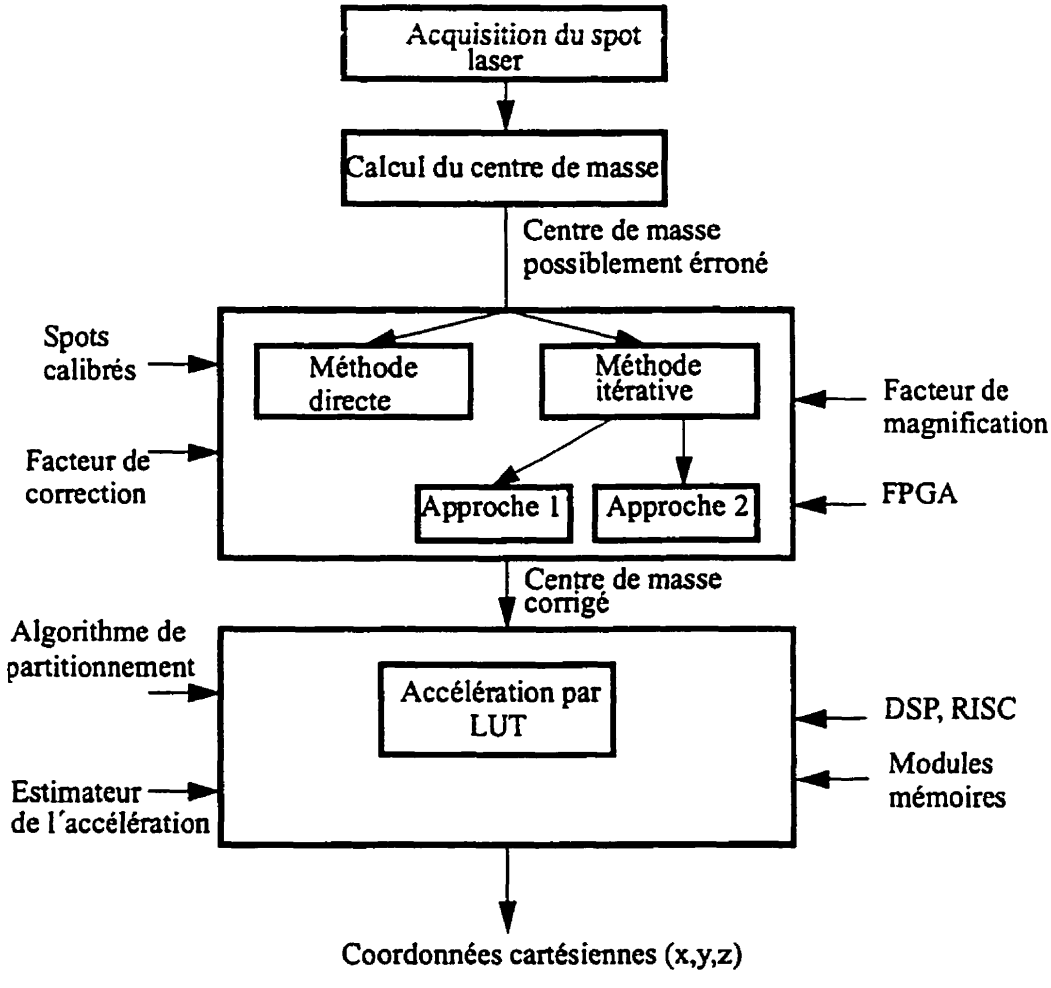

Figure 5.9: Diagramme bloc d'un système optique comprenant la correction d'erreurs et l'accélération des transformations géométriques.

Tel qu'illustré à la figure 5.9, le bloc de correction reçoit le centre de masse correspondant au centre de masse du spot courant. Il effectue sa correction suivant la méthode

directe ou itérative. Dépendemment de la performance désirée et des coûts associés, il est possible d'accélérer la méthode itérative en choisissant entre l'approche 1 ou l'approche 2. Après avoir corrigé le centre de masse, ce dernier est transmis au bloc d'accélération des transformations géométriques pour le calcul des coordonnées cartésiennes (x, y, z). Le bloc d'accélération exploite le résultat du partitionnement des transformations géométriques basées sur des LUT. Dépendemment de la quantité de mémoire disponible, les fonctions critiques seront accélérées par des tables LUT, alors que les autres seront implémentées en logiciel sur le processeur. Il est évident que la complexité des algorithmes de correction est plus grande que la complexité associée au calcul des coordonnées cartésiennes. Nous suggérons de confier le calcul des coordonnées cartésiennes au DSP (partition logicielle) et la correction du centre de masse à un accélérateur matériel tel qu'un FPGA (partition matérielle). La figure 5.10 illustre le partitionnement matériel-logiciel des tâches effectuées par un système optique. Le modèle d'architecture de système présenté à la figure 5.10 est appliquable à n'importe quel système d'acquisition optique. En raison du découplage apporté par un partionnement matériel-logiciel des tâches de correction de l'erreur et du calcul des coordonnées (x, y, z), l'architecture proposée est suffisamment flexible pour s'adapter à toute évolution technologique, tant du coté du DSP, des mémoires utilisées pour les LUT, ou du coté du FPGA.

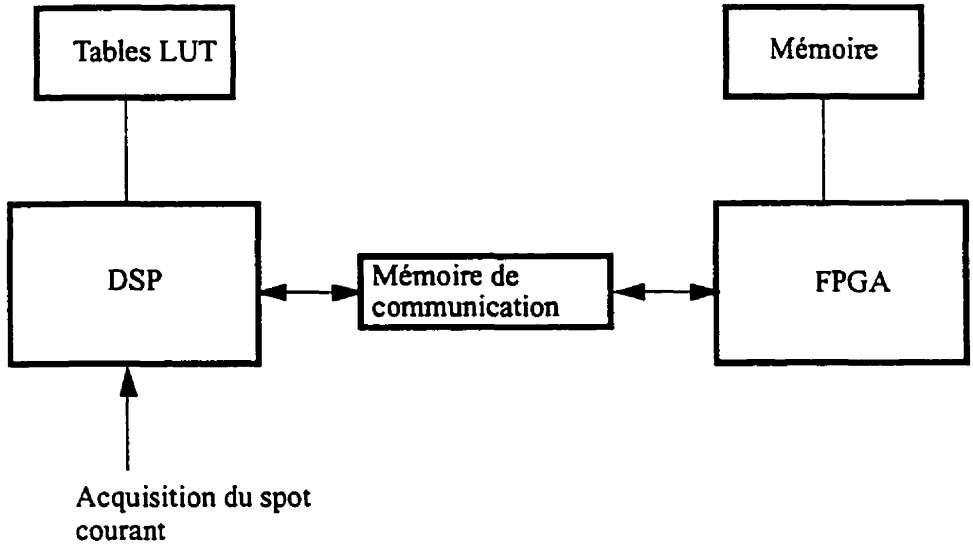

Figure 5.10: Modèle d'architecture de système pour une caméra optique de haute performance.

Conclusion

Dans cette thèse, nous avons abordé deux aspects importants reliés aux systèmes d'acquisition optique, à savoir la correction de l'erreur engendrée par une variation de réflectance, et l'accélération du calcul des coordonnées (x, y, z) . Dans le cas de la correction de l'erreur associée à une variation de réflectance, nous nous sommes concentrés sur les deux cas les plus fréquemment rencontrés, à savoir les sauts et les gradients de réflectance. Pour cela, nous avons développé deux méthodes de correction appelées respectivement méthode directe et méthode itérative. La méthode directe est constituée de deux phases: la phase de calibration et la phase de correction. La phase de calibration consiste à utiliser des cibles dont le profil de réflectance est connu et à effectuer des mesures sur un volume capable de contenir le plus grand objet que la caméra peut mesurer. Pour chaque mesure effectuée, un ensemble de paramètres sont calculés et conservés. La phase de correction consiste à effectuer des mesures sur des cibles inconnues et à corriger les erreurs causées par une variation de réflectance. Cette correction consiste principalement à reconstituer un estimé du profil inconnu de réflectance et, dépendemment de sa valeur, à appliquer l'équation de correction en se basant sur les informations collectées lors de la phase de calibration. La méthode directe se distingue par la rapidité de la correction apportée en raison de sa faible complexité, ce qui la rend bien adaptée aux systèmes optiques à hauts débits. Le coût monétaire de la méthode est fonction des coûts associés aux cibles de référence et au nombre de mesures à effectuer lors de la phase de calibration. Dépendemment du nombre de cibles de référence et du nombre de mesures à

effectuer lors de la phase de calibration, la méthode directe peut devenir lourde à utiliser. Son utilisation adéquate serait pour des applications où le volume de mesure est réduit et les profiles de réflectance ne sont pas nombreux. Contrairement à la méthode directe, la méthode itérative ne possède pas de phase de calibration. Elle consiste principalement à effectuer la recherche du facteur de magnification et de la position du centre du spot laser de référence qui minimisent une métrique donnée. Ce minimum, lorsqu'il est atteint, représente la position corrigée du spot laser reçu. La méthode itérative est caractérisée par une complexité de calcul élevée, comparée à celle de la méthode directe, en raison du processus de recherche effectuée. Cette complexité de calcul est principalement fonction de la largeur de pixel W et de la résolution du système R. Néanmoins, le coût monétaire associé est nul en raison de l'absence d'une phase de calibration.

Quant à l'accélération du calcul des coordonnées cartésiennes, nous avons contribué à deux niveaux complémentaires: accélération des algorithmes de correction proposés et accélération des transformations géométriques décrivant la caméra utilisée. Concernant l'accélération des algorithmes de correction, nous avons proposé une architecture de système matériel-logiciel basée sur un processeur (DSP, RISC, etc) et un coprocesseur (FPGA) qui implémente les algorithmes de correction. Nous avons proposé deux approches de calcul: l'approche 1 (section 5.3) qui consiste à calculer les énergies des pixels des spots de référence et l'approche 2 (section 5.4) qui consiste à précalculer les énergies des pixels des spots de référence et à les stocker dans une LUT. A chaque approche de calcul, sont associées deux sous-variantes: la sous-variante 1 qui consiste à exploiter plu-

sieurs processeurs élémentaires pour un même facteur de magnification et la sous-variante 2 qui consiste à exploiter plusieurs processeurs élémentaires pour plusieurs facteurs de magnification. Nous avons caractérisé chaque approche de calcul par une métrique de coût et une métrique de performance. En se basant sur ces deux métriques, nous avons montré que l'approche de calcul 1 limitait le coût, alorsque l'approche 2 maximise la performance. En combinant les métriques de coût et de performance, nous avons montré qu'il était possible de choisir une approche de calcul qui satisfasse une métrique de coût-performance en fonction des paramètres W et R. Pour les deux approches de calcul, nous avons proposé une architecture de système unique dont la flexibilité permet l'adaptation à toute évolution technologique tant du côté des processeurs que de celui des coprocesseurs. Comme le nombre de processeurs élémentaires est un facteur clé de performance, les deux approches de calcul vont bénéficier positivement de tout circuit FPGA très dense, telle que la famille de FPGA VIRTEX de Xilinx, qui permet l'intégration de dizaines de processeurs élémentaires. Quant à l'accélération des transformations géométriques, nous avons montré que l'usage des LUT constituait un moyen simple et efficace pour accélérer les calculs basés sur des fonctions trigonométriques et des développements en série. L'accélération par LUT est une approche très attrayante vu la baisse continuelle des coûts associés à la mémoire et l'augmentation significative de leurs performance (100 MHz et plus). Afin de quantifier cette accélération, nous avons proposé un modèle de prédiction du gain en vitesse, basé sur la loi d'Amdhal. Ce modèle tient compte des principaux paramètres liés à l'application et à l'architecture cible. Malgré que ce modèle ne tienne pas compte de certaines caractéristiques du processeur (nombre

d'étages du pipeline, latence, etc), les gains prédits sont très proches des gains obtenus sur les simulateurs de processeurs DSP cibles, tels que le C40 et le SHARC. Le modèle est applicable à un nombre quelconque de niveaux de hiérarchie mémoire et supporte directement les mémoires asynchrones. Dans le cas de l'utilisation de mémoires synchrones, il peut être facilement modifié pour tenir compte de la latence initiale associée à un accès mémoire. Il est évident que la situation idéale pour une application est qu'elle dispose d'une quantité de mémoire suffisante pour implémenter sous forme de LUT toutes ses fonctions critiques. Cependant, cette situation peut ne pas se présenter dépendemment des besoins en mémoire de l'application visée et des coûts associés. Il faut donc choisir parmi les fonctions disponibles, lesquelles seront implémentées sous forme de LUT, et lesquelles resteront en logiciel. Pour cela, nous avons développé un algorithme de partitionnement, qui à partir d'un ensemble donné de fonctions en logiciel, crée deux partitions afin de minimiser, si possible, le temps de traitement total: la partition des fonctions qui seront implémentées sous forme de LUT, et la partition des fonctions qui resteront en logiciel sur le processeur. L'algorithme proposé peut traiter autant les LUT à tailles fixes, que les LUT à tailles variables. Dans le cas de LUT à tailles fixes, l'algorithme est optimal. Cependant, dans le cas de LUT à tailles variables, l'optimalité de l'algorithme va dépendre de la méthode de groupage choisie. Notre approche de partitionnement peut être étendue à des applications dont les fonctions, potentiellement accélérables sous forme matérielle par un circuit FPGA, proviennent d'une bibliothèque. Chaque fonction de la bibliothèque est caractérisée par un coût et une performance lorsqu'elle est accélérée. Ceci permet d'élargir le spectre d'application de l'algorithme

de partitionnement proposé.

Les travaux effectués dans le cadre de cette thèse peuvent être approfondis et élargis afin d'atteindre un niveau supérieur de généralité. En effet, le problème de correction d'artefact a été abordé pour les deux situations les plus fréquemment rencontrées, à savoir les gradients et les sauts de réflectance. Les algorithmes de correction correspondant ont été conçu pour ne tenir compte que de ces deux situations. Il serait intéressant d'étendre le concept de correction d'artefact à des profils de réflectance autres que les gradients et les sauts. Ce problème pourrait être abordé en faisant appel à des techniques de reconstitution de signaux qui permettraient de reconstruire un estimé du signal de réflectance. L'analyse du déplacement du centre de gravité du signal de réflectance reconstituée, par rapport à son centre, pourrait nous renseigner sur la correction à apporter. Il est évident que les complexités de calcul associées à ces techniques sont supérieures à celles des algorithmes développés dans cette thèse, cependant il serait bon de comparer la qualité des corrections apportées, surtout en présence de bruit.

En ce qui concerne le problème de partitionnement abordé dans cette thèse, il est très intéressant de l'élargir à d'autres applications, ce qui conduirait à un algorithme de partitionnement plus général. Deux types d'applications méritent d'être analysées: les applications basées sur des fonctions provenant d'une bibliothèque (classe 1) et les applications basées sur des fonctions dont chacune d'elles est décrite par un ou plusieurs algorithmes (classe 2). La classe 1 représente une extension directe du problème de parti-

tionnement abordé dans cette thèse. En effet, les fonctions basées sur des LUTs sont remplacées par des fonctions en provenance de bibliothèques matériel-logiciel dont on connaît au préalable le coût et la performance dépendemment si elle est implémentée en logiciel ou en matériel. Ce type d'applications cible directement les systèmes HPRC (High-Performance Reconfigurable Computing), où chaque fonction du système est décrite par un algorithme lorsqu'elle est implémentée en logiciel sur le processeur et par un autre algorithme lorsqu'elle implémentée en matériel sur le coprocesseur (FPGA). L'aspect le plus important à analyser est l'effet de la reconfiguration du FPGA sur la performance du système. Si ce dernier est composé de plusieurs FPGA ou d'un seul FPGA supportant la reconfiguration dynamique, il est possible de charger une nouvelle fonction dans un FPGA pendant que la fonction courante occupe un autre FPGA. Dans une telle configuration, l'impact de la reconfiguration du FPGA sur la performance du système est quasi nulle. Dans le cas où un seul FPGA est disponible et qu'il ne supporte pas la reconfiguration dynamique, une analyse détaillée de l'impact de la reconfiguration du FPGA sur la performance du système doit être entreprise en fonction des principaux paramètres: vitesse de reconfiguration, taille du fichier de reconfiguration, quantité de données de reconfiguration transmises par cycle d'horloge, type de reconfiguration (sérielle ou parallèle), etc. Quant à la classe 2, celle-ci représente une généralisation du problème de partitionnement abordé dans cette thèse.

Sans perte de généralité, il est possible de distinguer trois situations potentielles:

1- chaque fonction est représentée par 2 algorithmes: son meilleur algorithme en logiciel

et son meilleur algorithme en matériel.

2- chaque fonction est représentée par 3 algorithmes: son meilleur algorithme en logiciel et ses deux meilleurs algorithmes en matériel, l'un favorisant la performance et l'autre le coût.

3- chaque fonction est représentée par quatre algorithmes: son meilleur algorithme en logiciel et ses trois meilleurs algorithmes en matériel, l'un favorisant la performance, un autre le coût et le dernier favorisant une combinaison coût-performance. Il est évident que ces trois situations ne constituent pas une liste exhaustive car il est aussi possible d'avoir des variantes au niveau logiciel. Cependant, nous pensons que ces trois situations sont celles les plus fréquemment rencontrées. Pour cette problématique, un aspect important à analyser est la manière d'aborder le problème de partitionnement dépendemment si l'on désire minimiser un temps de traitement, un coût ou une combinaison des deux. Cet aspect du problème amène à poser la question du choix de l'algorithme associé à une fonction dépendemment des critères à optimiser. L'impact des communications et des temps de reconfiguration des FPGAs doit être quantifié de façon précise afin de guider efficacement l'algorithme de partitionnement. Le problème à résoudre est complexe et peut constituer l'objet d'une recherche académique très intense, motivée par une évolution technologique rapide et d'applications de plus en plus exigeantes en performances.

Bibliographie

ALEXANDER, B.F. et CHEW-NG, K. (1991). Elimination of Systematic Error in Sub-pixel Accuracy Centroid Estimation. Optical Engineering, Vol. 30, 9, 1320-1331.

BACKES, P.G. et STEVENSON, W.H. (1990). A Comparison of Methods for Accurate Image Centroid Position Determination with Matrix Sensors. Journal of Laser Applications, Vol. 2, 2, 33-39.

BARIBEAU, R. et RIOUX, M. (1991). Influence of Speckle on Laser Range Finders. Applied Optics, Vol. 30, 20, 2873-2878.

BARIBEAU, R. et RIOUX, M. (1991). Centroid Fluctuations of Speckled Targets. Applied Optics, Vol. 30, 26, 3752-3755.

BARIBEAU, R., RIOUX, M. et GODIN, G. (1992). Color Reflectance Modeling Using a Polychromatic Laser Range Sensor. IEEE Transactions On Pattern Analysis and Machine Intelligence, Vol. 14, 2, 263-269.

BERALDIN, J.-A., BLAIS, F., RIOUX, M., DOMEY, J. et COURNOYER, L. (1992). Registered Range and Intensity at 10-Mega Samples per Second. Opt. Eng., Vol. 31, 1, 88-94.

BLAIS, F., RIOUX, M. et BERALDIN, J.-A. (1988). Practical Considerations for a Design of a High Precision 3-D Laser Scanner System. Proc. Soc. Photo-Opt. Instrum. Eng, 959, 225-246.

BLAIS, F., BERALDIN, J.-A., RIOUX, M., COUVILLON, R.A. et MACLEAN, S.G. (1993). Development of a Real-Time Tracking Laser Range Scanner for Space Applications. Workshop on Computer Vision for Space Applications, 161-173.

BLAIS, F. and RIOUX, M. (1986). Real-Time Numerical Peak Detector. Signal Processing, 11, 145-155.

BOULANGER, P. (1994). Multiscale Edge Detection Based on a New Geometrically Intrinsic Filter. SPIE Proceedings, Videometrics III, International Symposium on Photonic and Sensor and Controls for Commercial Applications, 2350, 264-278.

BUZINSKI, M., LEVINE, A. et STEVENSON, W.H. (1992). Laser Triangulation Range Sensors: A study of Performance Limitations. Journal of Laser Applications, Vol. 4, 1, 29-36.

DUPRAT, J. et MULLER, J.M. (1993). The CORDIC Algorithm: New Results for Fast VLSI Implementation. IEEE Transactions on Computers, Vol. 42, 2, 168-178.

El-Hakim, S.F. et Beraldin, J.-A. (1994). On the Integration of Range and Intensity Data to Improve Vision-Based Three-Dimensional Measurements. SPIE Proceedings, Videometrics III, 2350.

EORY, F.S. (1997). A Core-Based System-to-Silicon Design Methodology. IEEE Design&Test of Computers, Oct-Dec, 36-41.

ERCEGOVAC, M.D. et LANG, T. (1987). Fast Cosine/Sine Implementation Using ON-Lin CORDIC. 21st Annual Asilomar Conference on Signals, Systems & Computers, 222-226.

FOWKES, R.E. (1993). Hardware Efficient Algorithms for Trigonometric Functions. IEEE Transactions on Computers, Vol. 42, 2, 235-239.

GRADSHTEYN, I.S. et RYZHIK, I.M. (1992). Table of Integrals, Series, and Products, Corrected and Enlarged Edition. Academic Press, Inc.

GUPTA, R. et ZORIAN, Y. (1997). Introducing Core-Based System Design. IEEE Design&Test of Computers, Oct-Dec, 15-25.

HAUK, S. (1998). The Roles of FPGA's in Reprogrammable Systems. Proceedings of The IEEE, Vol. 86, 1, 615-638.

HAVILAND, G.L. et TUSZYNSKI, AL. A. (1980). A CORDIC Arithmetic Processor Chip. IEEE Transactions on Computers, Vol. C-29, 2, 68-79.

HENNESSY, J. et PATTERSON, D.A. (1990). Computer Architecture: A Quantitative Approach. Morgan Kauffmann Publishers, Inc.

HWANG, K. (1993). Advanced Computer Architecture: Parallelism, Scalability, Programmability. McGraw-Hill, Inc.

KATAYAMA, Y. (1997). Trends in Semiconductor Memories. IEEE Micro, Nov-Dec, 10-17.

KHALI, H. (1995). Modélisation d'un système optique à base d'une caméra autosynchronisée et étude de la compléxité de certaines transformations géométriques. Rapport de stage au CNRC, Groupe de recherche en microélectronique, Département de génie électrique, Ecole polytechnique de Montréal, Canada.

KHALI, H., Y. SAVARIA et HOULE, J.L (1997). Computational Limits of Homogeneous Acceleration Using Lookup Tables. High Performance Computing Systems (HPCS97), 345-351.

KHALI, H., SAVARIA, Y., HOULE, J.L., BERALDIN, J.A., BLAIS, F. et RIOUX, M.

(1995). A VLSI Chip for 3-D Camera Calibration. Canadian Conference on Electrical and Computer Engineering (CCECE95), 1, 120-123.

KIM, R.C. (1988). Design and Fabrication of Custom Beam Forming Optics for Optical Metrology and Signal Processing. Thèse de doctorat, Minnesota University.

KISHORE, K. et CAVALLARO, J.R. (1993). Numerical Accuracy and Hardware Tradeoffs for CORDIC Arithmetic for Special-Purpose Processors. IEEE Transactions on Computers, Vol. 42, 7, 769-779.

KLEINEMEIER, B. (1988). Measurement of CCD Interpolation Functions in the Subpixel Precision Range. SPIE Proceedings, Industrial Inspection, 1010, 158-165.

KOREN, I. et ZINATY, O. (1990). Evaluating Elementary Functions in a Numerical Coprocessor Based on Rational Approximations. IEEE Transactions on Computers, Vol. 39, 8, 1030-1037.

KOREN, I. (1993). Computer Arithmetic Algorithms. Prentice-Hall, Inc, New Jersey.

KUMAR, S., AYLOR, J.H., JOHNSON, B.W. et WULF, W.A. (1993). Exploring Hardware/Software Abstractions & Alternatives for Codesign. The 2nd International Workshop on Hardware/Software Codesign.

LANG, S., ROTH, G., GREEN, D.A. et BURHANPURKAR, V. (1994). Visual Measurement of Orientation Using Ceiling Features. Proceedings of the IEEE Instrumentation and Measurement Technology Conference, 552-555.

LAURIN, D.G. et RIOUX, M. (1993). Three Dimensional Object Tracking From Range Images Using Sine Wave Coding and Fourier Transform. SPIE Proceedings, Videometrics II, 2067, 252-267.

LEVINE, A.R. (1992). High Performance Optical Triangulation Ranging. Thèse de doctorat, Purdue University, USA.

MUNDY, J. L. et PORTER, G. B. (1987). A Three-Dimensional Sensor Based on Structured Light. Three-Dimensional Machine Vision, Part III, Kluwar Academic Publishers, 3-61.

POUSSART, D. (1986). Correction des données provenant d'u système de vision tridimensionnelle. Rapport technique, Laboratoire de vision et systèmes numériques, Département de génie électrique, Université Laval, Québec, Canada.

RIOUX, M. (1994). Digital 3-D Imaging: Theory and applications. SPIE Procedings, Videometrics III, International Symposium on Photonic and Sensors and Controls for Commercial Applications, 2350, 2-15.

RIOUX, M. (1984). Laser Range Finder Based on Synchronized Scanners. Applied Optics, 23, 3837-3844.

RIOUX, M., BECHTHOLD, G., TAYLOR, D. et DUGGAN, M. (1987). Design of a Large Depth of View Three-dimensional Camera for Robot Vision. Opt. Eng., Vol. 26, 12, 1245-1250.

RODRIGUES, M.R.D., ZURAWSKI, J.H.P. et GOSLING, J.B. (1981). Hardware Evaluation of Mathematical Functions. IEE Proceedings, Vol. 128, Pt. E, 4, 155-164.

SAVARIA, Y., BOIS, G., POPOVIC, P. et WAYNE, A. (1996). Computational Acceleration Methodologies: Advantages of Reconfigurable Acceleration Subsystems. SPIE Proceedings, 2914, 195-205.

SCHOWENGERDT, R. A. et WHITE, R. G. (1994). Effet of Point-Spread Functions on Precision Edge Measurement. J. Opt. Soc. Am., Vol. 11, 10, 2593-2603.

SMITH, J.E. et Weiss, S. (1994 Juin). PowerPC 601 and Alpha 21064: A Tale of Two RISCs. Computer Magazine, 46-58.

SOUCY, M., LAURENDEAU, D., POUSSART, D. et AUCLAIR, F. (1990). Behaviour

of the Center of Gravity pf a Refelected Gaussian Laser Spot Near a Surface Reflectance Discontinuity. Elsevier, Industrial Metrology, 1, 261-274.

STEVENSON, W. (1992). The Use of Laser Triangulation Probes in Coordinate Measuring Machines for Part Tolerance Inspection and Reverse Engineering. SPIE proceedings, 1821, 406-414.

SVETKOFF, D.J. (1991). Influence of Object Structure on the Accuracy of 3D Systems for Metrology. SPIE Proceedings, Optics, Illumination, and Image Sensing for Machine Vision VI, 1614, 218-230.

TAKAGI, N., ASADA, T. et YAJIMA, S. (1991). Redundant CORDIC Methods with a Constant Scale Factor for Sine and Cosine Computation. IEEE Transactions on Computers, Vol. 40, 9, 989-995.

VILLASENOR, J. et HUTCHINGS, B. (1998). The Flexibility of Configurable Computing. IEEE Signal Processing Magazine, Sept, 67-84.

WALTHER, J.S. (1971). A Unified Algorithm for Elementary Functions. Spring Joint Computer Conference, 379-385.

WENHENG, L. et PRASANNA, V.K. (1998). Utilizing the Power of High-Performance

Computing. IEEE Signal Processing Magazine, Sept, 85-100.

WESTE, N.H.E. et ESHRAGHIAN, K. (1993). Principles of CMOS VLSI Design: A Systems Perspective. Second Edition, Addisson Wesley.

Xilinx, (1998). Core Generator, Version 1.4.

Xilinx, (1999). The Programmable Logic Data Book.

ANNEXE I: Estimation du centre de masse du signal de réflectance reconstituée (méthode DIRECTE)

Cette annexe a pour but de développer certaines transformations mathématiques relatives à l'équation (3.7).

L'équation (3.7) est donnée par:

$$\bar{x} = \frac{\sum_{i=-d}^{d} iI(i)}{\sum_{i=-d}^{d} I(i)}.$$

Cette équation décrit la position du centre de masse du signal laser incident sur le détecteur optique en présence d'un saut de réflectance.

Tel que présenté à la section 3.7.2, j représente l'indice du pixel sur lequel le saut de réflectance est imagé, alors que les h pixels avant le pixel j ont observé une réflectance uniforme α et les $(2d-h)$ pixels après j ont observé une réflectance uniforme $(\alpha+\delta)$. Le numérateur de (3.7) peut alors être développé comme suit:

$$\sum_{i=-d}^{d} iI(i) = \sum_{i=-d}^{h} iI(i) + jI(j) + \sum_{i=j+1}^{d} iI(i) = \alpha \sum_{i=-d}^{h} i + jI(j) + (\alpha+\delta) \sum_{i=j+1}^{d} i.$$

Les séries arithmétiques de raison 1 impliquées dans le calcul du numérateur de (3.7) sont données par:

$$\alpha \sum_{i=-d}^{h} i = \alpha\left(-hd + \frac{h(h-1)}{2}\right).$$

$$(\alpha+\delta) \sum_{i=j+1}^{d} i = (\alpha+\delta)\left((j+1) + (j+2) + \ldots + (j+(2d-h))\right).$$

Cette série peut être réécrite comme suit:

$$(\alpha+\delta) \sum_{i=j+1}^{d} i = (\alpha+\delta)\left((2d-h)j + \frac{(2d-h)(2d-h-1)}{2}\right).$$

Le numérateur de (3.7) est alors égal à:

$$\sum_{i=-d}^{d} iI(i) = \alpha\left(-hd + \frac{h(h-1)}{2}\right) + jI(j) + (\alpha+\delta)$$

$$\left((2d-h)j + \frac{(2d-h-1)(2d-h)}{2}\right)$$