Titre: Title:	Élaboration d'un module de traitement d'image pour un stimulateur visuel cortical
Auteur: Author:	Louis-Xavier Buffoni
Date:	2004
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Buffoni, L.-X. (2004). Élaboration d'un module de traitement d'image pour un stimulateur visuel cortical [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7466/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/7466/
Directeurs de recherche: Advisors:	Mohamad Sawan
Programme: Program:	Non spécifié

NOTE TO USERS

UNIVERSITÉ DE MONTRÉAL

ÉLABORATION D'UN MODULE DE TRAITEMENT D'IMAGE POUR UN STIMULATEUR VISUEL CORTICAL

LOUIS-XAVIER BUFFONI

DÉPARTEMENT DE GÉNIE ÉLECTRIQUE

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION

DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES

(GÉNIE ÉLECTRIQUE)

JUILLET 2004

Your file Votre référence
ISBN: 0-612-97931-8
Our file Notre référence
ISBN: 0-612-97931-8

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

ÉLABORATION D'UN MODULE DE TRAITEMENT D'IMAGE POUR UN STIMULATEUR VISUEL CORTICAL

présenté par : BUFFONI Louis-Xavier
en vue de l'obtention du diplôme de : Maîtrise ès Sciences Appliquées
présenté au jury d'examen constitué de :

M. KHOUAS Abdelhakim, Doct., Président
M. SAWAN Mohamad, Ph.D., Directeur
M. ABOULHAMID El Mostapha, Ph.D., Membre

REMERCIEMENTS

Je tiens à remercier mon directeur de recherche, Mohamad Sawan, pour m'avoir accordé la chance de faire partie de cet ambitieux projet qu'est le stimulateur visuel cortical, pour m'avoir guidé et conseillé tout en m'affichant une totale confiance.

Merci à tous mes collègues de l'équipe PolySTIM pour leur bonne compagnie, et particulièrement à Jonathan Coulombe qui m'a pris sous son aile avant le début, et qui a continué à me conseiller tout au long de cette maîtrise. Merci également aux secrétaires pour leur bienveillance qui m'a sauvé en plusieurs occasions ... elles se reconnaîtront.

Je remercie toute ma famille, élargie, et particulièrement mes parents, Diane Comtois et Jean-François Buffoni, pour leur aide infaillible et leur intérêt.

Enfin, je tiens à remercier tous mes amis pour, entre autres, m'avoir rappelé qu'il y a un soleil, dehors. Un merci particulier à mes deux colocataires successifs, pour m'avoir enduré et inspiré grâce à leur autodiscipline exemplaire.

RÉSUMÉ

L'avancement de la technologie nous permet d'envisager qu'un jour, certaines pertes de fonctions physiologiques complexes comme la vision soient récupérées, du moins partiellement, grâce à la stimulation physiologique. Plusieurs avenues de recherches vont dans ce sens, mais une des plus prometteuses est la stimulation visuelle corticale, qui consiste à envoyer des impulsions de courant électrique dans le cortex visuel, ayant pour effet de produire des sensations lumineuses ponctuelles appelées phosphènes. La plupart des travaux ayant été effectués à ce jour concernent l'électronique du stimulateur, les microélectrodes, l'interface tissus-électrode, ou les travaux d'ordre purement neurologique. Le présent projet s'inscrit dans cet effort de recherche. Plus précisément, il s'agit d'élaborer un système permettant de transformer les images provenant d'une caméra transportée par l'aveugle en commandes de stimulation vers un stimulateur implanté dans le cortex. Cet objectif nécessite de déterminer une ou des façons plausibles de générer des percepts ordonnés dans le champ visuel, à partir des images, qui permettent à un usager de comprendre son environnement le plus efficacement possible. Nous verrons que la réponse est loin d'être triviale.

La première étape du projet a été de faire une recherche la plus exhaustive possible des connaissances actuelles dans le domaine biomédical concernant les phosphènes, leur aspect, leur organisation et surtout, lorsqu'ils sont évoqués en même temps, leur capacité à être identifiés comme représentant des formes. Les connaissances actuelles sur ce sujet étant encore rudimentaires, le système conçu permettra l'application de diverses stratégies pour transformer des images en patrons de phosphènes compréhensibles, avec un niveau de réalisme variable.

Ces stratégies sont constituées d'algorithmes de traitement d'image qui ont d'abord été implémentés de manière logicielle afin de les évaluer. Un système logiciel complet, faisant le lien entre la caméra et une future interface avec le contrôleur de l'impant, a été réalisé. Ce système est assorti d'un logiciel d'interface-usager permettant à un expérimentateur de le contrôler.

Parmi les traitements d'image choisis, un effort particulier a été mis dans la conception d'un algorithme de segmentation étant donné son utilité cruciale dans cette application. En effet, il sert simultanément à réduire la résolution de l'image de manière efficace et à la découper en zones de mêmes caractéristiques. Cet algorithme est décrit en détail. Avec des données de distance des objets fournies par un éventuel détecteur de distance, seules les zones stratégiques seront affichées sous forme de taches de lumière. Cela constitue une des approches les plus réalistes, car les phosphènes sont évoqués en mode « tout-ou-rien », et fournira certainement l'usager avec des indices utiles qui lui permettront de comprendre son environnement.

Le système de traitement d'image sous forme logicielle a été implémenté principalement pour évaluer la qualité des algorithmes, et également pour vérifier la compatibilité avec le logiciel d'interface-usager. Un système embarqué a également été développé pour être utilisé pour des tests in vivo, car l'évaluation des stratégies de traitement d'image nécessite que le système frontal du stimulateur visuel cortical soit portable. Dans son état actuel, il est remis sous la forme d'un groupe de modules logiciels et matériels codés en VHDL.

La version embarquée du système a également été complétée parce que les algorithmes implémentés en logiciel ne permettent pas le calcul d'un résultat en un temps suffisant pour être utilisé avec un stimulateur visuel. Une part de calcul a dû être laissée à un coprocesseur dédié. Il a été implémenté en VHDL et le design complet, intégré dans une architecture AMBA avec un processeur ARM, a été validé dans un environnement de simulation de co-design (logiciel et matériel simultanément). Les résultats montrent que les algorithmes sont fonctionnels et atteignent les performances voulues.

ABSTRACT

Continuous technological advances may soon allow some lost complex physiological functions like vision be recovered, at least partially, with physiological stimulation. Many research avenues are striving to meet this goal, but one of the most promising is the visual cortical stimulation, which consists in injecting current pulses in the visual cortex, producing ponctual visual sensations called phosphenes. Most of research teams have been focusing on designing microstimulators, microelectrodes, bio-electrode interfaces, or purely neurological researches. This project is part of this research effort. More precisely, it consists of elaborating a system to allow images captured by a video camera carried by the blind be transformed into stimulation command signals for an implant. In order to achieve this goal, we need to determine adequate ways of transforming images into percepts that would be disposed in the visual field, allowing a user to understand his surroundings. The answer to that issue is not trivial.

The first step of the project has been to seek, as exhaustively as possible, actual knowledge in the biomedical field about phosphenes, their aspect, their organization, and above all, their ability in being accurately identified as shapes when many of them are evoked at the same time. Knowledge in this field still being in its early beginnings, the system described in this master thesis implements multiple strategies in transforming images into intelligible phosphene patterns, with variable realism.

These strategies are constituted of image processing algorithms that have first been implemented in software for evaluation purposes. A complete software has been developed, making the link between the input device and an interface with the implant controller. It is also assorted with a user-interface software allowing its control.

Among the selected image processing techniques, particular attention has been paid to a segmentation algorithm because of its crucial use in this application. As a matter of fact, it is simultaneously used to efficiently reduce image resolution as well as to separate the image

into areas of same characteristics. This algorithm implementation is described in great details. With distance data obtained by an range finder, only the strategical areas will be displayed in the form of light patches. It constitutes one of the most realistic strategies, because phosphenes are evoked in an on/off fashion, and this will hopefully offer valuable clues about the user's surroundings.

The software image processing system has been principally implemented in order to evaluate the quality of the algorithms, and also to validate compatibility with the user-interface software. An embedded system has also been developed to be used for in vivo tests, because these tests require that the front-end subsystem of the visual cortical stimulator be portable. It is actually in the form of a set of software and *soft* hardware modules, that is coded in VHDL.

The embedded counterpart of the system has also been completed because the algorithms implemented in software could not meet speed performance needed to be used with a visual stimulator. A fraction of processing must have been left to a dedicated coprocessor. It has been implemented in VHDL and the complete design, integrated in an AMBA bus architecture with an ARM processor, has been validated in a codesing simulation environment (simultaneous software and hardware simulation). The algorithms functionality have been validated and their requested performance are met.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

LISTE DES SIGLES ET ABRÉVIATIONS

A	Amplitude de l'onde de stimulation
AHB	Advanced High-Speed Bus
ALU	Arithmetic-Logic Unit
AHB	Advanced High-performance Bus
APB	Advanced Peripheral Bus
AMBA	Advanced Microcontroller Bus Architecture
ASIC	Application Specific Integrated Circuit
CMC	Canadian Microelectronics Corporation
CMOS	Complementary Metal-Oxyde-Semiconductor
CNA	Convertisseur Analogique-Numérique
CV	Carte Visuotopique
DI	Durée d'interphase de l'onde de stimulation
DIT	Durée inter-train de l'onde de stimulation
DoG	Difference of Gaussians
DP	Durée de phase de l'onde de stimulation
DP-RAM	Dual-Port Random-Access-Memory
DSP	Digital Signal Processor
DT	Durée de train de l'onde de stimulation
DUT	Device Under Test
ENG	Electroneurogramme
F	Fréquence de l'onde de stimulation / Stimulation pulse frequency
FIFO	First-In First-Out
FPGA	Field Programmable Gate Array
GRM	Groupe de recherche en microélectronique
HLRG	Hybrid-Linkage Region Growing
I	Stimulation Current Amplitude
IPI	Inter-Pulse Interval
IPS	Image Processing System

ITI	Inter-Train Interval
LoG	Laplacian of Gaussian
LUT	Look-Up Table
MAC	Multiplying Accumulator
MEMS	Micro-Electrical Mechanical System
PC	Personal Computer
PD	Pulse duration
PE	Processing Element
PVFC	Phosphene Visual Field Coordinates
RAM	Random Access Memory
RISC	Reduced Instruction Set Computing
ROI	Region of Interest
RTL	Register-Transfer Level
SEQ	Set Equal
SIMD	Single-Instruction Multiple-Data
SLRG	Single-Linkage Region Growing
SLT	Set Lower Than
SoC	System on Chip
SSA	Stimulation Site Addresses
STI	Système de Traitement d'Image
SVC	Stimulateur Visuel Cortical
TI	Texas Instruments
TL	Train length
UAL	Unité arithmétique et logique
USB	Universal Serial Bus
VCS	Visual Cortical Stimulator
VFE	Visual Field Emulator
VHDL	Very high speed integrated circuit Hardware Description Langage
VM	Visuotopic Map

LISTE DES ANNEXES

INTRODUCTION

Près de 35 millions de personnes dans le monde sont affectées par une grave maladie de la vue. Leurs conséquences sont très problématiques pour le fonctionnement de ces personnes en société. En particulier pour ce qui a trait au déplacemement. Plusieurs solutions ont été envisagées au cours du temps pour réduire ce handicap, comme par exemple l'alphabet Braille pour rendre la lecture par voie tactile plus efficace. Récemment, de nouvelles techniques médicales ont vu le jour, comme la stimulation physiologique. La récupération de fonctions par cette méthode repose sur notre capacité à produire des réactions suite à la stimulation de cellules nerveuses. Cette technique pourrait être appliquée à la vision en stimulant électriquement certaines cellules nerveuses sur le chemin visuel. En particulier, il a été démontré que la stimulation du cortex visuel, centre principal de traitement de la vue dans le cerveau, générait des percepts lumineux appelés « phosphènes ». C'est sur ce principe que repose le fonctionnement d'un stimulateur visuel cortical (SVC) dont le but est de rendre une vision partielle à un patient dont le système visuel aurait été rendu non fonctionnel.

L'idée d'un tel stimulateur est simple à prime abord. Une caméra, fixée sur un monture de lunette portée par l'aveugle et regardant devant lui, prendrait des images du monde extérieur qui seraient ensuite traitées et transformées en commandes destinées à un implant situé dans la boîte crânienne. Celui-ci aurait la tâche de stimuler adéquatement le cortex visuel de manière à créer des percepts qui donneraient une idée à l'aveugle de ce que la caméra capte. De nos jours, les travaux qui portent sur un tel dispositif se concentrent sur le développement du stimulateur en tant que tel. Par contre, la partie frontale du système, c'est-à-dire la transformation des images en commandes de stimulation, n'est pas triviale car elle doit tenir en considération les contraintes physiologiques liées à la stimulation corticale, dont la compréhension est encore très limitée. Le projet de maîtrise présenté dans ce mémoire concerne le développement de cette partie frontale. Plus précisément, il s'agit d'envisager la façon dont les images seront transformées de manière à donner de l'information utile à un aveugle par le biais de son stimulateur visuel cortical, et ce, à la lumière des contraintes physiologiques connues.

Nous verrons que le grand nombre d'incertitudes liées à ces contraintes nécessite d'envisager un système permettant plusieurs stratégies de transformation des images. Le but est de concevoir un module flexible, faisant le pont entre la caméra et le stimulateur, qui permettra de procéder à des tests *in vivo* afin d'examiner réellement les stratégies employées et de plus tard les raffiner. Évidemment, la réussite de ces tests repose sur l'accès à un stimulateur robuste, lui-même testé en laboratoire, ce qui les repousse à une date se situant bien au-delà du cadre de la présente maîtrise. Le système qui a été développé constitue un premier pas vers la partie frontale d'un tel système. De plus, il se concentre sur l'aide à la mobilité. Enfin, une caméra vidéo est considérée comme dispositif d'entrée. D'autres avenues pourront être explorées dans le futur.

Le premier chapitre s'applique à exposer les résultats des études biomédicales précédentes portant sur le fonctionnement de la vision dans la perspective de la stimulation physiologique. Certains enjeux et défis ayant une influence directe sur le développement de la partie frontale d'un SVC seront mis en évidence.

Le chapitre 2 est un article soumis à la revue « Artificial Organs » qui présente d'abord une revue des efforts qui ont été faits par les diverses équipes de recherche en vue de développer la partie frontale d'un SVC. À la lumière du contexte physiologique, nous réaliserons qu'à cause des nombreuses incertitudes, il faudra envisager plusieurs stratégies de traitement des images. Ces stratégies sont présentées, et ensemble, elles constituent le cœur d'un système de traitement d'image dédié à un SVC.

Afin de valider les algorithmes constituant le module de traitement d'image, ces stratégies ont été implémentées de manière purement logicielle. Ce logiciel est présenté au chapitre 3, qui débute par une présentation générale des techniques usuelles de traitement d'image.

De tous les algorithmes implémentés au sein du système, le plus crucial est la segmentation d'image. Cet algorithme est décrit en détail dans un deuxième article, soumis à la revue « IEE

Medical and Biological Engineering and Computing », qui constitue le chapitre 4. Le design de cet algorithme a orienté l'implémentation de l'architecture d'un système embarqué, qui s'avèrera nécessaire pour la vérification in vivo du système de stimulation complet. Cette architecture est également présentée dans ce chapitre.

Finalement, en annexe se trouvent des détails supplémentaires sur l'implémentation logicielle et matérielle du module de traitement d'image pour le SVC. Les fichiers des sources s'y trouvent également.

CHAPITRE 1

CONTEXTE BIOMÉDICAL

1.1 Introduction

Nous abordons ce chapitre par la définition de la stimulation électrique et celle du système visuel humain. Les diverses approches pour la récupération de la vision sont ensuite examinées, avant de se pencher plus particulièrement sur celle qui nous intéresse, soit la stimulation corticale. Un bref historique expose le concept de phosphène, son apparence, son organisation spatiale, la manière dont il est évoqué par stimulation électrique, et les travaux de pointe dans le domaine de la reconnaissance de patrons de phosphènes. Suivent ensuite une description des stimulateurs visuels corticaux, ainsi qu'un survol des études portant sur le traitement d'image destiné à ces stimulateurs. La dernière section est consacrée plus précisément à l'organisation visuotopique et sur ce qui a été fait dans le cadre de ce projet pour représenter l'effet hypothétiquement perçu des patrons de phosphènes qui seront générés à la suite du traitement de l'image.

1.2 Dispositifs visuels

1.2.1 Stimulation neuronale

La stimulation neuromusculaire est couramment utilisée pour rétablir certaines fonctions du corps humain qui ont été rendues invalides par une maladie ou un accident. Les méthodes utilisées peuvent être classées en trois grandes catégories :

- l'injection de substances chimiques à un groupe de cellules nerveuses, modifiant ainsi les propriétés d'inhibition et d'exhibition de leurs activités synaptiques [23],
- la stimulation magnétique et qui crée un potentiel d'action par induction électromagnétique
- la stimulation électrique qui créent un potentiel d'action par le passage d'un courant ionique [29,75,78].

De nos jours, la stimulation par substance chimique n'est qu'au stade de développement, et peu d'applications de réhabilitation utilisent cette méthode [78]. Quant à la stimulation magnétique, bien qu'elle soit avantageuse en termes de non-invasibilité, elle est davantage utilisée pour le diagnostic de maladies neuromusculaires [35]. Aucun dispositif n'utilisant cette méthode n'est en mesure d'être assez précis. La stimulation électrique est la seule qui est présentement utilisée pour la réhabilitation de fonctions d'organes. On la retrouve entre autres dans les implants urinaires et cochléaires [58,74]. Elle nécessite l'emploi d'électrodes, et donc certaines considérations de biocompatibilité doivent être prises en compte étant donnée la présence d'un contact métal-tissu. Pour mieux comprendre les phénomènes liés à la stimulation neuromusculaire, il importe de préciser le fonctionnement d'une cellule nerveuse.

La figure 1.1 est un schéma simplifié d'une cellule nerveuse, ou neurone. Elle est composée de dendrites qui reçoivent les impulsions des neurones en amont, et de l'axone qui propage ces informations vers les neurones en aval. Les propriétés électrochimiques du neurone font en sorte qu'il conserve une différence de potentiel fixe entre l'extérieur et l'intérieur. Le bris momentané de cet équilibre crée ce qui est communément appelé un potentiel d'action, qui se propage rapidement à travers l'axone, jusqu'aux dendrites des cellules en aval à travers le milieu intercellulaire, riche en ions. Cette interface s'appelle la synapse [45].

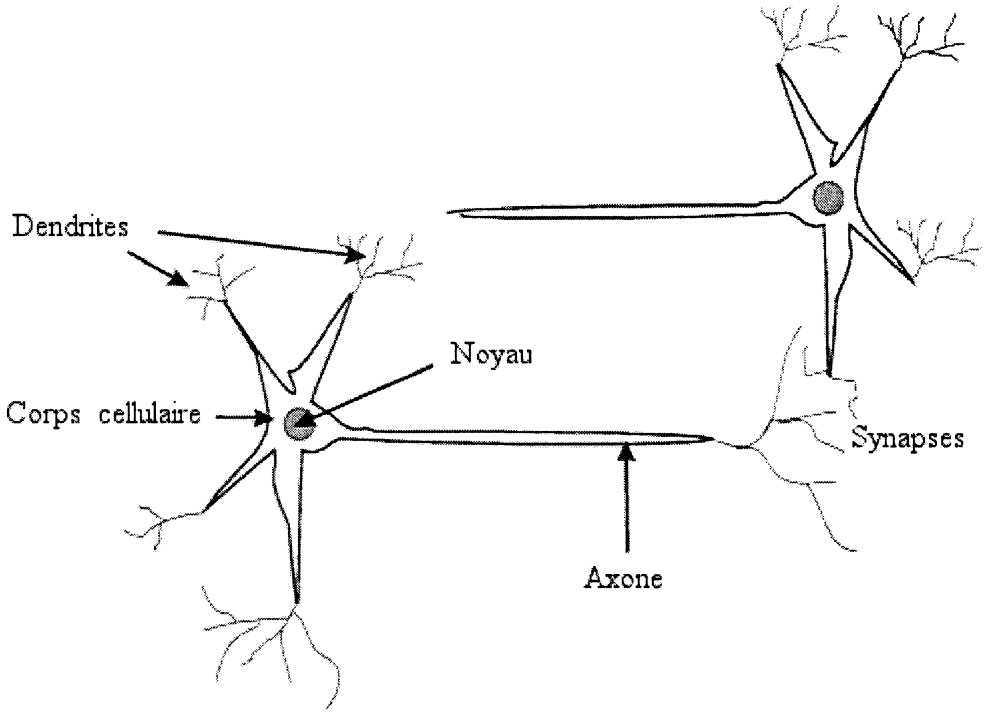


Figure 1.1 – Schéma simplifié d'une cellule nerveuse (tiré de [72]).

Le neurone agit comme un interrupteur : il décide de propager ou non un potentiel d'action en fonction de ses entrées, c'est-à-dire des potentiels d'action provenant des cellules qui sont reliées à ses dendrites. Tel est le but de la réhabilitation de fonctions nerveuses par la stimulation : créer un ensemble de potentiels d'actions formant un électroneurogramme (ENG) afin de remplacer celui échangé avec l'organe original. Il est alors propagé par les neurones subséquents.

Dans le cas de la stimulation électrique, c'est l'injection de courant, et donc le déplacement de charges, dans le milieu ionique qui déclenche le potentiel d'action. L'injection de charges peut être dommageable pour les tissus, c'est pourquoi le stimulus électrique est typiquement une onde biphasique, c'est-à-dire que le courant circulera dans les deux sens, afin d'introduire une quantité de charge nette nulle. La figure 1.2 illustre la forme d'onde utilisée

dans la plupart des applications biomédicales [69,87], et particulièrement la stimulation corticale.

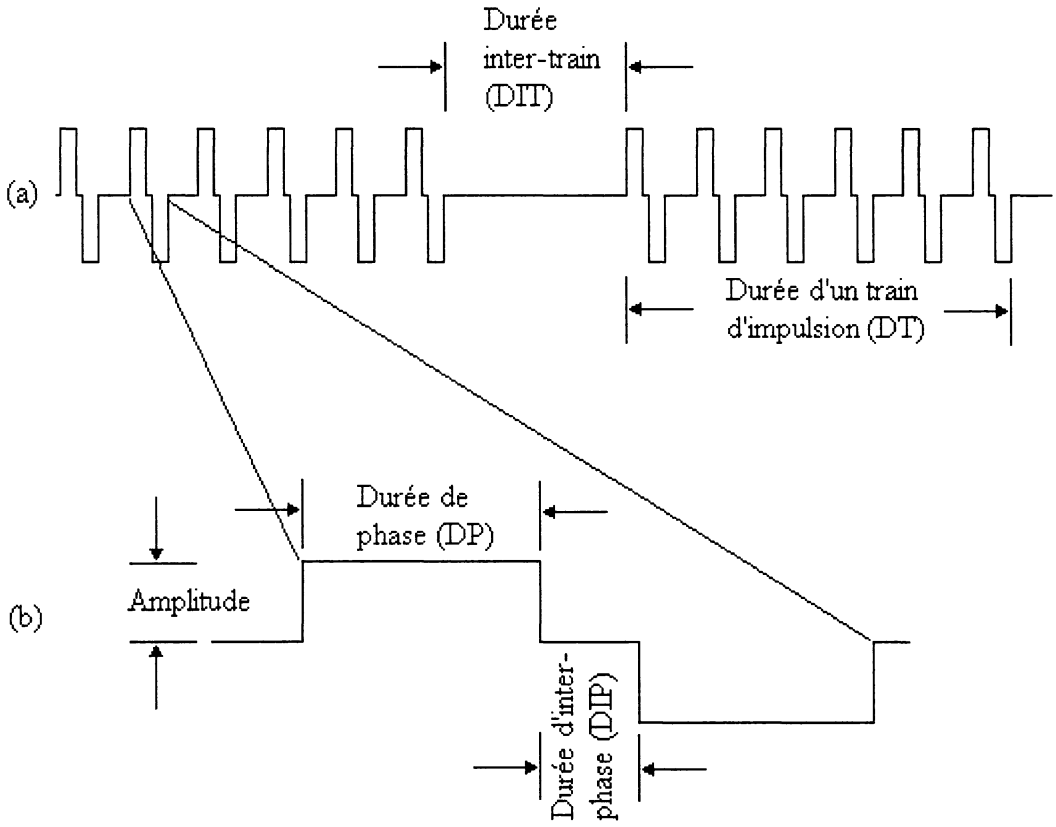


Figure 1.2 – Formes d'ondes utilisées pour la stimulatiom intracorticale.

1.2.2 Système visuel humain et maladies de la vision

La figure 1.3 illustre la structure du système visuel humain. L'œil a pour principale fonction de focaliser la lumière provenant de l'extérieur sur sa rétine, composée de capteurs. Deux sortes de cellules jouent ce rôle : les cônes, sensibles à la couleur et situés en majorité près de la fovea où la densité de capteurs est la plus grande, et les bâtonnets, insensibles à la couleur et situés sur toute la surface de la rétine. L'information visuelle provenant de ces cellules est traitée immédiatement par deux étages de cellules, les cellules ganglionnaires et les cellules bipolaires. Les signaux sont ensuite transportés par le nerf optique, composé d'environ 1 million de fibres, vers le chiasma optique, point de croisement entre les nerfs optiques des

deux yeux. L'information passe alors par les tractus optiques vers les corps genouillés latéraux où elle est traitée davantage. Elle arrive enfin au cortex visuel situé à l'arrière du cerveau, où plusieurs étages de traitement font le lien entre l'information visuelle et les autres parties du cerveau liées à son interprétation. L'information voyage dans les deux directions, ce qui implique un phénomène de rétroaction entre le cortex visuel et les couches de neurones en amont.

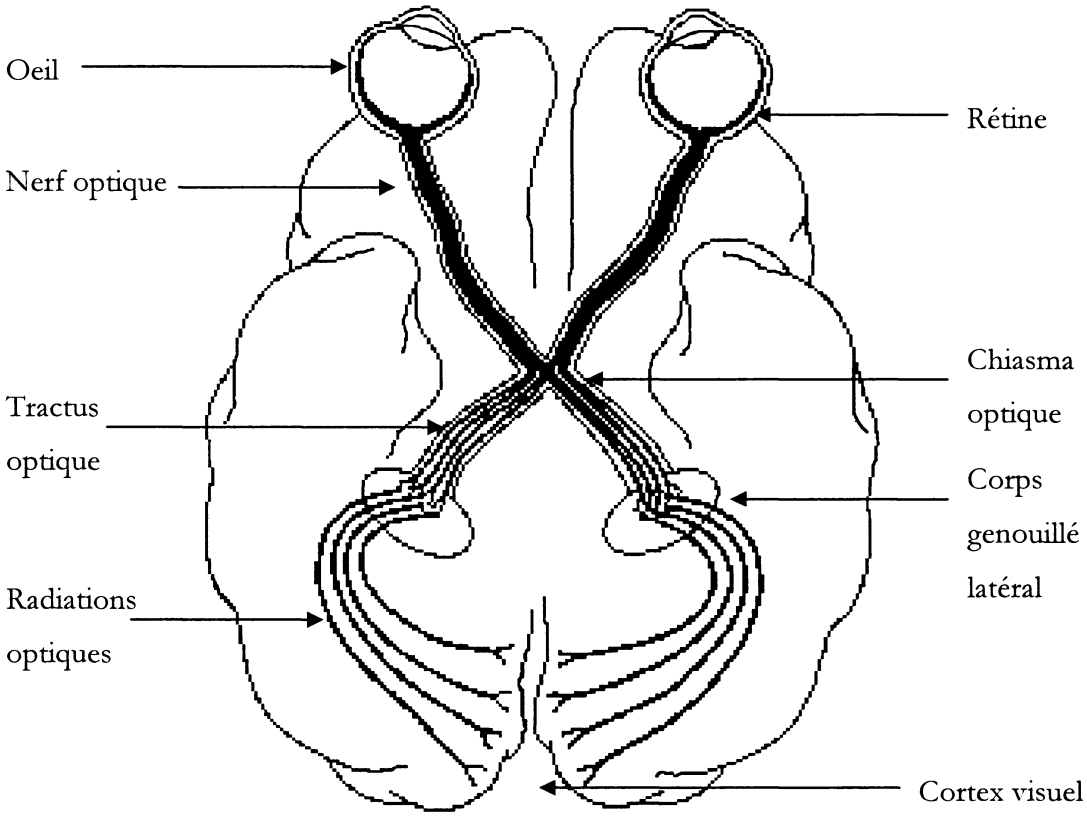


Figure 1.3 – Système visuel humain.

Dès la rétine, les signaux sont inhibés ou exhibés, de manière à faire ressortir les contrastes et les contours [54]. Au niveau cortical, l'information visuelle brute a déjà subi beaucoup d'autres traitements. L'information est encodée de manière très complexe : non seulement l'intensité, mais la couleur, l'orientation, le mouvement relatif, la différence stéréoscopique et

la dominance oculaire sont transportés [67]. Par contre, ces informations restent relativement locales, ce qui fait en sorte qu'il existe encore à cet étage une relation entre la position visuelle d'un stimulus et l'endroit sur la surface du cortex qui le traitera.

Les maladies de la vision les plus répandues en Amérique du Nord sont, dans l'ordre, la dégénérescence maculaire, le glaucome, la rétinopathie diabétique, rétinite pigmentaire et finalement la cataracte [11,59]. La dégénérescence maculaire est un trouble où la macula (centre de la rétine) cesse de répondre à la lumière. Le glaucome est une complication liée à l'humeur aqueuse dans l'œil qui ne s'écoule pas normalement. La rétinopathie diabétique est due au fait que des vaisseaux sanguins formés anormalement éclatent, endommageant la rétine. La cataracte est un problème au niveau du cristallin de l'œil qui s'épaissit, se durcit et devient opaque bloquant le passage de la lumière. Finalement, la rétinite pigmentaire est causée par la dégénérescence de la rétine. La déficience visuelle est quelque fois causée par des lésions au niveau du nerf optique, ce qui entraîne alors des pertes unilatérales de champ visuel ou par des lésions au niveau du chiasma optique ou plus loin, ce qui entraîne généralement des pertes bilatérales [72].

1.2.3 Avenues de recherche pour la récupération partielle de la vision

Dans le domaine de la restauration de la fonction visuelle par stimulation électrique, trois principales avenues ont été envisagées à ce jour en ce qui concerne l'endroit dans le chemin visuel (*visual path*) où la fonction est rétablie : la stimulation de la rétine, du nerf optique et du cortex visuel. Plus la fonction est rétablie tôt dans la chaîne, plus les percepts produits ont des chances de ressembler à la vision normale, étant donné qu'on bénéficie des étages de traitement naturels.

Les techniques de stimulation de la rétine se séparent en plusieurs sous-groupes : implants sous-rétinaux (sous la rétine), epi-rétinaux (sur la rétine), neurotransmetteurs (par injection chimique de glutamate stimulé par fibre optique) et hybrides (culture de neurones dans un MEMS). Cette avenue très populaire est étudiée par plusieurs groupes de recherche, dont les

équipes de Zrenner [38], Schubert et Chow pour les implants sous-rétinaux, Humayun et Liu [49], Rizzo et Wyatt [87] pour les implants epi-rétinaux, Iezzi pour l'implant neurotransmetteur, et Yagi pour l'implant hybride [50].

L'équipe de Veraart [85] étudie la stimulation du nerf optique, beaucoup moins populaire.

La stimulation corticale est quant à elle étudiée par les équipes de Normann de l'Universite du Utah [60], Schmidt du NIH [54], Troyk du Illinois Institute of Technology [55,57], de Dobelle [32], de McAllister du Wayne State University et de Sawan de l'École Polytechnique de Montréal [43].

Le tableau 1.1 résume les principaux avantages et inconvénients des trois principales avenues susmentionnées. Il est pertinent de définir ici l'organisation visuotopique, qui aura une grande importance dans la discussion qui suivra. Il s'agit d'un terme utilisé avec plus ou moins de rigueur dans la littérature, particulièrement en ce qui concerne les implants visuels. C'est un terme large qui concerne l'organisation des champs d'activité sur la surface du cortex, par exemple la symétrie par rapport à l'hémisphère, les frontières, les discontinuités et le degré de rétinotopie [21]. Nous y reviendrons plus tard. Pour l'instant, il faut noter qu'on considère que l'organisation visuotopique est bonne lorsqu'il y a un fort degré de corrélation entre la position des cellules de la rétine avec la position de la surface stimulée (degré de rétinotopie). Les implants rétiniens ont généralement une bonne organisation visuotopique étant donnée la proximité des photorécepteurs avec la surface de stimulation. Le nerf optique possède une mauvaise organisation à cause de la surface stimulée : la proximité des millions de fibres qui le compose rendent la stimulation sélective difficile. Le cortex visuel possède également une faible organisation visuotopique, dans ce cas à cause des multiples étages de traitement qui le séparent des photorécepteurs.

Le projet dans lequel s'inscrit ce mémoire concerne la mise en œuvre d'un système de stimulation du cortex visuel afin de rendre une vue partielle aux aveugles. La prochaine section est consacrée davantage à la stimulation corticale.

Tableau 1.1 – Avantages et inconvénients des différentes approches de recouvrement de la vision.

Approche	Avantages	Inconvénients
Rétine	Bénéficie du traitement rétinien, thalamique et cortical Bonne organisation visuotopique	Nécessite un chemin visuel fonctionnel Accès chirurgical difficile Bloque la rétine de l'irrigation de certains nutriments
Nerf optique	Peu de complications chirurgicales	Accès chirurgical très difficile Nécessite un nerf optique fonctionnel Mauvaise organisation visuotopique Matrice d'électrodes très complexe
Cortex visuel	Approprié pour toutes les maladies de la vision Site de l'implant robuste et protégé par le crâne Accès chirurgical facile Grande densité d'électrodes atteignable	Loin des photorécepteurs Mauvaise organisation visuotopique Encodage complexe des caractéristiques d'image

1.3 Microstimulation intracorticale

1.3.1 Phosphènes et organisation visuotopique

En 1918, Löwenstein et Borchardt rapportèrent que pendant une intervention chirurgicale au cerveau, le lobe occipital gauche du patient fut accidentellement stimulé électriquement, et il perçut un scintillement dans le champ visuel droit. Foerster [36] et Krause [52] reportèrent des phénomènes similaires en 1929. Cette importante découverte mit en évidence le fait que la stimulation électrique du lobe occipital induit des percepts visuels d'une part, et surtout que la position de ces percepts dans le champ visuel du patient dépend de l'endroit sur le cortex qui est stimulé. Le concept de phosphène était né : une sensation visuelle localisée

produite par la stimulation électrique ponctuelle d'un endroit du cortex visuel. Penfield et Jasper [65] stimulèrent électriquement le cortex visuel d'humains qui décrivirent les phosphènes perçus comme « des étoiles, des roues, des disques, des points, des raies, ou des lignes ». En 1972, Brindley [12-15] implanta 80 électrodes sur la surface du lobe occipital d'une patiente de 52 ans. Pendant les 3 années suivantes, il s'employa à cartographier la position des phosphènes dans le champ visuel. Par la suite, Dobelle implanta plusieurs patients avec des électrodes de surface, et les soumit à des tests d'habileté qui se révélèrent fonctionnels mais peu performants [30,31].

Les études précédentes utilisaient des électrodes de surface, avec des courants relativement grands (de l'ordre du milliampère). Au début des années 1980, Barlett et Doty [8] effectuèrent les premiers essais de stimulation intracorticale sur des primates. La stimulation intracorticale consiste à enfoncer des microélectrodes très fines, à la pointe exposée, de quelques millimètres à l'intérieur du cortex. Il en résulte des courants de stimulation de plusieurs ordres de grandeur inférieurs au courant des électrodes de surface pour évoquer des phosphènes (de l'ordre des microampères), et des phosphènes distincts pour une distance d'électrodes au moins 5 fois plus petite.

Les travaux les plus avancés dans la stimulation intracorticale sont ceux de Schmidt et al entre 1990 et 1996 [4,41,42,76]. Ils ont implanté 38 électrodes dans le cortex visuel d'une dame de 42 ans, aveugle depuis 22 ans à la suite d'un glaucome. Cela a permis de prouver que le cortex visuel peut réagir à la stimulation malgré plusieurs années d'inactivité, et que la stimulation sur une grande période de temps est viable car les percepts produits sont restés stables tout au long des 4 mois qu'a duré l'expérience. Ils ont aussi aidé à caractériser l'effet des paramètres d'ondes de stimulation (figure 1.2) sur l'apparence des phosphènes, que la patiente a décrit comme des points blanchâtres ou grisâtres, de la taille d'une pointe d'aiguille à celle d'un 5 cents tenu à bout de bras. D'autres données intéressantes sont rapportées. Des électrodes espacées de 500 µm produisent généralement deux phosphènes distincts alors qu'ils sont confondus lorsqu'elles sont séparées de 250 µm. Ils ont souvent une couleur orangée ou violette lorsque l'amplitude du courant est près du seuil d'apparition. Ils se

déplacent lorsque les yeux bougent. Quelques niveaux distincts de brillance peuvent être observés (entre 5 et 12). D'autres phénomènes ayant une incidence directe sur la conception d'un stimulateur comme l'accommodation à la stimulation et l'interaction entre phosphènes stimulés simultanément sont rapportés dans l'étude, et la première section du chapitre suivant y reviendra.

Un des buts de cette étude était de déterminer si la génération simultanée de plusieurs sites s'intégrait afin de produire des patrons intelligibles. Une caractéristique importante qui a été relevée est que la stimulation simultanée de 3 phosphènes ou plus les fait apparaître coplanaires et d'apparence semblable. Lorsque 6 phosphènes choisis de manière à être alignés de manière verticale ont été générés, la patiente rapporte qu'elle pourrait l'identifier à un I ou à la patte d'une lettre comme un M. Tel est l'état de l'art en matière de reconnaissance de patrons de phosphènes par microstimulation intracorticale.

Il est impensable dans un avenir proche de stimuler précisément un seul neurone à la fois, en imitant parfaitement l'information visuelle provenant des étages précédents. Un phosphène est le résultat de la stimulation simultanée d'un paquet de neurones conçus pour véhiculer des informations très différentes. Il faut envisager la vision par stimulation corticale comme la génération de patrons de phosphènes servant à donner des indices à l'aveugle pour qu'il puisse fonctionner. Les connaissances actuelles dans le domaine de la neurologie et de la microfabrication ne laissent pas croire qu'il sera un jour possible de faire vivre aux aveugles une expérience visuelle similaire à celle que nous vivons.

1.3.2 Systèmes de stimulation visuelle corticale

L'illustration de la figure 1.4 est un élégant schéma de principe d'un stimulateur visuel cortical. Toutes les équipes de recherche travaillant sur l'implant cortical, celle de Normann, Troyk, Schmidt, Dobelle et Sawan [4,31,60,64] suivent ce principe. Une caméra capte les images en face de l'aveugle. Ces images sont traitées et transformées en commandes de stimulation qui sont envoyées à l'implant via un lien inductif, qui envoie également l'énergie

pour alimenter l'implant. Ce dernier, totalement isolé à l'intérieur du crâne, décode les commandes et contrôle ses convertisseurs numérique-analogique (CNA) afin d'envoyer du courant à travers des électrodes micromachinées insérées dans le cortex, conformément aux paramètres d'onde requis. Des détails sur l'implant actuel du laboratoire PolySTIM, de même que sur son protocole de communication, peuvent être trouvées dans [72,17,27].

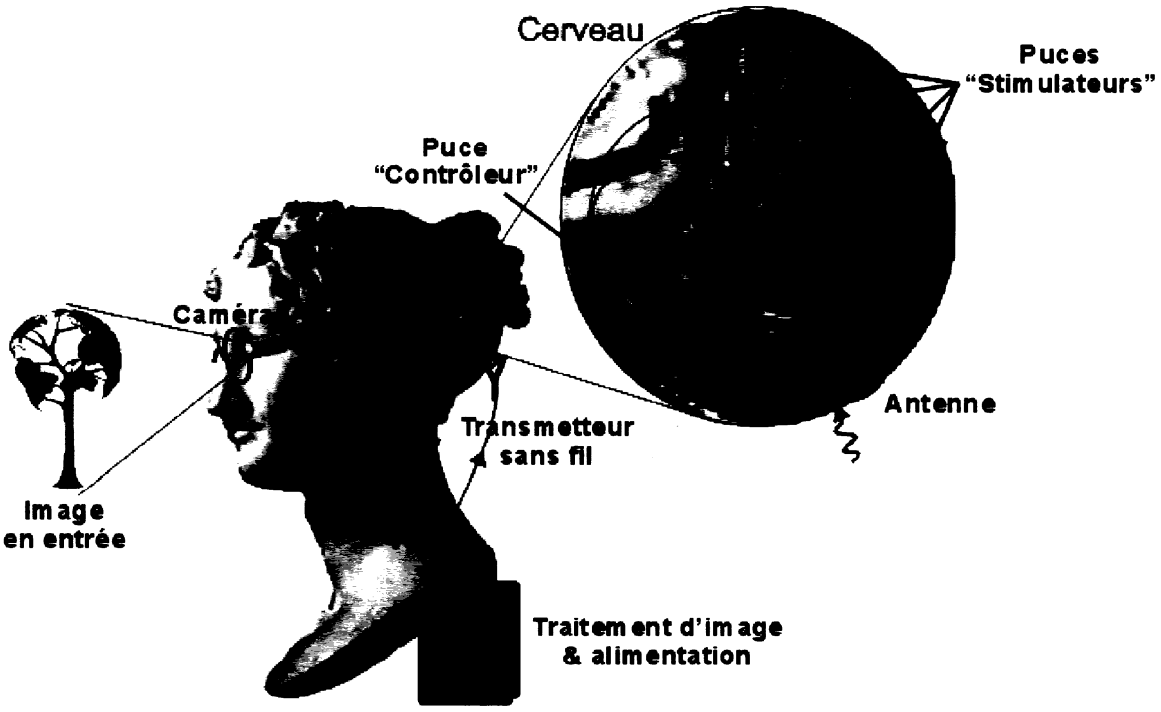


Figure 1.4 – Schéma de principe du système de stimulation visuelle corticale proposé par l'équipe PolySTIM.

1.3.3 Reconnaissance de patrons de phosphènes

Quelques équipes qui développent des systèmes de stimulation corticale se sont déjà penchés sur la faisabilité de tels systèmes au point de vue de la reconnaissance de patrons de faible résolution. Ces études peuvent se regrouper en diverses catégories. La première s'applique à déterminer les conditions minimales concernant les images de faible résolution avec des expériences psychophysiques sur des voyants, en simulant la vision sous forme de phosphènes. Citons les études de Hayes et al [44], et de Cha et al [22]. Cette dernière a fixé à

25x25 la taille minimale d'une matrice d'électrodes permettant d'obtenir une vision fonctionnelle. Une autre catégorie d'études s'applique à prévoir des traitements d'image dédiés à de tels systèmes. Gilmont et al [39] décrivent un algorithme de réduction de résolution pour un implant du nerf optique. Boyle et al [10] amènent le concept de cartes d'importance, basées sur des méthodes de détection des régions d'intérêt dans les images. Harvey et al [43] prévoient une égalisation d'histogramme et une réduction de résolution qui prend pour acquis que la brillance sera interprétée comme des tons de gris. Les résultats de ces travaux seront commentés au chapitre suivant. La troisième catégorie concerne les études physiologiques sur les champs réceptifs (*receptive fields*), et sont présentés dans la prochaine section.

1.3.4 Distribution spatiale des phosphènes

La position des phosphènes qu'il sera possible de générer par stimulation électrique dans le cortex dépend de la position des électrodes. À l'inverse, un champ réceptif est la mesure d'activité localisée dans le cortex, en réponse à un stimulus visuel. Les importants travaux de Hubel et Wiesel ont permis de comprendre davantage le fonctionnement du cortex visuel et des champs réceptifs. Au cours d'une expérience sur les champs réceptifs avec un chat en 1959 [46,47] et une autre avec un macaque en 1968 [48], ils trouvèrent que les cellules du cortex fonctionnent comme des interrupteurs, de la même manière que les cellules de la rétine, mais de manière relativement plus complexe, étant donné la plus grande quantité d'informations transportée. Plusieurs autres études ont été entreprises pour tenter de déterminer la position des champs réceptifs dans le cortex en fonction de la position des stimuli visuels dans la rétine [1,33,66,81,82].

Schwartz [77] a tenté de trouver une relation entre la position corticale et la position rétinienne grâce à une loi logarithmique [1,33,81,82]. Il en existe plusieurs variantes, la plus simple pouvant s'exprimer ainsi :

$$w = \ln(z + a) \tag{1.1}$$

avec w et z des nombres complexes où w représente les coordonnées corticales horizontales et verticales (en mm) et z les coordonnées rétiniennes (en degrés), et a est une constante réelle.

Cependant les recherches récentes de Warren et al, ainsi que de Maynard et al [57,86] ont montré que les caractéristiques des champs réceptifs dans une même colonne de neurones varient considérablement, ce qui les amène à affirmer qu'il n'existe pas de relation conforme (*conformal*) entre les positions corticales et rétiniennes, c'est-à-dire que la relation n'est pas aussi directe que l'entendait Schwartz. Cela signifie que la position des phosphènes devra être déterminée manuellement pour chaque patient.

En prenant l'hypothèse que l'organisation des phosphènes ressemble à l'organisation visuotopique, il est possible de concevoir le genre de carte de phosphènes qui pourront être évoqués suite à l'insertion d'un implant. Pour l'étude des traitements d'image pour un stimulateur visuel cortical, nous définissons une banque de données contenant les phosphènes disponibles et leurs caractéristiques, que nous appelons carte visuotopique.

1.4 Conclusion

Un stimulateur visuel cortical utilise la stimulation physiologique afin de rendre une vision partielle aux aveugles. Le traitement cortical de l'information visuelle est toutefois très complexe. Il n'est pas envisageable d'imiter le traitement normal fait au cours du chemin visuel en amont, d'autant plus qu'une électrode, aussi petite soit elle, stimule toujours plusieurs neurones à la fois, sans égard à leur fonction. Cette stimulation produit un phosphène, la perception d'un point lumineux localisé. L'information visuelle sera fournie à l'aveugle en créant des patrons de phosphènes représentant le monde d'une manière simplifiée. Ces patrons seront générés à partir de la carte visuotopique, définie à la fin de ce chapitre. Le choix des techniques de transformation de l'image en patron simple dépend de plusieurs facteurs qui ne sont pas encore élucidés. Le chapitre suivant donne un aperçu des diverses approches qui sont explorées à la lumière des connaissances actuelles.

CHAPITRE 2

STRATÉGIES DE TRAITEMENT D'IMAGE

2.1 Introduction

Un survol des recherches ayant trait à l'évocation de percepts lumineux par stimulation corticale a été fait au chapitre précédent. Il en ressort certains points importants :

1) La stimulation visuelle corticale est encore soumise à beaucoup d'incertitudes liées aux mécanismes physiologiques.
2) Il est peu envisageable pour l'instant de pouvoir stimuler les neurones du cortex de la manière que les étages de neurones en amont; en d'autres termes, cette technique ne permettra pas de simuler une sensation de vision comme nous la vivons, mais plutôt d'une manière beaucoup plus grossière.
3) Le percept stimulable élémentaire est le phosphène.
4) On considère l'information visuelle pertinente à envoyer à l'aveugle en termes de patrons de phosphènes, représentant des formes qui seront reconnaissables.
5) Les travaux de pointe dans la reconnaissance de patrons de phosphènes sont encore à leurs tout débuts, par contre les résultats obtenus jusqu'à présent sont encourageants.

Le but poursuivi ici est d'améliorer les aptitudes d'un aveugle à se déplacer. Cela pourrait être fait de plusieurs manières mais pour l'instant, cela sera envisagé en créant des patrons de phosphènes. C'est ce qu'on appelle en anglais une *bitmapped approach*. La prochaine étape consiste à déterminer la façon dont les images vues par la caméra devront être transformées pour générer ces patrons de phosphènes.

Le corps du présent chapitre est constitué d'un article, soumis au journal « Artificial Organs ». Il introduit le sujet avec une brève revue des faits présentés au chapitre précédent. De là sont dégagées les caractéristiques générales des traitements d'image appliqués à la

stimulation visuelle corticale. Les travaux effectuées à ce sujet par d'autres équipes de recherche envisagent le problème avec plus ou moins d'optimisme. Étant donné la grande possibilité de résultats possibles, il est apparu naturel de concevoir plusieurs stratégies de traitement d'image qui seront toutes implémentées dans le même système. Ces stratégies sont présentées et commentées dans cet article, en ordre décroissant d'optimisme.

2.2 « Multiple Image Processing Strategies Dedicated to Visual Cortical Stimulators: A Survey »

L.X. Buffoni, J. Coulombe, M. Sawan

PolySTIM Neurotechnologies Laboratory

Departement of Electrical Engineering, Ecole Polytechnique de Montreal

ABSTRACT

While multi-electrode devices are constantly evolving towards a state where complexity and reliability are adequate for providing a breakthrough in visual cortical stimulation that would allow the blind to recover partial vision, few research teams have yet focused on the development of the front-end subsystem that transforms an input image from a camera into stimulation commands. This paper gathers state-of-the-art knowledge about the appearance and organization of phosphenes, and previous work in image processing dedicated to visual cortical stimulation. Observations and hypothesis about important issues are pointed out, and six image processing strategies using a bitmapped approach are presented, from the most optimistic that use brightness modulation to emulate grayscale to the most conservative that use only on/off phosphene evocation. The last strategy uses an image segmentation algorithm along with distance knowledge, acquired from an hypothetic range finder. The strategies have been implemented both in a software model and in an embedded system. Experimental results demonstrating the timing characteristics of this system are presented to show its real-time capabilities.

KEYWORDS: Phosphene, Phosphene pattern recognition, Visual cortical stimulation, Intracortical microelectrodes, Scene representation, Image processing, Resolution reduction, Image segmentation.

I. INTRODUCTION

Since the early discovery that visual sensations can be evoked using direct electrical stimulation on the top of the visual cortex [12], researchers have been working to build a visual cortical stimulator (VCS) that will help blinds recover partial vision. Studies in interrelated topics have been conducted in order to develop such VCSs. They can be classified in the following categories: study of physiological responses to electrostimulation, development of microelectrodes, implementation of low-power microchips to be used in implanted devices, and evaluation of low resolution image recognition with sighted volunteers using simulated prosthetic vision.

We are developing a complete system for intracortical stimulation that will be used for *in vivo* testing (figure 2.1). A camera grabs images that are processed and transformed into stimulation commands to be sent to the implant via an inductive link. The implanted device allows multiple stimulation strategies [27]. Testing on living subjects will follow a bottom-up validation scheme: first, the stimulator itself needs to be tested alone in order to obtain reliable phosphene evocation, and then can be truly tested the front-end subsystem, comprising the input devices and the subsequent transformation into stimulation commands.

The purpose of this subsystem is to represent scenes of the real world into meaningful patterns of phosphenes. Yet, few significant advances have been made in the field of phosphene pattern recognition. It is often conceived that grayscale images can be directly generated in a patient's mind by evoking phosphenes which brightness match the corresponding image pixels, but results from biomedical studies do not share such optimism [76]. Therefore, several image processing strategies have to be considered, since *in vivo* testing is required to validate phosphene evocation and vision restoration. The strategies presented

in this paper are inspired by the description of phosphenes given by human volunteers in previous studies. Reliable stimulation techniques and a large number of implanted electrodes are assumed. Image processing is hereby envisioned as an aid for mobility, and a normal video camera is used as an input device.

A physiological description of the phosphenes and their organization will be given in section II to illustrate the phosphene pattern displaying issues. Previous work made by VCS research teams in the field of image processing is presented in section III, and some general characteristics of image processing for this application, derived from physiological constraints, are also reported. Examples of image processing strategies that could be used in a functional VCS are presented in Section IV, in an idealistic-to-realistic manner. From these strategies, segmentation of the image into homogeneous regions and distance knowledge will appear to be very important. All these strategies have been implemented in both software and in an embedded system, and some results are presented in section V.

II. PHOSPHENE APPEARANCE AND ORGANIZATION

Many studies have shown that spatial visual percepts are produced by electrical stimulation of the visual cortex [12,65,30]. Early experiments with humans revealed that these localized percepts, commonly called phosphenes, were described as stars, wheels, discs, spots, streaks or lines [12]. All of these were evoked by injecting current through electrodes on the pia-arachnoid surface. Dobelle implanted blind patients with arrays of surface electrodes, and reported that they could use it as a reading or displacement aid with fair success, although they are not yet as efficient as common methods used by the blind, such as Braille or trained dog for reading and walking [30,31]. Yet, little is known about recognition of complex patterns.

While more invasive, intracortical stimulation later proved to have many advantages over surface stimulation: much lower stimulation current had to be used to evoke phosphenes, but more importantly, their diameter was found to be smaller and their spatial density greater

[4, 41]. Schmidt *et al.* [76] implanted 38 electrodes in a 42-year-old woman who had been totally blind for 22 years secondary to glaucoma. Apart from demonstrating that the visual cortex of the subject remained responsive even after being blind for a long period of time, it brought interesting results on phosphenes' appearance and spatial organization.

They were described by the patient as small spots of light, with a size varying from a pin-point to a nickel at arm length. The stimuli were trains of biphasic current pulses with common parameters such as current amplitude (I), frequency (F), pulse duration (PD), inter-phase interval (IPI), train length (TL) and inter-train interval (ITI). At amplitudes near threshold, they are said to have colors, while at higher levels they become white, grayish or yellowish. Their diameter usually decreases at higher levels, but increases as TL is increased. They do not flicker when ITI is below 25 ms. Perceived phosphene brightness is influenced by a seemingly complex combination of I, PD, TL and F.

The study reported a fairly good relationship between the phosphene positions, determined with two methods, and the placement of microelectrodes on the cortex. However, it appears that the phosphene map cannot be computed precisely from the positions of the microelectrodes. This fact is confirmed by Warren *et al.* [86]. Their investigations on the receptive field positions in cats' striate cortex conclude that the relationship between cortical and visual space is not conformal as Schwartz' complex logarithmic transform [77]. Consequently, in the case of a visual prosthesis, the exact locations of phosphenes in the visual field will have to be manually determined after surgery for every patient.

Schmidt *et al.* [76] also reported that microelectrodes spaced 500 μm apart generate separate phosphenes, while microelectrodes spaced 250 μm apart typically do not. Moreover, interactions between pairs of phosphenes occur, and they will have to be dealt with in a functional visual stimulator.

Phosphenes have an apparent distance from the patient when they are generated alone. However, when three or more phosphenes are generated simultaneously, they appear

equidistant. This property, along with the fact that the phosphenes looked alike, made the subject state that phosphene patterns could be interpreted. When six phosphenes were simultaneously generated, closely spaced in a vertical orientation, the patient said it would be adequate to represent the letter "I" or the branch of a letter such as "M" [76].

A lot of processing of the retinal image is done prior to reaching the cortical level. Also, it has been shown that the primary areas of the visual cortex encode much more than just local intensity, but rather carry various information such as orientation, color, edges, texture and motion [51]. Furthermore, signal modultation occurs in primary visual cortex resulting from high-level processing [53]. Current technology clearly does not allow emulation of the natural visual system at a neuronal level. Neuroscientific knowledge is not sufficient, electrodes and stimulation techniques often excite many neurons at a time, achievable stimulation sites density is far from being sufficient for addressing every neuron, and modulation from feedback would require high complexity compared with today's stimulator capabilities.

Nevertheless, a bitmapped approach for phosphene patterns generation is still the only method that can currently be considered. This approach is commonly conceived as the displaying of phosphenes as image pixels, with levels of gray created by brightness modulation. However, this may be impossible to achieve and more minimalistic strategies should be available. Thus, multiple image processing strategies have been imagined and are presented in this paper. It is believed that even with the least optimistic strategy, the visual information provided to a blind will, with a bit of training, turn out to be a useful aid for mobility.

III. IMAGE PROCESSING FOR VISUAL STIMULATORS

A. Efforts toward vision recovery by electrical stimulation

Despite the lack of knowledge about phosphene pattern recognition, some research groups have already tried to evaluate the feasibility of artificial vision from an image processing point of view.

Cha *et al.* [22] used sighted volunteers in order to establish design specification for an electrode array. They created a phosphene simulator with a video camera, lenses, and a perforated mask. Visual acuity was evaluated from the volunteers' ability to perform reading and walking tasks. They found out that an array of 25x25 pixels would be sufficient to obtain useful visual information. However, this study makes the assumption that the phosphene map would be square and regular, with infinite levels of a monochromatic image. Hayes *et al.* [44] recently ran a similar study in which they showed that simple objects and symbols could be identified, even with a very small array.

In their investigation of the receptive fields in cat's visual cortex caused by visual stimuli, Warren *et al.* [86] bring additional data about the use of cortical stimulation in order to build a visual prosthesis. They found out that the relationship between cortical and visual coordinates was not conformal. This fact has also been confirmed by Maynard *et al.* [57]. By assuming that a phosphene has the same size as a receptive field, which is about 0.9 degrees, they deduced that a "display" of 625 distinct phosphenes could be obtained in an area of 19x19 degrees with an array of 25x25 electrodes, spaced 1.2 mm apart.

Harvey *et al.* [43] suggest histogram equalization as a mean to accentuate contrast. Image processing has to be suitable in most lighting situations, but this approach still takes for granted that multiple levels of grey can be precisely emulated by brightness modulation. Gilmont *et al.* [39] developed a new and efficient algorithm for resolution reduction by segmentation based on growth of zones, in the context of an optic nerve visual prosthesis. The segmented image is then fitted to a multi-sized grid, in order to reduce bandwidth. In

cortical visual stimulation, the grid is actually the phosphene map. Again, this procedure assumes infinite levels of grey.

Boyle *et al.* [10] emphasis on region of interest (ROI) detection. Locations in the image are assigned weigths according to their importance. These weights will be used to display only the most important features of the input image.

Recently, Troyk *et al.* [83] implanted a trained macaque with 152 microelectrodes and presented a model for the research on intracortical prosthesis, in which they challenged the traditional bitmapped approach of phosphene patterns evocation. They provided a realistic discussion that inspired the need for an image processing system that allows multiple strategies.

B. Specific image processing constraints

It is obvious that image data have to be transformed before being used to stimulate the cortex, even if a bitmapped approach is used. The following physiological constraints have to be considered:

- Low resolution: more electrodes give a better precision, but it is practically impossible to match the precision of a pixelized digital image even with a great number of electrodes. Phosphene size is not negligible.
- A narrow area of the visual field can effectively be stimulated, leading to tunneled vision.
- The display is not uniform: a quick look at the phosphene maps reported by [12,30,76,,84] shows that the available phosphenes are not necessarily distinct and disposed in a regular fashion.
- Reduced number of gray levels: Schmidt *et al.* report that at most 5 to 12 levels of brightness can be identified, and this number will likely be even lower in practice because of accommodation issues. Another important issue is that dim phosphenes are often overshadowed by bright phosphenes, and not perceived [76]. Finally, even

if the brightness level was perfectly controlled, no study has yet proven that it would be perceived as a level of grey in that stage of biological vision processing [83].

- There is a significant phosphene latency and minimum stimulation duration for a phosphene to be perceived. Consequently, the "frame rate", or phosphene pattern refresh rate, would be quite low [76].

Most recent advances in image processing are about the implementation of well-known mathematical operators for particular applications. Here are some characteristics that distinguish this application from most industrial image processing applications.

- In order to be used as a walking aid, image processing has to be suited for any kind of images: indoor/outdoor, day/night, objects/rooms/countryside/text.
- There is no or very little *a priori* knowledge.
- There is a need for efficient resolution reduction because of the small set of phosphenes available for representation.
- The system has to be aware of the phosphene map, unique for a given patient.
- Maximum control has to be given to the user. Efficient use of the stimulator will undoubtedly depend on learning, so it needs to be proactive. One interesting fact that can be exploited is that the user decides where he looks, unlike most industrial image processing applications.
- User-controlled zoom is an important feature.
- The highest level of image processing, leading to image understanding, is done by the patient's brain.

C. Cortical visual stimulator system

Our team is currently designing a VCS system that will be used for *in vivo* testing, presenting common principle with recently published designs [60,83]. This VCS includes an input device which is mounted on the subject's glasses and grabs information of the surrounding world. The input signal is then processed and transformed into implant commands. These commands are transmitted to the stimulating implant located inside the cranium, on the

visual cortex, via an inductive link. At this time, the preferred input device is a camera, but other types of devices could turn out to be more appropriate.

We describe in this paper the image processing system (IPS) which represents the link between acquired image data and the implant commands, whose communication protocol is described in [27]. Pixels are entered into the IPS, and the output is the set of stimulation site addresses (SSAs) to be activated for each frame. An SSA contains the identification code of a particular electrode, but also its associated pulse parameters. To be able to adequately choose which SSAs will be used, the IPS contains the corresponding phosphene visual field coordinates (PVFC), that would be determined manually for each patient. The main issue is to choose the appropriate PVFCs and their associated intensities for a given image.

In this paper, "scene representation" is used to designate the way in which the elements of an input image are symbolized into phosphene patterns, in order to make them identifiable by the blind. For example, displaying the object edges or displaying light and shade are two different scene representations, based on different hypothesis related to physiological mechanisms. An "image processing strategy" concerns a set of image processing operations that lead to a given scene representation.

The IPS under development will offer multiple image processing strategies because little is known about phosphene pattern recognition. On the other hand, it is mandatory not to constraint the system into pessimistic scene representation capabilities. It is hoped that this flexible implementation will be helpful in determining the best solution when experiments with human volunteers will be conducted. Next section discusses about these strategies, in an idealistic-to-realistic order.

IV. IMAGE PROCESSING STRATEGIES

In order to visualize in an accurate way the phosphene patterns resulting from image processing, an hypothetical representation in the visual field will be used. Figure 2.2 shows

an example of a randomly generated phosphenes map using white dots. This representation does not consider phosphenes interaction, and allows 256 levels of grey resulting of fully controlled brightness modulation by pulse parameters. Also, phosphene appearance is invariant. Phosphenes may overlap, which could be the result of two closely spaced electrodes (between 250 and 500 μm). The map of figure 2.2 contains 1500 phosphenes, but this number is irrelevant: the aim here is not to evaluate the quality of an image according to the number of phosphenes it contains.

Patterns are created following a bitmapped approach. A relationship is computed between PVFCs and image pixels. Processed images are then geometrically transformed into phosphene coordinates by scanning the available phosphenes and assigning them the intensity of the related pixel. Patterns represented on that kind of displays provide the most optimistic result that can be obtained for a given image processing.

A. Basic resolution reduction

Early research activities, dedicated to recover vision using cortical stimulation, thought of representing a scene by directly mapping the acquired image (figure 2.3a) in a lower resolution by mean of neighboring pixels (figure 2.3b). The emulation of the phosphene pattern (figure 2.3c) given by geometric transformation of the processed image of figure 2.3b over the phosphene map of figure 2.2 gives a nice result, but this strategy is clearly unrealistic for the following reasons:

- phosphenes are not square pixels disposed in an uniform matrix;
- a limited (if any) number of levels of grey may be generated with sufficient precision;
- neighboring phosphenes interaction (like brightness dimming) and other phenomena like accommodation make grayscale control even more difficult.

B. Enhanced resolution reduction

In order to enhance the resolution reduction quality, Gilmont *et al.* proposed a segmentation algorithm by growth of zones of same characteristics (figure 2.3d) and fitting of resulting images on a multi-scale grid [39]. In figure 2.3e, the processed image is transformed into a phosphene display instead of such a grid. Once applied to cortical stimulation, this strategy has as unrealistic features as the previous one.

C. Resolution reduction and edge highlighting

Marr reports that edge contrast is enhanced in the early stages of biological vision processing [54]. In this perspective, overlaying an edge image over a segmented one could improve contour and therefore give an even better perceptual result. This could have the benefit of efficiently using the dimming property of nearby phosphenes (figures 2.3f and 2.3g).

Every image processing strategy presented above made use of brightness modulation in order to emulate grayscale. Successful interpretation of grayscale created by phosphene brightness has not yet been proven. Therefore the next strategies are based on more conservative on/off phosphene evocation.

D. Threshold image

The simplest strategy that solely uses on/off phosphene evocation consists of setting the image pixels as black or white, whether they are above or below a defined threshold, as shown in figures 2.3h and 2.3i. The threshold value can be determined by user control or by common histogram analysis. Of course, a dark object in front of a white wall would be represented with most of the phosphenes evocated at full intensity. Another user control could allow displaying of inverted image threshold, or this can be done automatically by comparing the percentage of activated phosphenes. This technique does not allow removal of large, undesired regions, like a light background. This drawback could hide pertinent

visual information, and simultaneous evocation of a large amount of phosphene might be uncomfortable for the user.

E. Edge image

Dobelle *et al.* represented a scene by its image edges [31]. An edge image can be computed by applying a Sobel or Laplace operator, then filtered by an user-controlled threshold to remove edges that are not significant. This operation is followed by a geometric transformation algorithm to recreate the lines with corresponding phosphenes. This strategy, illustrated by figures 2.3j and 2.3k, can limit over-stimulation while carrying most useful scene information. However, while colinear phosphenes may look like an "I", complex curved lines might be confusing, especially at low resolution, or if great gaps exist between phosphenes.

F. Image segmentation and region selection

This strategy constitutes a fair enhancement to a threshold image. It consists of displaying selected regions from a segmented image as lit patches. Segmentation is the action of splitting an image into regions that have a strong correlation with objects or areas of the scene represented by the image. In most image processing applications, it is the first step toward image understanding. The natural vision mechanism can be considered as a 2-stage structure: the first is the low-level processing used to extract or enhance features, and the second is the high-level interpretation [54]. In the case of cortical stimulation, the device will naturally be responsible of the first stage, and leave the second stage to the blind's brain. However, the information carried at the cortical level is far more complex than simple image low-level processing, and it appears that it will only be possible to "connect" to the biological dataflow in an unnatural way, by displaying crude bitmapped phosphene patterns. The needed level of processing depends, among other things, on the number of electrodes and on our ability to control the percepts provoked by stimulation.

A particular attention has been paid on the development of an adapted segmentation algorithm because it is a basis for higher level processing. It can be used as an efficient resolution reduction method, as well as the first step for ROI display. The segmentation results in a structured representation of the image, allowing further higher-level processing. For example, it allows to eliminate useless regions, like a large wall that would occupies the field of view. Region selection should aim for intelligibility, predictability, patient comfort and user-control. The following considerations are derived from these four characteristics:

- Only a reasonable amount of phosphenes will be lit at the same time.
- Small regions may toggle from a frame to another, but large regions must be handled carefully.
- Displaying edgy parts of an image, that are described by Boyle *et al.* to be rich in information content [10], is not beneficial because of the low phosphene resolution. The user could decide whether they should be displayed or not. An edgy region will require the blind to zoom or to get closer.
- Object identification is only possible if the region filling is not excessive.
- The fact that the patient decides where he looks is an important proactive feature that would be efficiently exploited if the region selection was dependent of the characteristics of the central region. However, phosphenes in the central region should not always be activated, for comfort.

From these observations, it appears that a good criterion for choosing the regions to be displayed is distance. A real-time matrix-like range finder has to be used in conjunction with the camera, with its data corresponding to regions of the image. Then, the closest segmented regions would be lit, or those that are in a certain range of the central region. This last strategy would make good use of the blind's ability to control the direction in which he is looking.

Of course, since the phosphene organization is irregular, the same problem as edge displaying arises, because objects will not be easily recognizable by their region contours, as figures 2.3l to 2.3o show. With this approach, comprehension of the scene does not rely

solely on a still image, but rather on a sequence of images. If the aforementioned rules are followed, scanning of the scene will make patches and holes seen with different points of view, and it is likely that a blind will learn to benefit from this representation.

V. RESULTS

By taking a look at the simulated phosphene representation, it is clear that the best results are obtained from the strategies that use grayscale. However, they are inappropriate if phosphenes' brightness is not well controlled. In the case of the scene representation based on the image edges, it is obvious that the resulting quality is strongly dependent on the phosphene density in the visual space. For low resolution phosphene displays, a threshold image or a selected region image seem to be the best strategies. While image threshold is the simplest, it leaves unwanted details that have a negative effect over image intelligibility. On the other hand, selected regions of a segmented image from distance data reduce the scene to a very simplistic scene representation. It should be the most appropriate for early VCS prototypes. It is also the best way to integrate distance data with image data, distance being of utmost importance to understand a scene morphology and for mobility. By using discriminating criterion based on the regions' distance relative to the central region, a patient could fully understand a scene by scanning it. However, it demands a robust, stable segmentation algorithm in order to keep the regions constant. Also, permanent evocation of the phosphenes located in the center of the visual field could be an issue.

All these strategies have first been implemented in software for subjective evaluation, from which pictures of figure 2.3 were taken. Then, the algorithms have been ported to an embedded system. The phospenes map and its associated pixels-to-PVFC relationship is loaded upon configuration. A camera grabs an image that is processed according to the selected strategy. Then a geometric transformation of the processed image pixels into PVFCs is executed. A subsequent module will transform PVFCs into SSAs and send them to the stimulator according to the protocol specified in [27]. Table 2.1 contains computation time in the aforementionned implementation for the elementary operations that are used in

the strategies. They are classified as low-level and high-level operations, the former resulting from pixel neighborhood processing, while the latter concerns more complex data structures. Table 2.2 is a summary of these strategies, along with their constituting elementary operations and their total computation time. They have been obtained from simulation using a 100 MHz ARM processor model, along with a custom-made coprocessor. Further details can be found in [16]. An interesting fact to point out is that the different strategies use more or less the same elementary operations. Note that every strategy ends up with a geometric transformation, to convert the processed image into PVFCs. Segmentation has the longest execution time, but it is still fast enough for our needs (recall that phosphene latency and minimum stimulation duration will allow a "frame rate" well below video rate).

VI. CONCLUSION

A review of phosphenes' appearance, organization by electrical stimulation of the cortex has been presented. The physiological phenomena related to their evocation in the context of providing useful visual information to a blind have been pointed out. From these observations, various scene representation are taken into consideration. These representation spawned six different image processing strategies. They all rely on a bitmapped approach in phosphene pattern generation, but range from optimistic brightness modulated phosphene based to coarse on/off phosphene based scene representations that only give hints about the surrounding environment. However, the latter strategies are still believed to be meaningful when considered as a sequence of images describing the same scene. These various strategies have been implemented in both a software model and a portable embedded system that will eventually be used for *in vivo* testing. Such a system is resource limited in terms of area, power and frequency, therefore the algorithms have been implemented accordingly. Execution time results have been provided to demonstrate its real-time capabilities. Once more results from physiological studies on basic pattern recognition will be available, the least effective strategies will be dropped while the most effective ones will be enhanced.

ACKNOWLEDGEMENTS

The authors thank the Canadian Microelectronics Corporation for technical support, and NSERC and NATEQ for financial support.

REFERENCES

[1] Brindley GS., Lewin WS. "The Sensations Produced by Electrical Stimulation of the Visual Cortex," *Journal of Physiology*, 1968; 196:479-493.

[2] Coulombe J., Buffoni LX., Sawan M. "A Mixed-Signal IC for Multiple Cortical Stimulation Strategies", *Circuits and Systems, MWSCAS-2002*, 2002; 2(II):250-253.

[3] Schmidt EM., Bak MJ., Hambrecht FT., Kufta CV., O'Rourke DK., Vallabhanath, P. "Feasibility of a Visual Prosthesis for the Blind Based on Intracortical Microstimulation of the Visual Cortex", *Brain* 1996; 119:507-522.

[4] Penfield W., Jasper H. "Epilepsy and the Functional Anatomy of the Human Brain", *London:Churchill*, 1954; 116(26),404-406.

[5] Dobelle WH., Mladejovsky MG., Girvin JP. "Artificial vision for the blind: electrical stimulation of visual cortex offers hope for a functional prosthesis", *Science* 1974; 183:440-444.

[6] Dobelle WH., Mladejovsky MG., Evans JR., Roberts TS., Girvin JP. "Braille reading by a blind volunteer by visual cortex stimulation", *Nature* 1976; 259:111-112.

[7] Bak M., Girvin JP., Hambrecht FT., Kuft CV., Loeb GE., Schmidt EM. "Visual Sensations Produced by Intracortical Microstimulation of human Occipital Cortex", *Med Biol Eng Comput*, 1990; 28:257-259.

[8] Hambrecht T., Bak M., Kufta CV., O'Rourke DK., Schmidt EM., Vallabhanath, P. "Feasibility of a Visual Prosthesis for the Blind Utilizing Intracortical Microstimulation," *4th Vienna International Workshop on Functional Electrostimulation*, 1992; 1-8.

[9] Warren DJ., Fernandez E., Normann RA. "High-Resolution Two-Dimensional Spatial Mapping of Cat Striate Cortex Using a 100-Microelectrode Array," *Neuroscience*, 2001; 105(1):19-31.

[10] Schwartz, EL. "Computational Anatomy and Functional Architecture of Striate Cortex: A Spatial Mapping Approach to Perceptual Coding," *Vision Res.*, 1980; 20:645-669.

[11] Kandel ER, Schwartz JH, Jessel TM. *Principles of Neuroscience*, McGraw Hill, New York, 4th ed., 2000.

[12] Lamme VAF., Supèr H., Landman, R., Roelfsema, PR., Spekreijse, H., "The Role of Primary Visual Cortex (V1) in Visual Awareness", Vision Research, 2000; 40:1507-1521.

[13] Cha K., Horch KW., Normann RA. "Simulation of a phosphene field based visual prosthesis", *IEEE International Conference on Systems, Man and Cybernetics*, 1990; 921-923.

[14] Hayes JS., Yin VT., Piyathaisere D., Weiland JD., Humayun MS., Dagnelie G. "Visually Guided Performance of Simple Tasks Using Simulated Prosthetic Vision", *Artificial Organs*, 2003; 27(11):1016-1028.

[15] Maynard E. "Visual Prostheses", *Annual Review of Biomedical Engineering*, 2001; 3:145-168.

[16] Harvey JF., Roy M., Sawan M. "Visual Cortex Stimulator Prototype Based on Mixed-Signal Technology Devices", *IFESS*, 1999.

[17] Gilmont T., Verians X., Legat JD., Veraart Cl. "Resolution Reduction by Growth of Zones for Visual Prosthesis", *ICIP*, 1997; A:209-302.

[18] Boyle J., Maeder A., Boles W. "Inherent Visual Information for Low Quality Image Presentation", *APRS Workshop on Digital Image Computing*, 2003; 51-57.

[19] Troyk P., Bak M., Berg J., Bradley D., Cogan S., Erickson R., Kufta C., McCreery D., Schmidt E., Towle V. "A Model for Intracortical Visual Prosthesis Research", *Artificial Organs*, 2003; 27(11):1005-1015.

[20] Troyk, P.R. "Multi-Channel Transcutaneous Cortical Stimulation System," *NIH Final Report (Contract #N01-NS-7-2365)*, 2001.

[21] Normann R. "A Neural Interface for a Cortical Vision Prosthesis", *Vision Research*, 1999; 39:2577-2587.

[22] Marr D. *Vision*, W.H. Freeman&Co, New York, 1982.

[23] Buffoni LX., Sawan S., Coulombe J. "Design and Test of a Novel Image Processor Dedicated to Cortical Visual Stimulation", submitted to *Med. Biol. Eng. Comp.* 2004.

LIST OF FIGURES AND TABLES

FIGURES

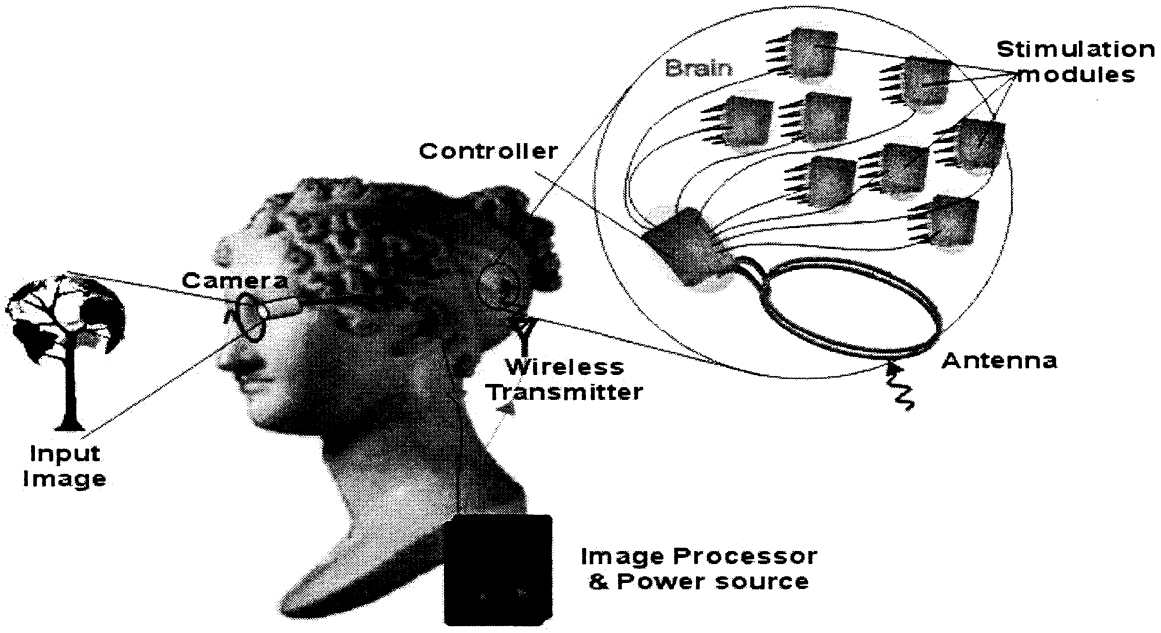


Fig. 2.1 – Complete visual cortical stimulation system. The image processing system is integrated in the external controller.

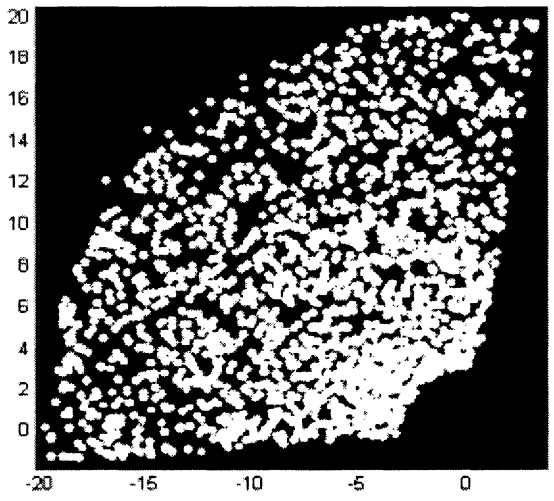
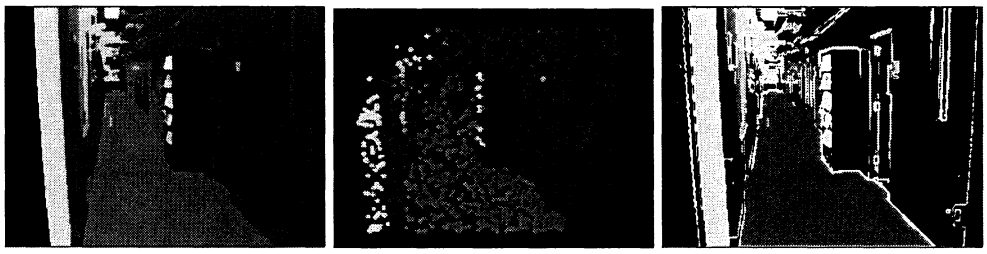


Fig. 2.2 – Artificial phosphene map: 1500 phosphenes, disposed between 10° and 185° of lateral angle, and from 3° to 20° of eccentricity. Phosphene diameter = 0.6°.

(a) (b) (c)



(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

(m) (n) (o)

Fig. 2.3 – Various image processing and representation strategies: a) original image, b) resolution reduction by mean of neighboring pixels, c) phosphene representation of (b), d) resolution reduction by growth of zones, e) phosphene representation of (d), f) resolution reduction with edges enhanced, g) phosphene representation of (f), h) image threshold, i) phosphene representation of (h), j) edge image from a Sobel filter and hysteresis filtering, k) phosphene representation of (j), l) segmented image, m) phosphene representation of (l), n) segmented region selection based on distance, o) phosphene representation of (n).

Table 2.1 – Execution time results for elementary operations used by the image processing strategies. Classification is done according to their complexity.

Operation	Code	Computation time (ms)
Low-level processing		
Resolution reduction	RR	0.079
Edge magnitude	EM	1.508
Threshold	T	0.615
High-level processing		
Geometric Transform (1000 phosphenes)	GT	0.54
Growth of zones	GOZ	93.61
High-level segmentation	S	128.27
Region selection (out of 30 remaining regions)	RS	0.06

Table 2.2 –Total execution time for the image processing strategies in embedded system. Constituting elementary operations are indicated with codes defined in table 2.1. Image size is 320x240 pixels.

Strategy	Operations	Computation time (ms)
Resolution reduction by mean of neighbor pixels	RR+GT	0.619
Resolution reduction by growth of zones	GOZ+GT	94.15
Resolution reduction and edge enhancement	EM+GOZ+GT	95.658
Threshold	T+GT	1.155
Edge image	EM+GT	2.048
Segmentation and region selection	EM+RR+S+RS+GT	130.457

2.3 Conclusion

Le cadre de la recherche faisant l'objet de ce mémoire a été défini davantage dans cet article. Nous y avons justifié l'approche choisie, qui est de déterminer les traitements d'image permettant la création de patrons de phosphènes (*bitmapped approach*). Il a été vu que les équipes de recherche dans ce domaine font généralement preuve d'un trop grand optimisme, pour des raisons évidentes. Par contre, il serait malvenu de se limiter à une représentation trop grossière. Certains mécanismes physiologiques pourraient jouer en notre faveur. Ainsi donc plusieurs stratégies ont été imaginées et décrites dans ce chapitre. Les premières consistent principalement à réduire la résolution des images, tandis que les dernières activent les phosphènes en mode tout-ou-rien. Celles-ci nécessitent davantage d'effort afin d'obtenir un effet acceptable, en particulier la segmentation d'image.

Le prochain chapitre décrit plus en détail la version logicielle du système qui a été présenté ici. Et en particulier, les bases des traitements d'image sont posées, ainsi que la méthodologie qui a fait en sorte que la recherche aboutisse aux résultats généraux qui viennent d'être présentés.

CHAPITRE 3

VERSION LOGICIELLE DU SYSTÈME DE TRAITEMENT D'IMAGE

3.1 Introduction

Ce chapitre est un complément d'information au chapitre précédent, plus particulièrement en ce qui concerne le traitement d'image. Une version entièrement logicielle du système de traitement d'image (STI) pour un SVC a été implémentée en C++, pour les raisons suivantes:

- Il faut valider les traitements d'image avant de les implémenter en système embarqué : l'exécution logicielle en temps réel donne une idée de la complexité des algorithmes et de la faisabilité de leur implémentation en système embarqué;
- L'appréciation subjective de plusieurs traitements nécessitent une séquence continue d'image, ce qui demande que le traitement se fasse en temps réel avec une caméra vidéo;
- Le fait que les algorithmes soient codés en C facilite leur portabilité vers un processeur dédié;
- L'implémentation logicielle du STI et d'une interface usager permet de valider le fonctionnement de la communication entre les deux processus.
- Nous donnons d'abord dans ce chapitre des détails sur les traitements d'image pertinents à notre application. Le contenu algorithmique du STI s'y trouve dévoilé. Parmi ces traitements se trouve l'opération critique qui est la segmentation, présentée au chapitre précédent. Les caractéristiques de cet algorithme ont déjà été données, mais le présent chapitre va plus loin en comparant les diverses avenues qui ont été explorées afin de développer un algorithme fonctionnel. Enfin, les détails de l'architecture du modèle logiciel du STI sont présentés.

3.2 Bases théoriques des traitements d'image

3.2.1 Méthodologie

Étant donné le grand nombre d'hypothèses reliées à la représentation des scènes en patrons de phosphènes, les spécifications du traitement d'image ne sont pas précises. Elles ont été énumérées autant que possible au chapitre 2. La qualité d'une approche donnée ne pourra réellement être évaluée qu'avec les commentaires d'un patient muni d'un stimulateur fonctionnel.

Afin de préparer le terrain à une telle validation, un système implémentant plusieurs stratégies de représentation a été conçu. L'élaboration de ces stratégies repose sur des traitements d'image ayant une plus ou moins grande complexité. L'idée est que la validation fonctionnelle *in vivo* de la représentation devrait suivre une approche *bottom-up* : les paramètres de bas niveau devraient être fixés de manière optimale avant de tenter l'essai d'une version plus autonome de la même représentation. C'est pourquoi les traitements d'image sont classés et présentés ici en ordre de complexité. Nous distinguons 4 grandes sphères de traitement : très-bas-niveau, bas-niveau, niveau moyen, et haut-niveau. La sphère de très-bas-niveau concerne les paramètres d'onde pour chaque électrode. Ces paramètres devront évidemment être optimisés avant de tenter de représenter des images en patrons de phosphènes. Un logiciel a déjà été mis au point par l'équipe PolySTIM afin de faire ces premiers tests, et donc nous ne donnerons pas davantage de détails à ce sujet. Il faut noter que les paramètres qui découleront de cette étape devront être incorporés à la carte visuotopique afin de permettre la conversion des PVFC en SSA. Rappelons qu'un SSA contient l'adresse de l'électrode, et ses paramètres d'onde pour l'évocation d'un phosphène stable, soient l'amplitude (A), la durée de phase (DP), la durée d'inter-phase (DI), la durée de train (DT), la durée inter-train (DIT) et la fréquence de répétition des trains (F). Les autres sphères font l'objet des sous-sections suivantes.

3.2.2 Traitements d'image de bas niveau

Les traitements de bas-niveau sont ceux qui généralement s'appliquent localement, sur des pixels voisins. Il s'agit typiquement de prétraitement. Pour notre application, ils ont deux fonctions : le rehaussement de la qualité d'image, et la représentation.

Le rehaussement d'image est au début de la chaîne. Il sert à rendre l'image apte à être traitée davantage par des algorithmes plus complexes. Il comprend des méthodes pour augmenter les contrastes, réduire le bruit, transformer l'intensité des pixels afin de les amener dans une plage raisonnable. Il a déjà été mentionné qu'il était nécessaire pour un STI dédié à un SVC de pouvoir s'adapter à une grande variété de scènes différentes, avec diverses conditions d'éclairage, et des caractéristiques à mettre en valeur qui peuvent être différentes selon la situation. Cela ne peut être possible qu'en préparant l'image à l'aide d'opérations de rehaussement.

À cause de raisons d'ordre biologique, la plupart de stratégies de représentation utilisent 2 niveaux de gris (noir et blanc). Pour ce faire, on utilise deux approches différentes : l'affichage des contours et l'affichage des zones « binarisées ». Il s'agit donc d'utiliser respectivement des algorithmes de détection de contours et de seuillage. Ces algorithmes font intervenir des opérations locales avec les pixels voisins, donc des fonctions de bas-niveau.

Pour les fonctions de rehaussement d'image, les algorithmes implémentés jusqu'à date sont l'égalisation d'histogramme global et le filtrage médian. L'histogramme de l'image est un tableau représentant, pour chaque valeur d'intensité (pour une image en tons de gris de 8 bits, il s'agit des valeurs 0 à 255), le nombre de pixels dans l'image ayant cette intensité. La figure 3.1 c) montre l'histogramme de l'image de la figure 3.1 a). L'égalisation d'histogramme consiste à obtenir une image dont la distribution des niveaux de gris est uniforme. Cela a pour effet d'améliorer le contraste pour les valeurs d'intensité près des maximums de l'histogramme, et de le diminuer près des minimums. La figure 3.1 b) montre le résultat de l'égalisation sur l'image de la figure 3.1 a), et la figure 3.1 d) montre le nouvel histogramme.

L'algorithme consiste à construire l'histogramme de l'image (qui est en quelque sorte une fonction de distribution de probabilité), d'en déduire l'histogramme cumulatif, et ensuite de calculer une fonction de transformation qui est ensuite appliquée à chaque pixel. Notons que Harvey de notre équipe avait déjà pensé à implémenter cet algorithme [43].

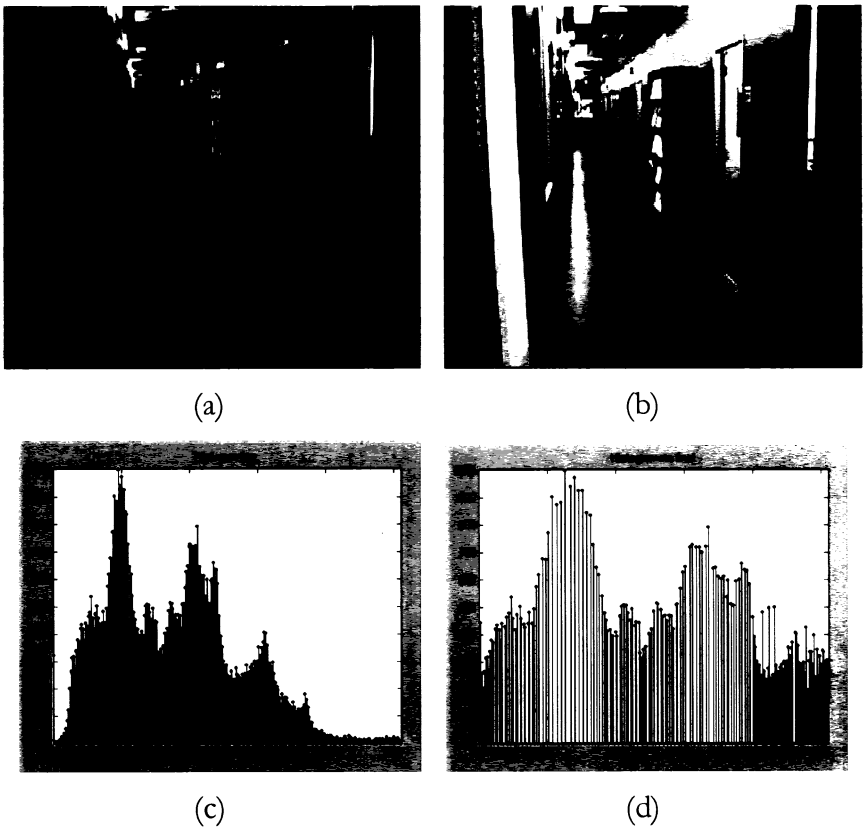


Figure 3.1 – Égalisation d'histogramme global : a) image originale, b) image traitée, c) histogramme original, d) histogramme obtenu après traitement. L'axe des ordonnées est le nombre de pixels à une intensité donnée. L'axe des abscisses est l'intensité, de 0 à 255 (8 bits).

Quant au filtrage médian, il sert à réduire le bruit. Il s'agit d'un traitement non-linéaire qui ne détériore pas les contours, et qui est particulièrement robuste contre le bruit d'impulsion, c'est-à-dire une corruption de l'image avec certains pixels noirs ou blancs. L'algorithme consiste à prendre la médiane des valeurs d'intensité des voisins. La figure 3.2 montre le résultat avec une fenêtre de 5x5 pixels. Bien qu'il semble simple, ce traitement demande

beaucoup de calcul car il faut trier les valeurs de la fenêtre pour tous les pixels, afin d'en retirer la médiane.

(a) (b)

Figure 3.2 – Filtrage médian : a) image originale, corrompue avec du bruit d'impulsion, b) image filtrée.

Les autres algorithmes de bas-niveau qui sont directement reliés à la représentation, soient l'application d'un seuil (*threshold*) et la détection de contours, ont déjà été expliqués au chapitre 2.

Les traitements présentés dans cette sous-section constituent un bagage de traitement autonome. En effet, pour les premiers tests avec une caméra (c'est-à-dire, après ceux qui utiliseront des patrons synthétiques pour l'étude de la reconnaissance de base), uniquement ces fonctions pourront être utilisées, car elles suffisent à rehausser l'image et à la binariser pour la représentation. Il ne suffira ensuite que de transformer les pixels en PVFCs, et ensuite de générer les commandes à l'implant. Les premières approches de représentation peuvent se contenter de ces fonctions, mais si elles s'avéraient être les plus prometteuses, il est fort à parier qu'un étage supplémentaire de complexité sera nécessaire afin d'optimiser les paramètres automatiquement. C'est le but des traitements de niveau moyen.

Avant de passer à la suite, il importe de mentionner que la plupart des opérations de bas-niveau sont aptes à être prises en charge, dans le cas où c'est justifié, par un accélérateur de calcul parallèle.

3.2.3 Traitements de niveau moyen

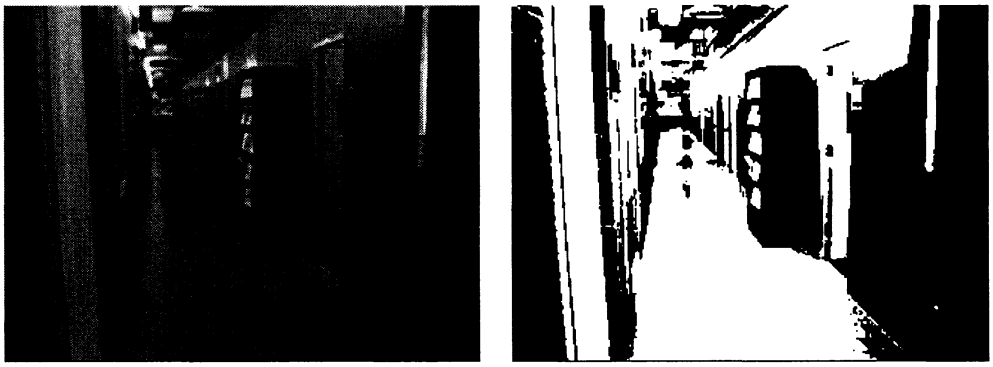
Les traitements de niveau moyen comprennent les techniques d'ajustement dynamique des paramètres des traitements de bas-niveau, ainsi que l'application locale de ceux-ci sur des régions ayant les mêmes caractéristiques de luminescence. Ils s'avèreront essentiels pour faire en sorte que les traitements de bas-niveau s'adaptent au plus grand nombre de situations possibles, ce qui est une caractéristique essentielle d'un STI dédié à un SVC. Voici ces techniques :

- Détermination automatique du seuil de binarisation de l'image;
- Détermination automatique de l'échelle de détection des contours;
- Rehaussement de l'image selon divers procédés : suppression du bruit, augmentation du contraste et égalisation locale d'histogramme en utilisant un voisinage adaptatif. Le voisinage adaptatif implique la segmentation de l'image en régions de caractéristiques semblables.

Les deux premières techniques servent à optimiser dynamiquement les paramètres d'opérations de bas-niveau. Elles seront utiles s'il est déterminé que les représentations de scènes par seuillage ou par contours sont pertinentes. La troisième technique utilise le voisinage adaptatif pour améliorer la qualité des opérations de bas-niveau mentionnées. Sonka et al [79] expliquent ces techniques en détail, ce qui ne sera pas fait ici car il s'agit d'une simple amélioration des algorithmes présentés précédemment. Il faut par contre retenir que le voisinage adaptatif peut être le résultat d'une segmentation, d'où l'importance de cette opération qui a été décrite au chapitre précédent. Nous l'avons classé dans la catégorie des traitements de niveau moyen.

La détermination automatique du seuil de binarisation s'appuie sur le fait que l'histogramme d'une image possède généralement 2 maximums : un pour l'arrière-plan et l'autre pour

l'avant-plan. Un algorithme servant à déterminer le seuil optimal pour l'image au complet a été développé. Il s'agit d'un algorithme itératif, qui crée deux régions, avant et arrière-plan. Pour chaque région, la moyenne des intensités est calculée, et un nouveau seuil est déterminé. Les pixels de l'image sont tous réassignés à chaque région selon le nouveau seuil, et l'opération est répétée. Généralement, il faut moins de 10 itérations pour parvenir à un seuil optimal. La figure 3.3 montre le résultat sur une image dont l'histogramme n'a pas été égalisé.



(a) (b)

Figure 3.3 – Détermination optimale du seuil global de binarisation : a) image originale, b) image binarisée en 5 itérations, avec un seuil automatique final de 89 (sur 255).

Cette technique donne des résultats douteux lorsqu'un objet est illuminé de côté, donnant un gradient d'intensité. La binarisation pourra être améliorée avec des techniques de voisinage adaptatif qui seront élaborées plus loin dans ce chapitre.

La représentation des scènes par contours nécessite un filtrage de l'image de contours pour enlever les contours faibles, dus à de petites variations d'intensité qui n'ont généralement rien à voir avec les contours des objets. Ce filtrage est une opération de seuillage normale. Son optimisation n'a pas été automatisée. Il est plus naturel que l'ajustement du seuil soit fait par le patient lui-même.

3.2.4 Traitements de haut-niveau

Les traitements de haut-niveau comprennent les tâches évoluées permettant de ne garder que les objets ou caractéristiques importantes de l'image et/ou de les schématiser. Celles qui s'avèreront utiles seront sélectionnables par le patient dans un STI portable, en fonction de la scène dans laquelle il se trouve (un aveugle a généralement une certaine connaissance de l'endroit dans lequel il se trouve). Il peut s'agir de l'automatisation des paramètres des algorithmes de niveau moyen, ou bien de modes de fonctionnement qui donne la priorité à certaines caractéristiques. On peut penser à des modes qui facilitent la lecture, d'autres les déplacements à l'intérieur, à l'extérieur, etc. Ces modes seraient basés sur des cartes d'importance, calculées à partir d'algorithmes permettant de relever les régions d'intérêt dans une image, algorithmes qui sont communs dans la littérature [39,52,80].

Pour le moment, la méthode de représentation idéale n'est pas encore identifiée. Il n'est donc pas justifié d'implémenter les traitements de haut-niveau. Ceux-ci devront être ajoutés dans une version ultérieure du STI.

3.3 Algorithmes de segmentation

3.3.1 Avenues envisagées

Les opérations de segmentation ont été introduites au chapitre précédent. Il en existe plusieurs techniques. En résumé, les techniques basées uniquement sur les contours sont généralement inappropriées, car elles utilisent soit un filtrage de l'image de contour, qui nécessite des conditions d'éclairage stables et connues, soit des méthodes de relaxation des contours qui sont très coûteuses en termes de calcul [71]. Les techniques basées sur un histogramme sont utiles pour identifier le sujet à l'avant-plan, dans des conditions d'éclairage connues [6,70]. Parmi les méthodes qui manipulent les régions, l'approche privilégiée utilise le fusionnement des régions (chapitre 2). D'autres techniques existent, comme le remplissage de bassin (*watershed*), mais sont réputées demander beaucoup de temps de traitement [79]. Les techniques plus avancées de segmentation ont besoin de connaissances sémantiques de

haut-niveau pour obtenir de bons résultats, ce qui n'est pas applicable ici car nous n'avons pas de connaissance a priori du contenu de l'image à segmenter [79].

Au moins quatre implémentations différentes d'algorithmes de segmentation par fusionnement de régions ont été tentées. Il s'agit de l'algorithme de *growth of zones*, développé par Gilmont et al [39], un algorithme de fusionnement simple (*region merging*), un autre utilisant en plus les données de contours (*region-and-edge*), et finalement le fusionnement par fonte de contours (*boundary melting*). La prochaine section les décrit et les compare brièvement.

3.3.2 Comparaison des différentes implémentations

L'algorithme *growth of zones* consiste à agglomérer les pixels (*clustering*) de manière à construire des régions [38]. Elles sont construites séquentiellement, à partir d'un pixel de départ (*seed pixel*). Le critère de fusion des pixels voisins dépend du gradient et de l'intensité moyenne (I_{moy}) des régions. Ce processus est très rapide et économe en mémoire, ayant une complexité O(MN) où M et N sont la hauteur et la largeur de l'image, mais il n'offre pas de structure hiérarchique des régions qui permette ensuite de manipuler celles-ci. Pour ce faire, il faudrait réanalyser l'image construite. D'autre part, il réagit mal au bruit car une région peut s'étendre par un petit trou dû au bruit et provoquer la fusion de deux grandes régions. Enfin, il laisse beaucoup de minuscules régions non fusionnées qui, à l'étape de la transformation en phosphènes, seront cachées ou pire, provoqueront l'affichage injustifié d'un phosphène.

Les trois autres méthodes sont basées sur des *blobs* (terme tiré de [88]), petites régions initiales de quelques pixels de côté. Cela permet de construire un graphe de taille raisonnable, cache les petits détails qui n'ont pas leur place à l'échelle des phosphènes, et diminue l'influence du bruit. Ils ont tous une complexité de $O(MN+MN/K^2*\log(MN/K^2))$, où K est la taille du côté d'un *blob*. Les temps d'exécution varient cependant beaucoup. La croissance des régions est parallèle, car les graphes sont mis à jour à chaque passe complète.

L'algorithme *region merging* utilise un graphe de régions et un graphe de contours. Le critère de fusion est basé sur I_{moy}. Lorsque deux régions sont fusionnées, elles sont chainées. Cela fait en sorte que le tableau de régions ne grandit jamais : celles-ci sont uniquement mises à jour. Par contre il faut parcourir la chaîne à chaque fois afin de trouver le début et/ou la fin, pour y accrocher la nouvelle région. Un des inconvénients majeurs est que la formation des *blobs* masque souvent les contours, permettant facilement à deux grandes régions distinctes de se fusionner.

L'algorithme *regions-and-edges* fonctionne de manière similaire, mais il calcule préalablement une image de contours avant de construire les *blobs*, afin de résoudre le problème de l'implémentation précédente. Les régions superposées à des contours forts sont marquées telles quelles et ne sont jamais fusionnées. Les critères d'appartenance sont I_{moy} et I_{moy}*l où l est la taille de la frontière commune. Deux régions séparées par un contour fort ont un l petit puisque qu'elles seront séparées par des régions marquées « contour ». Cela a pour effet d'éliminer les cas où 2 grandes régions différentes, mais séparées par un contour « troué », seraient tentées de se fusionner. Les régions de l'image ayant beaucoup de détails ne sont presque pas fusionnées, et une opération de post-traitement est utilisée afin de forcer leur fusion. Forcer une fusion est toujours dangereux, et dans ce cas-ci, c'est appliqué sur une grande partie de l'image. Cet algorithme est également le plus demandant en calculs.

L'algorithme *boundary melting* a été mis au point afin de résoudre les lacunes du précédent : au lieu du chaînage il utilise une structure hiérarchique des régions plus classique, l'information des contours est stockée dans les branches et non pas dans les régions, seules les valeurs des contours sont utilisées dans les critères, ce qui lui permet de ne manipuler que les branches. D'autres détails seront donnés au chapitre 4. Le tableau 3.1 résume succinctement les avantages et inconvénients de chaque implémentation. De plus, la figure 3.4 montre un exemple de résultats pour chaque algorithme.

Tableau 3.1 – Comparaison des implémentations de segmentation par fusionnement de régions.

Méthode	**Avantages**	**Inconvénients**
Growth of zones	Rapide Très économe en mémoire	Pas de structure hiérarchique Très sensible au bruit Beaucoup de très petites régions
Region merging	Accès relativement rapide aux parents des grandes régions Peu sensible au bruit Économe en mémoire	Récursif, basé sur l'intensité moyenne Les blobs lissent les contours Critère basé sur le périmètre peu efficace
Regions-and-edges	Utilise les données de contours brutes Économe en mémoire	Paramètre basé sur le périmètre difficile à contrôler Un nombre important de région ne sont pas fusionnées: doivent l'être en post-traitement Très récursif: longue mise à jour des caractéristiques de régions Les contours sont des objets en eux-même
Boundary melting	Meilleurs résultats que les autres méthodes basées sur des blobs Utilise les données de contours uniquement Structure hiérarchique plus classique	Moins économe en mémoire (mais elle est tout de même contenue) Post-traitement nécessaire (mais moins désastreux que pour l'algorithme *regions-and-edges*)

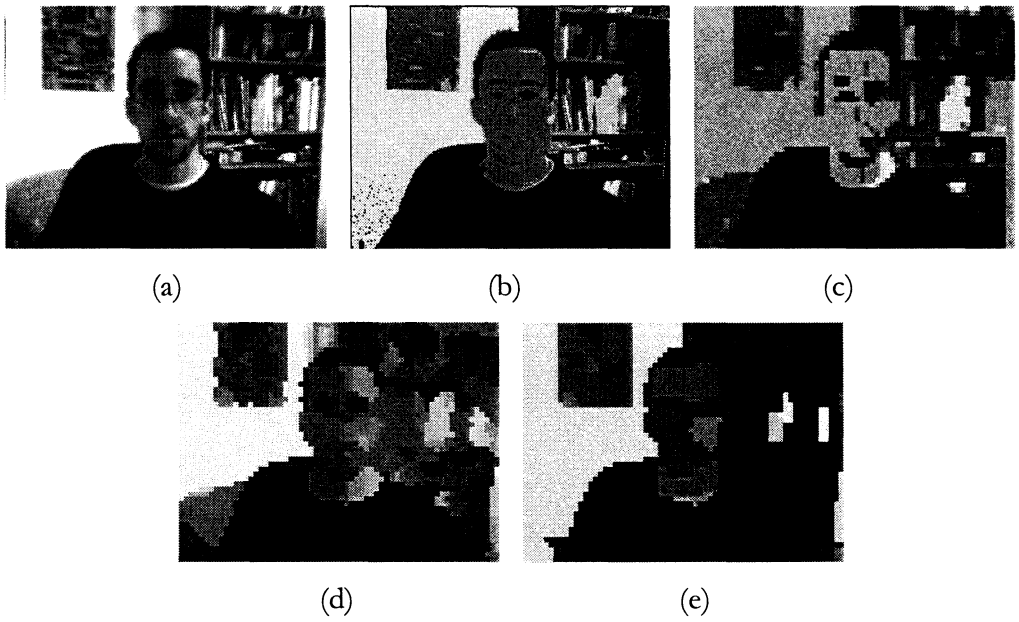


Figure 3.4 – Résultats de segmentation des différents algorithmes. L'image originale (a) est suivie respectivement par le résultat du *growth of zones* (b), *region merging* (c), *region-and-edges* (d) et *boundary melting* (e).

3.4 Modèle du système de traitement d'image sur PC

3.4.1 Architecture

La figure 3.5 est un schéma-bloc de la version logicielle du STI. Une caméra est connectée au module de traitement d'image, implémenté en C++ dans une application PC. Une seconde application, le logiciel usager, est également codée en C++ et peut s'exécuter sur un autre PC. Elle communique avec le STI via un port de communication sérielle. Elle sert à contrôler le STI (marche/arrêt, configuration des paramètres, chargement de la carte visuotopique) et à visualiser le résultat sous forme de phosphènes.

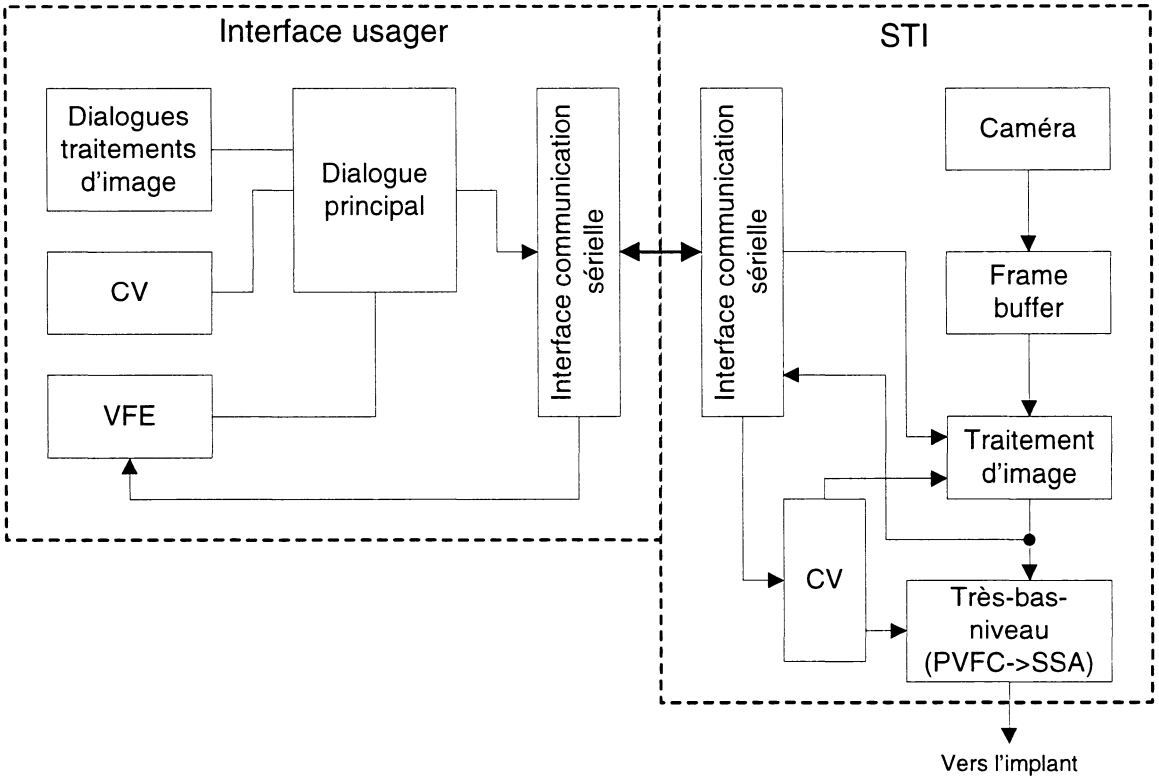


Figure 3.5 – Schéma-bloc de la version logicielle du STI (avec interface usager). L'étiquette CV désigne la carte visuotopique, et VFE le module de visualisation des patrons de phosphènes dans le champ visuel.

Le traitement d'image au sein du STI est implémenté dans une classe. Elle contient tous les algorithmes de traitement d'image et toutes les ressources qu'ils nécessitent, de même que les algorithmes de recalage de l'image traitée en PVFCs. Elle forme le cœur de ce qui doit éventuellement être porté en système embarqué. Tous les algorithmes ayant été présentés aux chapitres 2 et 3 y sont implémentés. La boucle principale a pour tâche de commander la capture d'une image et ensuite d'appeler la fonction de traitement. À chaque tour, elle vérifie le FIFO de communication sérielle. Davantage de détails d'implémentation sont donnés dans l'annexe A.

3.4.2 Programme d'interface usager

Le programme d'interface usager sert d'abord à configurer le STI. Elle permet d'envoyer les paramètres de traitement d'image selon le protocole de communication simple. Éventuellement, les paramètres seront choisis dans des boîtes de dialogue ergonomiques lorsque le STI sera rendu au stade de la fabrication.

Ce programme contient aussi la CV. Cette carte peut être chargée du disque dans le format de fichier MAP compatible avec le logiciel de génération de cartes aléatoires présenté au chapitre 1. C'est lui qui calcule la transformation des coordonnées du champ visuel en coordonnées d'image (pixel ou *blob*, selon la stratégie choisie). Il envoie ensuite ces coordonnées au STI qui s'en servira pour associer les phosphènes aux pixels de l'image traitée.

Enfin, le logiciel a la capacité d'afficher en temps presque réel le résultat du traitement grâce à l'émulateur de champ visuel (VFE). Il reçoit les intensités des PVFCs dans l'ordre, et les affiche à l'aide de Matlab qu'il contrôle, de la même manière que le programme de génération des CVs. L'apparence de cet affichage est semblable aux résultats « phosphénisés » du chapitre 2.

3.4.3 Communication

La communication entre les deux programmes se fait par port série, car il est facile à implémenter en matériel, utilise peu de plots d'entrées/sorties, et a un débit suffisant pour nos besoins. La lecture et l'écriture s'exécutent dans des fils d'exécution parallèle (*threads*) séparés. Lorsqu'un message est reçu, il est stocké et une fonction *callback* signale l'arrivée de données au *thread* principal. Celui-ci lit le message et prend l'action appropriée.

Le protocole est simple et est sujet à changement, c'est pourquoi il n'est pas décrit en détail ici : un mot de 8 bits signale la nature du message (marche/arrêt, choix de traitement, changement de paramètre, CV, ou signal de déboggage) et les mots suivants sont interprétés

conséquemment. Le signal de déboggage indique au STI qu'il doit envoyer les PVFCs calculés pour chaque image. Dans ce cas, le STI vérifie si le port est libre, le cas échéant il les place dans le tampon d'envoi et ils sont immédiatement transmis au logiciel d'interface. Le débit est d'environ 1000 PVFCs par seconde, ce qui fait que le VFE est rafraîchi une fois par seconde pour une CV de 1000 phosphènes. Pendant la transmission, le processeur est libre et le STI continue à traiter les images.

3.5 Conclusion

Les traitements d'image pertinents pour notre application ont été présentés d'une manière *bottom-up*. Parmi eux, la segmentation occupe un rôle important et ses différentes implémentations ont été décrites et comparées. Enfin, la structure de la version logicielle du STI a été présentée.

Lorsque le temps de fabriquer un système portable sera venu, il faudra valider la communication avec le matériel et adapter les paramètres et le protocole de communication aux traitements qui seront développés entre-temps. Quant à l'interface graphique, il faudra soigner son aspect et implémenter la conversion PVFC à SSA. Le prochain chapitre discute de l'algorithme de segmentation sélectionné et de l'implémentation embarquée du STI.

CHAPITRE 4

VERSION EMBARQUÉE DU SYSTÈME DE TRAITEMENT D'IMAGE

4.1 Introduction

Le chapitre précédent est une description de la version logicielle du système, dans laquelle davantage de détails ont été donnés au sujet des traitements d'image utilisés. La stratégie qui demande le traitement le plus complexe est celle qui choisit des zones à afficher. Pour ce faire, il faut appliquer une segmentation d'image. Plusieurs algorithmes de segmentation existent dans la littérature, mais les caractéristiques particulières à notre application exigent l'élaboration d'un algorithme spécfique. Les avenues explorées lors de son développement ont été présentées au chapitre précédent, mais il importe de décrire davantage son implémentation.

D'autre part, le but de cette recherche est de fournir des modules qui pourront être intégrés dans un système complet de stimulation visuelle corticale, afin de procéder à des tests *in vivo*. Puisqu'il s'agit de tester les aptitudes d'un aveugle muni d'un tel dispositif à se déplacer, il est primordial qu'il soit portable. Or, l'intégration directe de processeurs et de modules matériels dans la même puce, technique communément appelée *System on Chip* (SoC), est de plus en plus disponible. Il est donc pertinent d'envisager d'intégrer tous les modules de la partie frontale d'un tel stimulateur dans une seule puce, bien que des prototypes préliminaires pourront être fabriqués avec des composants discrets. Quoiqu'il en soit, le but poursuivi ici est d'offrir à l'équipe PolySTIM les modules de traitement d'image qui seront intégrés au sous-système frontal. La prochaine étape consiste donc à concevoir une version embarquée du système présenté au chapitre précédent.

La description de l'implémentation embarquée du système se trouve dans le corps du présent chapitre, constitué d'un article ayant été soumis au journal « IEE Medical and Biological

Engineering and Computing ». On y présente l'algorithme de segmentation introduit plus tôt d'une part, et l'implémentation de tout le design en système embarqué. Cette combinaison singulière est motivée par le fait que le principe de l'algorithme de segmentation est fortement dépendant de son implémentation. En effet, les stratégies de traitement d'image présentées au chapitre 2 sont généralement basées sur le calcul de convolutions. Pour atteindre une performance suffisante, l'algorithme de segmentation utilise au maximum les résultats obtenus de ces convolutions. Celles-ci occupant d'ailleurs une charge beaucoup trop grande pour un simple processeur embarqué, elles sont prises en charge par un module matériel faisant partie du système, dont le design interne sera également présenté dans l'article.

4.2 « Design and Test of a Novel Image Processor Dedicated to Visual Cortical Stimulation »

L.X. Buffoni, M. Sawan, J. Coulombe
PolySTIM Neurotechnologies Laboratory
Departement of Electrical Engineering, Ecole Polytechnique de Montreal

ABSTRACT

Visual cortical stimulation (VCS) consists of electrically evoking action potential in localized areas of the human cortex in order to help the blind recover partial vision. These phosphene based percepts must be generated according to images of the surrounding world. Such complex process is being validated by different research teams, but still contain some uncertainties, thus an image processing system implementing flexible strategies is needed. One of these strategies that is of particular interest for the VCS application was selected and implemented. It requires an image segmentation algorithm whose features are speed, stability among images in the same sequence, and no need for fine precision. An embedded system particularly suited for applying this algorithm, in addition to those that implement the other strategies, is presented. The proposed system uses an AMBA bus architecture, and includes a dedicated SIMD coprocessor which performs, among other tasks, fast convolution

operations. Experimental results of the proposed system show that real-time dedicated image processing for the VCS is achieved with an output rate of at least 7.5 images/second that is sufficient for this application.

KEYWORDS : Image processing, Segmentation algorithm, Edge detection, Region merging, Visual cortical stimulation, Embedded system, Parallel computing hardware, SIMD processor, AMBA bus, Hardware/software partition.

I. INTRODUCTION

Electrical stimulation in the visual cortex produces topographically mapped visual sensations, called phosphenes [12]. Stimulation in a particular location of the visual cortex evokes a phosphene in a particular location of the visual field. By placing many electrodes in different locations in the cortex and by appropriate stimulation, it would be possible to create recognizable patterns [31]. With technological advances in the fields of microelectronic and micro-fabrication, it becomes conceivable to create a visual cortical stimulator (VCS) that would allow the blind to recover partial vision by stimulating many locations simultaneously. Our team is currently achieving a VCS system (figure 4.1), presenting common principle with recently published designs [27,30,60,84]. It includes an image sensing device which is mounted on the subject's glasses and grabs information of the surrounding world. The input signal is then processed and transformed into implant commands. These commands are transmitted to the stimulating implant located inside the cranium, on the visual cortex, via an inductive link. At this time, the preferred input device for image sensing is a camera, but other types of devices could turn out to be more appropriate.

The biological vision system can be considered as a 2-stage structure, the first being the low-level processing used to extract or enhance features, such as edges, and the second being the high-level interpretation where, for example, objects of the scene are identified [54]. In the case of cortical stimulation, the device will naturally be responsible of the first stage, and

leave the second stage to the blind's brain. However, the information carried at the cortical level is far more complex than simple image low-level processing, and it seems that it will only be possible to "connect" to the biological dataflow in an unnatural way, by displaying crude bitmapped phosphene patterns. The needed level of processing depends amongst other things on the number of electrodes and on our ability to control the stimulated percepts.

The basic operation for most representation strategies is a segmentation, which consists of splitting an image into regions that have a strong correlation with objects or areas of the scene represented by the image. In most image processing applications, this is the first step toward image understanding [79]. A particular effort has been put on the development of an adapted segmentation algorithm because it can be used as an efficient resolution reduction method, as well as the first step for the selection of regions to display as phosphene patterns.

This paper concerns an image processing system (IPS) which represents the link between acquired image data and the implant commands, whose communication protocol is described in [27]. The proposed IPS offers multiple image processing and representation strategies because little is known about phosphene pattern recognition. Also, it is mandatory not to constrain the system into pessimistic representation capabilities. The goal of this flexible implementation is to help determine the best solution when experiments with human volunteers will be conducted.

Segmentation design is closely related to its algorithm implementation, so this paper presents both the segmentation algorithm and its corresponding embedded system that constitutes the proposed IPS. This embedded system will be integrated with other interface modules into a System on Chip (SoC) for efficient *in vivo* testing. In order to achieve real-time operation, a hardware on-chip coprocessor has also been implemented.

We present in Section II the segmentation algorithm that has been designed specifically for the visual cortical stimulation. Next, we describe in Section III the embedded system on

which it is implemented, and Section IV describes the design of the dedicated coprocessor in greater details. Results and discussion about the segmentation algorithm and its implementation are presented in Section V.

II. SEGMENTATION ALGORITHM BY BOUNDARY MELTING

A. Algorithm design considerations

In visual cortical stimulation, phosphene latency and minimum stimulation duration allow a frame rate well below video rate, therefore the segmentation algorithm shall have a throughput of at least 5 images per second. Also, in order to allow further processing, it must work over a high-level image representation. Stability is required so that phosphene patterns are meaningful for the patient, that is, similar results should be obtained from one image to another in a sequence. It must also have user-controlled parameters for scene adaptivity, and of course no semantic knowledge is available. The result does not need to be precise due to the coarse phosphene resolution.

There are three main types of segmentations. The first one is histogram-based, and is useful in the case where a main subject in the foreground has to be differentiated from the background. This technique relies on the fact that distributions of pixel intensities are usually distinct. The two other types of segmentation, edge-based and region-based, rely on the principle that a region is defined by a closed boundary. Edge-based methods consist of locating edges and to close them, while region-based methods try to grow regions from homogeneous parts until they hit a boundary. Segmentation without semantic knowledge is not easy, since edge elements often exist where no meaningful boundary exist, and absent where they should [5].

Edge-based segmentation requires that regions are enclosed in contours. However, an edge-image typically contains open edges, that have to be closed using advanced contour relaxation techniques [71]. These techniques usually need a lot of computation, and are most effective when the image is well controlled (lighting conditions, noise). But the disparity of

input images in our application may lead to different segmentation results, and therefore in poor stability.

Region-based methods either do splitting of large regions, merging of small regions, or both (split-and-merge). The segmentation algorithm suggested by Gilmont et al. for a visual prosthesis does region merging, in a clustering scheme [39]. In this fast process, regions grow sequentially from seed pixels. However, randomly selected seed pixels and sequential region growing often cause accidental merging between two large regions, thus making stability an issue. Also, the algorithm is more sensitive to noise, since clustering can occur through weak boundaries and accidentally merge two distinct regions. In addition, the process does not make use of a hierarchical representation, but works directly on the image buffer.

Phosphenes have non negligible size in the visual field, ranging "from a pinpoint to a nickel at arm's length" [76], therefore the final representation into phosphene patterns is always coarse. The final operation that geometrically transforms the image regions into phosphenes may end up by hiding small details or worse, activate wrong phosphenes. Region merging is preferred to region splitting because the corresponding average phosphene size can be used in order to determine the size of the initial regions, so they are in the same order of magnitude. It is useless to segment with more precision. Using an initial region size bigger than a pixel makes the algorithm less sensible to noise, and lower in computation.

Displaying phosphene patterns from edges is a representation strategy on its own [19], so some resources are already dedicated for edge extraction. On the other hand, a segmentation algorithm based solely on edges often needs a lot of computation for edge relaxation, while region merging is, as discussed earlier, faster but less precise. An hybrid algorithm based on region merging, but using edge information, seems to be a good compromise. Many segmentation algorithms use region and edge information, but they do it for improvement purposes [24,25,64]. In our case, it should be used for a speed-up purpose, since there will be no need for contour closing. Region merging alone, as it has been described in the previous

paragraph, fits in the Single Linkage Region Growing (SLRG) class, and is sequential by nature [7]. Working with edge information instead of region characteristics is beneficial since edge extraction is a local operation and is suited for parallel implementation. Such an algorithm fits in the Hybrid Linkage Region Growing (HLRG) class. The strategy employed is a variant of a region growing algorithm by boundary melting, presented in [79]. This algorithm is originally based on a supergrid data structure, which is not well suited for the representation of regions. A popular hierarchical data structure is the quadtree, abundantly described in [7,28], and the minimal adjacency table [26] is another interesting structure. Also, Yu et al. use an interesting hierarchical representation based on an UNION-FIND clustering scheme [88]. The chosen data structure incorporates this method with a planar-region adjacency graph [63]. Next section presents this data structure in greater details.

B. Description of the proposed algorithm

The algorithm starts with feature extraction, which is used to populate the first level of a data structure, based on region and edge information. Objects of this data structure are then processed in four consecutive steps leading to a complete hierarchical representation of the image. The first step is a straightforward removal of weak boundaries, from which a new level is built. Steps 2 and 3 are recursive refining steps, further removing boundaries based on regions' area and border length. A new level is created at every iteration. Step 4 is a an optional straightforward post-processing operation, performed to eliminate remaining oversegmented regions that would appear meaningless for the patient.

The data structure is a planar graph containing two types of objects: regions and boundaries. Regions contain data about their size and intensity, and a link to a parent of the next level. Boundaries contain information about edge intensity and border length, and are always connected between 2 regions. Figure 4.2 shows the data structure with regions (R) and boundaries (B), and their associated characteristics (S and L), at three different levels.

The segmentation process starts with small blocks of pixels, typically 4x4 or 8x8, which will be referred to as *blobs*, as in [88]. In the beginning, both horizontal and vertical edges between *blobs* are computed by convolving twice the image with standard Sobel masks, and are then stored in the boundary objects. Segmentation essentially works with boundaries because the merging criterions are based on edge data.

First, the graph is initialized by computing the edge image, and then the boundaries are filled with the number of weak edges, W, a decimal number, initially bound between 0 and 1. W is calculated from edge magnitude as follows:

$$W = 1 - \frac{|edge|}{255} \tag{4.1}$$

with |edge| being the mean of edge magnitude between *blobs*, from 0 to 255. Boundary length is set to 1. The 2 region pointers of each boundary are sequentially set to the corresponding regions.

The first step is the removal of weak edges, defined by an user-controllable threshold, T_1. The boundary objects are visited once. When $W < T_1$, they are labeled as melted, and the regions referenced by these boundaries are assigned a parent of the next level. The size of the newly merged regions are added to the size of the parent. Then the boundary graph is updated.

Boundary graph update starts by visiting every boundary object and updating their pointers to regions, so that they reference their parents. Then, for each boundary that is not labeled as melted, it is copied at the current location in the boundary list and every subsequent boundary object is checked to see if it references to the same regions. If it does, it is also labeled as melted, and its border size and number of weak edges are added to those of the current new boundary. Old boundaries are overwritten, so the size of the boundary list never exceeds the initial size.

Next, every boundary is visited again, but this time, a criterion based on the regions' area is used. Merging occurs if and only if

$$\frac{W}{\min(A_1, A_2)} > T_2 \tag{4.2}$$

where A_i is the region size, contained in region objects, and T_2 is a second threshold. The graph is then updated. This step can be done recursively, but it usually takes no more than two iterations before no new boundaries can be melted.

The last step uses a criterion based on the boundary length. Merging occurs if and only if

$$\frac{W}{l} > T_3 \tag{4.3}$$

where l is the boundary length and T_3 a third threshold. The graph is updated again, and the regions in the last level contain the whole segmented image. More information about threshold values is given in Section V.A.

This segmentation algorithm falls into HLRG class because it compares pairs of adjacent regions, but the merging criterions only need the information contained in the boundary objects. Since the process handles boundaries, it is not affected by the order in which they are processed. However, this kind of algorithm is very sequential in nature, and thus not suited for parallel implementation, except for the edge detection.

Since this class of algorithms lacks adaptivity to local area situations because it uses the same thresholds throughout the image, it produces areas where the number of regions is too large (oversegmentation), and others where it is too low (undersegmentation) [7]. As mentioned earlier, oversegmented parts are likely to be incomprehensible when transformed into phosphene patterns. This drawback is fixed with an optional post-processing step. It merges adjacent regions whose sizes are below another threshold, T_4. Again, the boundaries are visited and those with both referenced regions smaller than T_4 are labeled as melted, and the graph is updated afterward.

C. Analysis of the algorithm complexity

Let's define M the width and N the height of the image, and K, the size of the initial *blobs*. Computation of edge data has a complexity of O(MN). For a 320x240 image, experiments show that this represents about 88% of the total algorithm computation time, although this proportion will be dramatically reduced by using the dedicated coprocessor.

There are MN/K^2 initial regions, so there are less than $2MN/K^2$ horizontal and vertical boundaries. All four main steps of processing have a complexity of $O(MN/K^2)$, because each boundary is visited once. The boundary graph update has a complexity of

$$O\left(\frac{MN}{K^2}\log\left(\frac{MN}{K^2}\right)\right) \tag{4.4}$$

because it requires a graph search to group similar boundaries. Thus, the total algorithm complexity is

$$O\left(MN+\left(\frac{MN}{K^2}\right)\log\left(\frac{MN}{K^2}\right)\right) \tag{4.5}$$

High-level computation is typically very time consuming. The low-frequency system on which it is implemented makes computation time an issue. Graph representation of the image is possible here for 2 reasons: 1) low-level edge extraction will be computed by a hardware accelerator, 2) high-level computation manipulates *blobs* as initial regions, lowering the number of criterion evaluation, and reducing access to memory.

The algorithm is also memory efficient. A new region is only allocated when it is composed of at least two regions of the previous level, so the region list never contains more than twice the starting number of regions, which is 1200, for a 320x240 image with 8x8 *blobs* (M=320, N=240, K=8). A region uses 10 bytes, so the total list size would be 2x1200x10=24 Kb. Boundaries are computed at each level and overwrite those of the previous level, so the edge array size never exceeds the initial size of $(M-1)N/K^2$ horizontal plus $M(N-1)/K^2$ vertical elements. A single element uses 17 bytes, so the total array size would be 39.6 Kb.

III. IMPLEMENTATION IN AN EMBEDDED ARCHITECTURE

A. Description of a front-end system for cortical visual stimulators

A front-end system has been developed for the VCS. It implements various image processing strategies, as descripted in [19]. Along with this front-end system comes a user-interface software to be run on a PC.

The system owns the Visuotopic Map (VM), which defines all the needed phosphenes' characteristics: size, location in the visual field, stimulation pulse parameters and relation with image pixel coordinates. It is computed by the user-interface software and sent to the IPS during configuration.

Figure 4.3 is a dataflow diagram of the proposed IPS. A camera grabs an image which is then processed according to the selected strategy. Resulting images of all algorithms need a geometrical transformation, to transform pixels into phosphene visual field coordinates (PVFCs). These PVFCs must then be transformed into stimulation site addresses (SSAs), which is done by the PVFC-to-SSA module. This module is a dual-port RAM that contains the part of the VM that establishes relationship between PVFCs and SSAs. An SSA contains every phosphene's unique set of pulse parameters. A subsequent independent synchronization module will periodically read from the other side of the PVFC-to-SSA module and transform the selected SSAs into implant commands at a rate dictated by stimulation requirements.

PVFCs of a given frame can be sent out to the host interface via the serial link, and displayed in the Visual Field Emulator (VFE). This is a software program that owns a copy of the VM and uses the PVFCs to generate idealized phosphene images. Refresh rate is limited by communication time.

B. Actual architecture description

Image processing for cortical visual stimulators involves a lot of low-level operations. These operations need a small-sized data word, but a great amount of processing. This is especially true for 2D convolutions. Bosi et al. report that even for the powerful DSP processor C40 from TI, a 3x3 convolution mask takes 20 instruction cycles per pixel [9]. In the case of the Sobel mask used for the aforementioned segmentation algorithm, it would take 30 ms at 100 MHz clock frequency, which is unacceptable for our application. They suggest to exploit the inherent parallelism of such operations by developing a hardware accelerator. The same codesign methodology has been followed for our application, where a prototype has been built for future integration into an SoC. A RISC ARM7TDMI processor has been used in an AMBA bus architecture for the following reasons:

- An ARM core is small (~1 mm^2) and easy to integrate into an SoC;
- The AMBA standard is appropriate and widely used for SoC modules interconnections;
- With a hardware accelerator, this choice is sufficient for our needs.

The hardware/software partitioning is therefore obvious. The hardware acceleration module takes care of low-level, parallelizable tasks: mean of neighboring pixels, threshold and convolution. It executes the image preprocessing and puts it into the main processor's data memory. The latter does the high-level processing and the transformation of pixels into PVFCs, which are operations more suited to be executed by software, in a reasonable time. The last operation in the dataflow, the PVFC to SSA transformation, consists of identifying pulse parameters for a given PVFC. It is better implemented in hardware, in the form of a Dual-Port Random Access Memory (DP-RAM), where the back-end system reads the tagged SSAs from the other side.

Figure 4.4 shows the IPS blocks in an AMBA/AHB architecture with an ARM processor. Apart from the latter and its code and data memory, there are the AHB bus components, the UART interface for communication with the PC host, the dedicated coprocessor and its instruction controller. This last module is composed of a finite state machine and a 2 Kb

DP-RAM containing the instructions, loaded during configuration. It manages the coprocessor's execution: it allows configuration by the main processor, branch instructions and bus interface management. The coprocessor's output is connected to the bus by a master-type bus wrapper. When an image line is being outputted, the wrapper requests the bus to the arbiter. When it is granted, the line is transferred to the main memory in a burst transaction, according to AMBA/AHB specifications [2]. After completion of preprocessing of an image, an interruption is raised and the main processor starts the high-level processing. Next section presents the design of the coprocessor.

IV. DEDICATED COPROCESSOR

A. Coprocessor characteristics

The goal of building this coprocessor is to quickly execute low-level image operations that use neighboring pixels, which are inherently parallelizable. An appropriate architecture for these tasks is an array of processing elements (PE) working in a Single-Instruction Multiple-Data (SIMD) fashion [33,36,61]. Every PE does the same task, broadcasted by a controller, on its own data.

The hardware accelerator must be adapted to execute fast convolutions, and be flexible enough to implement various low-level operations. Many arithmetic operations are requested, and convolution masks weights must not be constrained. However, the data word width may be smaller than a normal processor. Finally, bottlenecks at its I/O must be avoided: input data must pass through the module and be processed in the same time.

B. The proposed coprocessor architecture

Figure 4.5a is a block diagram of the dedicated coprocessor. Image data is organized in a 2D mesh in order to allow easy access to neighboring pixels, as in [33]. A minimal degree of autonomy is needed for the algorithms, which means that there is no structure allowing to inhibit execution of some PEs [37]. An image line has 320 8-bit pixels. The mesh is

composed of 320x4 8-bit registers, so a complete line can be outputted in one clock cycle, providing high bandwidth to the design. Image input and output mechanism is inspired form the GRID system [61]. The first image line is placed in a shift register at the bottom of the mesh. Then, all the rows are shifted up. Data in the input register is shifted into the bottom row of the grid, and data in the top row is shifted to the output register. The input register is then filled with the second line of the image while the output register is purged to the main memory. This is repeated until the whole image has passed through the coprocessor.

Every register of the coprocessor grid contains a pixel value (8 bits), and is assorted with a multiplexer allowing to select input at next clock cycle: NORTH, SOUTH, EAST, WEST or IDLE. This selection is contained in the instruction word and is global for all registers of the mesh. It contains two stages of registers, one for the original data and the other for processed data. Registers on the mesh borders are connected to the other side, making the mesh a circular buffer of image data, so shifted data is never lost. The only exception occurs when a NORTH shift, along with an I_CONNECT instruction, makes the bottom row accept data from the input register instead of the top row.

There are 40 PEs sparsely disposed over the mesh, and they have access to the pixel value stored in their underlying register. Image processing operations that involve neighbor pixels are executed by appropriately shifting the pixel values in the mesh so the PEs are fed with appropriate data. Figure 4.5b shows the internal architecture of a PE, similar to that presented in [55]. It is a small, unpipelined combinatorial block containing three input selection multiplexers, a simple ALU, a multiply and accumulate (MAC) unit, a Barrel shifter and an output selection matrix. Inputs X and Y are routed towards the processing units, while input DIRECT is directly routed to the output matrix. It is thus possible to execute a mathematical operation and to store a value in a register at the same time. PEs also contain a 16 bit accumulator, a register file containing two 16 bit registers, and a working memory containing 32 slots of 16 bit words.

The instruction word contains 45 bits. Its various fields concern different resources, which allow very few decoding. It supports 3 parallel operations: a mathematical operation, a direct store, and a mesh operation. Figure 4.6 shows the instruction word's fields, and table 4.1 contains a description of these fields.

V. RESULTS AND DISCUSSION

A. Segmentation algorithm performance

Figure 4.7 shows the result of segmentation after each of the four main steps of processing. Parameter T_1, ranging from 0 to 1, defines the level over which an edge is considered weak. The homogeneous regions are merged together in this step. Parameters T_2 and T_3, also ranging from 0 to 1, further contribute to merge regions, especially those that are part of gradients between already merged regions. Because the second criterion is based on region size, it prevents two large regions from being merged together. The third criterion merges regions that are separated by a long but weak boundary. When T_i equals 1, all regions are merged, so the parameter values should be kept as low as possible, without causing excessive undersegmentation. At the end, a very busy region in terms of edge information remains oversegmented. Post-growing, which occurs in step 4, forces small remaining regions to be merged, that is whose size is below T_4. In figure 4.7e, all regions smaller than 10 were merged. The final image is coarse, but suited for representation into phosphene patterns. On the other hand, the algorithm is quite insensitive to slight differences in input images, making it appropriate for our application.

B. Embedded system performance

As mentioned earlier, the algorithms presented here were developed in order to maximize their parallelizable fraction. The ARM processor alone would be insufficient to match the needed throughput. Table 4.2 is a summary of the execution time results for various algorithms used by the IPS for visual cortical stimulation, obtained from a co-simulation model. The ARM and AMBA bus are clocked at 100 MHz, while the SIMD coprocessor is

clocked at 33 MHz. These results were taken using 320x240 images. Software execution times for the first four algorithms are presented for comparison. It is obvious that hardware acceleration is mandatory. Effective speed-up (SU_{eff}) for parallelizable algorithms is calculated using equation (6) [73]:

$$SU_{eff} = \frac{\#cycles_{accelerated}}{\frac{f_{normal}}{f_{accelerated}}(\#cycles_{normal})} \tag{4.6}$$

where $f_{normal}/f_{accelerated}$ is 1/3. The coprocessor maximum clock frequency obtained from automatic place and route in a Xilinx FPGA XCV2000E is 34.8 MHz. A new pipelined version of the PE design could allow faster clock frequency. The hardware processing time presented in table 4.2 includes data transfer from the coprocessor to the main memory. A line takes 80 cycles to be transferred, and this does not constitute a bottleneck for most algorithms, because data is processed during bus transfer until a new line is ready to be transferred.

Geometrical transformation execution time is proportional to the number of phosphenes. Results presented in table 4.2 used a 1000 phosphene VM. High-level segmentation is obviously the most time-consuming operation. However, reasonable results are obtained because the algorithm only manipulates *blobs* of 8x8 pixels in this case. The longest processing time occurs for segmentation: 1) the Sobel magnitude image average is calculated for each *blob* and transferred to memory, then 2) high-level segmentation is executed and finally, 3) PVFCs are outputted. Total computation time in this case is 1.55 + 128.3 + 0.54 = 130.4 ms, which is fast enough to display more than 5 images per second.

C. Dedicated coprocessor evaluation

The dedicated coprocessor uses 73736 LUTs in a Xilinx XCV2000E FPGA. Without any optimization, it can easily be clocked at 33 MHz. It is clocked at low frequency, partly because it is not pipelined, and also because the mesh's circular structure is problematic for place and route tools. It is possible that better results will be obtained when the module will be synthesized into a dedicated custom integrated circuit because of its regular structure.

However, long wire issues could occur when mesh's sides will be connected together. This wrapping feature was designed for maximum flexibility. However, one could decide to replace it by a margin of registers which sole purpose would be to keep mesh values from deletion while shifting EAST or WEST. The main weakness of the coprocessor is its excessive size. Optimizations are however possible, among others by reducing PE's features, number of PEs and mesh size.

VI. CONCLUSION

A cortical visual stimulation system is being developed in order to conduct *in vivo* tests with blind subjects. This paper presented the front-end system architecture that acquires images and transforms them into stimulation site addresses to be sent to the stimulator. These strategies use a lot of low-level image processing operations. The most important of them is the image segmentation, and it is also the basis higher level image processing. It was designed to fit this application's needs: it creates a hierarchical representation of the image and provides sufficient frame rate and stability in results of similar images in a sequence. It performs region merging of initial regions bigger than one pixel, giving coarse but fast results, because the sequential portion only works with blocks of pixels (*blobs*), and uses edge data. Edge extraction and other preprocessing operations are computed by dedicated hardware, exploiting their inherent parallelism. The dedicated hardware has been presented and its performances evaluated.

The complete image processing system architecture has been presented. The design is currently in the form of a set of hardware and software modules made to be integrated in a System on Chip, along with a camera and a range finder. This last item will surely prove to be essential for adequate selection of regions displaying, obtained from segmentation. The system will then be ready for *in vivo* testing. In the meanwhile, it is possible that new results in the field of phosphene patterns recognition and image processing for visual cortical stimulation will inspire other scene representation strategies.

ACKNOWLEDGEMENTS

The authors thank the Canadian Microelectronics Corporation for technical support, and NSERC and NATEQ for financial support.

REFERENCES

[1] BRINDLEY, G.S. and LEWIN, W.S. (1968): "The Sensations Produced by Electrical Stimulation of the Visual Cortex," *Journal of Physiology*, **196**, pp. 479-493.

[2] DOBELLE, W.H., MLADEJOVSKY, M.G., GIRVIN, J.P. (1974): "Artificial vision for the blind: electrical stimulation of visual cortex offers hope for a functional prosthesis", *Science*, **183**, pp. 440-444.

[3] TROYK, P.R. (2001): "Multi-Channel Transcutaneous Cortical Stimulation System," *NIH Final Report (Contract #N01-NS-7-2365).*

[4] NORMANN, R. (1999): "A Neural Interface for a Cortical Vision Prosthesis", *Vision Research*, **39**, pp. 2577-2587.

[5] COULOMBE, J., BUFFONI, L.X., SAWAN, M. (2002): "A Mixed-Signal IC for Multiple Cortical Stimulation Strategies", *Circuits and Systems, MWSCAS-2002*, **2**, pp. 250-253.

[6] MARR, D. (1982): "Vision", W.H. Freeman&Co, New York.

[7] SONKA, M., HLAVAC, V., BOYLE, R. (1999): "Image Processing, Analysis, and Machine Vision", Brooks/Cole Publishing Co, pp. 176-180.

[8] BALLARD, D.H. and BROWN, C.M. (1982): "Computer Vision", Prentice-Hall, Englewood Cliffs, NJ.

[9] ROSENFELD, A., HUMMEL, R.A., ZUCKER, S.W. (1976): "Scene labelling by relaxation operations", *IEEE Transactions on Systems, Man and Cybernetics*, **6**, pp. 420-433.

[10] GILMONT, T., VERIANS, X., LEGAT, J.D., VERAART, C. (1997): "Resolution Reduction by Growth of Zones for Visual Prosthesis", *ICIP*, **A**, pp. 209-302.

[11] SCHMIDT, E.M., BAK, M.J., HAMBRECHT, F.T., KUFTA, C.V., O'ROURKE, D.K., VALLABHANATH, P. (1996): "Feasibility of a Visual Prosthesis for the Blind Based on Intracortical Microstimulation of the Visual Cortex", *Brain* **119** pp. 507-522.

[12] BUFFONI, L.X., COULOMBE, J., SAWAN, M. (2004): "Multiple Image Processing Strategies Dedicated to Visual Cortical Stimulators: A Survey", Submitted to *Artificial Organs.*

[13] CHU, C.C. and AGGARWAL, J.K., (1993): "The Integration of Image Segmentation Maps Using Region and Edge Information", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **15**, pp. 1241-1252.

[14] CHOWDHURY, M.I. and ROBINSON, J.A. (2000): "Improving Image Segmentation Using Edge Information", *Electrical and Computer Engineering, 2000 Canadian Conference on*, **1**, pp. 312-316.

[15] PAVLIDIS, T. and LIOW, Y. (1990): "Integrating Region Growing and Edge Detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **12**, **3**, pp. 225-233.

[16] BARALDI, A. and PARMIGGIANI, F. (1996): "Single Linkage Region Growing Algorithms Based on the Vector Degree of Match", *IEEE Transactions on Geoscience and Remote Sensing*, **34**, pp. 137-148.

[17] CROSS, A.M. and MASON, D.C. (1988): "Segmentation of Remotely-Sensed Images by a Split-and-Merge Process", *Int. J. Remote Sensing*, **9**, **8**, pp. 1329-1345.

[18] COHEN, H.A. and DUONG C.H. (1996): "Gray-scale Image Segmentation Using a Parallel Graph-Theoretic Algorithm", *Australian Pattern Recognition Society, Segment 96, Sydney*, pp. 105-110.

[19] YU, W., FRITTS, J., SUN, F. (2002): "A Hierarchical Image Segmentation Algorithm", *ICME02 Proceedings, 2002 IEEE International Conference on*, **2**, pp. 221-224.

[20] PAVLIDIS, T. (1977): "Structural Pattern Recognition", Springer Verlag, Berlin.

[21] ARM (1999): "AMBA Specifications (Rev2.0)", A, ARM Limited. 230p. Specifications ARM IHI 001 1A.

[22] FOUNTAIN, T.J., "Introducing Local Autonomy to Processor Arrays", in FREEMAN, H. (Ed. 1988): "Machine Vision: Algorithms, Architectures and Systems" (Academic Press Inc., London, UK).

[23] OFFEN, R.J. (1985): "VLSI Image Processing" (McGraw-Hill Book Company, UK), pp. 99-123.

[24] MARRIOTT, P., KRALJIC, I.C., SAVARIA, Y. (1993): "Parallel Ultra Large Scale Engine SIMD Architecture For Real-Time Digital Signal Processing Applications", *ICCD98*, Austin, Texas.

[25] SAVARIA, Y., BOIS, G., POPOVIC, P., WAYNE, A. (1996): "Computational Acceleration Methodologies: Advantages of Reconfigurable Acceleration Subsystems", *Conference on High-Speed Computing, Digital Signal Processing using FPGAs. SPIE's Photonics East*, Boston, MA.

LIST OF FIGURES AND TABLES

at configuration (*instr_cfg*) and route them to the coprocessor at run-time. When the image is done, an interrupt is raised through *proc_nirq*.

Figure 4.5: a) Block diagram of the SIMD coprocessor: squares are the mesh registers, curved arrows indicate wrapping of data around the mesh, and "PE" boxes represent special registers that are overlaid by a PE. b) Internal architecture of a PE. Inputs and outputs are its memory ressources: accumulator (*accm*), register B and C (*regb*, *regc*), local memory (*mem*) and the underlying mesh registers (*mesh1*, *mesh2*).

Figure 4.6: SIMD coprocessor instruction word.

Figure 4.7: Segmentation result: a) original image, b) result after step1 (T_1 = 0.8), c) result after step 2 (T_2 = 0.8), d) result after step 3 (T_3 = 0.7), and e) result after post-processing (T_4 = 10 *blobs*).

Table 4.1: SIMD coprocessor instruction word fields.

Table 4.2: Execution time results of various image processing algorithms for 320x240 images. Column 4 shows the speed-up values for the algorithm that can be executed by the coprocessor, that is, all of them, except segmentation and geometric transform. Segmentation results exclude Sobel calculation. Execution times are those of the hardware (at 33 MHz) when applicable, or else those of the software at 100 MHz.

FIGURES AND TABLES

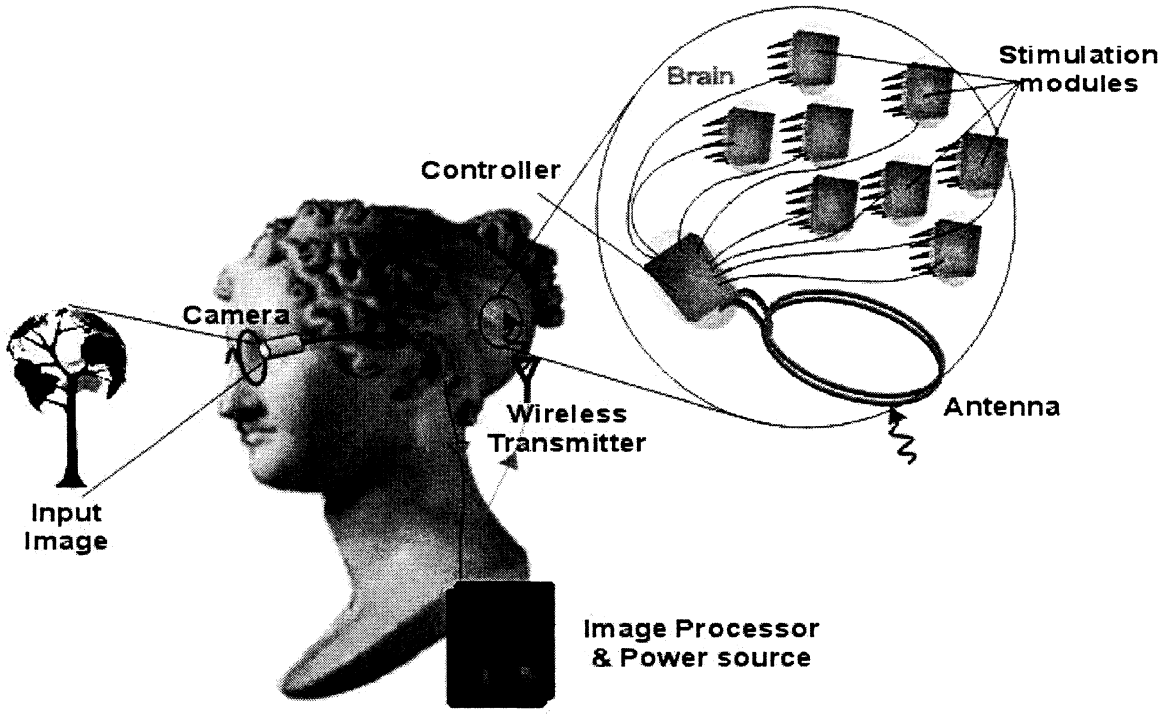


Fig. 4.1 – Complete visual cortical stimulation system. The image processing system is contained in the external controller.

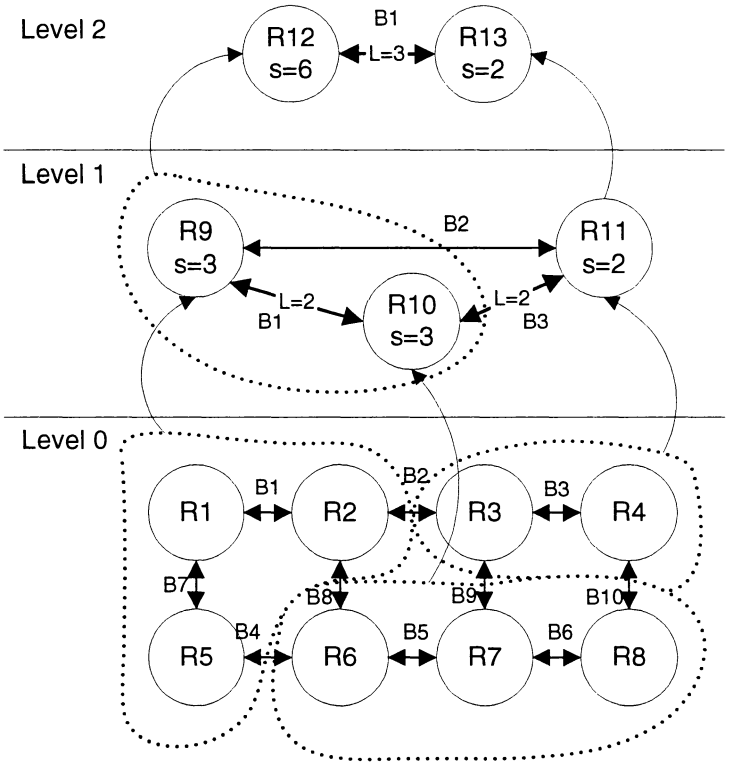


Fig. 4.2 – Planar graph representation of segmentation objects: regions (circles R) with their size S, and boundaries (arcs B) with their length L, at 3 levels of processing. S and L in level 0 equal 1, so they are not displayed. At the end of level 0, regions are grouped (dashed lines), corresponding parent regions are created in level 1, and new boundary objects are computed. Bold arcs represent boundaries whose length is greater than 1. Curved arrows are pointers to the parents.

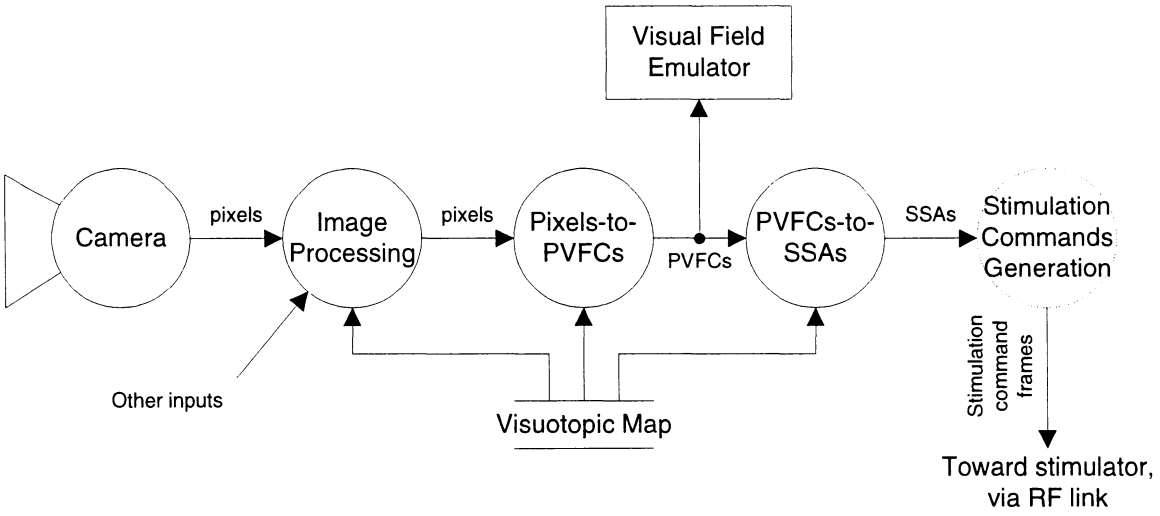


Fig. 4.3 – Dataflow diagram of the image processing system. Other types of input may later be added to the image processing unit.

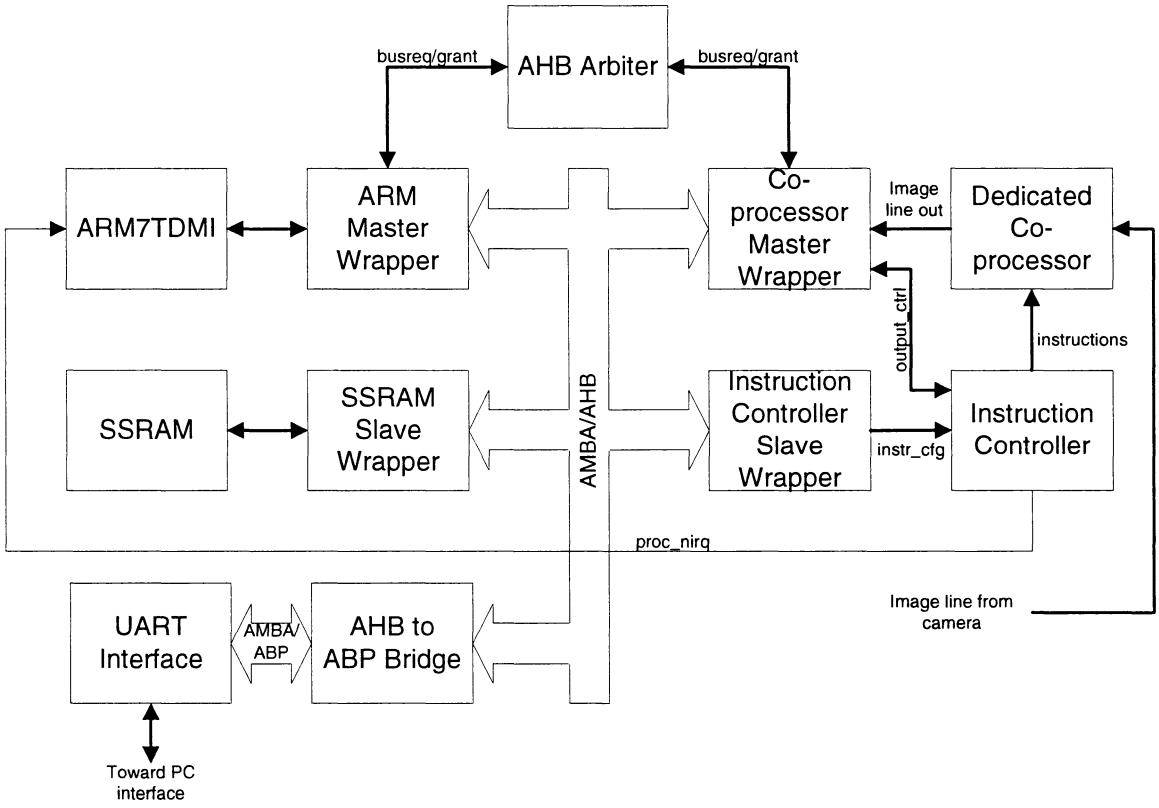


Fig. 4.4 – Block diagram of the image processing system in an AMBA/AHB bus architecture. An image line from camera is inputted in the dedicated coprocessor (on the right), and is outputted to its AMBA bus master-type wrapper, controlled by the instruction controller. The latter module receives coprocessor instructions at configuration (*instr_cfg*) and route them to the coprocessor at run-time. When the image is done, an interrupt is raised through *proc_nirq*.

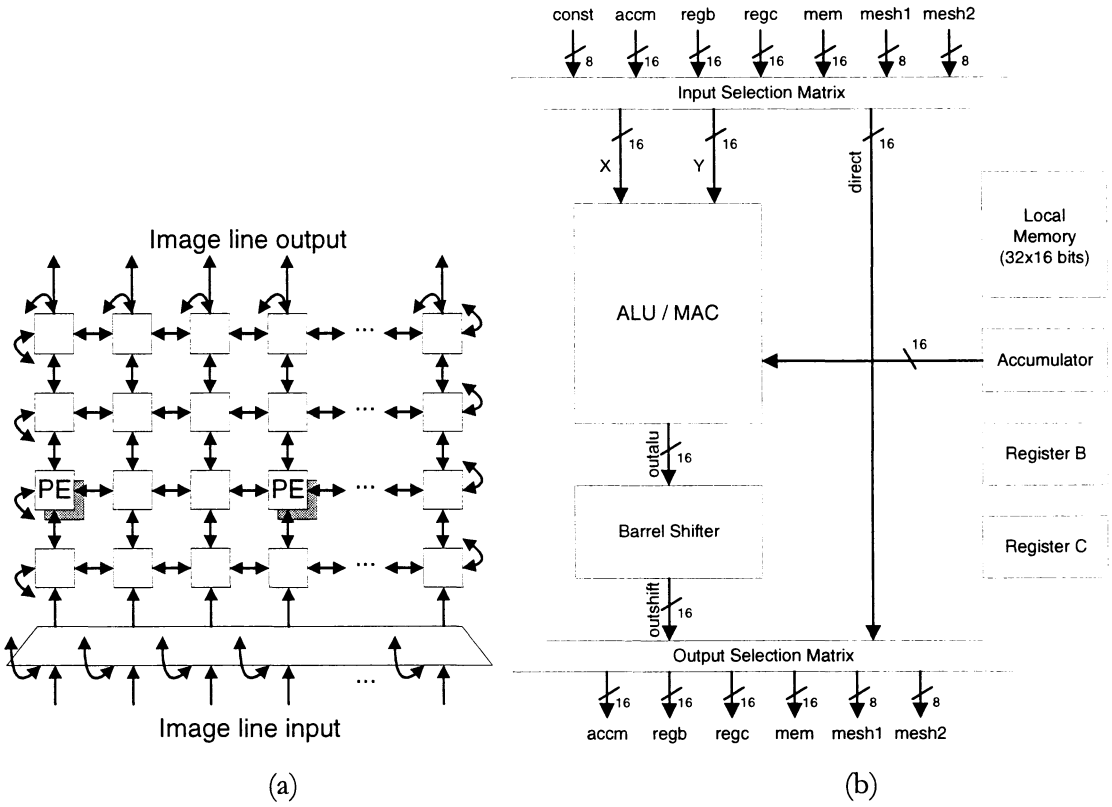


Fig. 4.5 – a) Block diagram of the SIMD coprocessor: squares are the mesh registers, curved arrows indicate wrapping of data around the mesh, and "PE" boxes represent special registers that are overlaid by a PE. b) Internal architecture of a PE. Inputs and outputs are its memory ressources: accumulator (*accm*), register B and C (*regb*, *regc*), local memory (*mem*) and the underlying mesh registers (*mesh1*, *mesh2*).

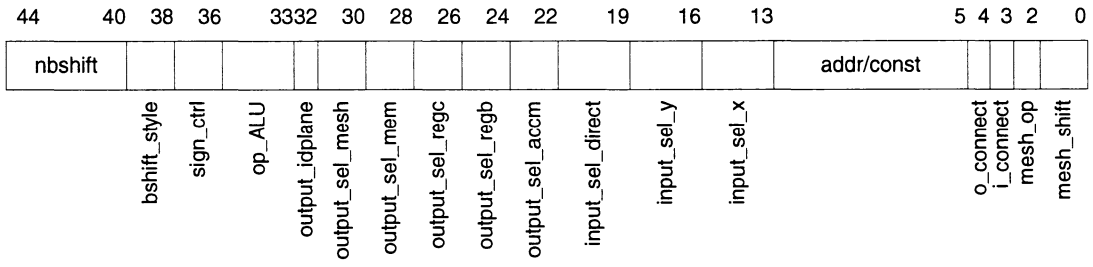


Fig. 4.6 – SIMD coprocessor instruction word.

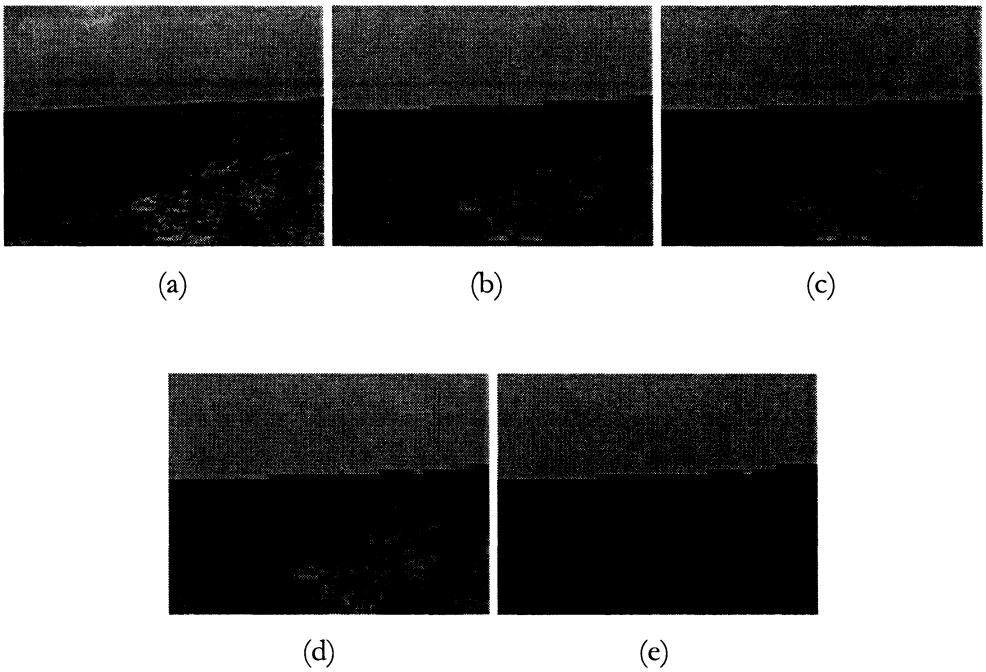


Fig. 4.7 – Segmentation result: a) original image, b) result after step1 ($T_1 = 0.8$), c) result after step 2 ($T_2 = 0.8$), d) result after step 3 ($T_3 = 0.7$), and e) result after post-processing ($T_4 = 10$ *blobs*).

Table 4.1 – SIMD coprocessor instruction word fields.

Field	Nb bits	Possible values	
		Meaning	**Binary**
mesh_shift	2	SHIFT_N	00
		SHIFT_O	01
		SHIFT_S	10
		SHIFT_E	11
mesh_op	2	MESH_IDLE	0
		MESH_SHIFT	1
flag_i_connect	1	WRAP	0
		IO_CONNECT	1
flag_o_connect	1	NO	0
		OUT_READY	1
addr/const	8	0x00 - 0xFF	
input_selx	3	SELIN_CONST	000
		SELIN_MEM	111
input_sely	3	SELIN_MESH1	010
		SELIN_MESH2	011
input_seldirect	3	SELIN_ACCM	100
		SELIN_REGB	101
		SELIN_REGC	110
output_selaccm	2	SELOUT_NONE	00
output_selregb	2	SELOUT_OUTPE	10
output_selregc	2	SELOUT_DIRECT	11
output_selmem	2		
output_selmesh	2		
output_idmesh	1	MESH_1	0
		MESH_2	1
op_alu	3	ALU_NAND	000
		ALU_AND	001
		ALU_NOR	010
		ALU_SLT	101
		ALU_SEQ	011
		ALU_INV	100
		ALU_MAC	111
signctrl	2	SIGN_S_S	00
		SIGN_S_U	01
		SIGN_U_S	10
		SIGN_U_U	11
bshiftstyle	2	SHIFT_LOGICAL	00
		SHIFT_ARITH	01
nbshift	5	[-16, 15]	

Table 4. 2 – Execution time results of various image processing algorithms for 320x240 images. Column 4 shows the speed-up values for the algorithm that can be executed by the coprocessor, that is, all of them, except segmentation and geometric transform. Segmentation results exclude Sobel calculation. Execution times are those of the hardware (at 33 MHz) when applicable, or else those of the software at 100 MHz.

Algorithm	Software Processing Time (nb of cycles)	Hardware Processing Time (nb of cycles)	Effective speed-up	Execution Time (ms)
Local Mean	7446000	2612	950.2	0.079
Threshold	3763238	20293	61.8	0.615
Sobel Magnitude	87520263	49768	586.2	1.508
Sobel Magnitude Mean	94966263	51162	618.7	1.550
Segmentation by boundary melting	12827100	-	-	128.27
Geometric Transform (1000 phosphenes)	54168	-	-	0.54

4.3 Conclusion

Dans ce chapitre ont d'abord été vues les caractéristiques motivant le design de l'algorithme de segmentation, dont le calcul constitue le goulot d'étranglement du système. Ensuite, l'implémentation de tout le système embarqué ainsi que de la segmentation elle-même a été décrite. Leur implémentation respective est inter-reliée, et les deux se justifient mutuellement.

Le système possède une architecture de bus AMBA avec un processeur ARM, pour des raisons qui ont été évoquées. Le processeur s'occupe du maximum de tâches, mais

l'exécution complète du moindre algorithme ne s'accomplit pas dans un temps raisonnable par lui seul. Pour décharger le processeur, une partition logicielle/matérielle a été faite. La fraction la plus apte à être prise en charge par un module de calcul parallèle implémenté en matériel est celle qui s'occupe des opérations de bas-niveau sur les images, comme la convolution par exemple. Un coprocesseur dédié, fortement parallélisé, optimisé pour les convolutions 3x3 tout en étant assez flexible pour exécuter d'autres algorithmes impliquant des pixels voisins, a été conçu et décrit dans l'article. Les résultats montrent le gain de vitesse impressionnant réalisé grâce à l'ajout de ce module. Les algorithmes s'exécutent dans un temps suffisant pour générer des adresses de phosphènes devant être envoyées à l'implant pour chaque image, c'est-à-dire à un rythme de plus de 5 images par seconde.

Plus tard, d'autres modules se joindront à ces modules et des tests *in vivo* pourront être effectués suite à leur intégration. Sur une même puce pourraient se trouver la caméra, un module de synchronisation et de transformation des SSAs en mots de commandes pour l'implant, un module d'ordonnancement pour l'égalisation de la puissance, le module de transmission vers l'implant, une interface avec un PC. De plus, un détecteur de distance et son système de traitement, dispositifs dont la présence serait très souhaitable pour la sélection de régions de phosphènes à activer, conformément aux recommandations faites au chapitre 2, devraient être intégrés. Ces modules sont ou seront le résultat du travail de recherche accompli par d'autres membres de l'équipe PolySTIM.

CONCLUSION

La stimulation visuelle corticale est une avenue très prometteuse dans la récupération de fonctions physiologiques complexes comme la vision. La plupart des travaux ayant été effectués à ce jour concernent l'électronique du stimulateur, les microélectrodes, l'interface tissus-électrode, ou la communication trans-crânienne par lien inductif. Le présent projet s'inscrit dans cet effort de recherche, mais conçoit le problème d'un point de vue plus général : comment générer des percepts, à partir d'un dispositif d'entrée, par exemple une caméra, qui permettent à un usager de comprendre son environnement le plus efficacement possible? Nous avons vu que la réponse est loin d'être triviale. Nous avons vu que les connaissances actuelles dans le domaine biomédical concernant les phosphènes, leur aspect, leur organisation et surtout, lorsqu'ils sont évoqués en même temps, leur capacité à être identifiés comme représentant des formes, sont encore très rudimentaires. La revue de littérature présentée au chapitre 1 et au début du chapitre 2 en fait foi.

Devant ces incertitudes, nous avons été amenés à imaginer diverses stratégies pour transformer des images en patrons de phosphènes compréhensibles, avec un niveau de réalisme variable. Pour visualiser les effets obtenus, un outil logiciel a été créé. Il affiche les patrons de phosphènes d'une manière idéalisée, à partir d'une distribution de coordonnées et de tailles de phosphènes qui, étant donné le manque de données expérimentales réelles, peut être générée de manière aléatoire, en suivant diverses lois de probabilité basées sur les résultats expérimentaux de la disposition des champs réceptifs dans le cortex visuel.

Les algorithmes de traitement d'image constituant les diverses stratégies ont d'abord été implémentés de manière logicielle afin de les évaluer. Un système logiciel complet, faisant le lien entre la caméra et une interface du contrôleur de l'implant, a été réalisé et décrit au chapitre 3. Ce système est assorti d'un logiciel d'interface-usager permettant à un expérimentateur de le contrôler. Son aspect graphique est cependant rudimentaire car les spécifications risquent de changer d'ici la fabrication d'un système complet.

Parmi les traitements d'image choisis, un effort particulier a été mis dans la conception d'un algorithme de segmentation étant donné son utilité cruciale dans cette application. En effet, il sert simultanément à réduire la résolution de l'image de manière efficace et à la découper en zones de mêmes caractéristiques, adressable grâce à sa structure de données évoluée. Avec des données de distance des objets fournies par un éventuel détecteur de distance, seules les zones stratégiques peuvent être affichées sous forme de taches de lumière. Cela constitue une des approches les plus réalistes pour commencer, car les phosphènes sont évoqués en mode « tout-ou-rien », et fournira certainement l'usager avec des indices utiles qui lui permettront de comprendre son environnement.

Implémenté de manière logicielle, cet algorithme ne permet pas le calcul d'un résultat en un temps suffisamment court pour être utilisé avec un stimulateur visuel. Il a fallu le porter en système embarqué, en s'arrangeant pour laisser une part de calcul à un coprocesseur dédié. Or, les diverses stratégies sont composées de traitement élémentaires semblables. Par exemple, l'affichage de zones sélectionnées utilisant la segmentation, ainsi que l'affichage des contours des objets, calculent toutes les deux des convolutions sur l'image. La convolution est une opération qu'il est facile d'envisager comme parallélisable, puisqu'elle s'applique sur un voisinage de pixels pour tous les pixels de l'image. La partition logicielle/matérielle s'en est donc inspirée. Un coprocesseur matériel, flexible mais optimisé pour les convolutions, a été implémenté et le design complet, intégrée dans une architecture de bus AMBA contrôlée par un processeur RISC ARM, a été validé dans un environnement de simulation de co-design (logiciel et matériel simultanément). Les résultats ont montré que les algorithmes sont fonctionnels et atteignent les performances voulues.

Le système de traitement d'image sous forme logicielle a été implémenté principalement pour évaluer la qualité des algorithmes, et également pour vérifier la compatibilité avec le logiciel d'interface-usager. Le système embarqué a quant à lui été développé pour être utilisé pour des tests in vivo. Cependant, plusieurs étapes sont nécessaires avant de s'y rendre. Dans son état actuel, il est remis sous la forme d'un groupe de modules logiciels et matériels codés en VHDL. Idéalement, afin d'être portable, et donc vérifiable puisque ce seront les aptitudes

pour aider à la mobilité qui seront évaluées, le système frontal complet devrait être intégré dans une seule et même puce. Outre les modules qui sont présentés dans ce mémoire, un détecteur de distance et son système de traitement devraient être implémentés, ainsi qu'un module de gestion des deux systèmes, une caméra fonctionnelle, une interface PC et une interface avec le reste du stimulateur visuel cortical. Tous ces modules devraient être intégrés dans un SoC et donc, synthétisés, placés et routés. D'ici ce temps, d'autres résultats, en particulier d'essais cliniques, viendront peut-être confirmer ou infirmer quelques hypothèses qui ont été mises de l'avant dans le présent mémoire, ce qui aura une influence sur ce qu'il aura apporté en termes concrets.

BIBLIOGRAPHIE

[1] ALEXANDER, D.M., SHERIDAN, P., BOURKE, P.D., KONSTANDATOS, O. 1997. « Global and local similarity of the primary visual cortex: mechanisms of orientation preference », *HELNET - International Workshop on Neural Networks.*

[2] ARM 1999. *AMBA Specifications.* Rev2.0. A

[3] ARM. *ARM7TDMI Technical Sheet.* In. ARM. [En ligne].

[4] BAK M., GIRVIN J.P., HAMBRECHT, F.T., KUFT, C.V., LOEB, G.E., SCHMIDT, E.M. 1990. « Visual Sensations Produced by Intracortical Microstimulation of Human Occipital Cortex », *Med Biol Eng Comput.* 28. 257-259.

[5] BALLARD, D.H. and BROWN, C.M. 1982. *Computer Vision*, Prentice-Hall, Englewood Cliffs, NJ.

[6] BANERJEE, S., EVANS, B.L. 2003. « A Novel Gradient Induced Main Subject Segmentation Algorithm for Digital Still Cameras ». *Proc. IEEE Asilomar Conf on Signals, Systems and Computers*, Pacific Grove.

[7] BARALDI, A. and PARMIGGIANI, F. 1996. « Single Linkage Region Growing Algorithms Based on the Vector Degree of Match ». *IEEE Transactions on Geoscience and Remote Sensing*, 34. 137-148.

[8] BARTLETT, J.R., DOTY, R.W. 1980. « An Exploration of the Ability of Macaques to Detect Microstimulation of Striate Cortex ». *Acta Neurobio. Exp.* 40. 713-727.

[9] BOSI, B., BOIS, G., SAVARIA, Y. 1999. « Reconfigurable Pipelined 2-D Convolvers for Fast Digital Signal Processing ». *IEEE Transactions on VLSI Systems*, 7:3.

[10] BOYLE, J., MAEDER, A., BOLES, W. 2003. « Inherent Visual Information for Low Quality Image Presentation ». *APRS Workshop on Digital Image Computing.* p.51-57.

[11 BRAILLE INSTITUTE. Braille Institute of America. In Braille Institute. [En ligne]. http://www.brailleinstitute.org (Page consultée le 7 mars 2004)

[12] BRINDLEY, G.S. and LEWIN, W.S. 1968. « The Sensations Produced by Electrical Stimulation of the Visual Cortex ». *Journal of Physiology* 196. 479-493.

[13] BRINDLEY, G.S. 1972. « The Variability of the Human Striate Cortex ». *J Physiol (London)* 225. 1-3. (cité par [83])

[14] BRINDLEY, G.S. 1973. « Sensory Effects of Electrical Stimulation of the Visual and Paravisual Cortex in Man ». *Handbook of Sensory Physiology.* Berlin :Springer-Veriag. VII:3. 583-94. (cité par [59])

[15] BRINDLEY, G.S., DONALDSON, P.E., FALCONER, M.A., RUSHTON, D.N. 1972. « The Extent of the Region of Occipital Cortex that when Stimulated Gives Phosphenes Fixed in the Visual Field ». *J Physiol* (London) 225. 57-58. (cité par [83])

[16] BUFFONI, L.X., SAWAN, S., COULOMBE, J. 2004. « Design and Test of a Novel Image Processor Dedicated to Cortical Visual Stimulation », submitted to *Med. Biol. Eng. Comp.*

[17] BUFFONI, L.X. 2001. *Conception de la partie numérique d'un implant visuel cortical*, rapport de projet de fin d'études, École Polytechnique de Montréal.

[18] BUFFONI, L.X., COULOMBE, J., SAWAN, M. 2003. « An Image Processing System Dedicated to Cortical Visual Stimulators ». *Electrical and Computer Engineering, 2003. IEEE CCECE 2003. Canadian Conference on.* vol. 3, pp. 1497-1500.

[19] BUFFONI, L.X., COULOMBE, J., SAWAN, M. 2004. « Multiple Image Processing Strategies Dedicated to Visual Cortical Stimulators: A Survey ». Submitted to *Artificial Organs.*

[20] CANADIAN MICROELECTRONICS CORPORATION. 2002. *Soft IP Authoring.*

[21] CARR C. 1999. Lecture 9 : Midbrain. In Wavecrest [En ligne].

[22] CHA, K., HORCH, KW., NORMANN, R.A. 1990. « Simulation of a Phosphene Field Based Visual Prosthesis ». *IEEE International Conference on Systems, Man and Cybernetics.* p.921-923.

[23] CHEN, J., WISE, K.D., HETKE, J.F., BLEDSOE, S.C.Jr. 1997. « A Multichannel Neural Probe for Selective Chemical Delivery at the Cellular Level ». *IEEE Trans. Biomed.* Eng. 44:8. 760-769.

[24] CHOWDHURY, M.I. and ROBINSON, J.A. 2000. « Improving Image Segmentation Using Edge Information ». *Electrical and Computer Engineering, 2000 Canadian Conference on*, 1, p.312-316.

[25] CHU, C.C. and AGGARWAL, J.K. 1993. « The Integration of Image Segmentation Maps Using Region and Edge Information ». *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15, p.1241-1252.

[26] COHEN, H.A. and DUONG C.H. 1996. « Gray-scale Image Segmentation Using a Parallel Graph-Theoretic Algorithm ». *Australian Pattern Recognition Society*, Segment 96, Sydney, p.105-110.

[27] COULOMBE, J., BUFFONI, L.X., SAWAN, M. 2002. « A Mixed-Signal IC for Multiple Cortical Stimulation Strategies ». *Circuits and Systems, MWSCAS-2002*, 2, p.250-253.

[28] CROSS, A.M., MASON, D.C. 1988. « Segmentation of Remotely-Sensed Images by a Split-and-Merge Process ». *Int. J. Remote Sensing*, 9:8, 1329-1345.

[29] DAVEY, K., LUO, L., ROSS, D.A. 1994. « Toward Functional Magnetic Stimulation (FMS) Theory and Experiment ». *IEEE Transactions on Biomedical Engineering.* 41:11. 1024-1030.

[30] DOBELLE, W.H., MLADEJOVSKY, M.G., EVANS, J.R., ROBERTS, T.S., GIRVIN, J.P. 1976. « Braille reading by a blind volunteer by visual cortex stimulation » *Nature* 259. p.111-112.

[31] DOBELLE, W.H., MLADEJOVSKY, M.G., GIRVIN, J.P. 1974. « Artificial vision for the blind: electrical stimulation of visual cortex offers hope for a functional prosthesis ». *Science*, 183, pp.440-444.

[32] DOBELLE, W.H. 2000. « Artificial Vision for the Blind by Connecting a Television Camera to the Visual Cortex ». *ASAIO Journal*, 46. p.3-9.

[33] DUFF, M.J.B. 1983. *Computing Structures for Image Processing*, Academic Press Inc., London.

[34] ENDO, S., TOYAMA, H., KIMURA, Y., ISHII, K., SENDA, M., KIYOSAWA, M., UCHIYAMA, A. 1997. « Mapping visual field with positron emission tomography by mathematical modeling of the retinotopic organization in the calcarine cortex ». *IEEE Transactions on Medical Imaging.* 16:3.

[35] ESSELLE, K.P., STUCHLY, M.A. 1992. « Neural Stimulation with Magnetic Fields : Analysis of Induced Electric Fields ». *IEEE Transactions on Biomedical Engineering.* 39:7. 693-700.

[36] FOERSTER, O. 1929. « Beiträge zur Pathophysiologie des Sehspäre ». *J. Psychol Neurol (Ppz).* Vol 39. 463-85. (cité par [1-59])

[37] FREEMAN, H. 1988. *Machine Vision: Algorithms, Architectures and Systems*, Academic Press Inc., London.

[38] GEKELER F., SCHWAHN H., STETT A., KOHLER K., ZRENNER, E. 2001. « Subretinal Microphotodiodes to Replace Photoreceptor-Function. A Review of the Current State ». *Les Seminaires Ophtalmologiques d'IPSEN.* Tome 12. p.77-95.

[39] GILMONT, T., VERIANS, X., LEGAT, J.D., VERAART, C. 1997. « Resolution Reduction By Growth of Zones For Visual Prosthesis ». *ICIP*, vol. A, p.209-302.

[40] HALLUM, L.E., TAUBMAN, D.S., SUANING, G.J., MORLEY, J.W., LOVELL, N.H. 2003. « A Filtering Approach to Artificial Vision: a Phosphene Visual Tracking Task ». *Proceedings of the World Congress on Medical Physics and Biomedical Engineering (WC2003)*, August 24-29, 2003, Sydney, Australia.

[41] HAMBRECHT, T., BAK, M., KUFTA, C.V., O'ROURKE, D.K., SCHMIDT, E.M., VALLABHANATH, P. 1992. « Feasibility of a Visual Prosthesis for the Blind Utilizing Intracortical Microstimulation ». *4th Vienna International Workshop on Functional Electrostimulation*, p1-8.

[42] HAMBRETCH, F.T. 1995. « Visual Prostheses Based on Direct Interfaces with the Visual System ». *Baillière's Clinical Neurology*, 4:1. 147-164

[43] HARVEY, J.F., ROY, M., SAWAN, M. 1999. « Visual Cortex Stimulator Prototype Based on Mixed-Signal Technology Devices ». submitted to *IFESS*.

[44] HAYES, J.S., YIN, V.T., PIYATHAISERE, D., WEILAND, J.D., HUMAYUN, M.S., DAGNELIE, G. 2003. « Visually Guided Performance of Simple Tasks Using Simulated Prosthetic Vision ». *Artificial Organs.* 27:11. 1016-1028.

[45] HUBEL, D.H. 1995. *Eye, Brain, and Vision.* New York : Scientific American Library, 240 p.

[46] HUBEL, D.H., WIESEL, T.N. 1959. « Receptive Fields of Single Neurones in the Cat's Striate Cortex ».. *J. Physiol. (Lond.)* 148. 574-591.

[47] HUBEL, D.H., WIESEL, T.N. 1962. « Receptive Fields, Binocular Interaction and Functional Architecture in the Cat's Visual Cortex ». *J. Physiol. (Lond.)* 160. 106-154.

[48] HUBEL, D.H., WIESEL, T.N. 1968. « Receptive Fields and Functional Architecture of Monkey Striate Cortex ». *J Physiol.* 195. 215-243.

[49] HUMAYUN, M. 1999. « Pattern Electrical Stimulation of the Human Retina ». *Vision Research.* 39. 2569-2576.

[50] KANDA, H., YAGI, T., ITO, Y., TANAKA, S., WATANABE, M., UCHIKAWA, Y. 1999. « Efficient Stimulation Inducing Neural Activity in Retinal Implant ». *Int. Conf. on Systems, Man and Cybernetics, Proceedings of IEEE.* IV, 409-413.

[51] KANDEL, E.R., SCHWARTZ, J.H., JESSEL, T.M. 2000. *Principles of Neuroscience*, McGraw Hill, New York, 4th ed.

[52] KRAUSE F. 1924. « Die Sehbahnen in Chirurgischer Beziehung und die Faradische Reizung des Sehzentrums ». *Klin Wschr.* 3. 1260-1265. (cité par [1-59])

[53] LAMME, V.A.F., SUPÈR, H., LANDMAN, R., ROELFSEMA, P.R., SPEKREIJSE, H. 2000. « The Role of Primary Visual Cortex (V1) in Visual Awareness ». *Vision Research*, 40. 1507-1521.

[54] MARR D. 1982. *Vision*, New York : W.H. Freeman&Co.

[55] MARRIOTT, P., KRALJIC, I.C., SAVARIA, Y. 1998. « Parallel Ultra Large Scale Engine SIMD Architecture For Real-Time Digital Signal Processing Applications ». *ICCD98*, Austin, Texas.

[56] MATHWORKS. 1984-2004. *Matlab Product Documentation : External Interfaces*, Version 6, In. Matlab Support. [En ligne]. http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/apiext.pdf (page consultée le 6 juillet 2004)

[57] MAYNARD, E. 2001. « Visual Prostheses ». *Annual Review of Biomedical Engineering.* 3, p.145-168.

[58] McDERMOTT, H. 1989. « An Advanced Multiple Channel Cochlear Implant ». *IEEE Trans. Biomed. Eng.* 36:7. 789-797.

[59] NAEYAERT, K. 1990. *La cécité et la déficience visuelle au Canada*. [Ottawa] : Statistiques Canada. 3 : no. cat.82-615.

[60] NORMANN, R.A. 1999. « A Neural Interface for a Cortical Vision Prosthesis ». *Vision Research*. 39. 2577-2587.

[61] OFFEN, R.J. 1985. *VLSI Image Processing*, McGraw-Hill Book Company, UK.

[62] OSBERGER, W., MAEDER, A. 1998. « Automatic identification of perceptually important regions in an image using a model of the human vision system ». *14th International Conference on Pattern Recognition*, Brisbane, Australia, p701-704.

[63] PAVLIDIS, T. 1977. *Structural Pattern Recognition*, Springer Verlag, Berlin.

[64] PAVLIDIS, T., LIOW, Y. 1990. « Integrating Region Growing and Edge Detection ». *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:3, p.225-233.

[65] PENFIELD, W., JASPER, H. 1954. « Epilepsy and the Functional Anatomy of the Human Brain ». London : Churchill. 116:26 . p404-406.

[66] PERUS, M. 2001. « Visual Memory ». *Cognitive neuroscience*, p77-79.

[67] PRITZKER INSTITUTE OF MEDICAL ENGINEERING. IntraCortical Visual Prosthesis. In Neuroprosthetic Research Laboratory. [En ligne]. http://neural.iit.edu/visualprosthesis2.htm (Page consultée le 15 avril 2002)

[68] REGIMBAL, S., LEMIRE, J.F., SAVARIA, Y., BOIS, G., ABOULHAMID, E.M., BARON, A. 2002. « Aspect Partitioning for Hardware Verification Reuse ». *IWSOC2002*, p.49-58.

[69] RONNER, S.F., LEE, B.G. 1983. « Excitation of Visual Cortex Neuron by Local Intracortical Microstimulation ». *Experimental Neurology*. 81:1. 376-395.

[70 ROSENFELD, A., DE LA TORRE, P. 1983. « Histogram concavity analysis as an aid in threshold selection ». *IEEE Transactions on Systems, Man and Cybernetics*, 13:3, 231-235.

[71] ROSENFELD, A., HUMMEL, R.A., ZUCKER, S.W. 1976. « Scene labelling by relaxation operations ». *IEEE Transactions on Systems, Man and Cybernetics*. 6, p.420-433.

[72] ROY M. 1999. *Conception et réalisation d'un prototype d'implant visuel cortical*, Mémoire de maîtrise en génie électrique, École Polytechnique de Montréal.

[73] SAVARIA, Y., BOIS, G., POPOVIC, P., WAYNE, A. 1996. « Computational Acceleration Methodologies: Advantages of Reconfigurable Acceleration Subsystems ». *Conference on High-Speed Computing, Digital Signal Processing using FPGAs.* SPIE's Photonics East, Boston, MA.

[74] SAWAN, M., DUVAL, F., HASSOUNA, M.M., LI, J.S. ELHILALI, M.M., LACHANCE, J., LECLAIRE, M., POURMEHDI, S. et MOUINE, J. 1992. « Computerized Transcutaneous Control of a Multichannel Implantable Urinary Prosthesis ». *IEEE Trans. Biomed. Eng.* 39:6. 600-609.

[75] SAYPOL, J.M., ROTH, B.J., COHEN, L.G., HALLETT, M. 1991. « A Theoretical Comparison of Electric and Magnetic Stimulation of the Brain ». *Annals of Biomedical Engineering.* 19:3. 317-328.

[76] SCHMIDT, E.M., BAK, M.J., HAMBRECHT, F.T., KUFTA, C.V., O'ROURKE, D.K., VALLABHANATH, P. 1996. « Feasibility of a Visual Prosthesis for the Blind Based on Intracortical Microstimulation of the Visual Cortex ». *Brain.* 119. 507-522.

[77] SCHWARTZ, E.L. 1980. « Computational Anatomy and Functional Architecture of Striate Cortex: A Spatial Mapping Approach to Perceptual Coding ». *Vision Res.*, 20, p.645-669.

[78] SELIGMAN, L.J. 1982. « Physiological Stimulators : From Electric Fish to Programmable Implants ». *IEEE Transactions on Biomedical Engineering.* BME-29:4. 270-284.

[79] SONKA, M., HLAVAC, V., BOYLE, R. 1999. *Image Processing, Analysis, and Machine Vision.* Brooks/Cole Publishing Co. p176-180.

[80] STENTIFORD, F. 2001. « An estimator for visual attention through competitive novelty with application to image compression ». *Picture Coding Symposium 2001*, 25-27 April 2001, Seoul, Korea.

[81] SWINDALE, N.V. 1996. « The development of topography in the visual cortex: a review of models ». *Network.* 7. 161-247.

[82] TKACZYK, E. 2001. « Pressure Hallucinations and Patterns in the Brain ». *Morehead Electronic Journal of Applicable Mathematics*, Issue 1.

[83] TROYK, P., BAK, M., BERG, J., BRADLEY, D., COGAN, S., ERICKSON, R., KUFTA, C., McCREEDY, D., SCHMIDT, E., TOWLE, V. 2003. « A Model for Intracortical Visual Prosthesis Research ». *Artificial Organs.* 27:11. 1005-1015.

[84] TROYK, P.R. 2001. *Multi-Channel Transcutaneous Cortical Stimulation System.* NIH Final Report (Contract #N01-NS-7-2365).

[85] VERAART, C., RAFTOPOULOS, C., MORTIMER, J.T., DELBEKE, J., PINS, D., MICHAUX, G., VANLIERDE, A., PARRINI, S., WANET-DEFALQUE, M.C. 1998. « Visual Sensations Produced by Optic Nerve Stimulation Using an Implanted Self-Sizing Spiral Cuff Electrode ». *Brain Research.* 813 p.181-186.

[86] WARREN, D.J., FERNANDEZ, E., NORMANN, R.A. 2001. « High-Resolution Two-Dimensional Spatial Mapping of Cat Striate Cortex Using a 100-Microelectrode Array ». *Neuroscience.* 105:1. 19-31.

[87] WYATT, J.L., RIZZO, J.F. 1996. « Ocular Implants for the Blind ». *IEEE Spectrum.* 33, 47-53.

[88] YU, W., FRITTS, J., SUN, F. 2002. « A Hierarchical Image Segmentation Algorithm ». *ICME02 Proceedings, 2002 IEEE International Conference on*, 2, p.221-224.

ANNEXE A

COMPLÉMENT D'INFORMATION SUR LA VERSION LOGICIELLE DU SYSTÈME DE TRAITEMENT D'IMAGE

A.1 Architecture générale

L'architecture logicielle de la version logicielle du système de traitement d'image a été présentée au chapitre 3. Cependant davantage de détails doivent être donnés afin de permettre efficacement la suite des travaux, c'est-à-dire l'utilisation de la structure existante et le choix des algorithmes à tester pour la génération de patrons de phosphènes.

Le logiciel est constitué de 4 grands morceaux :

- La capture d'image
- Le traitement d'image en temps réel
- La communication sérielle avec logiciel d'interface
- L'affichage des patrons de phosphènes en temps réel

Au centre de tout cela se trouve la boucle principale de fonctionnement, ainsi que le dialogue principal. Ce dernier est inclus dans la classe CSti_mfcDlg, un objet unique qui est créé dans la fonction main() et qui gère les messages envoyés par le système d'exploitation, c'est-à-dire Windows. Cette classe n'est pas portable sur un autre système. Sa fonction d'initialisation appelle une fonction de la classe CFrameBuffersIO afin d'initialiser la caméra, et ensuite elle lance la boucle de la classe CBclProc.

La classe CBclProc contient la boucle principale. Elle possède une fonction d'initialisation qui crée les objets système permettant la mise en place d'un fil d'exécution parallèle (*thread*). Lorsque celui-ci est lancé, on lui spécifie une fonction de départ (boucleProc), qui est une fonction qui boucle à l'infini.

Voici les tâches que cette fonction accomplit pendant un tour de boucle :

1. Vérification du buffer de communication d'entrée, géré par la classe de communication CHighLevelComm.
2. Gestion de l'arrêt de la boucle, commandé par un événement signalé par le dialogue principal.
3. Capture d'une image de la caméra, par le biais de la classe CFrameBuffersIO.
4. Traitement de l'image, par la classe CImageProcessing.
5. Si désiré, envoi des PVFCs vers le programme d'interface par le biais de la classe CHighLevelComm.
6. Si désiré, affichage de l'image sous forme de phosphènes via la classe CVFEMatlab.

Toutes les classes qui viennent d'être mentionnées sont des classes dont un seul objet global est instantié au démarrage. Elles gèrent chacune un des grands morceaux du programme. La figure A.1 est un diagramme de classe, dans lequel on voit la classe CBclProc qui appelle des fonctions des objets globaux. Chacun de ces objets est constitué ou dérivé d'autres classes. Les sections suivantes décrivent l'implémentation de chacun de ces objets. La prochaine section commence avec la capture d'image.

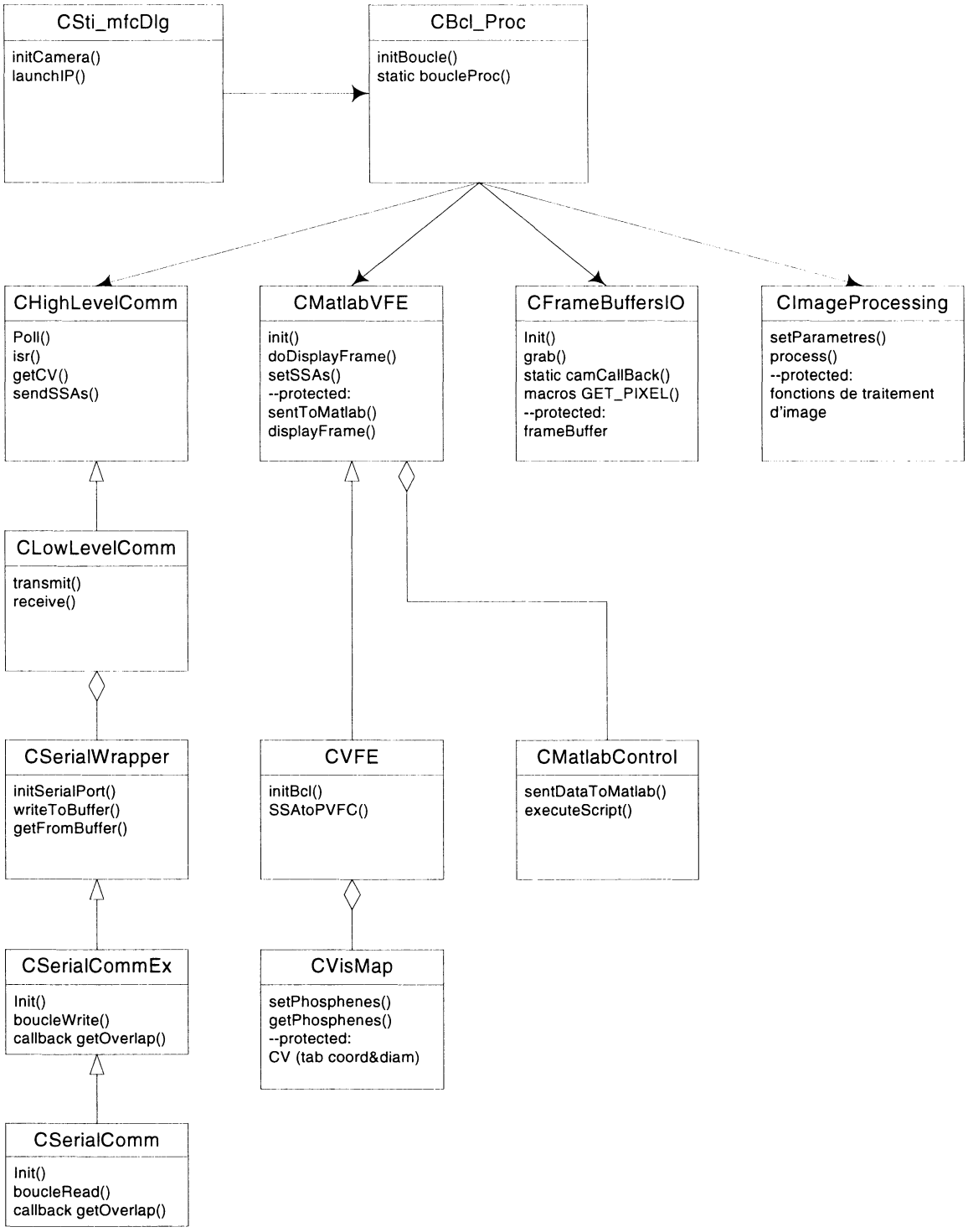


Figure A.1 – Diagramme de classe de la version logicielle du système de traitement d'image

A.2 Capture d'image de la webcam

La capture d'image est orchestrée par la classe CFrameBuffersIO. Celle-ci possède toutes les fonctions de gestion de la caméra, qui est pour l'instant une simple webcam. La fonction initFrameGrabber() est doit être appelée avant la capture. Elle cherche une caméra branchée sur le système, appelle un dialogue de configuration (pour la résolution entre autres), connecte le tampon d'image interne à une fenêtre de visualisation, et enfin spécifie une fonction dynamique (*callback*), qui est appelée par le système dès qu'une image a fini d'être capturée.

La fonction dynamique, frameCallback(), reçoit un pointeur à une structure, de laquelle les valeurs des pixels de l'image sont extraites. Elles sont copiées dans un tableau défini et alloué dans la classe. La fonction appelle ensuite la fonction process() de la classe CImageProcessing, qui modifie directement les pixels de ce tableau. Lorsque process() retourne, les valeurs des pixels du tableau, qui sont maintenant traités, sont recopiés dans le tampon interne d'image, ce qui fait que l'image qui apparaît à l'écran est traitée.

La classe définit des macros d'accès aux pixels, écrites en commandes de préprocesseur. Ainsi, même si la capture d'image change, par exemple dans le cas où le code de traitement d'image est porté sur une autre plateforme, celui-ci restera fonctionnel en autant que les macros appropriées soient redéfinie.

Enfin, la classe possède la faculté de charger et sauvegarder des images directement sur le disque, à des fins de débogage. Cela se fait via la classe Crecorder. Un maximum de 100 images peuvent être sauvegardées dans une séquence. Il est à noter que la fréquence de rafraîchissement apparaît à ce moment moins rapide, étant donné le temps nécessaire à la sauvegarde.

A.3 Traitement d'image en temps réel

La classe CImageProcessing exécute le traitement d'image. Sa fonction publique est process(), et elle appelle les fonctions de traitement internes. Il s'agit d'une classe de développement, où un grand nombre de traitements ont été implémentés afin d'être évalués en temps réel.

Certains traitements nécessitent un tampon d'image temporaire. Celui-ci est alloué lorsque la variable de préprocesseur correspondante est égale à 1. Des macros d'accès lui sont également associées. Sinon, les algorithmes utilisent le tampon d'image principal, dont les macros d'accès sont définies dans CFrameBuffersIO.

Des ressources sont également allouées pour les opérations de transformation géométrique, qui transforment les pixels de l'image finale en adresse de phosphènes (SSA). Dans le cas de l'affichage par le VFE, que ça soit par le VFE inclus dans ce programme ou bien le VFE du logiciel d'interface, accédé par communication sérielle, la boucle principale va chercher ces SSAs via la fonction getPtrIntensitesPh(), et les place dans le buffer de communication.

Le tableau A.1 est une liste des traitements qui ont été implémentés en C et qui se trouvent dans cette classe.

Tableau A.1 – Liste des traitements implémentés dans la classe CImageProcessing.

Pré-traitement
Filtrage médian
Moyennage local
Filtrage gaussien
Égalisation d'histogramme
Seuil statique
Seuil dynamique optimal
Détection des contours
Détection des contours par différence de Gaussiennes
Détection des contours directionnelle Sobel
Détection des contours module du masque Sobel
Segmentation
Segmentation Growth of zones
Segmentation par fusion de régions
Segmentation par fusion de régions et détection de contours
Segmentation par fonte de contours
Post-traitement
Sélection des régions selon diverses stratégies
Adaptation des paramètres de segmentation
Analyse automatique des régions
Transformation géométrique
Transformation géométrique à partir des pixels
Transformation géométrique à partir d'un graphe
Affichage
Affichage des régions
Affichage des contours
Affichage des contours filtrés

A.4 Communication sérielle avec logiciel d'interface

La communication est accédée par la classe de CHighLevelComm. Cette classe est dérivée de la classe CLowLevelComm, et constitue une enveloppe de haut-niveau pour le transfert de paramètres d'algorithmes de traitement d'image, de la carte visuotopique, et des SSAs devant être affichés par le programme d'interface usager. La méthode de communication lui est transparente. La fonction poll(), appelée par la boucle principale, vérifie si des données ont été reçues. Elle appelle la fonction isr() qui décortique ces données et gère le tampon de réception.

La classe CLowLevelComm contient des méthodes de bas-niveau : gestion des tampons de données, fonctions d'écriture et de lecture. La communication sérielle a été choisie pour cette application, pour des raisons qui ont été données au chapitre 3. Elle contient et appelle les méthodes de l'objet de la classe CSerialCommWrapper, qui enveloppe la communication sérielle.

La classe CSerialCommWrapper contient des fonctions de transfert de données, qui appellent elle-mêmes des fonctions des classes desquelles elle est dérivée, soit successivement CSerialCommEx et CSerialComm. CSerialCommEx est une extension de la classe CSerialComm, implémentant un tampon d'envoi, alors que cette dernière a été écrite simplement pour gérer un tampon de réception.

Ces deux classes créent chacune pour elle-même un fil d'exécution parallèle qui écrit constamment ce qu'il y a dans le tampon de données sur le port série, et inversement. Le port série fonctionne en mode Overlap, c'est-à-dire de manière asynchrone. La boucle de transfert appelle une commande du pilote de périphérique du port, WriteFile(), qui retourne immédiatement, et attend ensuite que l'événement Overlap soit signalé. Le signalement de cet événement est également géré dans une autre fonction de la classe, liée au pilote lors de l'ouverture du port. Lorsqu'il est signalé, la boucle analyse le résultat du transfert et envoie une message d'erreur le cas échéant.

A.5 Affichage des patrons de phosphènes en temps réel

L'affichage des patrons de phosphènes se fait normalement dans le logiciel d'interface usager, après avoir reçu les SSAs du STI. Mais étant donnée la grande latence introduite par la communication, il s'est avéré plus simple pour le développement d'algorithmes de traitement d'image de l'implémenter directement dans le STI, sous une condition de préprocesseur.

Cela se fait via la classe CMatlabVFE. Celle-ci est dérivée de la classe virtuelle CVFE qui fixe les fonctions nécessaires pour l'interface d'un VFE, implémenté d'une quelconque manière, au reste du STI. Le VFE implémente une boucle d'affichage qui s'exécute de manière parallèle, dans son propre fil d'exécution. La fonction doDisplayFrame(), appelée par la boucle principale, copie les SSAs retournés par le traitement d'image dans un tampon interne. La classe contient donc ce tampon interne, ainsi qu'une méthode de transformation en coordonnées visuelles (PVFCs). Cette méthode doit avoir connaissance de la carte visuotopique, c'est pourquoi elle la possède, sous la forme d'un objet de la classe CCarteVis. Quant à l'affichage, il se fait dans un engine Matlab. Le contrôle de Matlab est géré par la classe CMatlabControl, membre de CMatlabVFE.

La boucle d'affichage roule constamment, prend les données de son tampon interne, les transforme en PVFC et les transfère vers Matlab. Il appelle ensuite une fonction de CMatlabControl qui fait exécuter un script Matlab, placé dans un répertoire qui lui est accessible. Ce script dessine des cercles pleins aux positions, et selon le diamètre et l'intensité, indiqués par les PVFCs et la carte visuotopique.

Toutes les sources sont disponibles dans l'annexe C.

A.6 Note au sujet du logiciel d'interface

Le logiciel d'interface est remis sous une forme simple, car il n'est pas encore pertinent de raffiner son apparence et ses fonctionnalités puisque nous sommes encore loin des

tests in vivo permettant d'évaluer la représentation du monde sous forme de patrons de phosphènes. Dans son état actuel, il contient la carte visuotopique (CCarteVis), le VFE dont la description se trouve dans la section précédente, ainsi que les mêmes classes de communication sérielle présentées plus haut. Son interface très simple permet néanmoins d'envoyer des données brutes au STI, de même que la carte visuotopique, et permet également de fixer un fanion indiquant qu'il s'attend à recevoir des SSAs. Il les affiche alors dans le VFE.

ANNEXE B

COMPLÉMENT D'INFORMATION SUR LE COPROCESSEUR DÉDIÉ

B.1 Objectifs du coprocesseur dédié

- La machine à concevoir devrait être capable d'effectuer des convolutions d'image rapidement, dans le but de :
  - Appliquer des masques de rehaussement d'image (pré-traitement);
  - Exécuter des algorithmes de segmentation et de détection des contours;
  - Calculer des statistiques sur les images qui passent (ex. histogramme).
- Le traitement devrait se faire au passage des données, à la manière d'une machine systolique, avec une grande bande passante. La machine pourrait éventuellement être incorporée à l'électronique d'une caméra CMOS et fournir des images prétraitées au reste du système.
- Elle devrait être assez flexible pour permettre l'implémentation de divers algorithmes, avec des paramètres différents.
- Elle devrait être rapide. Un système de traitement d'image complet pour un SVC, effectuant des tâches complexes de haut niveau, devrait avoir à sa disposition des images prétraitées à un grand débit.
- Elle devrait consommer peu de courant afin d'être portable.

B.2 Méthodologie de conception

1. Étude des algorithmes de traitement d'image de bas-niveau :
   a. Seuillage,
   b. Convolution 3x3,

   c. Convolution 5x5,
   d. Convolution 7x7,
   e. Soustraction d'image,
   f. Détection de passage par zéro,
   g. Laplacien d'une Gaussienne (LoG),
   h. Approximation par une différence de Gaussienne (DoG),
   i. Détection de contours avec une DoG et passage par zéro;
2. Design de l'architecture d'un module de calcul parallèle SIMD;
3. Modèle de haut-niveau en C++;
4. Détermination des tailles de registres, opérations de l'UAL, contrôle du signe, décaleur de Barillet;
5. Écriture d'un compilateur symbolique pour le modèle de haut niveau;
6. Validation des algorithmes avec le modèle de haut niveau;
7. Évaluation des performances des algorithmes implémentés pour la machine (en nombre d'instructions)
8. Sauvegarde des instructions en format logique;
9. Modèle de niveau RTL en VHDL;
10. Simulation des sous-blocs (UAL et décaleur de Barillet);
11. Simulation de l'ensemble avec un banc d'essai de haut-niveau en *e* (Specman);
12. Synthèse du coprocesseur avec Synplify (pour implémentation dans un FPGA Xilinx Virtex2000E);
13. Optimisations;
14. Implémentation d'un système embarqué destiné à la plate-forme ARM7TDMI;
15. Validation en temps réel grâce à la plate-forme de prototypage rapide et à l'émulateur de champ visuel.

Les étapes précédentes ont été effectuées jusqu'à la validation en temps réel grâce à la plateforme ARM7TDMI (étape 15). Celle-ci montre des problèmes de fonctionnement lorsqu'elle est surmontée d'un circuit de communication USB. D'autre part, ce coprocesseur trouvera probablement sa place dans un système embarqué dont le processeur est un DSP.

Les DSP de TI sont beaucoup plus rapides, et consomment moins de puissance qu'un processeur ARM.

Il est connu qu'un système de traitement d'image pour un SVC doit être capable de faire des opérations de bas-niveau. Des opérations classiques de ce genre ont été implémentées avec Matlab à l'étape 1. Une machine dédiée a ensuite été conçue en suivant les étapes classiques de développement d'un bloc matériel *soft IP* [20] (étapes 2 à 11). Elle a été validée avec un banc d'essai en *e*, et synthétisée pour un FPGA afin de relever les caractéristiques de performances.

Le traitement d'image de bas-niveau se révélera insuffisant pour obtenir des patrons de phosphènes compréhensibles par un aveugle, avec un nombre raisonnable d'électrodes. Un système plus poussé a été conçu en utilisant un processeur pour exécuter les tâches de haut-niveau.

Une question essentielle est : pourquoi concevoir une machine de calcul dédiée pour les opérations de bas-niveau, si celles-ci peuvent être exécutées par un processeur dans un système plus complet, de plus haut-niveau, en respectant le temps réel? Les résultats du chapitre 4 montrent qu'en utilisant le processeur ARM uniquement, le temps réel ne pouvait pas être atteint.

B.3 Résultats

B.3.1 Explication succincte du fonctionnement de la machine

L'accélérateur de calcul fonctionne à la manière des processeurs SIMD (*Single-Instruction Multiple-Data*) : plusieurs élément de traitement (*processing elements – PE*) exécutent la même instruction en parallèle, sur des données différentes. Il consiste en une grille 2D d'éléments de mémoire de 8 bits, à l'intérieur de laquelle traverse l'image. Chaque élément est muni d'un multiplexeur recevant les données des éléments voisins, de telle sorte que l'image peut être décalée dans toutes les directions. L'ensemble de ces MxN éléments est référé à une *grille*

dans le code, et à une *grille* dans le texte. La grille est circulaire, et la machine en comporte deux, pour stocker des résultats intermédiaires. Dans l'implémentation présente, M = 320 et N = 4. Ainsi, cela devrait être suffisant pour que la machine traite des images entières (320x240), sans avoir besoin de la séparer en plusieurs fenêtres.

Les PEs sont très simples. Ils possèdent quelques fonctions logiques, ainsi qu'une unité de multiplication-accumulation (MAC). La configuration actuelle en comporte 40, disposés de manière stratégique sur la grille. Le gain de vitesse théorique est donc de 40.

L'entrée de la machine est une ligne d'image, de 320x8 bits. L'image est traitée au passage, à la manière d'une machine systolique.

L'architecture détaillée est présentée à la section B.4.

B.3.2 Traitements d'image

Les traitements d'images de bas-niveau qui ont été implémentés sont décrits dans cette sous-section. La section B.6 décrits plus en détail la façon dont ils sont exécutés par la machine d'accélération.

B.3.2.1 Seuillage

Le seuillage consiste à binariser les pixels d'une image autour d'un seuil. La figure suivante montre une binarisation autour de 100 (sur 255).

(a) (b)

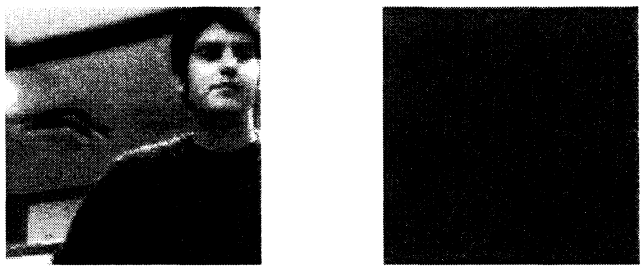
Figure B.1 – Seuillage : a) image originale, b) image binarisée.

La stratégie d'implémentation sur la machine consiste à insérer 2 lignes de l'image, et à calculer le résultat pour chaque pixel en décalant l'image vers la gauche (ouest) et ensuite vers la droite (est). Chaque PE s'occupe de 8 pixels par ligne.

Nous définissons la métrique suivante pour évaluer la performance des algorithmes en termes du nombre d'instructions. Une image prend 240 cycles pour traverser la machine, sans traitement. Nous calculons le nombre d'instructions par image, moins 240. Ou plus simplement, le nombre d'instructions par ligne, moins 1. Dans le cas du seuillage, nous obtenons (18 instructions / 2 lignes) –1, soit 8 instructions / ligne. Puisqu'il y a 40 PEs et 320 pixels par lignes, l'algorithme prend 1 instruction par pixel, et est donc optimal.

B.3.2.2 Laplacien d'une Gaussienne (LoG)

Un traitement avec le Laplacien d'une Gaussienne est une convolution avec un masque gaussien dérivé 2 fois et normalisé. Nous prenons ici un masque 5x5. L'image suivante montre le résultat. Les valeurs peuvent être négatives, si bien que tous les pixels ont été additionnés à 128. Ce genre de traitement est utilisé pour la détection de contours, car ceux-ci se retrouvent aux passages par zéro.



(a) (b)

Figure B.2 – Laplacien d'une Gaussienne : a) image originale, b) image traitée.

La convolution 5x5 comporte 25 multiplications-accumulations, qui sont exécutées en un cycle par la machine. L'algorithme développé (décrit en détail en annexe) a une efficacité de 26 instructions par pixel. Un léger *overhead* est dû au fait qu'il faut replacer les pixels de l'image avant de traiter la ligne suivante. Une version améliorée de l'algorithme pourrait ramener l'efficacité à 25 instructions.

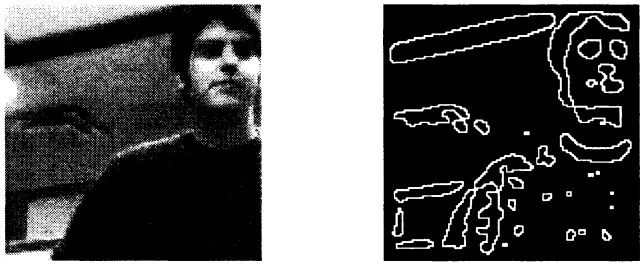
B.3.2.3 Approximation par différence de Gaussienne (DoG)

Il a été démontré que le Laplacien d'un Gaussienne à grand σ (nécessaire pour l'élimination des contours non significatifs) s'approxime très bien en soustrayant deux images lissées avec des Gaussiennes à différents σ [54]. De plus, le lissage successif d'une image est équivalent au lissage de l'image originale avec un facteur $k\sqrt{2}$. Pour obtenir des résultats intéressants, nous avons effectué un lissage avec un masque 7x7, 6 fois. Une convolution 7x7 comporte 49 multiplications-accumulations et est effectuées en 50 instructions / pixel. Les résultats successifs sont sauvegardés dans des grilles différentes, si bien qu'à la fin, la grille 2 contient l'image de niveau 5, et la grille 1 contient l'image de niveau 6. Ces deux images seront plus tard soustraites l'une de l'autre afin de détecter les contours.

B.3.2.4 Détection des passages par zéro

Pour la détection des passages par zéro, il s'agit de faire passer une fenêtre 2x2. Si une valeur à l'intérieur de cette fenêtre comporte un signe différent, un point de contour est associé au

coin supérieur gauche. L'image suivante montre le résultat de la détection des passages par zéro, à partir du résultat de la soustraction des images de niveau 5 et 6 obtenues dans la section précédente.



(a) (b)

Figure B.3 – Détection des passages par zéro d'une DoG : a) image originale, b) contours.

Les images des 2 grilles sont soustraites et le signe de chaque pixel est placé dans la mémoire locale de 16 bits, 2 lignes à la fois (2 instructions / pixel). Ensuite, la mémoire est parcourue et le résultat est placé dans la grille (6 instructions / pixel).

B.3.3 Résultats de simulation RTL

La simulation RTL avec le banc d'essai en *e* retourne exactement les mêmes images que celles qui ont été obtenues du modèle de haut-niveau.

B.3.4 Résultats de synthèse

Synthétisé pour un FPGA Xilinx XCV2000E, le coprocesseur occupe 73736 éléments de logique (LUT). Sans optimisation, il peut fonctionner à une fréquence de 34.8 MHz.

B.3.5 Optimisations futures

La fréquence et la surface du coprocesseur pourraient être améliorés de plusieurs façons. D'abord, l'UAL pourrait être allégée des opérations qui ne sont pas utilisées. D'autre part, la fréquence pourrait être augmentée en pipelinant sa logique. Le nombre de PE pourrait être augmenté pour accroître le parallélisme de la machine, ou bien diminué pour réduire sa surface. Enfin, la grille est faite de telle sorte que les données passent de l'autre côté. Cela nécessite des longs fils qui peuvent ralentir le circuit. Il serait possible de remplacer cette fonctionnalité par des registres en périphérie qui serviraient à conserver les valeurs.

La machine a été laissée dans son état actuel pour lui laisser un maximum de flexibilité. Une fois intégrée, le remaniement de quelques constantes dans le code VHDL ferait en sorte de l'adapter davantage à l'utilisation qui lui est destinée.

B.4 Description détaillée de l'architecture

B.4.1 Principe de fonctionnement

La machine présentée ici est un accélérateur de calcul destiné à effectuer des opérations de bas-niveau sur des images. Elle fonctionne avec le principe des processeurs SIMD, *Single-Instruction Multiple-Data*, où chaque élément de traitement (*Processing Element, PE*) exécute la même instruction sur des pixels différents. Elle est principalement constituée d'une grille 2D d'éléments de mémoire, munis de multiplexeurs faisant en sorte que le contenu (un pixel de l'image), peut se déplacer vers ses 4 voisins. Des PEs sont disposés dans cette grille. Pour l'instant, elle en possède 16, ce qui donne en première approximation un gain de vitesse théorique de 16. La machine est optimisée pour effectuer des convolutions rapides, notamment grâce à l'unité de multiplication-accumulation (MAC) des PEs, et à cause de l'architecture générale qui facilite les opérations sur les voisinages de pixels.

B.4.2 Description de l'architecture générale

Le schéma suivant montre l'architecture du système.

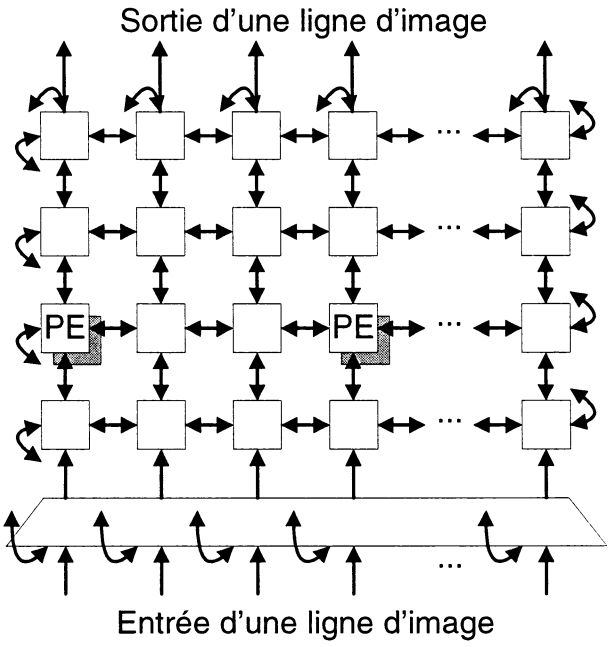


Figure B.4 – Architecture générale de l'accélérateur de calcul.

La grille d'éléments de mémoire est composée de 320x4 registres de 8 bits, accueillant chacun une valeur de pixel. Comme il a été mentionné plus tôt, les valeurs peuvent être décalées d'une case dans une des 4 directions (N, O, S, E) avec une instruction appropriée. Toutes les valeurs se déplacent en même temps. Il y a 2 grilles superposées, de manière à pouvoir stocker des images intermédiaires. Le tampon est circulaire, ce qui fait que les valeurs à une extrémité se retrouvent de l'autre côté, à l'exception du côté SUD qui reçoit ou bien les valeurs du côté NORD, ou bien une ligne d'image en entrée (selon la valeur du fanion FLAG_I_CONNECT du mot d'instruction, décrit plus loin). Ainsi, les données ne sont pas perdues suite à un décalage. Les cases ombragées sont des registres spéciaux, surmontés par un PE. Avec une instruction appropriée, les valeurs de sortie des PEs sont enregistrées dans ces cases. La disposition des PEs dans cette configuration s'est avérée très efficace pour l'exécution optimale de la plupart des algorithmes choisis.

Le système reçoit un mot d'instruction, qui est envoyé aux grilles et aux PEs. Il reçoit également, sur le côté SUD de la grille, une ligne de l'image. Il s'agit d'un vecteur de 320x8 bits. Lorsque le fanion FLAG_I_CONNECT du mot d'instruction est '1', jumelé à une opération de décalage vers le NORD, la ligne d'image en entrée remplace la ligne de registres du côté NORD qui devrait normalement y revenir, étant donnée la circularité de la mémoire. Lorsque le fanion FLAG_O_CONNECT du mot d'instruction est '1', un signal de sortie indique à l'interface externe qu'il est temps de récupérer la ligne d'image sortante, présente sur le côté NORD.

B.4.3 Architecture d'un élément de traitement (PE)

La figure B.5 ci-dessous montre l'architecture d'un élément de traitement, qui est grossièrement basée sur l'implémentation du système PULSE [55].

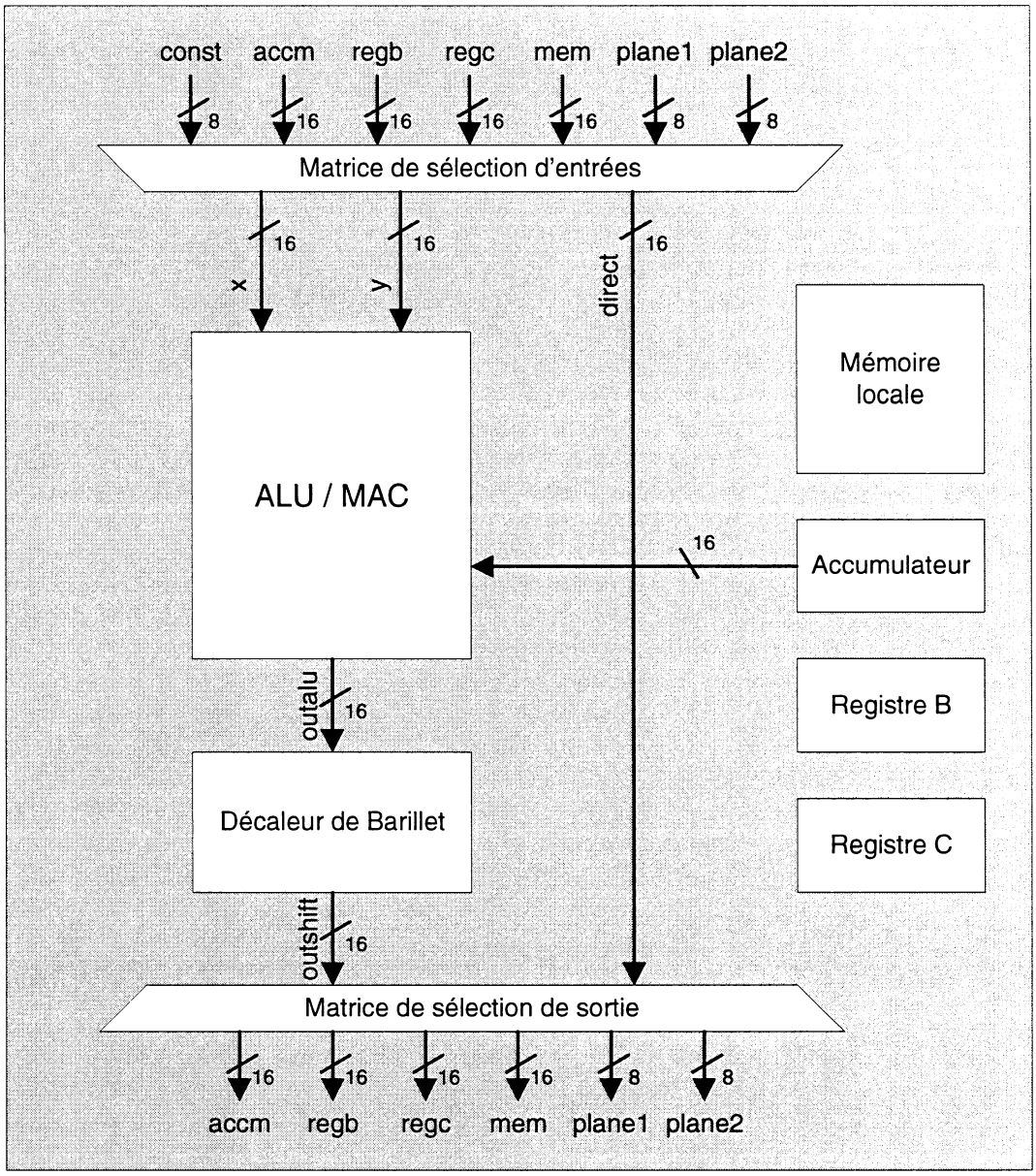


Figure B.5 – Architecture d'un élément de traitement.

Il s'agit d'un bloc combinatoire non pipeliné, contenant trois multiplexeurs de sélection d'entrée, un UAL simple, une unité multiplicateur-accumulateur (MAC), un décaleur Barillet, et une matrice de sélection de sortie. Les entrées X et Y sont routées vers les unités de calcul, alors que l'entrée DIRECT est routée directement vers la matrice de sortie. Il est ainsi possible d'exécuter un calcul en même temps que l'on stocke directement une valeur dans un

registre. Le PE possède également un accumulateur 16 bits, un fichier de registres à 2 registres à 2 ports de 16 bits, et une mémoire locale à un port, contenant 32 mots de 16 bits.

Les multiplexeurs d'entrées assignent les signaux X, Y et DIRECT parmi les sources possibles : registre de la grille 1, de la grille 2, accumulateur, fichier de registres, mémoire locale, ou valeur constante comprise dans le mot d'instruction.

L'UAL exécute des opérations logiques simples sur les entrées X et Y de 16 bits : NAND, AND, NOR, NOT, SLT, SEQ. Les deux dernières sont *Set if Lower Than* et *Set if Equal*, et sont très utiles pour la détection des passages par zéro, nécessaires pour la détection de contours basée sur la dérivé seconde (voir la section B.6).

L'unité MAC reçoit en entrée deux valeurs de 8 bits, i.e. les 8 bits les moins significatifs des entrées X et Y (rappelons que les registres de la grille comportent 8 bits). Elle additionne le produit de ces deux valeurs au contenu de l'accumulateur. Un champ du mot d'instruction permet de choisir le contrôle du signe pour le produit (s*s, s*u, u*s, u*u).

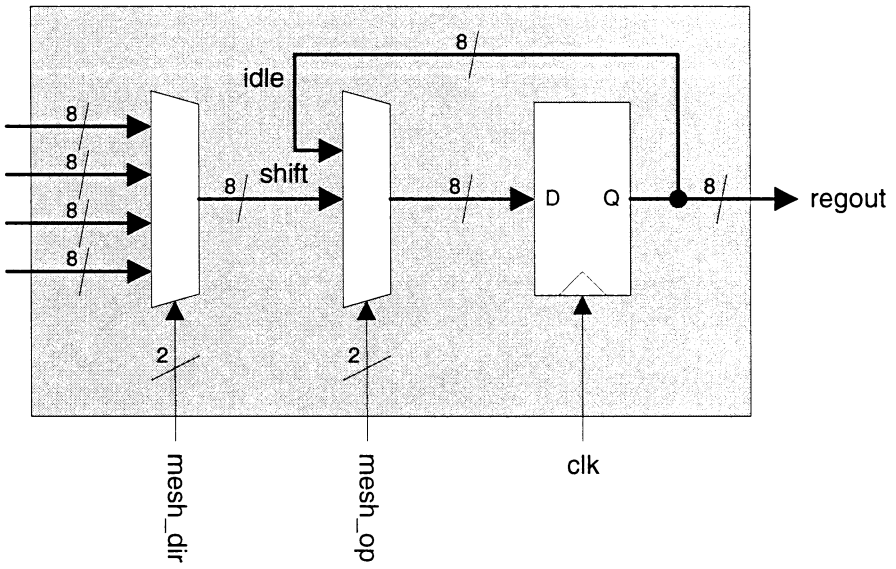
Le décaleur de Barillet permet des décalages logiques et arithmétiques, entre –16 et 15 (les décalages négatifs sont vers la droite). Il ne fait pas de doute qu'une optimisation future consistera à diminuer sa plage et sa flexibilité.

La matrice de sortie permet de stocker la valeur directe ou la valeur calculée (sortie du décaleur) dans une des destinations possibles : registre de la grille 1, de la grille 2, accumulateur, fichier de registres et mémoire locale.

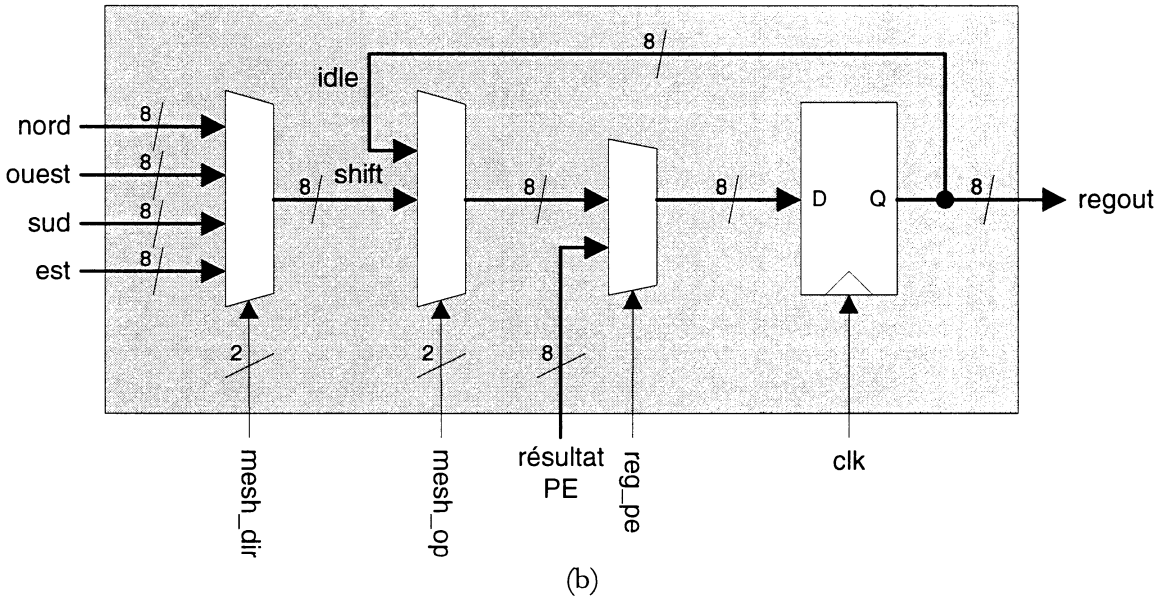
L'adresse de la mémoire est sélectionnée par la valeur du champ ADDR/CONST du mot d'instruction (5 bits les moins significatifs). Ce champ sert également de constante en entrée; il est donc impossible d'accéder à la mémoire locale si on utilise l'entrée constante (à moins qu'il ne s'agisse du même nombre). La mémoire locale ne peut pas être lue et modifiée dans la même instruction parce qu'elle n'a qu'un port.

B.4.4 Description d'un élément de mémoire de la grille

La figure B.6 a) montre un élément de mémoire normal de la grille. Les éléments de mémoire sous-jacents et voisins aux PEs sont spéciaux, étant donné qu'ils peuvent recevoir la valeur de sortie du PE. La figure B.6 b) montre cette configuration spéciale.



(a)



(b)

Figure B.6 – Élément de mémoire de la grille : a) normal, b) spécial.

Ainsi, les éléments spéciaux ont la particularité de recevoir une valeur externe, avec le signal de commande approprié. Ce signal de commande est global pour toute la grille, et est généré par un simple décodeur qui l'active si l'instruction commande un enregistrement de données des PEs dans la grille. Nous avons dit que ces registres spéciaux étaient non seulement situés sous les PEs, mais à leurs voisins également. La raison est que lorsqu'il y a sauvegarde d'un calcul dans la grille en même temps qu'un décalage des données, il doit être sauvegardé dans le registre voisin. Le choix du registre est déterminé par le décodeur. Ainsi, le décalage de la grille et les calculs des PEs peuvent toujours se faire en parallèle.

B.4.5 Description du mot d'instruction

Le mot d'instruction comporte 45 bits. Ses divers champs s'appliquent sur des ressources différentes, ce qui fait que très peu de décodage est nécessaire. Il permet 3 opérations en parallèle : un calcul (avec les entrées X et Y), un stockage direct (entrée DIRECT), et une opération de grille.

La figure B.7 montre la répartition des champs du mot d'instruction. Le tableau B.1 contient une description des champs.

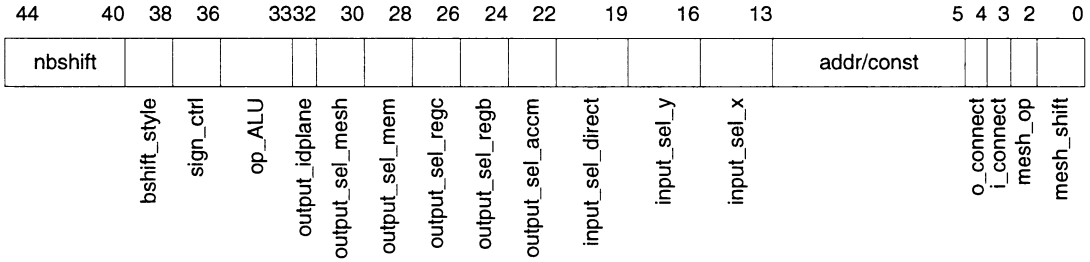


Figure B.7 – Mot d'instruction.

Tableau B.1 – Champs du mot d'instruction.

Champ	Nb bits	Valeurs possibles	
		Signification	Binaire
mesh_shift	2	SHIFT_N	00
		SHIFT_O	01
		SHIFT_S	10
		SHIFT_E	11
mesh_op	2	MESH_IDLE	0
		MESH_SHIFT	1
flag_i_connect	1	WRAP	0
		IO_CONNECT	1
flag_o_connect	1	NO	0
		OUT_READY	1
addr/const	8	0x00 - 0xFF	
input_selx	3	SELIN_CONST	000
		SELIN_MEM	111
input_sely	3	SELIN_MESH1	010
		SELIN_MESH2	011
input_seldirect	3	SELIN_ACCM	100
		SELIN_REGB	101
		SELIN_REGC	110
output_selaccm	2	SELOUT_NONE	00
output_selregb	2	SELOUT_OUTPE	10
output_selregc	2	SELOUT_DIRECT	11
output_selmem	2		
output_selmesh	2		
output_idmesh	1	MESH_1	0
		MESH_2	1
op_alu	3	ALU_NAND	000
		ALU_AND	001
		ALU_NOR	010
		ALU_SLT	101
		ALU_SEQ	011

		ALU_INV	100
		ALU_MAC	111
signctrl	2	SIGN_S_S	00
		SIGN_S_U	01
		SIGN_U_S	10
		SIGN_U_U	11
bshiftstyle	2	SHIFT_LOGICAL	00
		SHIFT_ARITH	01
nbshift	5	[-16, 15]	

B.5 Modèle de haut-niveau en C++

B.5.1 Introduction

Un modèle de haut-niveau de l'accélérateur de calcul a été codé en C++ orienté-objet. Chaque module matériel est représenté par un objet, de même que le mot d'instruction, un contrôleur et un compilateur. L'architecture logicielle de ce modèle est décrite dans la sous-section B.5.2.

Ce modèle était nécessaire dans la conception de la machine. D'abord, il a permis de valider le fonctionnement des grilles. Ensuite, il a permis de déterminer les outils de calcul nécessaires dans le PE. Des algorithmes ont pu être implémentés, et des constatations ont été faites par rapport aux besoins en termes de ressources, de manière à rendre l'implémentation de la plupart des algorithmes optimale. Par exemple, la disposition des PEs dans la grille, la flexibilité des matrices de sélection des entrées et des sorties, les opérations de l'UAL, la présence d'une unité MAC, la gestion du contrôle du signe pour la multiplication, la flexibilité et la méthode de décalage du décaleur Barillet, la taille de la mémoire, etc.

Le modèle de haut-niveau permet de faire un lien entre les algorithmes développés avec Matlab et les algorithmes tels qu'ils seront exécutés par la machine. L'écriture d'algorithmes

dédiés à la machine est faite via un compilateur symbolique, qui lit un fichier texte et le transforme en instructions exécutables par le modèle. Plus de détails au sujet de ce compilateur sont donnés dans la sous-section B.5.3. Lorsqu'un onglet spécial est coché, le modèle sauvegarde, sous forme de fichiers texte, l'image originale, l'image finale, et les instructions en binaire. Ces trois fichiers seront utilisés plus tard par le banc d'essai en *e* du modèle VHDL pour vérification. Ainsi, la méthodologie normale de développement d'algorithmes de traitement d'image pour l'accélérateur présenté ici est celle-ci :

1. développement de l'algorithme sur Matlab,
2. écriture en langage symbolique pour la machine,
3. vérification du résultat,
4. impression des instructions en format binaire pour utilisation matérielle.

B.5.2 Diagramme de classes

La figure B.8 montre le diagramme de classes du modèle objet de haut-niveau de la machine.

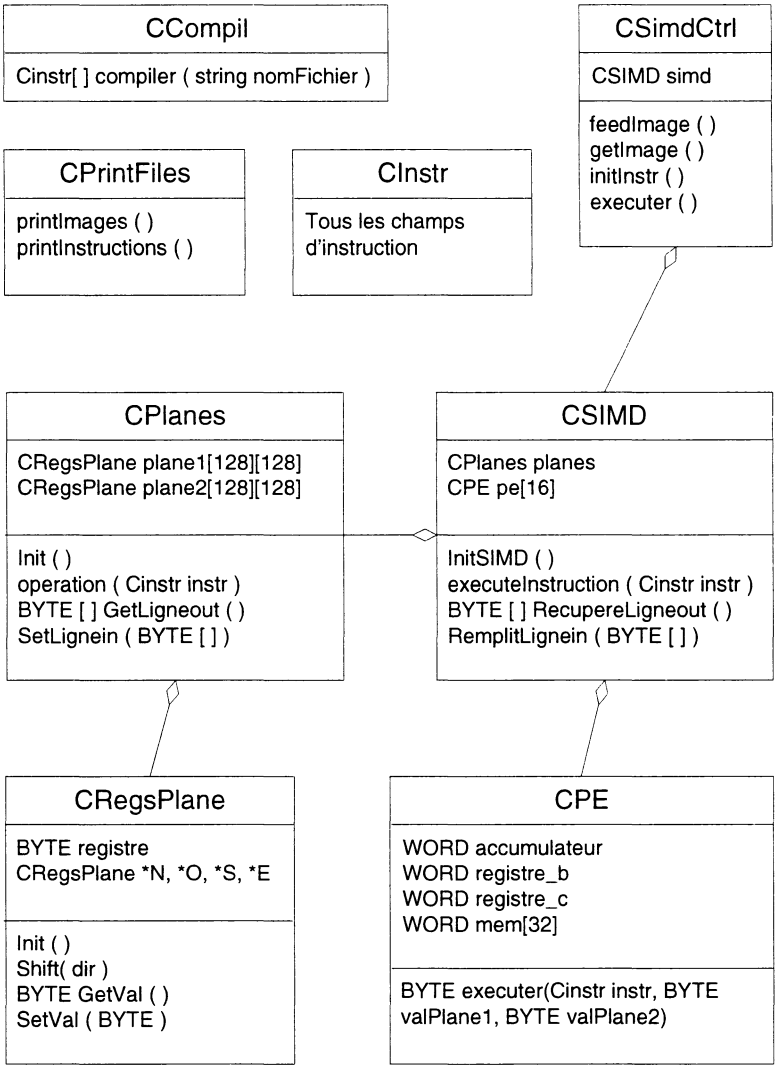


Figure B.8 – Diagramme de classes du modèle objet de haut-niveau.

L'objet CPlanes comporte 2 tableaux de 320x4 éléments de mémoire CRegsPlane, gère les opérations de grilles, comme les décalages, et offre des méthodes d'accès pour l'entrée/sortie des lignes d'images et l'enregistrement des résultats de calcul des PEs. La classe CPE contient les ressources d'un PE et simule son exécution. L'objet CSimd contient l'objet CPlanes et les objets CPE. Sa fonction d'initialisation instancie et branche les PEs selon la configuration choisie. La configuration retenue est celle qui est présentée dans le schéma-bloc de la figure B.4 (voir section B.4.2). Il va sans dire qu'une modification à cette configuration nécessite un nouvel algorithme pour un traitement d'image donné. Il possède

une fonction qui appelle la fonction d'exécution de chaque PE avec les bonnes données, appelle l'exécution des opérations des grilles, sauvegarde les résultats de calcul aux bons endroits dans le grille et gère les entrées/sorties, de manière à refléter fidèlement ce qui se passera dans la version matérielle. L'objet CCtrlSimd possède l'objet CSimd. Il s'occupe des fournir et récupérer des lignes d'image au SIMD, et gère les boucles d'instructions. Il représente le système embarqué devant contrôler l'accélérateur de calcul.

La classe CInstr est une structure contenant les champs d'instruction. La classe CPrintFiles sert à sauvegarder les images et les instructions sous forme de fichiers textes, lisibles par le banc de test matériel en *e*. La classe CCompil sert quant à elle à interpréter les fichiers texte contenant des instructions en langage symbolique.

B.5.3 Langage symbolique et compilateur

Le modèle de haut-niveau est muni d'un compilateur d'instructions écrites en langage symbolique. L'ajout d'une telle fonctionnalité s'est avérée indispensable lorsqu'il a fallu traduire des algorithmes de traitement d'image écrits en Matlab pour une exécution séquentielle à des instructions pour l'accélérateur de calcul exécutés de manière parallèle. L'écriture d'algorithmes de la moindre complexité directement dans le code devenait rapidement très laborieuse.

Un langage symbolique et son compilateur ont donc été conçus afin de faciliter l'élaboration des algorithmes pour la machine. Il s'agit d'un langage d'assemblée qui demande cependant d'avoir une connaissance suffisante de l'architecture de la machine. Une ligne de code permet de faire exécuter 1 à 3 opérations en parallèle à l'intérieur d'un cycle : calcul et/ou stockage direct et/ou opération de grilles. Ces opérations parallèles sont séparées par le symbole « || ». Voici un exemple :

```
p1 c, b macsu $128, l-10 || b, c || sh N i
```

Cette ligne possède 3 opérations en parallèle. La première est une opération de calcul : le registre b est considéré signé et est multiplié à la constante 128 (en décimale), le tout est additionné à l'accumulateur « a » (car c'est une opération MAC). Le résultat est décalé de 10 bits vers la droite (décalage logique) et stocké dans la grille 1, ainsi que dans le registre c. Ainsi, le matériel sélectionne le registre b pour l'entrée X, la constante 0x80 pour l'entrée Y, l'opération MAC et le contrôle du signe S*U pour l'unité de calcul, un décalage logique de 10 bits vers la droite pour le décaleur Barillet, et la grille 1 et le registre c conserveront le résultat du calcul. La 2^{e} opération parallèle est un stockage direct : la valeur de c est stockée dans b. Notons que l'opération est légale en autant que les destinations des 2 opérations soient différentes, i.e. qu'on ne peut stocker 2 résultats différents dans la même ressource. Le compilateur comporte des mécanismes de vérification pour éviter cela. Ainsi, à la fin de l'instruction, le registre b contiendra la valeur initiale de c, et le registre c contiendra le résultat de la MAC avec la valeur initiale du registre b. Le matériel sélectionne le registre c pour l'entrée DIRECT, et assigne ce signal au registre b. La 3^{e} opération parallèle est une opération de grille. Il s'agit d'un décalage (« sh » pour *shift*) vers le NORD, avec fanion FLAG_I_CONNECT égal à '1'.

Voici la syntaxe générale d'une ligne de code :

```
{ [destination(s)], [source_1] [opération] [source_2]{,
[style_décalage][nombre_bits_décalage]}}{ || [destination(s)],
[source]}{ || [opération_grilles]{ [direction_décalage]}{
[FLAG_IO_CONNECT]}}
```

Une structure de boucle a également été implantée. Il s'agit de boucles simples, sans conditions à évaluer. La syntaxe est :

```
for 3
. . .
end
```

Le tableau B.2 résume les symboles acceptés.

Tableau B.2 – Symboles valides du langage d'assemblée pour l'accélérateur de calcul.

Symboles	**Description**
Ressources	
a	accumulateur
b	registre b
c	registre c
m(x)	mémoire locale (adresse)
$x	constante
p1	grille 1
p2	grille 2
Opérations	
nand	NAND
and	AND
nor	NOR
slt	SLT
seq	SEQ

inv	NOT
mac	MAC S*S
macss	MAC S*S
macsu	MAC S*U
macus	MAC U*S
macuu	MAC U*U
Décalage	(x = [-16,15])
lx	Logique
ax	Arithmétique
Grilles	
sh	Décalage
N	arg. NORD
O	arg. OUEST
S	arg. SUD
E	arg. EST
i	I_CONNECT = 1
o	O_CONNECT = 1
Autres	
for x	boucle x fois jusqu'à *end*
end	fin de boucle

B.6 Algorithmes de traitement d'image

B.6.1 Introduction

Les algorithmes implémentés pour l'accélérateur de calcul ont été décrits dans la section B.3.2. La présente section a pour but de décrire en détail l'exécution de ces algorithmes par la machine.

B.6.2 Seuillage

Le seuillage consiste à couvrir tous les pixels de la grille 1. Pour chaque pixel, une opération *Set Lower Than* (SLT) est exécutée entre la valeur du pixel et une valeur constante de seuil, dans ce cas-ci, 100 (sur 255). La figure B.9 montre l'ordre de visite des pixels par un PE pour 2 lignes. Puisqu'il y a 40 PEs et que chaque PE couvre 8 pixels par ligne, les 320 pixels d'une ligne d'image sont couverts. L'algorithme prend un cycle par pixel (l'opération SLT et un décalage de la grille en parallèle, comme le montre le listing 3.1). Le seuillage d'une image complète (320x240 pixels) prend donc 1920 cycles d'horloge et est optimal.

16	15	14	13	12	11	10	9
1	2	3	4	5	6	7	8

Figure B.9 – Ordre de visite des pixels pour le seuillage, pour un PE donné.

```
p1, p1 slt $100 || sh 0
```

Listing 3.1 – Opération de seuillage.

B.6.3 Convolution 3x3

La convolution d'image avec un masque 3x3 consiste en ceci : pour chaque pixel, on multiplie ses voisins immédiats par le coefficient du masque correspondant et on les additionne. Cela se fait très efficacement par la machine grâce à son unité MAC. En un cycle, on multiplie la valeur du registre de la grille sous le PE par le bon coefficient et on l'additionne à la valeur de l'accumulateur, tout en décalant la grille pour la prochaine valeur.

Le résultat complet est sauvegardé dans la 2[e] grille. Une convolution 3x3 prend donc 9 cycles par pixel. La figure B.10 montre l'ordre de visite du voisinage pour un pixel donné. À la fin, on se retrouve déjà en position pour calculer la valeur du prochain pixel, à droite. Le listing 3.2 montre une opération de MAC et l'opération finale de sauvegarde du résultat dans la grille.

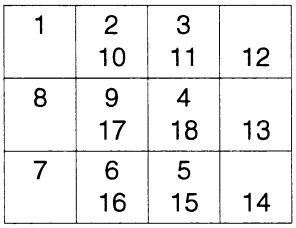


Figure B.10 – Ordre de visite des pixels d'une convolution 3x3, pour un pixel donné.

a)

```
a, $1 macsu p1 || sh 0
```

b)

```
p2, $1 macsu p1 || sh S
```

Listing 3.2 – a) Opération de MAC, b) sauvegarde du résultat dans la *grille*.

Après les 8 pixels de la ligne, il faut exécuter un décalage des pixels vers l'est afin de replacer l'image pour l'introduction de la prochaine ligne dans la grille. Cela rajoute donc 7 cycles par ligne d'image, et donne une efficacité de 9.875 cycles par pixel, ce qui est presque optimal. L'algorithme pourrait être amélioré en introduisant une ligne supplémentaire de l'image originale, et en calculant le résultat de deux lignes à la fois. On ne perdrait alors qu'une instruction de décalage par ligne, et cela donnerait un coût de 9.1 cycles par pixel.

Il faut faire très attention ici au problème de la représentation des nombres. Un lissage gaussien ne se fait qu'avec des coefficients positifs, et la norme est unitaire, de telle sorte que le résultat se représentera toujours sur 8 bits. Il est donc possible de conserver la nouvelle image directement dans la grille. Il faut également s'assurer qu'il n'y a pas de risque de dépassement dans l'accumulateur de 16 bits, ce qui est généralement le cas pour une convolution 3x3. Typiquement, on multiplie les coefficients par une puissance de 2, afin d'augmenter la précision, et ce n'est qu'à la fin où le résultat est divisé (par le décaleur de Barillet) avant d'être enregistré dans la grille.

Cependant, un Laplacien de Gaussienne (LoG) comporte toujours des valeurs négatives. Le résultat est alors un nombre entier signé qu'il n'est pas possible de sauvegarder adéquatement dans un registre de 8 bits. Typiquement, un masque de ce type est utilisé pour trouver les contours, qui se trouvent alors aux passages par zéro. Une bonne stratégie est donc d'appliquer le masque sur une petite partie de l'image, mais de sauvegarder les résultats dans la mémoire locale de 16 bits du PE. Ensuite, la détection de passages par zéro est effectuée sur les valeurs de la mémoire (voir plus loin) et on revient placer les résultats (0x00 ou 0xFF) dans la grille.

B.6.4 Convolution 5x5

La convolution 5x5 suit le même principe que la convolution 3x3. La figure B.11 illustre l'ordre de visite des pixels voisins. Elle prend 25 cycles par pixel, plus 8 cycles par ligne pour replacer l'image, ce qui donne 26 cycles par pixel. Une amélioration similaire à celle décrite dans la sous-section précédente pourrait être faite afin de rendre l'algorithme optimal.

15	14	13	12	11
16	17	24	23	10
1	18 1	25	22	9
2	19	20	21	8
3	4	5	6	7

Figure B.11 – Ordre de visite des pixels pour une convolution 5x5.

B.6.5 Convolution 7x7

La convolution 7x7 fonctionne selon le même principe que les autres. La figure B.12 illustre l'ordre de visite des pixels voisins. Elle prend 49 cycles par pixel, plus 8 cycles par ligne pour replacer l'image. Étant donné la grandeur des coefficients, il faut parfois sauvegarder un résultat temporaire dans le registre c pour éviter un possible dépassement (*overflow*). Dans le présent exemple, cela rajoute 6 cycles par pixel (6 stockages intermédiaires). Le coût total de cet algorithme est donc de 56 cycles par pixel.

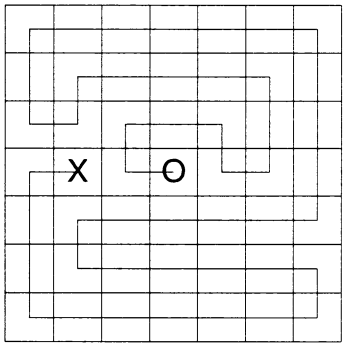


Figure B.12 – Ordre de visite des pixels pour une convolution 7x7 (le « X » est la position de départ, le « O » la position d'arrivée où le résultat est enregistré).

B.6.6 Différence de Gaussienne

Il est possible de montrer qu'un Laplacien de Gaussienne (LoG) s'approxime bien en soustrayant deux images lissées avec des gaussiennes d'échelle légèrement différente ($\sigma_1/\sigma_2 \approx 1.6$). Cette opération est appelée Différence de Gaussiennes (DoG) [54]. Le but est qu'il est possible de calculer des LoG à grande échelle avec des petits masques. La détection de contours qui en résulte ne conserve alors que les arêtes importantes. D'autre part, il est possible de montrer également que l'application d'un lissage gaussien d'échelle σ sur une image déjà lissée avec le même masque équivaut à un lissage d'échelle $\sigma\sqrt{2}$ sur l'image originale. Ainsi, on applique successivement le même masque sur l'image originale. Les deux dernières images obtenues pourront ensuite être soustraites et cela donnera une bonne approximation d'un LoG à grande échelle.

Un algorithme de DoG a été implémenté. Il applique un masque gaussien 7x7, $\sigma_0 = 1.0$, sur l'image, qu'il conserve dans la grille 2. La sauvegarde dans la grille est permise étant donné que le résultat du lissage est toujours compris entre 0 et 255, pour tous les pixels. Il applique ensuite de nouveau le même masque sur cette image, et conserve le résultat dans la grille 1. Cela est fait successivement 6 fois. À la fin, la grille 1 contient l'image de niveau 6, et la grille 2 contient l'image de niveau 5. Il ne reste qu'à les soustraire pour obtenir un DoG de $\sigma = \sigma_0(\sqrt{2})^{6-1} = 6$. Cependant les images soustraites ne peuvent pas être placées dans la grille, étant données les considérations de représentation des nombres mentionnées plus tôt. Les 6 lissages coûtent 6 * 1024 pixels/lissage * 56 cycles/pixel = 344064 cycles.

B.6.7 Détection des passages par zéro

Détecter les passages par zéro en recherchant les zéros est une approche bien naïve et inefficace. Un vrai détecteur de passages par zéro consiste à utiliser une fenêtre 2x2. Pour chaque bloc de valeurs, on regarde leur signe. S'ils n'ont pas tous le même signe, on identifie ce bloc comme un contour en assignant 0xFF au pixel en haut à gauche [79].

Après l'application de l'algorithme de DoG, les grilles contiennent chacune une image qu'il faut soustraire l'une de l'autre. Le résultat de la soustraction ne peut pas être conservé dans la grille. Au lieu de cela, on parcourt deux lignes en soustrayant les valeurs des 2 grilles, en déterminant leur signe avec une opération SLT, et en les stockant dans la mémoire. La figure B.13 montre l'ordre de parcours des pixels, et l'adresse de la mémoire à laquelle ils sont sauvegardés. Le listing 3.3 montre les opérations effectuées pour chacun.

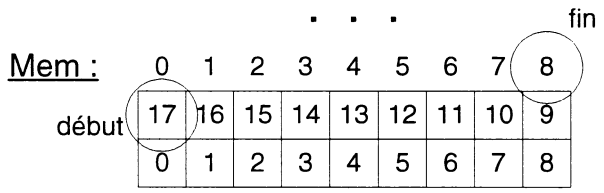


Figure B.13 – Ordre de parcours des pixels pour la détermination du signe. Les chiffres représentent l'adresse de la mémoire dans laquelle sont stockés les signes. Le cercle « début » indique le début du parcours pour le marquage des contours. À la fin, le pixel marqué du cercle « fin » se trouve sous le PE, prêt à calculer les passages par zéro de la prochaine ligne.

```
a, p1
b, $-1
c, $0
a, p2 macus b || p1, $0 || sh 0
m(0), a slt c || a, p1
a, p2 macus b || p1, $0 || sh 0
m(1), a slt c || a, p1
. . .
```

Listing 3.3 – Soustraction, détermination du signe et sauvegarde en mémoire.

Lorsque le signe de 2 lignes a été sauvegardé en mémoire, on détermine si les pixels de la ligne du haut sont des contours. Pour chaque pixel, on prend une fenêtre 2x2 et on compare les valeurs de la mémoire. Les fonctions logiques SEQ, AND et NAND sont bien utiles ici. Les valeurs sont comparées par paire avec la fonction SEQ. Dès qu'il y a un changement de signe, elle retourne 0x00. Ces valeurs sont ensuite NAND-ées ensemble, ce qui fait que dès qu'il y a un 0x00 (i.e. un changement de signe), le résultat est 0xFF. Pendant ce temps, le *grille* est décalé afin d'aller porter le résultat au bon endroit. Ensuite, on répète les opérations pour la ligne suivante. Les valeurs de la ligne inférieures sont remplacées par les valeurs de la nouvelle ligne. Le listing 3.4 montre les opérations nécessaires pour détecter le passage par zéro d'une fenêtre. L'algorithme prend 66 opérations par ligne. La détection des contours avec cette méthode prend donc 344064 + 127*66 = 352446 cycles en tout, soit environ 11 ms @ 33 MHz.

```
c, m(17)
a, m(16) seq c
b, m(0) seq c
a, a and b || c, m(0)
c, m(1) seq c
p1, a nand c || sh 0
```

Listing 3.4 – Détection des passages par zéro d'une fenêtre 2x2 conservée en mémoire.

B.7 Description du modèle RTL en VHDL

B.7.1 UAL

B.7.1.1 Description fonctionnelle

L'UAL reçoit deux mots de 16 bits et retourne un mot de 16 bits. Le signal *opAlu* sélectionne l'opération logique à effectuer. Les opérations disponibles sont NAND, AND, NOR, INV, SLT, SEQ.

B.7.1.2 Interfaces au niveau système

- Opérandes : 16 bits.
- Résultat : 16 bits.
- Sélection de l'opération : 3 bits.

B.7.1.3 Interfaces logiques

Le tableau B.3 donne le détail des interfaces logiques de l'UAL.

Tableau B.3 – Interfaces logiques de l'UAL.

Port	Taille	Direction	Description
Interface opérandes			
x	16	entrée	opérande 1
y	16	entrée	opérande 2
Interface résultat			
outalu	16	Sortie	résultat
Interface contrôle			
opalu	3	entrée	sélection de l'opération

B.7.1.4 Commentaires

L'UAL contient des opérations (NOR et INV) qui ne sont utilisées par aucun algorithme présentement implémenté. Elles ont été conservée afin que le PE soit suffisamment flexible, à peu de frais.

B.7.2 Unité de multiplication-accumulation (MAC)

B.7.2.1 Description fonctionnelle

L'unité MAC reçoit deux mots, b et c, ainsi que la valeur de l'accumulateur, a, et retourne a + (b * c). Un signal de contrôle du signe est également présent pour la multiplication. Sont

permises les multiplications s*s, s*u, u*s et u*u. L'addition est toujours considérée signée (cela n'a d'influence que sur l'interprétation de l'*overflow*).

B.7.2.2 Interfaces au niveau système

- Opérandes b et c: 8 bits.
- Opérande a : 16 bits.
- Résultat : 16 bits. La multiplication de b et c donne un mot de 16 bits, additionné à un autre mot de 16 bits.
- Contrôle du signe : 2 bits.

B.7.2.3 Interfaces logiques

Le tableau B.4 donne le détail des interfaces logiques de l'unité MAC.

Tableau B.4 – Interfaces logiques de l'unité MAC.

Port	Taille	Direction	Description
Interface opérandes			
x	8	entrée	opérande b
y	8	entrée	opérande c
a	16	entrée	accumulateur (a)
Interface résultat			
outalu	16	sortie	résultat b*c+a
Interface contrôle			
signctrl	2	entrée	contrôle du signe

B.7.2.4 Commentaires

Il n'est pas certain que le style d'écriture en VHDL du contrôle du signe pour la multiplication donne des résultats optimaux en termes de ressources. Il faudra investiguer davantage.

B.7.3 Décaleur de Barillet

B.7.3.1 Description fonctionnelle

Le décaleur de Barillet (*Barrel shifter*) opère des décalages de bits sur des mots de 16 bits. Il s'agit de divisions ou de multiplications rapides par des puissances de 2. Pour l'instant, deux styles de décalage sont implémentés, soient le décalage logique et le décalage arithmétique. Le décalage logique insère des zéros, alors que le décalage arithmétique vers la droite insère la valeur du bit le plus significatif (conserve le signe). Un signal de contrôle gère le style de décalage, un autre indique la taille du décalage (entre –16 et 15).

B.7.3.2 Interfaces au niveau système

- Opérande : 16 bits.
- Résultat : 16 bits.
- Style de décalage : 2 bits. Il s'agit de « logique » ou « arithmétique ».
- Taille du décalage : 5 bits. C'est un entier signé entre –16 et 15.

B.7.3.3 Interfaces logiques

Le tableau B.5 donne le détail des interfaces logiques du décaleur de Barillet.

Tableau B.5 – Interfaces logiques du décaleur de Barillet.

Port	Taille	Direction	Description
Interface opérandes			
wordin	16	entrée	mot d'entrée
Interface résultat			
wordout	16	sortie	mot de sortie
Interface contrôle			
bshiftstyle	2	entrée	style de décalage
nbshift	5	entrée	nombre de bits

B.7.3.4 Commentaires

Aucun algorithme n'a nécessité le style de décalage en rotation. Le décaleur est composé de 2 étages de multiplexeurs. Il est probablement beaucoup trop flexible pour nos besoins, et donc coûteux en ressources. Une optimisation possible de la machine consisterait à le simplifier. Il serait alors peut-être envisageable de mettre plus de PEs dans la machine.

B.7.4 Grilles

B.7.4.1 Description fonctionnelle

La grille est un tapis circulaire d'élément de mémoire, constitués de multiplexeurs et d'un registre de 8 bits. Ils sont disposés en grille 2D de 320x240 éléments, et chaque élément est connecté à ses 4 voisins, de telle sorte que par une opération appropriée, les données de chaque élément peuvent se déplacer dans l'une ou l'autre des 4 directions : NORD, SUD, EST, OUEST. Aux frontières, les données voyagent jusqu'à la frontière opposée. La frontière NORD est connectée à un port de sortie (*ligneout*). La frontière SUD est connectée à un multiplexeur qui sélectionne ou bien la frontière NORD, ou bien le port d'entrée *lignein*. Un bit du mot d'instruction permet de faire la sélection. Des ports d'entrée et de sortie sont reliés à des éléments de mémoire spéciaux, servant d'interface avec les PEs. Le module *grilles* contient en fait 2 grilles, la seconde étant utilisées pour conserver des résultats intermédiaires.

B.7.4.2 Interfaces au niveau système

- Lignes d'image : vecteurs de 320x8 bits, entrée et sortie, branchés respectivement aux bords SUD et NORD de la grille 1.
- Instruction : champs du mot d'instruction contrôlant les grilles. Le champ PLANE_OP contrôle l'opération des grilles, qui peut être l'inactivité ou le décalage dans une des 4 directions cardinales. Le champ PLANE_SHIFT indique la direction de décalage des données dans le cas d'une opération de décalage. Le champ FLAG_I_CONNECT indique si le côté SUD est en mode circulaire (branché au bord côté) ou s'il est branché à la ligne d'image en entrée *lignein*. Le champ

FLAG_O_CONNECT ne sert quant à lui qu'à indiquer à l'interface de sortie de la machine qu'une ligne valide est prête à être récupérée sur la frontière NORD.

- Interface PE : vecteurs de N_PEx8 bits. Il y a 40 PEs. Un vecteur contient les données des registres spéciaux de la grille 1, l'autre celles de la grille 2, et un vecteur reçoit les données de sortie des PEs, branché à l'interne aux registres spéciaux. Des signaux de contrôle indiquent s'il faut enregistrer ces données, et dans quelle grille.

B.7.4.3 Interfaces logiques

La figure B.14 montre les interfaces logiques des grilles, résumées dans le tableau B.6.

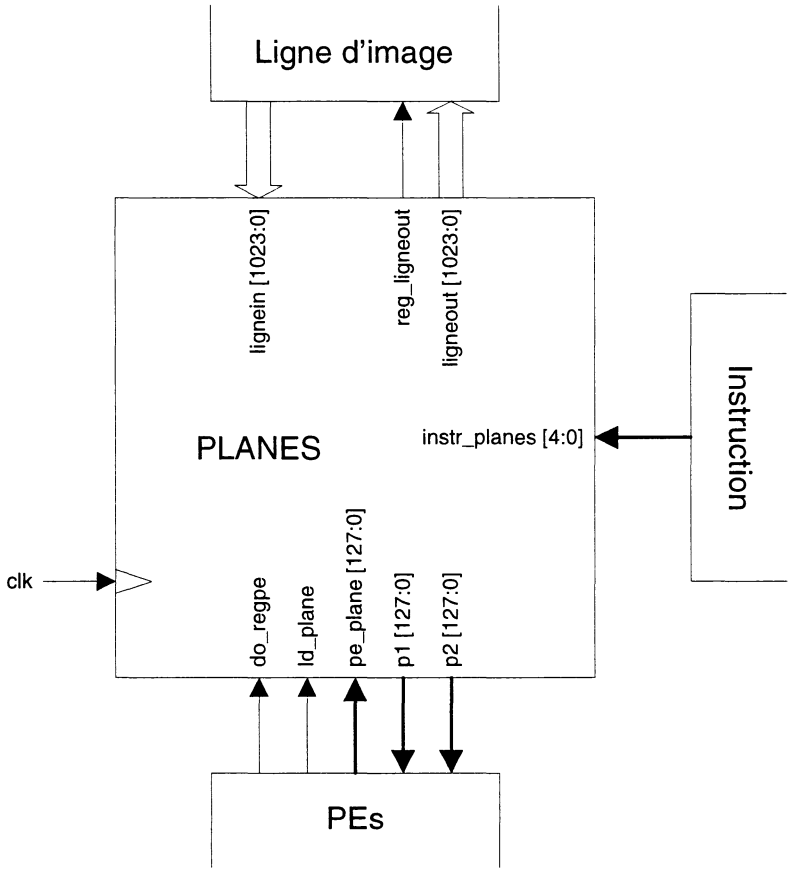


Figure B.14 – Interfaces logiques des grilles.

Tableau B.6 – Interfaces logiques des grilles.

Port	Taille	Direction	Description
Clk	1	entrée	Horloge
Interface lignes d'image			
lignein	128*8	entrée	ligne d'entrée
ligneout	128*8	sortie	ligne de sortie
reg_ligneout	1	sortie	signal de sortie valide
Interface instruction			
instr_grilles	5	entrée	PLANE_OP
			PLANE_SHIFT
			FLAG_IO_CONNECT
Interface PE			
do_regpe	1	entrée	doit registrer la sortie des PEs
id_grille	1	entrée	ID du grille devant registrer
pegrille	16*8	entrée	résultat des PEs vers les grilles
p1	16*8	sortie	sorties de la grille 1 vers les PEs
p2	16*8	sortie	sorties de la grille 2 vers les PEs

B.7.4.5 Commentaires

Les grilles constituent la plus grosse entité de la machine, grossièrement bâtie sur une mémoire. Cette mémoire contient environ 2.5 Ko, ce qui est très grand pour une mémoire embarquée. Ils pourraient être remplacés par une structure pyramidale, dont l'étage supérieur a un côté 2 fois plus petit que celui de l'étage précédent, et qui se prête bien aux algorithmes qui diminuent la résolution, comme les DoG. Une petite économie de mémoire serait réalisée. Une solution intéressante, permettant de justifier une aussi grande quantité de mémoire embarqué serait de jumeler la machine avec une caméra. Ainsi les grilles pourraient remplacer le tampon d'image, et renvoyer des images prétraitées ou non, en temps réel.

B.7.5 PE

B.7.5.1 Description fonctionnelle

Le PE est un élément de traitement. Il est constitué d'une matrice de sélection des entrées, une matrice de sélection des sorties, un UAL, une unité MAC, un accumulateur 16 bits, un fichier de registres comprenant 2 registres de 16 bits chacun, et une mémoire locale de 32 mots de 16 bits. Son architecture est décrite en détail dans la section B.4.3. Il n'est pas pipeliné. Ainsi, toutes les opérations se font en un cycle d'horloge. Il y a 3 entrées simultanées : 2 pour un calcul, et une autre pour du stockage direct dans ses ressources.

B.7.5.2 Interfaces au niveau système

- Interface grilles : mot de 8 bits provenant de la grille 1, mot de 8 bits provenant de la grille 2, sortie de 8 bits allant vers les grilles.
- Instruction : champs du mot d'instruction contrôlant les PEs. Il s'agit des champs pour contrôler l'UAL, la gestion du signe de l'unité MAC, la gestion du décaleur de Barillet, la sélection des entrées et la sélection des sorties. Un autre champ s'appelle ADDR/CONST, et est branché au port d'adresse de la mémoire locale, et comme source d'entrée externe.

B.7.5.3 Interfaces logiques

La figure B.15 montre les interfaces logiques du PE, résumées dans le tableau B.7.

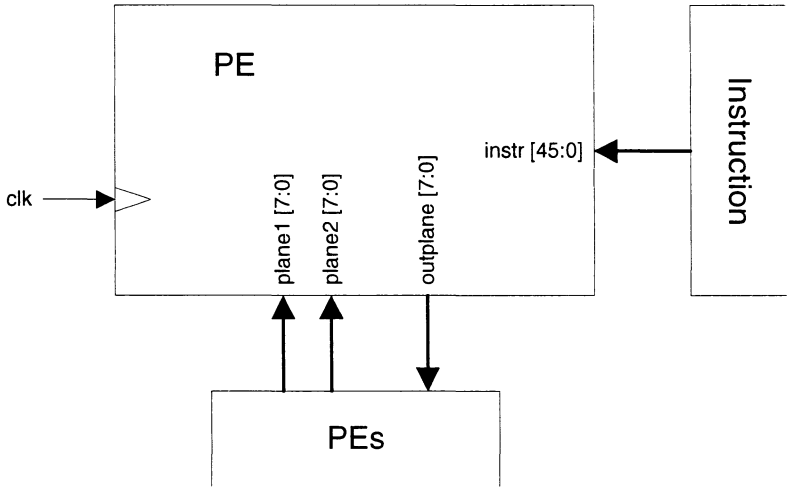


Figure B.15 – Interfaces logiques du PE.

Tableau B.7 – Interfaces logiques du PE.

Port	Taille	Direction	Description
Clk	1	entrée	horloge
Interface instruction			
Instr	45	entrée	mot d'instruction PE
Interface grilles			
grille 1	8	entrée	valeur du registre sous-jacent de la grille 1
grille 2	8	entrée	valeur du registre sous-jacent de la grille 2
outgrille	8	sortie	résultat du PE vers les grilles

B.7.5.5 Commentaires

Outre la complexité de ses composants UAL, MAC et décaleur de Barillet, la complexité des matrices de sélection des entrées/sorties pourraient être elles aussi diminuées légèrement. Une plus petite complexité des PEs permettrait d'en disposer davantage dans la machine. Cependant, la capacité de stocker directement des opérandes dans ses ressources simultanément à un calcul permet de faire de grandes économies de cycles dans la plupart des algorithmes implémentés.

B.7.6 SIMD

B.7.6.1 Description fonctionnelle

Le SIMD est le module de haut-niveau. Il contient la grille et les PEs, décrits plus haut. Ses interfaces sont les lignes d'image d'entrée et de sortie de la grille, ainsi que les mots d'instruction.

B.7.6.2 Interfaces au niveau système

- Lignes d'image : vecteurs de 320x8 bits, entrée et sortie, branchés respectivement aux bords SUD et NORD de la grille 1. Un signal de sortie, branché directement sur le fanion FLAG_O_CONNECT du mot d'instruction, indique si la sortie doit être registrée.
- Mot d'instruction : 45 bits.

B.7.6.3 Interfaces logiques

La figure B.16 montre les interfaces logiques du SIMD, résumées dans le tableau B.8.

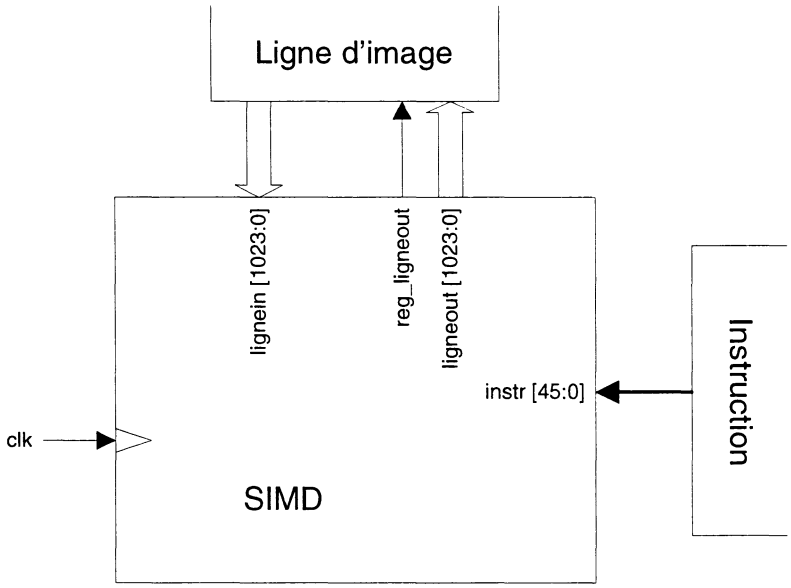


Figure B.16 – Interfaces logiques du SIMD.

Tableau B.8 – Interfaces logiques du SIMD.

Port	Taille	Direction	Description
clk	1	entrée	horloge
Interface lignes d'image			
lignein	128*8	entrée	ligne d'entrée
ligneout	128*8	sortie	ligne de sortie
regligneout	1	sortie	signal de sortie valide
Interface instruction			
instr	45	entrée	mot d'instruction

B.8 Description du banc de test en e

Le banc d'essai pour la machine a été codé en langage ***e*** et vérifié avec l'outil Specman Elite. Son architecture suit les principes de base énoncés dans [68]. Le design est entouré d'un anneau de vérification, défini par l'unité (*unit*) *simd_vring* (fichier *simd_vring.e*). L'anneau

contient deux générateurs, *simd_inst_gen* et *simd_lignein_gen*, correspondant aux deux interfaces d'entrée (voir les spécifications, section 4.1.6), et un moniteur, *simd_ligneout_mon*, correspondant à l'interface de sortie. Les générateurs sont des unités qui contiennent les méthodes pour charger des données de fichiers texte (images ou série d'instructions) et injecter les stimuli en appelant des routines de bas-niveau, et des signaux d'événements qui déclenchent ces méthodes de manière concourante. Le moniteur contient des méthodes pour récupérer les données de sortie du module en vérification (*Device Under Test* – DUT) en appelant une routine de bas-niveau, sauvegarder ces données dans un fichier texte, charger des données d'un fichier contenant le résultat attendu (généré par le modèle de haut-niveau décrit dans la section 2.3) et comparer. Les erreurs sont affichées dans la console de commande de Specman.

Les fichiers *simd_hdl_fct.e* et *simd_hdl_low.e* contiennent les routines et définitions requises pour l'interface de bas-niveau avec le DUT. Les routines sont incluses dans les unités dérivées des générateurs et du moniteur. Le générateur d'instruction contient la routine *send_instruction()* qui applique l'instruction reçue en argument. Le générateur de ligne d'image contient la routine *applique_ligne()* qui met la ligne d'image reçue en argument à l'entrée *lignein*, et le moniteur contient la routine *recupere_ligne()* qui fonctionne de façon similaire, mais dans l'autre sens.

Le fichier *simd_glogic.e* colle le tout ensemble. Une extension de l'anneau de vérification est implémentée dans le fichier de configuration *simd0.e*. Ce fichier contient toutes les méthodes de test de haut niveau. Typiquement, un test exécute les opérations suivantes :

1. appel des méthodes des générateurs qui s'occupent du chargement des fichiers de données de stimulation;
2. exécution du traitement par le démarrage concourant de la méthode de traitement du générateur d'instructions, de la méthode d'application successive des lignes d'image du générateur de lignes d'image, et de la méthode de récupération successive des lignes d'image en sortie du moniteur (ces deux dernières méthodes procèdent lorsque

les fanions FLAG_I_CONNECT et FLAG_O_CONNECT de l'instruction courante du générateur d'instructions sont respectivement '1');

3. attente de la fin du traitement;
4. appel des méthodes suivantes du moniteur pour la vérification : sauvegarde des résultats et comparaison avec les résultats attendus.

Il faut noter la différence entre les termes « appel » et « démarrage » de méthodes. Le premier est un appel de fonction qui donne le contrôle à la fonction appelée jusqu'au retour. Cela est utilisé pour les fonctions de gestions de fichiers. Le deuxième est le lancement d'une opération dans un différent fil d'exécution (*thread*). Le contrôle est immédiatement remis à la fonction appelante. Celle-ci va typiquement lancer plusieurs opérations simultanément, et attendre la fin qui est signalée par un événement. Ce sont des opérations de stimulation ou de récupération de données de sortie du DUT, qui consomment du temps de simulation.

Divers tests ont été implémentés, notamment un test qui exécute des mouvements de registres, vérifie la mémoire et le fonctionnement des grilles, un test qui exécute et vérifie toutes les opérations de l'UAL, et des traitements d'image, implémentés dans le modèle de haut-niveau.

ANNEXE C

SOURCES

Le fichier Readme.txt explique l'arborescence des répertoires des sources sur le CD.